Testing Phase Space Properties of Synchronous Dynamical Systems with Nested Canalyzing Local Functions

Daniel J. Rosenkrantz
University at Albany – SUNY
Albany, NY
drosenkrantz@gmail.com

Madhav V. Marathe*
Biocomplexity Institute of Virginia Tech
Blacksburg, VA
mmarathe@vt.edu

S. S. Ravi†
Biocomplexity Institute of Virginia Tech
Blacksburg, VA
ssravi0@gmail.com

Richard E. Stearns
University at Albany – SUNY
Albany, NY
thestearns2@gmail.com

ABSTRACT

Discrete graphical dynamical systems serve as effective formal models for simulations of agent-based models, propagation of contagions in social networks and study of biological phenomena. A class of Boolean functions, called nested canalyzing functions (NCFs), has been used as a good model of certain biological phenomena. Motivated by these biological applications, we study a variety of analysis problems for synchronous graphical dynamical systems (SyDSs) over the Boolean domain, where each local function is an NCF. We present intractability results for some properties as well as efficient algorithms for others. In several cases, our results clearly delineate intractable and efficiently solvable versions of problems.

KEYWORDS

Discrete dynamical systems, Boolean functions, Nested canalyzing functions, Phase space properties, Complexity, Algorithms.

ACM Reference Format:

Daniel J. Rosenkrantz, Madhav V. Marathe, S. S. Ravi, and Richard E. Stearns. 2018. Testing Phase Space Properties of Synchronous Dynamical Systems with Nested Canalyzing Local Functions. In Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. Andre, S. Koenig (eds.), Stockholm, Sweden, July 2018, IFAAMAS, 10 pages.

1 INTRODUCTION

1.1 Motivation

Discrete graphical dynamical systems, which are generalizations of cellular automata (CA) [16, 45], serve as an effective formal model for multi-agent systems (see, e.g., [41, 46]). They have also been used in many other contexts, including simulations of agent-based models, propagation of contagions in social networks, study of biological phenomena, and game theoretic settings (see, e.g., [9, 10, 21, 23, 32, 34, 43]). Here, we focus on synchronous discrete dynamical systems (SyDSs). Informally, a SyDS consists of an undirected graph whose vertices represent entities (agents) and edges represent local interactions among entities. Each vertex v has a Boolean state value and a local transition function f_v whose inputs are the current state of v and those of its neighbors; the output of f_v is the next state of v. The vector consisting of the state values of all the nodes at each time instant is referred to as the configuration of the system at that instant. In each time step, all nodes of a SyDS compute and update their states synchronously. Starting from a (given) initial configuration, the time evolution of a SyDS consists of a sequence of successive configurations. The SyDS formalism with different classes of local transition functions has been used in applications such as disease propagation in urban areas, diffusion of innovations, etc. (see, e.g. [6, 10, 43]).

In this paper, we study a class of graphical dynamical systems motivated by applications in systems biology. Many researchers have analyzed such models (see e.g., [14, 31, 39]); others have investigated their stability (see e.g., [17, 20, 26, 37]). Since the work by Waddington [44], the term canalization has been used to describe the stability of a biological system with changes in external conditions. In 1969, Kauffman [17] introduced a Boolean network model to explain the stability of gene regulatory networks. Kauffman found that the use of one class of Boolean functions (which he called canalyzing Boolean functions) in the model captured many observed properties of gene regulatory networks, including stability. The subclass of nested canalyzing functions (NCFs) was introduced later by Kauffman et al. [19] to facilitate a rigorous analysis of the Boolean network model for gene regulatory networks. A precise definition of NCFs (and a more general version of NCFs) is given in Section 2.1. Many researchers have studied mathematical properties of NCFs and have alluded to the importance of NCFs in modeling biological phenomena (e.g., [19, 20, 26–30]).

We consider several analysis problems for graphical dynamical systems whose node functions are NCFs. We use the

* Also with the Computer Science Department at Virginia Tech.
† Also with the University at Albany – SUNY.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. Andre, S. Koenig (eds.),
, July 2018, Stockholm, Sweden.

Problem	Result(s)
Reachability	PSPACE-complete even when the maximum node degree and the treewidth of the underlying graph are bounded (Section 3).
Predecessor Existence	NP-complete even when the maximum node degree is 3. The corresponding counting problem is #P-complete (Section 4). (The problem and the counting version are efficiently solvable when the maximum node degree is 2 [25].)
Fixed Point Existence	NP-complete even when the maximum node degree is 3. The corresponding counting problem is #P-complete (Section 5). (The problem and its counting version are efficiently solvable when the maximum node degree is 2 [35].)
Garden of Eden Existence	Efficiently solvable; when the answer is "yes", such a configuration can also be found efficiently (Section 6).

Table 1: Summary of Results Presented in the Paper

term NCF-SyDS to denote a SyDS where each local transition function is an NCF. Such analysis problems are studied by considering the **phase space** of the SyDS, which is a directed graph with one vertex for each possible configuration and a directed edge ([illegible]) from a vertex [illegible] to vertex [illegible] if the SyDS can transition from the configuration corresponding to [illegible] to the one corresponding to [illegible] in one time step. When an NCF-SyDS has a one step transition from a configuration [illegible] to a configuration [illegible] we say that [illegible] is the **successor** of [illegible] and that [illegible] is a **predecessor** of [illegible] Since NCF-SyDSs are deterministic, each configuration has a unique successor; however, a configuration may have zero or more predecessors. Each self loop in the phase space of a SyDS represents a **fixed point** of the actual system, that is, a configuration in which the system will stay forever. Also, any vertex in the phase space with no incoming edges represents a **Garden of Eden** (GE) configuration. Such a configuration cannot be reached during the evolution of a SyDS; it can only occur as an initial configuration.

1.2 Contributions and Their Significance

Our contributions (shown in Table 1) are explained below.

(1) The reachability problem asks whether a given NCF-SyDS starting from a given configuration [illegible] will reach another given configuration [illegible]. This problem formalizes the question whether a system modeled by an NCF-SyDS may reach an undesirable configuration in the future. (For example, in the disease propagation context, [illegible] may represent a situation in which a large number of agents are infected.) In Section 3, we show that the reachability problem for NCF-SyDSs is PSPACE-complete even when the maximum node degree and the treewidth [11] of the underlying graph are constants.

(2) Given a configuration [illegible] the goal of the predecessor existence problem is to determine whether [illegible] has a predecessor configuration. An algorithm for this problem is useful in determining how a system reached the configuration [illegible] if [illegible] is an undesirable one (e.g., one in which many agents are infected), measures to prevent the system from reaching [illegible] can be undertaken. In Section 4, we show that the predecessor existence problem for NCF-SyDSs is NP-complete even when the maximum node degree of the underlying graph is three. The reduction used in the proof also enables us to conclude that the problem of counting the number of predecessors of an NCF-SyDS is #P-complete. This result is tight since it is known that when the maximum node degree is two, the predecessor existence problem as well as the corresponding counting version can be solved efficiently for any SyDS, regardless of the local transition functions [7, 25].

(3) Recall that a **fixed point** of a SyDS is configuration [illegible] which is its own successor; thus, if a SyDS reaches [illegible] it stays in that configuration forever. Again, in the context of epidemics, fixed points in which only a small number of agents are infected are useful, since the number of infections does not grow once the system reaches such a configuration. In Section 5, we consider the fixed point existence problem for NCF-SyDSs. We show that this problem is NP-complete even when the maximum node degree of the underlying graph is three. The reduction also enables us to conclude the hardness of the counting version of the problem. This result is also tight; when the maximum node degree is two, the fixed point existence problem as well as the corresponding counting version can be solved efficiently for any SyDS, regardless of the local transition functions [35].

(4) In Section 6, we consider the Garden of Eden (GE) existence problem for NCF-SyDS. In contrast to the other analysis problems, we show that the GE existence problem can be solved efficiently, even when the local functions are generalized NCFs. (This class of NCFs is defined in Section 2.1.) Our result (Theorem 6.1), which characterizes the existence of GE configurations in SyDSs with generalized NCFs, leads to a simple algorithm for the GE existence question. However, the proof of the result requires an intricate analysis.

Due to length restrictions, only proof sketches are given in the paper. A complete version that includes all proofs is available as [36].

1.3 Related Work

Computational aspects of testing phase space properties of discrete dynamical systems and multi-agent systems have been addressed by many researchers. For example, Barrett et al. [4, 5, 8] studied reachability problems as well as existence questions for fixed points and GE configurations under the sequential update model; here, a permutation of the vertices is also given, and state updates are carried out in the order specified by the permutation. Bounds on the lengths of transients and cycles in restricted versions of dynamical systems under the sequential update model are established in [32]. A good discussion of complexity results for multi-agent systems appears in the well known text by Wooldridge [46]. Tosic [41, 42] presented results for fixed point enumeration problems for systems with special forms of local transition functions. Kosub and Homan [24] presented dichotomy results that delineate computationally intractable and efficiently solvable versions of counting fixed points, based on the class of allowable local transition functions. The predecessor existence problem for deterministic and stochastic SyDSs was considered in [6, 7]. These references present hardness results for various restricted graph structures (e.g., grid graphs) and for various restricted families of local transition functions (e.g., k-threshold functions for any $k \geq 2$). Problems similar to predecessor existence have also been considered for cellular automata [12, 15].

We [35] introduced the notion of graph predicates to specify very general forms of phase space properties. There, it was shown that for many graph predicates (e.g., those which model problems such as fixed point and GE existence), the analysis problem can be solved in polynomial time when the underlying graph is treewidth-bounded and the local transition functions are r-symmetric[1] for some fixed integer r. As we explain in Section 2.4, NCFs are, in general, not r-symmetric for any fixed r. Moreover, our efficient algorithm for GE existence (Section 6) does not require any restriction on the underlying graph. Thus, our result for GE existence is not implied by the results of [35].

The class of Boolean networks introduced in [19] to model many biological phenomena is also a variant of the SyDS model. Results for many analysis problems under the Boolean network model appear in [1, 2, 18, 39, 40]. In [33], the reachability problem for SyDSs was shown to be PSPACE-hard for the Boolean network model where each local function is from {AND, OR}. Since AND and OR are both NCFs, this shows the computational intractability of reachability for dynamical systems under the Boolean network model where the local functions are NCFs. It should be noted that in the Boolean network model, the underlying graph of a dynamical system is directed while our work uses undirected graphs. Moreover, our result holds for a very restricted class of graphs, namely those whose maximum node degree and treewidth are both constants. It is not clear whether the result in [33] can be readily modified to hold for this restricted class.

[1] The definition of r-symmetric functions is given in Section 2.4.

2 DEFINITIONS AND PROBLEM FORMULATIONS

2.1 Nested Canalyzing Functions

As mentioned earlier, the class of nested canalyzing functions (NCFs), was introduced in [19] to model the behavior of certain biological systems. We follow the presentation in [26] in defining such a Boolean function. (For a Boolean value b, the complement is denoted by $\bar{b}$.)

Definition 1. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a set of n Boolean variables. Let σ be a permutation of $\{1, 2, \ldots, n\}$. A Boolean function $f(x_1, x_2, \ldots, x_n)$ over X is nested canalyzing in the variable order $x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}$ with canalyzing values $a_1, a_2, \ldots, a_n$ and canalyzed values $b_1, b_2, \ldots, b_n$ if f can be expressed in the following form:

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} b_1 & \text{if } x_{\sigma(1)} = a_1 \\ b_2 & \text{if } x_{\sigma(1)} \neq a_1 \text{ and } x_{\sigma(2)} = a_2 \\ \vdots & \vdots \\ b_n & \text{if } x_{\sigma(1)} \neq a_1 \text{ and } \ldots \\ & \quad x_{\sigma(n-1)} \neq a_{n-1} \text{ and} \\ & \quad x_{\sigma(n)} = a_n \\ \bar{b}_n & \text{if } x_{\sigma(1)} \neq a_1 \text{ and } \ldots \\ & \quad x_{\sigma(n)} \neq a_n \end{cases}$$

For convenience, we will use a computational notation introduced in [38] to represent NCFs. For $1 \leq j \leq n$, line j of our representation has the following form:

$$x_{\sigma(j)} : a_j \longrightarrow b_j$$

We say that $x_{\sigma(j)}$ is the canalyzing variable that is tested in line j with a_j and b_j denoting respectively the canalyzing and canalyzed values in line j as before, $1 \leq j \leq n$. The above line is interpreted as follows: if the value of $x_{\sigma(j)} = a_j$, then the value of the function is b_j; otherwise, we consider the next line in the description. We refer to each such line as a rule. When none of the conditions "$x_{\sigma(j)} = a_j$" is satisfied, we have line $n+1$ with the "Default" rule for which the canalyzed value is $\bar{b}_n$:

Default: $\bar{b}_n$

We will refer to the above specification of an NCF as the simplified representation and assume (without loss of generality) that each NCF is specified in this manner. The simplified representation provides the following convenient computational view of an NCF. Lines defining an NCF are considered sequentially in a top-down manner. The computation stops at the first line where the specified condition is satisfied, and the value of the function is the canalyzed value on that line. We now present an example of an NCF using the two representations mentioned above.

Example 1: Consider the Boolean function $f(x_1, x_2, x_3) = x_1 \vee (\bar{x}_2 \wedge x_3)$. This function is nested canalyzing using the identity permutation σ on $\{1, 2, 3\}$ with canalyzing values 1, 1, 1 and canalyzed values 1, 0, 1. We first show how this function can be expressed using the syntax of Definition 1.

$$f(x_1, x_2, x_3) = \begin{cases} 1 & \text{if } x_1 = 1 \\ 0 & \text{if } x_1 \neq 1 \text{ and } x_2 = 1 \\ 1 & \text{if } x_1 \neq 1 \text{ and } x_2 \neq 1 \text{ and } x_3 = 1 \\ 0 & \text{if } x_1 \neq 1 \text{ and } x_2 \neq 1 \text{ and } x_3 \neq 1 \end{cases}$$

A simplified representation of the same function is as follows.

x_1 : 1 $\longrightarrow$ 1
x_2 : 1 $\longrightarrow$ 0
x_3 : 1 $\longrightarrow$ 1
Default: 0

Additional conventions regarding NCFs: The canalyzed value for the rule labeled "Default" is always the complement of the canalyzed value on the line that immediately precedes that rule. So, for simplicity, we will omit the "Default" rule in specifying an NCF. To save space in presenting examples and proofs, we list successive rules along a line separated by commas. Thus, a linear representation of the NCF shown in Example 1 (with the "Default" rule omitted) is as follows: x_1 : 1 $\longrightarrow$ 0, x_2 : 1 $\longrightarrow$ 0, x_3 : 1 $\longrightarrow$ 1.

Generalized NCFs: One of the problems considered in this paper involves determining whether a given NCF-SyDS has a GE configuration. To extend the applicability of this algorithm, we allow local functions to be in the form of generalized NCFs, where rules are specified only for a subset of the variables. A precise definition of generalized NCFs is given below.

Definition 2. *A generalized NCF is a function represented as either a constant or an NCF representation of a subset (not necessarily proper) of the function's variables.*

Thus, every NCF is a generalized NCF; however, the converse is not true. We note that 1-decision lists studied in the context of computational learning [22] are the same as generalized NCFs.

Example 2: The constant function which takes on the value 0 for every combination of inputs can be represented as a generalized NCF using the following single rule:

Default: 0

As another example, a generalized NCF specification for a function $f(x_1, x_2, x_3, x_4)$ is as follows.

x_1 : 0 $\longrightarrow$ 1
x_3 : 1 $\longrightarrow$ 1
Default: 0

In this case, the function does not depend on the values of variables x_2 and x_4.

If a generalized NCF specifies a constant function (i.e., a function which has the value 0 for all inputs or 1 for all inputs), we will indicate that using just the "Default" rule. Otherwise (i.e., there is at least one rule involving a variable), we can assume without loss of generality that the canalyzed value specified in the "Default" rule is the complement of that specified on the line that immediately precedes the "Default" rule; in such cases, we omit the "Default" rule for simplicity.

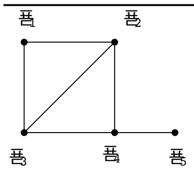
2.2 Synchronous Boolean Dynamical Systems

Let $\mathbb{B}$ denote the Boolean domain {0,1}. A Synchronous Dynamical System (SyDS) $\mathcal{S}$ over $\mathbb{B}$ is specified as a pair $\mathcal{S} = (G_{\mathcal{S}}, \mathcal{F})$, where (i) $G_{\mathcal{S}}(V_{\mathcal{S}}, E_{\mathcal{S}})$, an undirected graph with $|V_{\mathcal{S}}| = n$, represents the underlying graph of the SyDS, with node set $V_{\mathcal{S}}$ and edge set $E_{\mathcal{S}}$, and (ii) $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is a set of functions in the system, with f_i denoting the local transition function associated with node v_i, $1 \leq i \leq n$.

Each node of $G_{\mathcal{S}}$ has a state value from $\mathbb{B}$. Each function f_i specifies the local interaction between node v_i and its neighbors in $G_{\mathcal{S}}$. The inputs to function f_i are the state of v_i and those of the neighbors of v_i in $G_{\mathcal{S}}$; function f_i maps each combination of inputs to a value in $\mathbb{B}$. This value becomes the next state of node v_i. It is assumed that each local function is specified as an NCF or generalized NCF using the notation discussed in Section 2.1. In a SyDS, all nodes compute and update their next state synchronously. Other update disciplines (e.g., sequential updates) for discrete dynamical systems have also been considered in the literature (e.g., [5, 32]). At any time t, the configuration $\mathcal{C}^t$ of a SyDS is the n-vector $(s_1^t, s_2^t, \ldots, s_n^t)$, where $s_i^t \in \mathbb{B}$ is the state of node v_i at time t ($1 \leq i \leq n$).

Example 3: Consider the graph shown in Figure 1. In defining local transition functions for the corresponding SyDS as NCFs, we use the name of a node to be the variable representing its state.
(1) The function f_1 at v_1 is the OR function (i.e., $v_1 \vee v_2 \vee v_3$) with the following NCF description: v_1 : 1 $\longrightarrow$ 1, v_2 : 1 $\longrightarrow$ 1, v_3 : 1 $\longrightarrow$ 1.
(2) The function f_2 at v_2 is the AND function (i.e., $v_1 \wedge v_2 \wedge v_3 \wedge v_4$) with the following NCF description: v_1 : 0 $\longrightarrow$ 0, v_2 : 0 $\longrightarrow$ 0, v_3 : 0 $\longrightarrow$ 0, v_4 : 0 $\longrightarrow$ 0.
(3) The function f_3 at v_3 is $v_1 \vee \overline{v_2} \vee v_3 \vee \overline{v_4}$ whose NCF description is: v_1 : 1 $\longrightarrow$ 1, v_2 : 0 $\longrightarrow$ 1, v_3 : 1 $\longrightarrow$ 1, v_4 : 0 $\longrightarrow$ 1.
(4) The function f_4 at v_4 is the AND function (i.e., $v_3 \wedge v_4 \wedge v_5$) with the following description: v_3 : 0 $\longrightarrow$ 0, v_4 : 0 $\longrightarrow$ 0, v_5 : 0 $\longrightarrow$ 0.
(5) The function f_5 at v_5 is $\overline{v_4} \wedge \overline{v_5}$ whose NCF representation is: v_4 : 1 $\longrightarrow$ 0, v_5 : 1 $\longrightarrow$ 0.

We specify a configuration by listing the states of the nodes in the order v_1 through v_5. Assume that the initial configuration of the system is (0, 1, 0, 1, 1). During the first time step, v_3 remains in state 0 while the states of the other nodes change in the following manner: v_1 changes to 1 (since its neighbor v_2 is in state 1), v_2 changes to 0 (since its neighbor v_1 is in state 0), v_4 changes to 0 (since its neighbor v_3 is in state 0) and v_5 changes to 0 (since both v_4 and v_5 are in state 1). Thus, the configuration at time 1 is (1, 0, 0, 0, 0). The configuration at time 2 can be seen to be (1, 0, 1, 0, 1). Subsequently, while the state values of nodes v_1 through v_4 remain unchanged, the state value of v_5 gets complemented at each time step. Thus, the system cycles between the two configurations (1, 0, 1, 0, 1) and (1, 0, 1, 0, 0). □



Initial Config.:	(0, 1, 0, 1, 1)
Config. at time 1:	(1, 0, 0, 0, 0)
Config. at time 2:	(1, 0, 1, 0, 1)
Config. at time 3:	(1, 0, 1, 0, 0)
Config. at time 4:	(1, 0, 1, 0, 1)

Note: The system cycles between the two configurations at times 3 and 4.

Figure 1: An Example of a SyDS.

2.3 Problem Formulations

We consider a number of analysis problems for SyDSs whose local functions are specified as NCFs (or generalized NCFs). Precise definitions of these problems are given below.

I. Reachability Problem:

Instance: An NCF-SyDS $\mathcal{S}$ with underlying graph $G(V, E)$; two configurations $\mathcal{I}$ and $\mathcal{B}$ of $\mathcal{S}$.
Question: Does $\mathcal{S}$ starting from $\mathcal{I}$ reach $\mathcal{B}$?

II. Fixed Point Existence:

Instance: An NCF-SyDS $\mathcal{S}$ with underlying graph $G(V, E)$.
Question: Does $\mathcal{S}$ have a fixed point, that is, a configuration $\mathcal{C}$ such that the successor of $\mathcal{C}$ is $\mathcal{C}$ itself?

III. Predecessor Existence:

Instance: A NCF-SyDS $\mathcal{S}$ with underlying graph $G(V, E)$; a configuration $\mathcal{C}$ of $\mathcal{S}$.
Question: Does $\mathcal{C}$ have a predecessor, that is, a configuration $\mathcal{C}'$ such that the successor of $\mathcal{C}'$ is $\mathcal{C}$?

IV. Garden-of-Eden Existence:

Instance: A SyDS $\mathcal{S}$ with underlying graph $G(V, E)$ and generalized NCF local functions.
Question: Does $\mathcal{S}$ have a GE configuration, that is, a configuration $\mathcal{C}$ which has no predecessor?

2.4 NCFs and Symmetric Functions

As mentioned in Section 1.3, several references have addressed the analysis problems formulated above for r-symmetric Boolean functions (e.g., [3, 4, 6, 7, 25, 35]). A Boolean function f with ℓ inputs is *symmetric* if the value of the function depends only on the number of inputs which have the value 1 and not on the order in which the values are specified. Examples of symmetric functions include AND, OR, NAND, NOR, XOR, etc. A Boolean function f with ℓ inputs is r-*symmetric* if the inputs can be partitioned into r subsets such that the value of the function depends only on the number of 1-valued inputs in each subset. For example, it is observed in [25] that the class of bi-threshold functions is 2-symmetric. The results for analysis problems presented in the above references assume that each local function is r-symmetric for some fixed r. We present an example to show that NCFs are, in general, different from r-symmetric functions for fixed values of r.

Example 4: Consider the Boolean function $f(x_1, x_2, x_3) = x_1 \vee (\overline{x_2} \wedge x_3)$. An NCF representation for this function was given in Example 1. The function is not symmetric since $f(1, 0, 0) = 1$ while $f(0, 1, 0) = 0$. We can also argue that function f is not 2-symmetric by considering each possible partition of $\{x_1, x_2, x_3\}$ into two subsets. Suppose the partition is $\{x_1, x_2\}$ and $\{x_3\}$. Note that $f(1, 0, 0) = 1$ while $f(0, 1, 0) = 0$; in both assignments, the number of 1-valued inputs in the subset $\{x_1, x_2\}$ is 1. In a similar way, we can rule out the other partitions of $\{x_1, x_2, x_3\}$ into two subsets.

The above example can be generalized to show that there are NCFs with n variables which are not $n - 1$-symmetric. Thus, the results presented in this paper for NCF-SyDSs are not implied by the known results [35] for SyDSs with r-symmetric functions for fixed r.

3 COMPLEXITY OF REACHABILITY

Here, we establish the computational intractability of the reachability problem for NCF-SyDSs. To prove this result, we use a reduction from the Linear Bounded Automaton (LBA) Acceptance problem (i.e., given a deterministic LBA M and a string x, does M accept x?) which is known to be PSPACE-complete [13].

Theorem 3.1. There exist constants d_0 and k_0 such that the Reachability problem for NCF-SyDSs is PSPACE-complete, even when the maximum node degree of the underlying graph is d_0 and the treewidth of the graph is $\leq k_0$.

Proof: It is easy to see that the problem is in PSPACE. We show the PSPACE-hardness of reachability via a reduction from the LBA acceptance problem. Suppose the LBA contains n cells. Then, the underlying graph of the constructed SyDS consists of n clusters of nodes, with the i^{th} cluster representing the i^{th} cell of the LBA tape. This node cluster encodes the tape symbol on the i^{th} cell, as well as whether the tape head is residing on that cell, and if so, the state of the LBA. Thus, the SyDS configuration corresponds to an instantaneous description of the LBA. The transition function of the LBA is captured by appropriate NCF local transition functions so that successive configurations of the SyDS correspond to successive instantaneous descriptions of the LBA. In each step of the SyDS, the state of a given node of the SyDS changes if and only if the corresponding element in the LBA's instantaneous description changes. In the simulation of the LBA by the constructed SyDS, the LBA accepts its input string in t steps if and only if the SyDS reaches a specified configuration in t steps. Details of this construction are given below.

Let $M = (Q, \Sigma, \Sigma', q_0, q_y, \delta)$ denote the given deterministic LBA where Q is the (finite) set of states, Σ is the tape alphabet, $\Sigma' \subset \Sigma$ is the input alphabet, $q_0 \in Q$ is the initial state, $q_y \in Q$ is the accepting state and $\delta : (Q \times \Sigma) \longrightarrow (Q \times \Sigma \times \{L, R, N\})$ is the transition function. Given the current state and the current symbol scanned by the (read-write) head, δ specifies the next state, the symbol to be written on the cell scanned by the head and the direction of head movement (left or right by one tape cell or stay on the same cell). Let $x = a_1 a_2 \ldots a_n$ be the input string given to M with $a_1 = \$$ and $a_n = ©$ being the endmarkers.

An instantaneous description (ID) of M consists of the current state, the contents of the tape cells and the position of the head. M starts at q_0 with its head on the tape cell containing $a_0 = \$$. We represent the ID at time zero by the vector $\mathcal{I}_M = \langle (q_0, a_0), a_1, \ldots, a_{n+1} \rangle$. We may assume without loss of generality that if M accepts x then it replaces all the symbols on the tape cells between the endmarkers with the symbol b, moves the head to the cell containing \$, and cycles in state q_y. Thus, the unique accepting ID can be represented by the vector $\mathcal{B}_M = \langle (q_y, \$), b, \ldots, b, © \rangle$.

Given M and input string x, we create a SyDS $S_{M,x}$ and two configurations $\mathcal{I}_S$ and $\mathcal{B}_S$ such that $S_{M,x}$ starting from configuration $\mathcal{I}_S$ reaches configuration $\mathcal{B}_S$ if and only if M accepts x.

Let $n = |x|$, $m = |\Sigma|$, and $k = |Q|$. SyDS $S_{M,x}$ contains $(n+2)m + (n+2)km$ nodes, which can be viewed as being arranged into n clusters, each with $m + km$ nodes. Within cluster i, $1 \le i \le n$, there are m nodes, denoted as $\{s_{i,a} \mid a \in \Sigma\}$, which we refer to as passive nodes. Also within cluster i there are km nodes, denoted as $\{t_{i,q,a} \mid q \in Q, a \in \Sigma\}$, which we refer to as active nodes. Intuitively, node $s_{i,a}$ having value 1 corresponds to tape cell i containing symbol a, and node $t_{i,q,a}$ having value 1 corresponds to the tape head residing on tape cell i in state q, with tape cell i containing symbol a.

We say that a configuration of $S_{M,x}$ is valid if it satisfies the following three conditions: (1) For each cluster i, exactly one of the passive nodes in the cluster has value 1. (2) Exactly one active node of $S_{M,x}$ has value 1. (3) If a given node $t_{i,q,a}$ has value 1, then the node $s_{i,a}$ also has value 1.

We define a bijection h from IDs of M onto the set of valid configurations of $S_{M,x}$, as follows. Node $s_{i,a}$ has value 1 iff tape cell i contains tape symbol a, and node $t_{i,q,a}$ has value 1 iff the tape head resides on tape cell i in state q, with tape cell i containing symbol a.

SyDS $S_{M,x}$ will be constructed so that if ID α_1 of the LBA is followed by ID α_2, then configuration $h(\alpha_1)$ of $S_{M,x}$ is followed by configuration $h(\alpha_2)$.

The nodes in each cluster are interconnected as a clique, and are connected to all the nodes in adjacent clusters. Thus, the maximum node degree Δ is $3m(k+1) - 1$, and the treewidth w is at most $2m(k+1) - 1$.

We now give the local transition functions of $S_{M,x}$, explaining how they operate when evaluated on a valid configuration. First, we give the NCF representation for a passive node, say node $s_{i,a}$. The first km lines of the NCF representation test all the active nodes in cluster i. If any of these nodes has value 1, then the transition function of LBA M determines the new contents of tape cell i. More specifically, the line in the NCF representation that tests variable $t_{i,q,a'}$ is

$$t_{i,q,a'} : 1 \longrightarrow b$$

where b is 1 iff $\delta(q, a') = (p, a, d)$ for some p and d.

Since in a valid configuration of $S_{M,x}$, at most one of the canalyzing variables in the above lines will equal 1, the above lines can be written in any order.

If the above km canalyzing variables are all 0, then the tape head is not on cell i, so the contents of tape cell i will be unchanged by the next LBA transition. So, the next three lines of the NCF representation keep the value of node $s_{i,a}$ unchanged. Let a_1 and a_2 be two tape symbols different from tape symbol a. Note that in any valid configuration, at least one of the nodes s_{i,a_1} and s_{i,a_2} has value 0. The next three lines of the NCF representation are: $s_{i,a} : 1 \longrightarrow 1$, $s_{i,a_1} : 0 \longrightarrow 0$, $s_{i,a_2} : 0 \longrightarrow 0$.

Note that for any valid configuration, at least one of the above $km + 3$ lines will satisfy its test condition, so the remaining lines of the NCF representation can be arbitrary.

We now give the NCF representation for an active node, say node $t_{i,q,a}$. The first km lines of the NCF representation test all the active nodes in cluster i. If any of these nodes has value 1, then the transition function of LBA M determines the new value of node $t_{i,q,a}$. More specifically, the line in the NCF representation that tests variable $t_{i,q',a'}$ is

$$t_{i,q',a'} : 1 \longrightarrow b$$

where b is 1 iff $\delta(q', a') = (q, a, S)$.

If the above km canalyzing variables are all 0, then the tape head is not on cell i, so the contents of tape cell i will be unchanged by the next LBA transition. The next $m - 1$ lines of the NCF representation check whether the current contents of tape cell i is not tape symbol a, in which case the contents of tape cell i after one transition is not a. Thus, for each $a' \neq a$, we have the line: $s_{i,a'} : 1 \longrightarrow 0$.

If all the above tests fail, and this point in the NCF evaluation is reached, then a is the tape symbol on cell i, and the tape head is not on cell i. So, we next test whether the tape head will move onto cell i in state q.

If $i > 1$, we have an NCF line for each possibility of the tape head moving to the right onto cell i in state q. Thus, for each (q', a') such that $\delta(q', a') = (q, a'', R)$ for some a'', we have the line: $t_{i-1,q',a'} : 1 \longrightarrow 1$.

If $i < n$, we have an NCF line for each possibility of the tape head moving to the left onto cell i in state q. Thus, for each (q', a') such that $\delta(q', a') = (q, a'', L)$ for some a'', we have the line: $t_{i+1,q',a'} : 1 \longrightarrow 1$.

If all the above tests fail, and this point in the NCF evaluation is reached, then node $t_{i,q,a}$ should be set to 0. The next two lines of the NCF representation accomplish this. Let j be the index of an adjacent cluster, and let a_1 and a_2 be any two tape symbols. The next two lines of the NCF representation are: $s_{j,a_1} : 0 \longrightarrow 0$, $s_{j,a_2} : 0 \longrightarrow 0$.

Note that for any valid configuration, at least one of the above lines will satisfy its test condition, so the remaining lines of the NCF representation can be arbitrary.

We now consider the reachability problem for $S_{M,x}$. The initial configuration $\mathcal{I}_S$ of $S_{M,x}$ is constructed from the initial ID $\mathcal{I}_M$, so we construct $\mathcal{I}_S$ to be $h(\mathcal{I}_M)$. Similarly, the final configuration $\mathcal{B}_S$ of $S_{M,x}$ is constructed from the final ID $\mathcal{B}_M$, so we construct $\mathcal{B}_S$ to be $h(\mathcal{B}_M)$. Thus, $S_{M,x}$ reaches the required configuration $\mathcal{B}_S$ iff M accepts x. ∎

4 PREDECESSOR EXISTENCE

Theorem 4.1. *The predecessor existence problem for NCF-SyDSs is NP-complete even when the maximum node degree of the underlying graph is 3.*

Proof sketch: It is easy to see that the predecessor existence problem is in NP. We show NP-hardness via a parsimonious reduction from 3SAT.

Suppose the given 3SAT formula f has n variables and m clauses. The reduction constructs an NCF-SyDS S and a configuration C. The underlying graph G of S contains $n + m$ nodes. For each variable, there is a node, which we denote as v_i, $1 \leq i \leq n$. For each clause, there is a node, which we denote as w_j, $1 \leq j \leq m$. There is an edge between each node for a clause and the nodes for the variables occurring in that clause.

We first describe the local transition function for the nodes corresponding to the variables of the 3SAT formula f. For each node v_i, the first line of the NCF representation for the local transition function at v_i is: $v_i : 0 \longrightarrow 1$.

Each subsequent line of the function at v_i corresponds to a clause in which the variable corresponding to v_i appears. For each such clause node w_j such that the variable corresponding to v_i appears in the clause corresponding to w_j, the local function for v_i has the following line: $w_j : 0 \longrightarrow 1$.

We now describe the local transition function for the nodes corresponding to the clauses of the 3SAT formula f. For each w_j, the first line of the NCF for w_j is: $w_j : 1 \longrightarrow 0$.

This line is followed by a line for each literal occurring in clause j. If a given literal is positive, say x_k, then the corresponding line is: $v_k : 1 \longrightarrow 1$; if a given literal is negative, say $\bar{x}_k$, then the corresponding line is: $v_k : 0 \longrightarrow 1$.

The constructed configuration C has the value 1 for every node. It is easy to see that the construction can be carried out in polynomial time. It can be verified that the configuration C has a predecessor iff the given 3SAT instance is satisfiable.

It is well known that 3SAT is NP-complete even when each variable occurs in at most three clauses [13]. Using a reduction from this restricted version of 3SAT, it can be verified that in the underlying graph of the SyDS resulting from the above construction, the maximum node degree is 3. Thus, the predecessor problem remains NP-complete for NCF-SyDSs even when the maximum node degree is 3. ■

Theorem 4.1 is tight with respect to maximum node degree of the underlying graph. This is because when the maximum node degree is 2, the predecessor existence problem can be solved efficiently for all local transition functions [25].

It can be also seen that the above reduction is parsimonious; that is, the number of predecessors of the configuration C constructed in the above proof is equal to the number of satisfying assignments of the 3SAT formula f. Since the counting problem for 3SAT is #P-complete, we have:

Corollary 1. *The problem of counting the number of predecessors of a given configuration of an NCF-SyDS is #P-complete.* ■

5 FIXED POINT EXISTENCE

Theorem 5.1. *The fixed point existence problem for NCF-SyDSs is NP-complete even when the maximum node degree of the underlying graph is 3.*

Proof sketch: It is easy to see that the fixed point existence for SyDSs is in NP. We show NP-hardness via a parsimonious reduction from 3SAT. Without loss of generality, we assume that each variable of the given 3SAT instance occurs (positively or negatively) in at least one clause.

Suppose the given 3SAT formula f has n variables and m clauses. The reduction constructs a SyDS S whose underlying graph G contains $n + m$ nodes. For each variable, there is a node, which we denote as v_i, $1 \leq i \leq n$. For each clause, there is a node, which we denote as w_j, $1 \leq j \leq m$. There is an edge between each node for a clause and the nodes for the variables occurring in that clause.

We first discuss the NCF representation of the functions at the nodes corresponding to the variables of the given 3SAT instance. For each v_i, the first line of the NCF for v_i: $v_i : 0 \longrightarrow 0$.

The subsequent lines of the NCF representation for the function at v_i are constructed as follows. For each clause node w_j such that v_i appears in the clause corresponding to w_j, we have the line: $w_j : 1 \longrightarrow 1$.

We now present the NCF representation of the functions at the nodes corresponding to the clauses of the given 3SAT instance. For each clause node w_j, the first line of the NCF representation for w_j is: $w_j : 0 \longrightarrow 1$.

This is followed by a line for each literal occurring in clause j. If a given literal is positive, say x_k, then the corresponding line is: $v_k : 1 \longrightarrow 1$; if a given literal is negative, say $\bar{x}_k$, then the corresponding line is: $v_k : 0 \longrightarrow 1$.

It can be seen that the construction can be carried out in polynomial time. It can be shown that the resulting SyDS has a fixed point iff the given 3SAT instance is satisfiable.

It is known that 3SAT is NP-complete even when each variable occurs in at most three clauses [13]. Using a reduction from this restricted version of 3SAT, it can be verified that the maximum node degree of the underlying graph is 3. Thus, the fixed point existence problem for NCF-SyDSs is NP-complete when the maximum node degree is 3. ■

The above hardness result is also tight with respect to maximum node degree since it follows from the results in [35] that when the maximum node degree is 2, the fixed point existence can be solved efficiently. Further, it can also be seen that the above reduction is parsimonious. Thus:

Corollary 2. *The problem of counting the number of fixed points of an NCF-SyDS is #P-complete.* ■

6 GARDEN OF EDEN EXISTENCE

We now consider the Garden-of-Eden (GE) existence problem. To develop our algorithm for this problem, we need to first define a new operation (called projection) on NCFs.

Definition 3. *Given a Boolean function f, a variable x and a Boolean value b, the* projection *of f on $x = b$, denoted by $f_{x=b}$, is the function on the remaining variables whose value on any assignment α to these variables is the value of f when α is extended to a complete assignment for f by setting the variable x to the value b.*

The projection operation is used in the proof of our result for the GE existence problem for SyDSs whose local functions are generalized NCFs. A statement of this result is as follows.

Theor em 6.1. A SyDS whose local transition functions are all generalized NCFs has a GE conguration unless the generalized NCF for each node involves exactly one canalyzing variable, and each node occurs as a canalyzing variable in exactly one of these generalized NCFs.

Moreover, when the local transition functions are specied as generalized NCFs, if a GE conguration exists, then such a conguration can be constructed in linear time.

Before presenting a proof sketch for Theorem 6.1, we note that the rst part of the theorem provides the following simple two-step algorithm to check whether a GE conguration exists in a SyDS where each node function is a generalized NCF.

---

1. If there is any local function whose number of variables is /= 1, output "Yes" and stop.
2. (Here, each local function has exactly only one variable.) If a node occurs in two or more local functions, output "Yes"; otherwise, output "No".

A proof of Theorem 6.1 and the construction of a GE conguration when one exists, require an intricate analysis involving the edges of the graph and the local functions of the nodes. A sketch of the proof is given below.

Proof sketch for Theorem 6.1: Let 퓔 be a given SyDS where each local transition function is specied as a generalized NCF. Let 퐿 denote the number of nodes of 퓔, and 퓓 denote the set of nodes. For convenience, we let 퓧 ∈ 퓓 denote both a node and its corresponding variable. Let 퓓 denote the set of congurations of 퓔. For any conguration 퓓, we let 퓔(퓓) denote the successor conguration of 퓓.

For a conguration 퓔 of 퓔 and a conguration 퓓 on a set of variables 퓔 ⊆ 퓓, we say that 퓔 and 퓓 are compatible if for every node 퓧 ∈ 퓔, 퓔(퓧) = 퓓(퓧), and incompatible if there exists a node 퓧 ∈ 퓔 such that 퓔(퓧) /= 퓓(퓧).

We now describe an algorithm to construct a GE conguration. The algorithm proceeds in stages. Stage 1 might report that no GE conguration exists, and then exit the algorithm. Otherwise, a given stage either returns a GE conguration and exits the algorithm, or the given stage is completed, and the next stage begins.

After Stage 퓚 where 퓚 ≥ 0, the following objects will have been constructed: (i) A set of 퓚 nodes, which we refer to as predecessor nodes, and denote as 퓓퓚. (ii) A set of 퓚 nodes, which we refer to as successor nodes, and denote as 퓔퓚. (Sets 퓓퓚 and 퓔퓚 are not necessarily disjoint.) (iii) A conguration 퓓퓚 on node set 퓓퓚. We refer to 퓓퓚 as a predecessor pattern. (iv) A conguration 퓔퓚 on node set 퓔퓚. We refer to 퓔퓚 as a successor pattern.

Initially, 퓓⁰ and 퓔⁰ are empty, and 퓓⁰ and 퓔⁰ contain no components. For each node 퓧, let 퓕퓧,0 denote the given NCF representation of the transition function for 퓧. At the end of a given stage, say stage 퓚, 퓕퓧,0 will have been transformed into a generalized NCF function, denoted as 퓕퓧,퓚, representing the projection of 퓕퓧,0 onto the variables in 퓓 – 퓓퓚, obtained by setting each variable in 퓓퓚 to its value in 퓓퓚.

Let 𝓑퓚 be the set of congurations of 퓔 that are compatible with predecessor pattern 퓓퓚. Note that 𝓑퓚 contains 2^(퓛–퓚) congurations, and that 퓓 – 𝓑퓚 contains 2^퓛 – 2^(퓛–퓚) congurations. We refer to the congurations in 𝓑퓚 as eligible congurations, and those in 퓓 – 𝓑퓚 as ineligible congurations.

The constructed objects can be seen to have the following two properties: (i) For every ineligible conguration 퓓 ∈ 퓓 – 𝓑퓚, its successor 퓔(퓓) is incompatible with 퓔퓚. (ii) If 퓚 > 0, let 퓧 be the last node added to 퓔퓚. Then there is at least one eligible conguration 퓓 ∈ 𝓑퓚 whose successor conguration 퓔(퓓) has (퓔(퓓))(퓧) = $\overline{퓔퓚(퓧)}$, so that 퓔(퓓) is incompatible with 퓔퓚.

A consequence of these two properties is that after the completion of stage 퓚 where 퓚 > 0, there is at least one conguration that is an extension of 퓔퓚, and is a GE conguration. From Property 1, if a conguration that is compatible with 퓔퓚 has a predecessor, this predecessor must be an eligible conguration. However, there are only 2^(퓛–퓚) eligible congurations. From Property 2, the successor of at least one of the eligible congurations is incompatible with 퓔퓚. Thus, there are at most 2^(퓛–퓚) – 1 congurations whose successor is compatible with 퓔퓚. Since there are 2^(퓛–퓚) congurations that are compatible with 퓔퓚, at least one of these congurations has no predecessor, and so is a GE conguration.

Each stage that completes without exiting adds one more variable to the successor pattern. If a given stage 퓚 exits with a GE conguration, this GE conguration is an extension of the current successor pattern 퓔퓚. Additional details and an ecient implementation of the algorithm appear in [36]. ■

7 FUTURE WORK

There are two useful future research directions. One direction is to consider restrictions on the dynamical system that can lead to ecient algorithms for the analysis problems considered in this paper. Another direction is to develop algorithms that work well in practice, even though their running times may be exponential in the worst case. For problems that are eciently solvable, it would be of interest to see if the algorithms can be extended to more general versions along the lines of [35].

Acknowledgments: We thank the referees of AAMAS 2018 for providing valuable suggestions. This work has been partially supported by DTRA CNIMS (Contract HDTRA1-11-D-0016-0001), NSF DIBBS Grant ACI-1443054 and NSF BIG DATA Grant IIS-1633028. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

REFERENCES

[1] T. Akutsu, M. Hayashida, W. Ching, and M. K. Ng. 2007. Control of Boolean Networks: Hardness results and algorithms for tree structured networks. Journal of Theoretical Biology 244 (2007), 670–679.

[2] T. Akutsu, S. Kosub, A. Melkman, and T. Tamura. 2012. Finding a Periodic Attractor of a Boolean Network. IEEE/ACM Trans. Comput. Biol. Bioinformatics 9, 5 (Sep. 2012), 1410–1421.

[3] C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. 2003. On Special Classes of Sequential Dynamical Systems. Annals of Combinatorics 7 (2003), 381–408. Issue 4.

[4] C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. 2003. Reachability problems for sequential dynamical systems with threshold functions. Theor. Comput. Sci. 295 (2003), 41–64.

[5] C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. 2006. Complexity of Reachability problems for finite discrete dynamical systems. J. Comput. Syst. Sci. 72, 8 (2006), 1317–1345.

[6] C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. 2011. Modeling and Analyzing Social Network Dynamics Using Stochastic Discrete Graphical Dynamical Systems. Theoretical Computer Science 412, 30 (2011), 3932–3946.

[7] C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and Mayur Thakur. 2007. Predecessor Existence Problems for Finite Discrete Dynamical Systems. Theoretical Computer Science 386, 1–2 (2007), 3–37.

[8] C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and P. T. Tosic. 2001. Gardens of Eden and Fixed Points in Sequential Dynamical Systems. In DM-CCG. HAL - INRIA, Paris, France, 95–110.

[9] C. L. Barrett, H. S. Mortveit, and C. M. Reidys. 2000. Elements of a theory of simulation II: Sequential Dynamical Systems. Appl. Math. Comput. 107, 2-3 (2000), 121–136.

[10] C. L. Barrett and C. M. Reidys. 1999. Elements of a theory of simulation I: Sequential CA Over Random Graphs. Appl. Math. Comput. 98, 3 (1999), 241–259.

[11] H. L. Bodlaender. 1993. A Tourist Guide through Treewidth. Acta Cybernetica 11, 1-2 (1993), 1–22.

[12] B. Durand. 1995. A Random NP-Complete Problem for Inversion of 2D Cellular Automata. Theor. Comput. Sci. 148, 1 (1995), 19–32.

[13] M. R. Garey and D. S. Johnson. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co., San Francisco, CA.

[14] A. Ghaffarizadeh, G. J. Podgorski, and N. S. Flann. 2017. Applying attractor dynamics to infer gene regulatory interactions involved in cellular differentiation. Biosystems 155 (2017), 29–41.

[15] F. Green. 1987. NP-Complete Problems in Cellular Automata. Complex Systems 1, 3 (1987), 453–474.

[16] H. Gutowitz. 1989. Cellular Automata: Theory and Experiment. North Holland, Ameterdam, The Netherlands.

[17] S. Kauffman. 1969. Metabolic Stability and epigenesis in randomly constructed genetic nets. J. Theoretical Biology 22, 3 (1969), 437–467.

[18] S. Kauffman. 1993. The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, New York, NY.

[19] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. 2003. Random Boolean network models and the yeast transcriptional network. Proc. National Academy of Sciences (PNAS) 100, 25 (Dec. 2003), 14796–14799.

[20] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein. 2004. Genetic networks with canalyzing Boolean rules are always stable. Proc. National Academy of Sciences (PNAS) 101, 49 (Dec. 2004), 17102–17107.

[21] M. Kearns. 2008. Graphical Games. In Algorithmic Graph Theory, N. Nissan, T. Roughgarden, E. Tardos, and V. Vazirani (Eds.). Cambridge University Press, New York, NY, Chapter 7, 159–178.

[22] M. J. Kearns and V. V. Vazirani. 1994. An Introduction to Computational Learning Theory. MIT Press, Cambridge, MA.

[23] D. Kohler and N. Friedman. 2009. Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge, MA.

[24] S. Kosub and C. M. Homan. 2007. Dichotomy Results for Fixed Point Counting in Boolean Dynamical Systems. In Proc. ICTCS. World Scientific, Singaore, 163–174.

[25] C. J. Kuhlman, A. Kumar, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. 2013. Analysis Problems for Special Classes of Bi-threshold Dynamical Systems. In Proc. Workshop on Multi-Agent Interaction Networks (MAIN 2013). ACM Sheridan Press, New York, NY, 26–33. (Held in conjunction with the 12th Intl. Conference on Autonomous Agents and Multiagent Systems (AAMAS)).

[26] L. Layne. 2011. Biologically Relevant Classes of Boolean Functions. Ph.D. Dissertation. Department of Mathematics, Clemson University.

[27] L. Layne, E. Dimitrova, and M. Macauley. 2012. Nested Canalyzing Depth and Network Stability. Bulletin of Mathematical Biology 74, 2 (2012), 422–433.

[28] Y. Li and J. O. Adeyeye. 2012. Sensitivity and Block Sensitivity of Nested Canalyzing Functions. arXiv:1209.1597v1 [cs.DM]. (Sept. 2012).

[29] Y. Li, J. O. Adeyeye, and R. C. Laubenbacher. 2011. Nested Canalyzing Functions And Their Average Sensitivities. arXiv: 1111.7217v1 [cs.DM]. (Nov. 2011).

[30] Y. Li, J. O. Adeyeye, D. Murrugarra, B. Aguilar, and R. C. Laubenbacher. 2013. Boolean nested canalizing functions: A comprehensive analysis. Theoretical Computer Science 481 (2013), 24–36.

[31] A. A. Melkman and T. Akutsu. 2013. An improved satisfiability algorithm for nested canalyzing functions and its application to determining a singleton attractor of a Boolean network. Journal of Computational Biology 20, 12 (2013), 958–969.

[32] H. S. Mortveit and C. M. Reidys. 2007. An Introduction to Sequential Dynamical Systems. Springer, Berlin, Germany.

[33] M. Ogihara and K. Uchizawa. 2015. Computational Complexity Studies of Synchronous Boolean Finite Dynamical Systems. In Theory and Applications of Models of Computation – 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings. Springer, New York, NY, 87–98.

[34] C. H. Papadimitriou and T. Roughgarden. 2003. Equilibria in Symmetric Games. Report, Stanford University. (2003).

[35] D. J. Rosenkrantz, M. V. Marathe, H. B. Hunt III, S. S. Ravi, and R. E. Stearns. 2015. Analysis Problems for Graphical Dynamical Systems: A Unified Approach Through Graph Predicates. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015. ACM Sheridan Press, New York, NY, 1501–1509.

[36] D. J. Rosenkrantz, M. V. Marathe, S. S. Ravi, and R. E. Stearns. 2017. Testing Phase Space Properties of Synchronous Dynamical Systems with Nested Canalyzing Local Functions. Technical Report, Biocomplexity Institute of Virginia Tech. (2017).

[37] C. Seshadhri, A. M. Smith, Y. Vorobeychik, J. R. Mayo, and R. C. Armstrong. 2016. Characterizing short-term stability for Boolean networks over any distribution of transfer functions. Physical Review E 94, 1 (2016), 012301.

[38] R. E. Stearns, D. J. Rosenkrantz, S. S. Ravi, and M. V. Marathe. 2017. An Elementary Proof of the Upper Bound on the Average Sensitivity of Nested Canalyzing Functions. Technical Report, Network Dynamics and Simulation Science Laboratory, Biocomplexity Institute of Virginia Tech, Blacksburg, VA. (2017).

[39] T. Tamura and T. Akutsu. 2007. An 퀶(1.787퀻)-Time Algorithm for Detecting a Singleton Attractor in a Boolean Network Consisting of AND/OR Nodes. In Fundamentals of Computation Theory, 16th International Symposium, FCT 2007, Budapest, Hungary, August 27-30, 2007, Proceedings. Springer, New York, NY, 494–505.

[40] T. Tamura and T. Akutsu. 2008. An Improved Algorithm for Detecting a Singleton Attractor in a Boolean Network Consisting of AND/OR Nodes. In Algebraic Biology, Third International Conference, AB 2008, Castle of Hagenberg, Austria, July 31-August 2, 2008, Proceedings. Springer, New York, NY, 216–229.

[41] P. T. Tosic. 2010. On the complexity of enumerating possible dynamics of sparsely connected Boolean network automata with simple update rules. In Automata 2010 - 16th Intl. Workshop on CA and DCS. HAL - INRIA, Paris, France, 125–144.

[42] P. T. Tosic. 2017. Phase Transitions in Possible Dynamics of Cellular and Graph Automata Models of Sparsely Interconnected Multi-Agent Systems. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017. ACM Sheridan Press, New York, NY, 474–483.

[43] T. W. Valente. 1996. Social network thresholds in the diffusion of innovations. Social Networks 18 (1996), 69–89.

[44] C. H. Waddington. 1942. Canalyzation of development and the inheritance of acquired characters. Nature 150, 14 (1942), 563–565.

[45] S. Wolfram. 1987. Theory and Applications of Cellular Automata. World Scientific, Singapore.

[46] M. Wooldridge. 2002. An Introduction to Multi-Agent Systems. John Wiley & Sons, West Sussex, UK.