Synchronous Dynamical Systems on Directed Acyclic Graphs: Complexity and Algorithms

DANIEL J. ROSENKRANTZ, University of Virginia, Charlottesville, USA
MADHAV V. MARATHE, University of Virginia, Charlottesville, USA
S. S. RAVI, University of Virginia, Charlottesville, USA
RICHARD E. STEARNS, University of Virginia, Charlottesville, USA

Discrete dynamical systems serve as useful formal models to study diffusion phenomena in social networks. Several recent articles have studied the algorithmic and complexity aspects of some decision problems on synchronous Boolean networks, which are discrete dynamical systems whose underlying graphs are directed, and may contain directed cycles. Such problems can be regarded as reachability problems in the phase space of the corresponding dynamical system. Previous work has shown that some of these decision problems become efficiently solvable for systems on directed acyclic graphs (DAGs). Motivated by this line of work, we investigate a number of decision problems for dynamical systems whose underlying graphs are DAGs. We show that computational intractability (i.e., **PSPACE**-completeness) results for reachability problems hold even for dynamical systems on DAGs. We also identify some restricted versions of dynamical systems on DAGs for which reachability problem can be solved efficiently. In addition, we show that a decision problem (namely, Convergence), which is efficiently solvable for dynamical systems on DAGs, becomes **PSPACE**-complete for Quasi-DAGs (i.e., graphs that become DAGs by the removal of a *single* edge). In the process of establishing the above results, we also develop several structural properties of the phase spaces of dynamical systems on DAGs.

CCS Concepts: • **Theory of computation → Abstract machines;**

Additional Key Words and Phrases: Discrete dynamical systems, directed acyclic graphs, reachability, Convergence Guarantee, complexity, algorithms

ACM Reference Format:
Daniel J. Rosenkrantz, Madhav V. Marathe, S. S. Ravi, and Richard E. Stearns. 2024. Synchronous Dynamical Systems on Directed Acyclic Graphs: Complexity and Algorithms. *ACM Trans. Comput. Theory* 16, 2, Article 11 (June 2024), 34 pages. https://doi.org/10.1145/3653723

A preliminary version of this article appeared in the *Proceedings of AAAI 2021* [38].
This work is partially supported by University of Virginia Strategic Investment Fund Award SIF160, CMMI-1745207 (EAGER), OAC-1916805 (CINES), CCF-1918656 (Expeditions) and IIS-1908530.
Authors' addresses: D. J. Rosenkrantz, M. V. Marathe, S. S. Ravi, and R. E. Stearns, University of Virginia, Biocomplexity Institute, University of Virginia, P. O. Box 400298, Charlottesville, VA 22904, USA; e-mails: drosenkrantz@gmail.com, marathe@virginia.edu, ssravi0@gmail.com, thestearns2@gmail.com.

ACM 1942-3454/2024/06-ART11
https://doi.org/10.1145/3653723

1 INTRODUCTION

1.1 Background and Motivation

Discrete dynamical systems provide a formal model for the study of diffusion phenomena in networks. Examples of such phenomena include diffusion of social contagions (e.g., information, opinions, fads) and epidemics [1, 13, 31, 33]. Informally, such a dynamical system[1] consists of an underlying (social or biological) network, with each node having a state value from a domain $\mathbb{B}$. In this article, we assume that the underlying graph is directed and that the domain is binary (i.e., $\mathbb{B} = \{0,1\}$). The propagation of the contagion is modeled by a collection of Boolean local functions, one per node. For any node v, the inputs to the local function f_v at v are the current states of v and those of its in-neighbors (i.e., the nodes from which v has an incoming edges); the output of f_v is the state of v at the next time instant. We consider the **synchronous** update model, where all nodes evaluate their local functions and update their states *in parallel*. These dynamical systems are referred to as **synchronous dynamical systems (SyDSs)** in the literature (e.g., [1] and [37]). In applications involving systems biology, such systems are also referred to as **synchronous Boolean networks** (e.g., [2], [26], and [33]). Throughout this article, we will use the term *SyDS* to denote such a system. In general, the local functions may be deterministic (e.g., threshold models used in social systems [24]) or stochastic (e.g., the SIR model of disease propagation [21]). Our focus is on deterministic local functions.

For a SyDS with n nodes, the **configuration** at time τ is a vector $(s_1^\tau, s_2^\tau, \ldots, s_n^\tau)$, where s_i^τ is the state value of node v_i at time τ, $1 \leq i \leq n$. As the nodes evaluate and update their local functions, the configuration of the system evolves over time. If a SyDS transitions from a configuration C to a configuration C' in one time step, we say that C' is the **successor** of C and that C is a **predecessor** of C'. A configuration C which is its own successor is called a **fixed point**. Thus, once a SyDS reaches a fixed point, no further state changes occur at any node.

When using SyDSs as models of social or biological phenomena, it is of interest to study whether a given SyDS starting from a specified initial configuration may reach certain (desirable or undesirable) configurations. For example, in the context of information propagation, where the state value 1 indicates that a node has received the information propagating through the network, one may be interested in configurations where many nodes are in state 1. On the other hand, in contexts such as disease propagation where the state of 1 indicates that a node has been infected, configurations where many nodes are in state 1 are undesirable. One can study such **configuration reachability** problems by considering another directed graph, called the **phase space**, of the dynamical system [31]. Each node in the phase space is a configuration and there is a directed edge from a node X to node Y if the SyDS can transition from the configuration represented by X to that represented by Y in one step. Thus, the configuration reachability problem becomes a directed path problem in phase space. For a SyDS with n nodes where the state of each node is from $\{0, 1\}$, the number of possible configurations is 2^n; thus, the size of the phase space of a SyDS is *exponential* in the size of the SyDS. There has been a considerable amount of research whose goal is to understand when such reachability problems are solvable in time that is polynomial in the size of the description of SyDS and when they are computationally intractable (see, e.g., [2], [7], [13], [32], and [33], and the references cited therein).

Several papers (e.g., [2] and [32]) have presented computational intractability results for reachability problems for SyDSs on directed graphs. Very recently, Chistikov et al. [13] studied two reachability problems (called the *Convergence and Convergence Guarantee*[2]) in the context of

[1] A formal definition is provided in Section 2.

[2] Informal definitions of these problems are provided in Section 1.2; the formal definitions appear in Section 2.5.

opinion propagation. They show that these problems are PSPACE-complete for general directed graphs. They also show that the Convergence problem can be solved efficiently when the underlying directed graph is *acyclic* (i.e., it does not have any directed cycle) *regardless* of the local functions at each node. Following standard graph theoretic terminology, we refer to directed acyclic graphs as **DAGs** [16]. Many optimization problems (e.g., finding a longest simple directed path) which are NP-hard for general directed graphs are known to be efficiently solvable for DAGs [23]. Several computational problems for discrete dynamical systems on DAGs have been addressed in the literature; these problems will be discussed in Section 1.3. Thus, it is of interest to study the complexity of Reachability and related problems for SyDSs whose underlying graphs are DAGs. We refer to such SyDSs as *DAG-SyDSs* and our results are summarized below.

1.2 Summary of Results and Their Significance

(1) *Results on the Structure of the Phase Space.* For any DAG-SyDS, we show that the length of every phase space cycle is a power of 2. Further, if the number of levels in the underlying DAG of a given SyDS is L, then no phase space cycle is longer than 2^L, and no transient (i.e., a directed path leading to a directed cycle in the phase space) is longer than $2^L - 1$ (Theorem 3.4). Moreover, we show that these bounds are achievable (Theorem 3.5). As discussed in Item (5) below, such structural properties of phase spaces are useful in solving reachability problems; if the maximum lengths of transients and cycles are bounded by polynomial functions of the size of a given SyDS, then reachability problems for that SyDS can be solved efficiently (by running the SyDS for a polynomial number of steps).

(2) *Complexity of the Reachability Problem for DAG-SyDS.* It was shown by Chistikov et al. [13] that the Convergence problem (i.e., given a SyDS $\mathcal{S}$ and an initial configuration C, does $\mathcal{S}$ starting from C reach a fixed point?) is PSPACE-complete for SyDSs on directed networks. They also showed that the Convergence problem can be solved efficiently for DAG-SyDSs, regardless of the local functions. We show that the reachability problem (i.e., given a SyDS $\mathcal{S}$ and two configurations C and $\mathcal{D}$, does $\mathcal{S}$ starting from C reach $\mathcal{D}$?), which is similar to the Convergence problem, is PSPACE-complete for DAG-SyDSs even when each local function is symmetric.[3] To our knowledge, no hardness result is currently known for the Reachability problem for DAG-SyDSs. Our proof of this result involves two major steps. The first step uses a reduction from the **Quantified Boolean Formulas (QBF)** problem [23] to show that the Reachability problem for DAG-SyDSs is PSPACE-complete even when each local function is r-symmetric[4] for some constant r (Theorem 4.1). For the second step, we define the concept of a **configuration embedding** of one SyDS into another, and use this concept to show that for any fixed value of r, the reachability problem for DAG-SyDSs with r-symmetric local functions is polynomial-time reducible to the Reachability problem for DAG-SyDSs with symmetric local functions (Corollary 4.6). The PSPACE-completeness of the Reachability problem for DAG-SyDSs with symmetric local functions (Theorem 4.7) follows directly from Theorem 4.1 and Corollary 4.6.

The difference between the complexities of Convergence and Reachability problems for DAG-SyDSs is due to the following: For any DAG-SyDS with L levels, regardless of the local functions, the length of any transient leading to a fixed point is bounded by L. (This is a slight restatement of Proposition 1 in [13].) Thus, the Convergence problem for DAG-SyDSs is efficiently solvable. On the other hand, we show that one can construct DAG-SyDSs with appropriate local functions such that the length of a transient and that of the cycle the transient leads to are both exponential in L

[3]Symmetric Boolean functions [17] are defined in Section 2.4.
[4]The definition of r-symmetric functions is given in Section 2. Symmetric Boolean functions are 1-symmetric. As will be explained in Section 2, the local functions used in [13] are 2-symmetric.

(Theorem 3.5). The idea behind this construction enables us to establish PSPACE-hardness result for the Reachability problem for DAG-SyDSs.

(3) *Complexity of the Convergence Problem for Quasi-DAG-SyDSs.* As mentioned above, the Convergence problem is efficiently solvable for DAG-SyDSs regardless of the local functions. However, we show that the complexity of the problem changes when we consider "Quasi-DAG"-SyDSs, where the underlying graph becomes a DAG when exactly one directed edge is removed. Specifically, we show using a reduction from the Reachability problem for DAG-SyDSs that the Convergence problem is PSPACE-complete for Quasi-DAG-SyDSs with symmetric local functions (Theorem 5.4).

(4) *Complexity of the Convergence Guarantee Problem.* It was shown in [13] that the Convergence Guarantee problem (i.e., given a SyDS $\mathcal{S}$ on a directed graph, does $\mathcal{S}$ reach a fixed point from every initial configuration?) is PSPACE-complete for SyDSs on general directed graphs. For DAG-SyDSs, we show that the problem is Co-NP-complete, even when restricted to DAGs where each local function is 2-symmetric (Theorem 6.1). We also show that this result is tight; that is, for DAG-SyDSs where each local function is symmetric, the Convergence Guarantee problem can be solved efficiently (Theorem 6.7).

(5) *Efficient Solvability of Reachability Problem for Special Classes of DAG-SyDSs.* We show that the Reachability problem is efficiently solvable for DAG-SyDSs whose local functions are from two specific classes of Boolean functions, namely monotone functions and **nested canalyzing functions**[5] **(NCFs)**. For DAG-SyDSs with monotone local functions, we show that every cycle in the phase space is a fixed point (Theorem 7.1). For DAG-SyDSs with NCF local functions, we show that each phase space cycle is of length at most 2 and that the maximum length of a transient is $2L - 1$, where L is the number of levels in the DAG (Theorem 7.7). These properties imply the efficient solvability of the Reachability problem for these two cases (Theorems 7.2 and 7.8, respectively).

(6) *Complexity of Garden of Eden Existence for DAG-SyDSs.* We show that the **Garden of Eden (GE) Existence** problem (i.e., given a DAG-SyDS, does it have a configuration which does not have a predecessor?) is, in general, NP-complete (Proposition 8.4). However, for DAG-SyDSs where each local function is r-symmetric for some *fixed* r, we show that the GE Existence problem can be solved efficiently (Corollary 8.7).

(7) *Complexity of Other Decision Problems for DAG-SyDSs.* We also establish complexity results for Fixed Point Existence (i.e., given a DAG-SyDS, does it have a fixed point?) and Predecessor Existence (i.e., given a DAG-SyDS $\mathcal{S}$ and a configuration C, does C have a predecessor?). In particular, these problems are shown to be NP-complete even for two-level DAGs and the corresponding counting problems are shown to be #P-complete (Propositions 9.1 and 9.2).

Brief statements of the above results are provided in Table 1.

1.3 Related Work

Reachability problems for various models of discrete dynamical systems have been widely studied in the literature. For example, complexity results for reachability and related problems for Hopfield neural nets were studied in [22], [34], and [35]. Problems related to the diffusion of opinions and other contagions have also been studied under various discrete dynamical systems models (see e.g., [4], [10], [11], [13], and [15] and the references contained therein).

Motivated by dynamical system models for large-scale simulations, Barrett et al. [7] studied the reachability problem for discrete dynamical systems where the underlying graph is undirected and the update model is sequential (i.e., nodes compute and update their state values in a specified order). Synchronous dynamical systems on directed graphs serve as useful models for biological

[5]This class of Boolean functions will be defined in Section 7.3.

Table 1. Brief Statements of the Main Results Presented in the Paper

Problem	Results for DAG-SyDSs
Phase Space properties	The length of each phase space cycle is a power of 2 and the maximum lengths of any transient and cycle in the phase space are $2^L - 1$ and 2^L respectively, where L is the number of levels (Theorem 3.4). There are DAG-SyDSs that achieve these bounds (Theorem 3.5).
Reachability	(a) PSPACE-complete even for symmetric DAG-SyDSs (Theorem 4.7). (b) Efficiently solvable for monotone DAG-SyDSs (Theorem 7.2) and for DAG-SyDSs where each local function is an NCF (Theorem 7.8).
Convergence	PSPACE-complete for Quasi-DAG-SyDSs (Theorem 5.4).
Convergence Guarantee	(a) Co-NP-complete for DAG-SyDSs where each local function is 2-symmetric (Theorem 6.1). (b) Efficiently solvable for DAG-SyDSs where each local function is symmetric (Theorem 6.7).
Garden of Eden Existence	(a) NP-complete even for two-level DAG-SyDSs (Proposition 8.4). (b) Efficiently solvable for r-symmetric DAG-SyDS, where r is a *fixed* integer (Theorem 8.6).
Fixed Point Existence	NP-complete even for two-level DAG-SyDSs (Proposition 9.1).
Predecessor Existence	NP-complete even for two-level DAG-SyDSs (Proposition 9.2).

phenomena [26]. Many researchers have studied reachability and related problems for such dynamical systems (see, e.g., [2], [32], and [33] and the references cited therein). The results in [37] readily imply that the Reachability problem for SyDSs whose local functions are NCFs and whose directed graphs may contain cycles is PSPACE-complete.

Many researchers have presented results for various computational problems for dynamical systems on DAGs. We now provide a summary of these results. Akutsu et al. [2] showed that the problem of controlling a synchronous Boolean network so that it reaches a desired configuration is NP-hard for general directed graphs but is efficiently solvable when the underlying graph is a directed tree. This problem is different from the reachability problem considered here; in particular, the variables that are used to control a network are external to the network and the goal of the controller is to choose appropriate values for those variables at each time step. Materassi and Salapaka [30] considered the problem of inferring the edges in the underlying DAG for a dynamical system given time-series data. A similar problem is considered by Cliff et al. [14] where the underlying DAG is assumed to represent relationships between latent variables of a probabilistic graphical model. Creager et al. [18] point out that dynamical systems on DAGs provide a unifying framework for studying fairness issues that arise when a learning algorithm must interact with a dynamically changing environment. Arnold et al. [3] discuss methods for comparing the effectiveness of several modeling approaches when the underlying causal model for an epidemic can be represented as a DAG. To our knowledge, the reachability problem for DAG-SyDSs was first considered in [28]. They showed that when each local function is a bi-threshold function (i.e., each node has two threshold values to control the $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions), the problem can be solved efficiently. By establishing bounds on the lengths of cycles and transients, they showed that the reachability problem can be efficiently solved for such DAG-SyDSs.

2 DEFINITIONS AND NOTATION

2.1 Graph Theoretic Definitions and Notation

Given a directed graph, we say that node u **directly precedes** node v if the graph contains an edge from u to v, that u **precedes** v if the graph contains a path (possibly with no edges) from u

to v, and that u **properly precedes** v if the graph contains a path with at least one edge from u to v. Note that the local transition function for a given node v in a SyDS over a directed graph is a function of v and the nodes that directly precede v.

Definition 2.1. Let $G(V, E)$ be a DAG. The *level* of a node v in G is the maximum number of edges in any directed path to v.

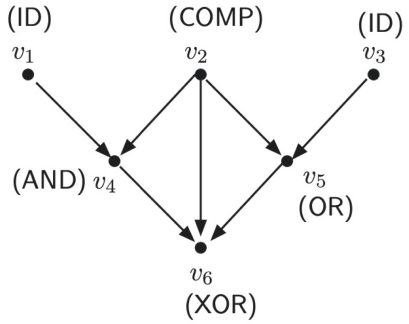
Suppose a given DAG $G(V, E)$ has L levels. For each j, $0 \leq j < L$, we let $\mathcal{L}_j$ denote the set of nodes at level j, and let $\mathcal{L}'_j$ denote the nodes whose level is at most j.

2.2 SyDSs on Directed Graphs

Let $\mathbb{B}$ denote the Boolean domain {0,1}. In this article, a **Synchronous Dynamical System (SyDS)** $\mathcal{S}$ over $\mathbb{B}$ is specified as a pair $\mathcal{S} = (G, \mathcal{F})$, where (i) $G(V, E)$, a directed graph with $|V| = n$, represents the underlying graph of the SyDS, with node set V and (directed) edge set E, and (ii) $\mathcal{F} = (f_1, f_2, \ldots, f_n)$ is a collection 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 has a state value from $\mathbb{B}$. Each Boolean function f_i specifies the local interaction between node v_i and its in-neighbors in G. The inputs to function f_i are the state of v_i and those of the in-neighbors of v_i in G; 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 can be evaluated in polynomial time. In a SyDS, all nodes compute and update their next state *synchronously*. Other update disciplines (e.g., sequential updates) have also been considered in the literature (e.g., [31]). At any time τ, the **configuration** C of a SyDS is the n-vector $(s_1^\tau, s_2^\tau, \ldots, s_n^\tau)$, where $s_i^\tau \in \mathbb{B}$ is the state of node v_i at time τ $(1 \leq i \leq n)$. A **DAG-SyDS** is a SyDS whose underlying graph is a DAG.

Example. An example of a DAG-SyDS is shown in Figure 1. Here, the local functions at nodes v_1 through v_6 are Identity, Complement, Identity, AND, OR, and XOR, respectively. (The Identity and Complement functions have only one input; they return respectively the input value and its complement.) The Boolean expression for each local function is shown in the table in Figure 1. Suppose the initial configuration of the SyDS is $(1, 0, 1, 0, 0, 0)$; that is, at time $\tau = 0$, the states of nodes v_1, v_2, v_3, v_4, v_5 and v_6 are 1, 0, 1, 0, 0, and 0, respectively. Recall that the inputs to the local function at a node v are the current state of v and those of its in-neighbors (if any). Since the local functions at v_1 and v_3 are Identity functions, the states of v_1 and v_3 will remain 1 in all subsequent time steps. Since the local function at v_2 is the Complement function, the state of v_2 at time $\tau = 1$ is 1. Using the local functions at v_4, v_5 and v_6, one can verify that their states at time $\tau = 1$ are 0, 1 and 0 respectively. Thus, the configuration at time $\tau = 1$ is $(1, 1, 1, 0, 1, 0)$. Similarly, the configuration at times $\tau = 2$, $\tau = 3$ and $\tau = 4$ are $(1, 0, 1, 0, 1, 0)$, $(1, 1, 1, 0, 1, 1)$ and $(1, 0, 1, 0, 1, 1)$, respectively. The configuration at time $\tau = 5$ is the same as the configuration at time $\tau = 1$. So, beginning at time $\tau = 1$, the SyDS cycles between the four configurations at times $\tau = 1, 2, 3, 4$.

A Note Regarding the SyDS Model. In our SyDS model discussed above, for each node v, the state of v is one of the inputs to the local transition function f_v at v. Other references that present results for SyDSs over directed graphs (e.g., [2], [32], and [33]) allow a local function f_v to use or ignore the state of v. We now provide a justification for our model in the context of DAG-SyDSs. Specifically, we note that the dynamical behavior of DAG-SyDSs where for each node v, the state of v is not an input to the local function f_v, is rather uninteresting. To see why, assume that the node set of such a DAG-SyDS has been partitioned into k levels, denoted by $\mathcal{L}_0$, $\mathcal{L}_1$, . . ., $\mathcal{L}_{k-1}$, as indicated in Section 2.1. Note that the nodes in $\mathcal{L}_0$ have no incoming edges. Thus, when the local function f_v at a node $v \in \mathcal{L}_0$ does not use the state of v, there is no input to f_v; in other words,



Node	Local Function
v_1	$f_1(s_1) = s_1$
v_2	$f_2(s_2) = \neg(s_2)$
v_3	$f_3(s_3) = s_3$
v_4	$f_4(s_1, s_2, s_4) = s_1 \wedge s_2 \wedge s_4$
v_5	$f_5(s_2, s_3, s_5) = s_2 \vee s_3 \vee s_5$
v_6	$f_6(s_2, s_4, s_5, s_6) = s_2 \oplus s_4 \oplus s_5 \oplus s_6$

Fig. 1. An example of a DAG-SyDS. The local functions at nodes v_1 through v_6 are Identity, Complement, Identity, AND, OR and XOR respectively. In the figure, we abbreviate Identity as "ID" and Complement as "COMP". The table on the right shows the local function at each node as a Boolean formula. In the table, we use s_i to denote the state of node v_i, $1 \leq i \leq 6$. The symbol '$\oplus$' denotes the XOR operator.

the function f_v is a constant[6] function. Thus, the states of the nodes in $\mathcal{L}_0$ do not change after time step 0. The states of the nodes in $\mathcal{L}_1$ do not change after time step 1 since their inputs do not change after time step 0. Inductively, for each i, $0 \leq i \leq k$, the states of nodes in $\mathcal{L}_i$ do not change after time step i. In other words, regardless of the initial configuration, such a DAG-SyDS reaches a fixed point after k time steps. For this reason, our results are for DAG-SyDSs where for each node v, the state of v is an input to the local function f_v. Many papers that study SyDSs on undirected graphs (e.g., [7], [27], and [31]) also use this assumption, and the (undirected) self loop around each node v is not included in the underlying graph. We follow the same convention for DAG-SyDSs; that is, we omit the (directed) self loops around the nodes.

2.3 Additional Definitions and Notation Regarding SyDSs

We now define some additional concepts related to SyDSs. If there is a one step transition of a SyDS from a configuration C_1 to a configuration C_2, we say that C_1 is a **predecessor** of C_2 and that C_2 is the **successor** of C_1. Since the SyDSs considered in this article are deterministic, every configuration has a *unique* successor. However, a configuration may have zero or more predecessors.

A **fixed point** of a SyDS is a configuration C which is its own successor. Thus, when a system reaches a fixed point, it stays in that configuration forever. A **garden of Eden (GE)** configuration is one that has no predecessor. Thus, a GE configuration may only occur as a starting configuration of a SyDS; it cannot be the successor of any configuration.

Recall that the **phase space** of a SyDS is a directed graph in which each node represents a configuration and a directed edge (x, y) indicates that the configuration represented by y is the successor of that represented by x. A **transient** in phase space is a simple directed path P whose last node is part of a directed cycle which does not contain any other node of P. The **length** of a phase space cycle is the number of edges in the cycle, and the **length** of a phase space transient is the number of edges in the transient. In the phase space, each fixed point is self-loop (i.e., a cycle of length 1) and each GE configuration is a node with indegree zero.

Definition 2.2. For the local transition function f_v of a given node v of a directed SyDS, we call the variable corresponding to the source node of each incoming edge to v an **incoming variable** of f_v, and call the variable corresponding to v the **self-variable** of f_v.

[6]Note that all local functions considered in this paper are deterministic.

Let $\mathcal{S}$ be a SyDS. For a given configuration C and node v, we let $C(v)$ denote the state of node v in C. For a given configuration C and set of nodes Y, we let $C[Y]$ denote the projection of C onto Y. We assume that the initial configuration of the system occurs at time 0. For a given initial configuration C and nonnegative integer i, we let C_i denote the configuration of $\mathcal{S}$ at time i.

For a given initial configuration C, we say that a given node v is **stable** at time t if for all $i \geq t$, $C_i(v) = C_t(v)$. Also, we say that a given node v is **alternating** at time t if for all $i \geq 0$, $C_{t+2i}(v) = C_t(v)$ and $C_{t+2i+1}(v) = \overline{C_t(v)}$; in other words, the state of v alternates between 0 and 1.

2.4 Definitions of Some Classes of Boolean Functions

Here, we provide definitions of several classes of Boolean functions used in this article. These definitions are from [7], [17], and [26].

Consider assignments α and β to a set of Boolean variables X. We say that $\alpha \leq \beta$ if for every variable $x \in X$, $\alpha(x) \leq \beta(x)$. A Boolean function f is **monotone** if for every pair of assignments α and β to its variables, $\alpha \leq \beta$ implies that $f(\alpha) \leq f(\beta)$. A **monotone SyDS** is a SyDS whose local transition functions are all monotone.

A **symmetric** Boolean function is one whose value does not depend on the order in which the input bits are specified; that is, the function value depends only on how many of its inputs are 1. For example, the XOR function is symmetric since the value of the function is determined by the parity of the number of 1's in the input. A special class of symmetric functions are threshold functions. For each integer $k \geq 0$, a k-**threshold function**[7] has the value 1 iff at least k of the inputs are 1. A symmetric function with q inputs can be represented by a table of size $q + 1$, with entry i of the table specifying the value of the function when the number of 1's in the input is i, $0 \leq i \leq q$.

In symmetric Boolean functions, we don't know whether a specific input is 0 or 1; we only know the number of 1's in the input. A similar situation arises in anonymous symmetric pure strategy graphical games where each player's strategy is from {0,1}. A player X doesn't know whether a specific neighbor chose 0 or 1; X knows how many neighbors chose 1. Convergence in SyDSs corresponds to reaching a fixed point where everyone is satisfied with the outcome. This is similar to a Nash equilibrium in the game theoretic setting. In this sense, the convergence problem is related to the problem of Nash equilibria in such games and reachability is useful in understanding whether players can achieve certain equilibria (see e.g., [9], [19], [20], and [25]).

A Boolean function f is r-**symmetric** if the inputs to f can be partitioned into at most r groups such that the value of f depends only on how many of the inputs in each of the r groups are 1. Note that symmetric functions defined above are 1-symmetric. For each node v, the majority function used in [13] to study opinion diffusion is defined as follows: Let $N(v)$ denote the set of in-neighbors of v, that is, nodes from which v has an incoming edge. At any given time step, let $A(v) \subseteq N(v)$ and $D(v) = N(v) - A(v)$ denote the in-neighbors of v whose states agree with that of v and differ from that of v, respectively. If $|A(v)| \geq |D(v)|$, then the value of the local function at v is the current state of v; otherwise (i.e., $|D(v)| > |A(v)|$), the value of the function is the complement of the current state of v. This function can be seen to be 2-symmetric: one group contains just the node v and the other group contains all the in-neighbors of v. The value of the function is determined by the number of 1's in these two groups. However, the function is not symmetric. To see this, consider a node v with 5 in-neighbors. Suppose we want to specify the value of the local function at v knowing only that three inputs to the function have the value 1. It can be verified that the value of the local function in this case is the complement of the current state of v. In other words, to specify the value of the function in this case, one needs to know the value of v as well.

[7]These threshold functions are special forms of **weighted threshold functions** [17], where the weight of each input is 1.

An r-symmetric function with q inputs can be represented by a table of size $O(q^r)$, with each row specifying the value of the function for one combination of the the number of 1's in each of the r groups. A SyDS is r-symmetric if each of its local transition functions is r'-symmetric for some $r' \leq r$.

We also consider DAG-SyDSs where each local function is from the class of **Nested Canalyzing Functions** (NCFs) introduced in [26]. This class of functions is defined in Section 7.3.

2.5 Problem Formulations

We now provide formal specifications of the problems considered in this article. These definitions are from [6], [7], and [13]. In these problems, it is assumed that each local function is represented in such a way that the size of the representation is polynomial in the number of inputs. Examples of such representations are Boolean expressions and table representation of r-symmetric functions (mentioned in Section 2.4).

(a) **Reachability**

Instance: A SyDS $\mathcal{S}$ specified by an underlying graph $G(V, E)$ and a local transition function f_v for each node $v \in V$, and two configurations C and $\mathcal{D}$.

Question: Starting from configuration C, does $\mathcal{S}$ reach configuration $\mathcal{D}$?

(b) **Convergence**

Instance: A SyDS $\mathcal{S}$ specified by an underlying graph $G(V, E)$ and a local transition function f_v for each node $v \in V$, and a configuration C.

Question: Starting from configuration C, does $\mathcal{S}$ reach a fixed point?

(c) **Convergence Guarantee**

Instance: A SyDS $\mathcal{S}$ specified by an underlying graph $G(V, E)$ and a local transition function f_v for each node $v \in V$.

Question: Does $\mathcal{S}$ reach a fixed point from *every* initial configuration?

(d) **Garden of Eden Existence**

Instance: A SyDS $\mathcal{S}$ specified by an underlying graph $G(V, E)$ and a local transition function f_v for each node $v \in V$.

Question: Does $\mathcal{S}$ have a GE configuration?

(e) **Predecessor Existence**

Instance: A SyDS $\mathcal{S}$ specified by an underlying graph $G(V, E)$ and a local transition function f_v for each node $v \in V$; a configuration C for $\mathcal{S}$.

Question: Does C have a predecessor, that is, is there a configuration C' such that $\mathcal{S}$ has a one step transition from C' to C?

(f) **Fixed Point Existence**

Instance: A SyDS $\mathcal{S}$ specified by an underlying graph $G(V, E)$ and a local transition function f_v for each node $v \in V$.

Question: Does $\mathcal{S}$ have a fixed point?

The above problems will be explored for DAG-SyDSs in the subsequent sections of this article.

2.6 Organization of This Article

The remainder of this article is organized as follows: Section 3 establishes bounds on the lengths of transients and cycles in the phase spaces of DAG-SyDSs. Section 4 establishes the PSPACE-completeness of the Reachability problem for DAG-SyDSs. Section 5 shows that the Convergence problem is PSPACE-complete for Quasi-DAG-SyDSs. Section 6 shows that the Convergence Guarantee problem is Co-NP-complete for DAG-SyDSs. Section 7 shows that the Reachability problem is efficiently solvable for two special classes of DAG-SyDSs. Section 8 presents our results for the Garden of Eden Existence problem. Section 9 establishes complexity results for Fixed Point Existence and Predecessor Existence problems. Some conclusions and directions for future work are provided in Section 10.

3 SOME PHASE SPACE PROPERTIES OF DAG-SYDSS

In this section, we present some properties of the phase spaces of DAG-SyDSs. These properties are independent of the local functions at the nodes of the DAG-SyDS. We begin with two lemmas that are useful in establishing these properties.

Lemma 3.1. *For a given DAG-SyDS, and a given initial configuration, suppose that the node values of all incoming edges to a given node v are stable at time t. Then node v is either alternating at time t, or stable at time $t+1$.*

Proof. Given the stable value of each incoming variable, the local transition function for node v at any time t', where $t' \geq t$ is a function of only one variable: the self-variable v. The only possible functions of a single variable are a constant function, the identity function, or the complement function. If the local transition function for v is a constant function, then the value of v can possibly change between t and $t+1$, but for any $t' \geq t+1$, remains unchanged. If the local transition function for v is the identity function, then the value of v never changes from its value at time t. If the local transition function for v is the complement function, then the value of v alternates between complementary values. □

Lemma 3.2. *For any DAG-SyDS, each level 0 node is either alternating at time 0 or stable at time 1.*

Proof. A level 0 node has no incoming edges, so the result follows from Lemma 3.1. □

We now show that the phase spaces of DAG-SyDSs may contain exponentially large cycles. The idea behind the construction used to prove this result is also used later in proving Theorem 3.5.

Proposition 3.3. *For every $n > 1$, there is an n node DAG-SyDS whose phase space graph is a cycle of length 2^n.*

Proof. For a given $n > 1$, we construct the DAG-SyDS S_n to be an n-bit counter as follows: The underlying graph contains n levels, one node per level. For each node, there is an incoming edge from the nodes on each of the lower levels. The transition function for each node is the function that retains the current value of the node if any of the lower order bits is 0, and changes the value of the node if all of the lower order bits are 1. In particular, we note that the level 0 node alternates.

Suppose a given configuration of S_n is interpreted as the binary encoding of an integer k, $0 \leq k < 2^n$. Then, the successor configuration encodes the integer $k+1 \pmod{2^n}$. Thus, the phase space of S_n is a cycle of length 2^n. □

For any SyDS, any infinitely long phase space path consists of a transient (possibly of length 0), followed by an infinite number of repetitions of a basic cycle. We now show that for any DAG-SyDS, the length of every phase space cycle is a power of 2. Moreover, the lengths of the longest

transient and the longest phase space cycle are each bounded by an exponential function of the number of levels in the underlying graph of the SyDS.

Theorem 3.4. *For a DAG-SyDS, the length of every phase space cycle is a power of 2. Moreover, if the number of levels of a given DAG-SyDS is L, then no phase space cycle is longer than 2^L, and no transient is longer than $2^L - 1$.*

Proof. The proof is by induction on the number of levels.

Basis. Suppose that there is only one level. Then, the underlying graph has no edges. By Lemma 3.1, either all phase cycles are of length 1 or all are of length 2. Also, no transient is of length greater than 1.

Induction Step. Suppose that the result holds for a given value of L, and consider a DAG-SyDS $\mathcal{S}$ with $L+1$ levels (numbered 0 through L). Let $\mathcal{S}_L$ be the DAG-SyDS with L levels obtained by deleting $\mathcal{L}_L$ (the nodes in level L) from $\mathcal{S}$. Let C be a given configuration of $\mathcal{S}$.

For non-negative integer i, recall that C_i denotes the configuration of $\mathcal{S}$ at time i, when started from configuration C. Since the underlying graph in S has no edges from the level L nodes to the nodes in $\mathcal{S}_L$, $C_i[\mathcal{L}'_{L-1}]$ is the configuration at time i when S_L starts in configuration $C_0[\mathcal{L}'_{L-1}]$.

From the induction hypothesis, when $\mathcal{S}_L$ is started in configuration $C_0[\mathcal{L}'_{L-1}]$, the length of the phase space cycle that is reached is 2^k, where $0 \le k \le L$, and the length of the transient leading to this cycle is some value j such that $j \le 2^L - 1$.

Let v be a level L node of $\mathcal{S}$. The sequence of values taken on by v can be classified as belonging to one of the following three cases:

Case 1. $C_{j+2^k}(v) = C_j(v)$: Then, for every $i \ge j$, $C_{i+2^k}(v) = C_i(v)$. Thus, the values of $C_i(v)$, where $i \ge j$, form a cycle, whose period is 2^k. So, after a transient of length at most j, where $j \le 2^L - 1$, the value of node v cycles with a cycle whose length is a power of 2 and is at most 2^L.

Case 2. $C_{j+2^k}(v) = \overline{C_j(v)}$ and $C_{j+2^{k+1}}(v) = C_j(v)$: Then, for every $i \ge j$, $C_{i+2^{k+1}}(v) = C_i(v)$. Thus, the values of $C_i(v)$, where $i \ge j$, form a cycle, whose period divides is 2^{k+1}. So, after a transient of length at most $2^L - 1$, the value of node v cycles with a cycle whose length is a power of 2 and is at most 2^{L+1}.

Case 3. $C_{j+2^k}(v) = \overline{C_j(v)}$ and $C_{j+2^{k+1}}(v) = \overline{C_j(v)}$: Then, for every $i \ge j + 2^k$, $C_{i+2^k}(v) = C_i(v)$. Thus, the values of $C_i(v)$, where $i \ge j + 2^k$, form a cycle, whose period divides 2^k. Note that C_{j+2^k} is part of this cycle. So, after a transient of length at most $2^{L+1} - 1$, the value of node v cycles with a cycle whose length is a power of 2 and is at most 2^L.

Considering all the nodes in $\mathcal{S}$, the length of the transient beginning at C is at most $2^{k+1} - 1$, and the length of the cycle reached from C divides 2^{k+1}.

Since this holds for all configurations C, the inductive hypothesis holds for $\mathcal{S}$. □

We now show that there are matching lower bounds on cycle and transient length. We show this by constructing a DAG-SyDS that incorporates the counter from Proposition 3.3 to produce a long cycle, and repeated applications of Case 3 from the proof of Theorem 3.4 to produce a long transient.

Theorem 3.5. *For every $L \ge 1$, there is a DAG-SyDS with L levels whose phase space contains a transient of length $2^L - 1$, leading to a cycle of length 2^L.*

Proof. The constructed SyDS $\mathcal{S}$ contains $2L$ nodes, which we refer to as $X = \{x_0, x_1, \ldots, x_{L-1}\}$ and $Y = \{y_0, y_1, \ldots, y_{L-1}\}$. For each i, nodes x_i and y_i will occur on level i.

The underlying graph of $\mathcal{S}$ has directed edges (x_i, x_j) and (x_i, y_j) for each i and j such that $0 \le i < j < L$, and also has directed edges (y_i, y_{i+1}) for each i such that $0 \le i < L - 1$. An example of this DAG for $L = 4$ is shown in Figure 2.

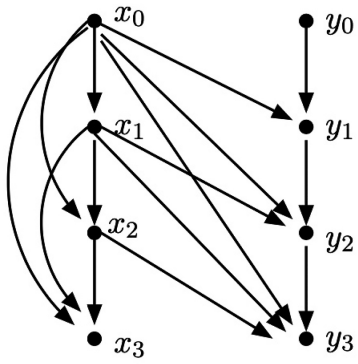


Fig. 2. An example of the DAG constructed in the proof of Theorem 3.5 with $L = 4$.

The local transition function at x_0 is the complement function so that node x_0 alternates. The local transition function for each other node in X is the function that retains the current value of the node if any of the incoming variables equals 0, and changes the value of the node otherwise. Thus, node set X behaves as a counter, similar to the nodes in the construction of Proposition 3.3.

The local transition function for node y_0 is the constant function 1. The local transition function for each node y_i, $1 \leq i \leq L-1$, is 1 iff $(y_i = 1)$ or $(y_{i-1} = 1$ and the incoming variables $\{x_0, x_1, \ldots, x_{i-1}\}$ encode the integer $2^i - 2)$.

Note that the L nodes in X give rise to a phase space cycle of length 2^L.

Consider the initial configuration C, where all nodes have the value 0. It can be seen that by induction on i, for each node y_i, $0 \leq i \leq L-1$, and each nonnegative j, $C_j(y_i)$ equals 0 if $j < 2^{i+1}-1$, and equals 1 if $j \geq 2^{i+1} - 1$. Since y_{L-1} does not change from 0 to 1 until configuration C_{2^L-1}, the length of the transient from C is $2^L - 1$. □

We will also use the following result which is a slight restatement of Proposition 1 in [13]:

Theorem 3.6. *For a DAG-SyDS, the length of a transient leading to a fixed point does not exceed the number of levels of the SyDS.*

4 REACHABILITY

4.1 Overview

In this section, we establish the complexity of the Reachability problem for DAG-SyDSs with symmetric local functions. Our proof uses two major steps. In the first step (Section 4.2), we show the PSPACE-hardness for DAG-SyDSs whose local functions are r-symmetric for a constant value of r. In the second step (Section 4.3), we show that the Reachability problem resulting from the first step can be reduced to the same problem for DAG-SyDSs where each local function is symmetric. We begin with the first step.

4.2 Reachability Problem for DAG-SyDSs with r-Symmetric Local Functions

This section establishes the following result:

Theorem 4.1. *The reachability problem is PSPACE-complete for DAG-SyDSs where each local function is r-symmetric for some $r \leq 6$.*

Proof. It is easy to see that the problem is in PSPACE. The proof of PSPACE-hardness is via a reduction from the **Quantified Boolean Formulas (QBF)** problem which is known to be PSPACE-complete [23]. Let F denote the given quantified Boolean formula. Let f denote the

Boolean expression that is quantified. Without loss of generality, we can assume that f is a 3SAT formula [23]. Let $X = \{x_0, x_1, \ldots, x_{n-1}\}$ be the set of variables of f, and $\{c_0, c_1, \ldots, c_{m-1}\}$ be the set of clauses of f. Let F be $(Q_{n-1}x_{n-1}) \cdots (Q_1x_1)(Q_0x_0)\, f$, where each Q_i is either $\forall$ or $\exists$.

We give a reduction where the constructed DAG-SyDS is 6-symmetric (i.e., each local function is r-symmetric for some $r \leq 6$). Our proof uses a sequential evaluation of a QBF; this type of reduction has been used in the literature (see e.g., [8] and [12]) to establish hardness results for other problems.

For each i, $0 \leq i < n$, we use the notation $\otimes_i$ to denote a binary Boolean operation as follows: If Q_i is $\forall$, then $\otimes_i$ is AND; if Q_i is $\exists$, then $\otimes_i$ is OR.

For the reduction, we construct a reachability problem instance whose SyDS $\mathcal{S}$ has an underlying graph with $2n + m + 2$ nodes, on $2n + 2$ levels.

SyDS $\mathcal{S}$ contains the following nodes. We let $Y = \{y_0, y_1, \ldots, y_n\}$ be a set of $n + 1$ nodes, $R = \{r_0, r_1, \ldots, r_{n-1}\}$ be a set of n nodes, $W = \{w_0, w_1, \ldots, w_{m-1}\}$ be a set of m nodes, and h be an extra node. The initial configuration C for the constructed problem instance has the states of all nodes equal to 0.

The goal configuration $\mathcal{D}$ for the constructed problem instance has $h = 1$, $r_{n-1} = 1$, each $r_i = 0$ for $0 \leq i < n - 1$, each $y_i = 0$, and each w_j equal to 1 iff the corresponding clause c_j contains a positive literal.

Node set Y will function as a cyclical counter, with incoming edges and local transition functions similar to those for the n nodes in the construction presented in our proof of Proposition 3.3. For each i, $0 \leq i < n$, node y_i will correspond to the variable x_i in f.

Each node $w_i \in W$ will correspond to clause c_i.

Each node $r_i \in R$ will correspond to the quantified Boolean formula $(Q_ix_i) \cdots (Q_0x_0)\, f$. In particular, at some point in the operation of $\mathcal{S}$, node r_{n-1} will have the value of the quantified Boolean formula F.

Node h will serve as a control node. Once the value of node h equals 1, it remains 1 forever more, forces node r_{n-1} to retain its value, and forces all the other nodes in R to have value 0.

The underlying graph of $\mathcal{S}$ has directed edges (y_i, y_j) for each i and j such that $0 \leq i < j \leq n$. For each node w_j, there are incoming edges from those Y nodes corresponding to the variables occurring in clause c_j. Node h has incoming edges from the $n + 1$ nodes in Y. Node r_0 has an incoming edge from y_0, h, and from each node in W. For each node r_i, $1 \leq i < n$, there are incoming edges from r_{i-1}, h, and each y_j where $0 \leq j \leq i$. An example of the DAG resulting from this construction is shown in Figure 3.

The local transition function for each node in Y is the function that retains the current value of the node if any of the incoming variables equals 0, and changes the value of the node otherwise.

The local transition function for each node $w_i \in W$ is the same as clause c_i, but using the values of the incoming variables.

The local transition function for h is as follows: If $h = 1$, then 1. If $h = 0$ and the $n + 1$ incoming edges from Y encode the integer $2^n + n - 1$, then 1. Otherwise, 0.

The local transition function for r_0 is as follows: If $h = 1$, then 0. Otherwise, if $y_0 = 1$, then the AND of the incoming variables from W. Otherwise, operation $\otimes_0$ applied to r_0 and the AND of the incoming variables from W.

The local transition function for r_i , $1 \leq i < n - 1$, is as follows: If $h = 1$, then 0. Otherwise, if the $i + 1$ incoming edges from Y encode the integer $2^i + i$, then r_{i-1}. Otherwise, if the incoming edges from Y encode the integer i, then operation $\otimes_i$ applied to r_i and r_{i-1}. Otherwise, r_i.

The local transition function for r_{n-1} is as follows. If $h = 0$ and the n incoming edges from Y encode the integer $2^{n-1} + n - 1$, then r_{n-2}. If $h = 0$ and the n incoming edges from Y encode the integer $n - 1$, then operation $\otimes_{n-1}$ applied to r_{n-1} and r_{n-2}. Otherwise, r_{n-1}.

CNF Formula for a QBF Instance: In this example, the number of variables (n) is 3 and the number of clauses (m) is 2.

Variables: x_0, x_1, x_2

Clauses:

$c_0 = (x_0 \vee \overline{x_1} \vee x_2)$

$c_1 = (\overline{x_0} \vee x_1 \vee \overline{x_2})$

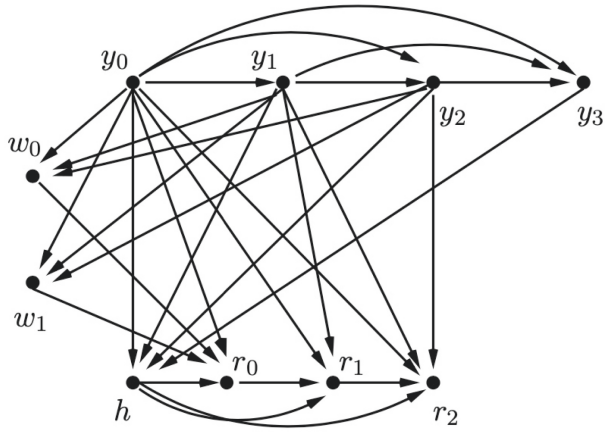


Fig. 3. An example of the DAG constructed in the reduction from the CNF formula of a QBF. The number of variables (n) and the number of clauses (m) in the CNF formula are 3 and 2 respectively. There are $2n+m = 8$ levels in the directed graph, numbered 0 through 7. Nodes y_0, y_1 and y_2 are in levels 0, 1, and 2, respectively. Nodes y_3, w_0 and w_1 are all in level 3. Node h is in level 4. Nodes r_0, r_1 and r_2 are in levels 5, 6, and 7, respectively.

Note that SyDS $\mathcal{S}$ is 6-symmetric; the local transition functions for nodes r_i, $1 \leq i \leq n-1$, contain 6 symmetry classes.

For $0 \leq i < n$, we let F_i be a Boolean function of $n-i-1$ variables, as follows:

$$F_i(x_{n-1}, x_{n-2}, \ldots, x_{i+1}) \;=\; (Q_i x_i) \cdots (Q_1 x_1)(Q_0 x_0)\, f(X)$$

Note that F_{n-1} equals F, the value of the given quantified Boolean formula.

Table 2 illustrates the dataflow as the constructed SyDS $\mathcal{S}$ for $n = 4$ goes through its initial sequence of transitions. Each row of the table corresponds to a configuration of $\mathcal{S}$, starting with the constructed initial configuration $\mathcal{C}$. Each column of the table corresponds to a single node, except for the column labeled W, which corresponds to the entire node set W. Note that configuration $\mathcal{C}_{32}$ equals the constructed goal configuration $\mathcal{D}$, except possibly for the value of node r_3. In particular, $\mathcal{C}_{32}$ equals $\mathcal{D}$ iff the value of the given quantified Boolean formula F is 1.

We now establish the correctness of the reduction.

Let β be a tuple, possibly empty, of Boolean values. Let $l(\beta)$ denote the degree of β, i.e., the number of variables in β. Let $k(\beta)$ denote the integer encoded by β. (If β is empty, then $k(\beta) = 0$.) For β such that $l(\beta) < n$, let $j(\beta) = n - l(\beta) - 1$, and let $t(\beta) = (k(\beta) + 1)2^{n-l(\beta)} + n - l(\beta)$.

In what follows, the tuple β corresponds to the values of the outermost $n-i-1$ variables, namely $x_{n-1}, x_{n-2}, \ldots, x_{i+1}$, of the quantified Boolean formula F.

For any $t \geq 0$, let X^t be the assignment to the variables X of f where $X^t(x_i) = C_t(y_i)$, $0 \leq i \leq n-1$. From the proof in Theorem 3.3, the first n nodes of $\mathcal{S}$ undergo a phase space cycle whose length is 2^n. Thus, all 2^n possible assignments to X occur during this cycle. In particular, all 2^n assignments to X are in the set $\{X^t \mid 0 \leq t < 2^n\}$. Specifically, for any assignment α to X, $X^{k(\alpha)} = \alpha$.

For any assignment α to X, let $W(\alpha)$ be the assignment of values to the nodes set W where w_i equals the value of clause c_i for assignment α, $0 \leq i \leq m-1$. Then, for all $t \geq 0$, $C_{t+1}[W] = W(X^t)$.

Table 2. Initial Configuration Sequence of Constructed SyDS for Four Variables

C_i	r_3	r_2	r_1	r_0	W	h	y_4	y_3	y_2	y_1	y_0
$C_0 = C$	0	0	0	0	$--0--$	0	0	0	0	0	0
C_1	0	0	0	0	$W(0000)$	0	0	0	0	0	1
C_2	0	0	0	$f(0000)$	$W(0001)$	0	0	0	0	1	0
C_3	0	0	0	$F_0(000)$	$W(0010)$	0	0	0	0	1	1
C_4	0	0	$F_0(000)$	$f(0010)$	$W(0011)$	0	0	0	1	0	0
C_5	0	0	$F_0(000)$	$F_0(001)$	$W(0100)$	0	0	0	1	0	1
C_6	0	0	$F_1(00)$	$f(0100)$	$W(0101)$	0	0	0	1	1	0
C_7	0	$F_1(00)$	$F_1(00)$	$F_0(010)$	$W(0110)$	0	0	0	1	1	1
C_8	0	$F_1(00)$	$F_0(010)$	$f(0110)$	$W(0111)$	0	0	1	0	0	0
C_9	0	$F_1(00)$	$F_0(010)$	$F_0(011)$	$W(1000)$	0	0	1	0	0	1
C_{10}	0	$F_1(00)$	$F_1(01)$	$f(1000)$	$W(1001)$	0	0	1	0	1	0
C_{11}	0	$F_2(0)$	$F_1(01)$	$F_0(100)$	$W(1010)$	0	0	1	0	1	1
C_{12}	$F_2(0)$	$F_2(0)$	$F_0(100)$	$f(1010)$	$W(1011)$	0	0	1	1	0	0
C_{13}	$F_2(0)$	$F_2(0)$	$F_0(100)$	$F_0(101)$	$W(1100)$	0	0	1	1	0	1
C_{14}	$F_2(0)$	$F_2(0)$	$F_1(10)$	$f(1100)$	$W(1101)$	0	0	1	1	1	0
C_{15}	$F_2(0)$	$F_1(10)$	$F_1(10)$	$F_0(110)$	$W(1110)$	0	0	1	1	1	1
C_{16}	$F_2(0)$	$F_1(10)$	$F_0(110)$	$f(1110)$	$W(1111)$	0	1	0	0	0	0
C_{17}	$F_2(0)$	$F_1(10)$	$F_0(110)$	$F_0(111)$	$W(0000)$	0	1	0	0	0	1
C_{18}	$F_2(0)$	$F_1(10)$	$F_1(11)$	$f(0000)$	$W(0001)$	0	1	0	0	1	0
C_{19}	$F_2(0)$	$F_2(1)$	$F_1(11)$	$F_0(000)$	$W(0010)$	0	1	0	0	1	1
C_{20}	F_3	$F_2(1)$	$F_1(11)$	$f(0010)$	$W(0011)$	1	1	0	1	0	0
C_{21}	F_3	0	0	0	$W(0100)$	1	1	0	1	0	1
...					...						
C_{31}	F_3	0	0	0	$W(1110)$	1	1	1	1	1	1
C_{32}	F_3	0	0	0	$W(1111)$	1	0	0	0	0	0

For notational convenience, we use the following convention in the rest of the proof. The symbol x_i denotes a variable when used as part of the given quantified formula F. When x_i is used in the context of the tuple β, x_i refers to the Boolean value of the corresponding component of β. The difference between the usage of x_i as a variable and as a Boolean value will be clear from the context.

For any given tuple β of $n - i - 1$ Boolean values corresponding to values of the variables x_{n-1}, x_{n-2}, . . ., x_{i+1}, note that $j(\beta) = i$. Moreover, the value of $F_{j(\beta)}(\beta) = F_i(x_{n-1}, x_{n-2}, \ldots, x_{i+1})$ is determined by the 2^{i+1} assignments to X occurring at times $k(\beta)2^{n-l(\beta)}$ through $(k(\beta) + 1)2^{n-l(\beta)} - 1$.

Claim 1. *For all j', $0 \leq j' < n$, for all β such that $j(\beta) = j'$, $F_{j(\beta)}(\beta) = C_{t(\beta)}(r_{j(\beta)})$.*

We prove the claim by induction on j'.

Basis Step. Suppose that $j' = 0$. Consider any β such that $j(\beta) = 0$. Then, $l(\beta) = n - 1$, $t(\beta) = 2k(\beta) + 3$, and the variables in β are $(x_{n-1}, x_{n-2}, \ldots, x_1)$. By definition,

$$F_0(x_{n-1}, x_{n-2}, \ldots, x_1) = (Q_0 x_0)\, f(x_{n-1}, x_{n-2}, \ldots, x_1, x_0).$$

Thus,

$$F_0(x_{n-1}, x_{n-2}, \ldots, x_1) \;=\; f(x_{n-1}, x_{n-2}, \ldots, x_1, 0) \otimes_0 f(x_{n-1}, x_{n-2}, \ldots, x_1, 1).$$

Given the values $x_{n-1}, x_{n-2}, \ldots, x_1$ occurring in β, let $\gamma^0 = (x_{n-1}, x_{n-2}, \ldots, x_1, 0)$ and $\gamma^1 = (x_{n-1}, x_{n-2}, \ldots, x_1, 1)$. Then, $F_0(\beta) = f(\gamma^0) \otimes_0 f(\gamma^1)$. Note that $\gamma^0 = X^{k(\gamma^0)}$ and $\gamma^1 = X^{k(\gamma^1)}$. Also,

$k(\gamma^0) = 2k(\beta)$ and $k(\gamma^1) = 2k(\beta)+1$. Thus, at time $2k(\beta)+1$, $W = W[\gamma^0]$; and at time $2k(\beta)+2$, $W = W[\gamma^1]$. Since at time $2k(\beta)+1$, $h = 0$, and $y_0 = 0$, the local transition function for r_0 sets $C_{2k(\beta)+2}(r_0)$ to be the AND of the values of the incoming edges from W at time $2k(\beta)+1$. Thus, $C_{2k(\beta)+2}(r_0) = f(\gamma^0)$. Since at time $2k(\beta)+2$, $h = 0$, and $y_0 = 0$, the local transition function for r_0 sets $C_{2k(\beta)+3}(r_0)$ to be result of $\otimes_i$ applied to $C_{2k(\beta)+2}(r_0)$ and the AND of the values of the incoming edges from W at time $2k(\beta)+2$. Thus, $C_{2k(\beta)+3}(r_0) = f(\gamma^0) \otimes_i f(\gamma^1) = F_0(\beta)$. This proves the claim for $j' = 0$.

Inductive Step. Now assume that the claim holds for a given value of j', $0 \le j' < n-1$. We want to prove that the claim holds for $j'+1$. Consider any β such that $j(\beta) = j'+1$. We need to show that $F_{j'+1}(\beta) = C_{t(\beta)}(r_{j'+1})$. We first note that $l(\beta) = n-j'-2$, $t(\beta) = (k(\beta)+1)2^{j'+2} + j' + 2$, and the variables in β are $(x_{n-1}, x_{n-2}, \ldots, x_{j'+2})$. (If $j' = n-2$, then β contains no variables.) By definition, $F_{j'+1}(\beta) = (Q_{j'+1}x_{j'+1}) \cdots (Q_1x_1)(Q_0x_0)f(X)$. Thus, referring to the values of variables $x_{n-1}, x_{n-2}, \ldots, x_{j'+2}$ in β,

$$F_{j'+1}(x_{n-1}, x_{n-2}, \ldots, x_{j'+2}) = F_{j'}(x_{n-1}, x_{n-2}, \ldots, x_{j'+2}, 0) \otimes_{j'+1} F_{j'}(x_{n-1}, x_{n-2}, \ldots, x_{j'+2}, 1).$$

Given the values of $x_{n-1}, x_{n-2}, \ldots, x_{j'+2}$ occurring in β, let $\gamma^0 = (x_{n-1}, x_{n-2}, \ldots, x_{j'+2}, 0)$ and $\gamma^1 = (x_{n-1}, x_{n-2}, \ldots, x_{j'+2}, 1)$. Then,

$$F_{j'+1}(\beta) = F_{j'}(\gamma^0) \otimes_{j'+1} F_{j'}(\gamma^1).$$

Note that $t(\gamma^0) = (k(\gamma^0)+1)2^{j'+1} + j' + 1$. Since $k(\gamma^0) = 2k(\beta)$, $t(\gamma^0) = (2k(\beta)+1)2^{j'+1} + j' + 1 = k(\beta)2^{j'+2} + 2^{j'+1} + j' + 1$. Also, $k(\gamma^1) = 2k(\beta)+1$, and $t(\gamma^1) = (k(\gamma^1)+1)2^{j'+1} + j' + 1 = (2k(\beta)+2)2^{j'+1} + j' + 1 = (k(\beta)+1)2^{j'+2} + j' + 1$. By the inductive hypothesis, $F_{j'}(\gamma^0) = C_{t(\gamma^0)}(r_{j(\gamma^0)}) = C_{t(\gamma^0)}(r_{j'})$, and $F_{j'}(\gamma^1) = C_{t(\gamma^1)}(r_{j(\gamma^1)}) = C_{t(\gamma^1)}(r_{j'})$.

At time $t(\gamma^0) = k(\beta)2^{j'+2} + 2^{j'+1} + j' + 1$, the $j'+2$ incoming edges to $r_{j'+1}$ from Y encode the integer $2^{j'+1} + j' + 1$, so the local transition function for $r_{j'+1}$ evaluates to be the value of $r_{j'}$. Thus, $C_{t(\gamma^0)+1}(r_{j'+1}) = C_{t(\gamma^0)}(r_{j'}) = F_{j'}(\gamma^0)$. The local transition function for $r_{j'+1}$ has $r_{j'+1}$ retain its current value until $t(\gamma^1) = (k(\beta)+1)2^{j'+2} + j' + 1$. At time $t(\gamma^1) = (k(\beta)+1)2^{j'+2} + j' + 1$, the $j'+2$ incoming edges to $r_{j'+1}$ from Y encode the integer $j'+1$, so the local transition function for $r_{j'+1}$ evaluates to be the result of applying operation $\otimes_{j'+1}$ to $r_{j'+1}$ and $r_{j'}$. Thus, $C_{t(\gamma^1)+1}(r_{j'+1}) = C_{t(\gamma^1)}(r_{j'+1}) \otimes_{j'+1} C_{t(\gamma^1)}(r_{j'}) = F_{j'}(\gamma^0) \otimes_{j'+1} F_{j'}(\gamma^1) = F_{j'+1}(\beta)$. However, note that $t(\beta) = (k(\beta)+1)2^{j'+2} + j' + 2 = t(\gamma^1) + 1$. Thus, $F_{j'+1}(\beta) = C_{t(\beta)}(r_{j'+1})$, and Claim 1 follows.

Note that each node in Y and W cycles with a period that divides 2^{n+1}. Since $\mathcal{D}$ has all nodes in Y equal to 0, and this only occurs at a time that is a multiple of 2^{n+1}, $\mathcal{D}$ is reachable iff it is reachable at a time that is a multiple of 2^{n+1}.

Note that $C(h) = 0$ and $\mathcal{D}(h) = 1$. Furthermore, node h is stable with value 1 at time $2^n + n$, and h is equal to 0 at all prior times. Thus, $\mathcal{D}$ is reachable iff it is reachable at a time t such that $t \ge 2^n + n$. Considering node set Y, $\mathcal{D}$ is reachable iff it is reachable at a time that is a nonzero multiple of 2^{n+1}.

Note that every node of $\mathcal{S}$ has the same value for every time that is a nonzero multiple of 2^{n+1}. Thus, $\mathcal{D}$ is reachable iff it is reachable at time 2^{n+1}, i.e., iff $\mathcal{D} = C_{2^{n+1}}$.

Recall that $\mathcal{D}(h) = C_{2^{n+1}}(h)$ and $\mathcal{D}[Y] = C_{2^{n+1}}[Y]$. Since $C_{2^{n+1}-1}[Y]$ consists of all 1's, each $C_{2^{n+1}}(w_j)$ equals 1 iff the corresponding clause c_j contains a positive literal. Thus, $\mathcal{D}[W] = C_{2^{n+1}}[W]$. The nodes in $R - \{r_{n-1}\}$ are stable with value 0 at time $2^n + n + 1$, and so have the same value in $\mathcal{D}$ and $C_{2^{n+1}}$. Thus, $\mathcal{D} = C_{2^{n+1}}$ iff $\mathcal{D}(r_{n-1}) = C_{2^{n+1}}(r_{n-1})$. Since $\mathcal{D}(r_{n-1}) = 1$, $\mathcal{D} = C_{2^{n+1}}$ iff $C_{2^{n+1}}(r_{n-1}) = 1$.

Note that node r_{n-1} is stable at time $2^n + n$, so $C_{2^{n+1}}(r_{n-1}) = C_{2^n+n}(r_{n-1})$. Thus, $\mathcal{D}$ is reachable iff $C_{2^n+n}(r_{n-1}) = 1$. Let β be the empty tuple. As has been shown above, $F_{j(\beta)}(\beta) = C_{t(\beta)}(r_{j(\beta)})$. Note that $l(\beta) = 0$, $k(\beta) = 0$, $j(\beta) = n-1$, and $t(\beta) = (k(\beta)+1)2^{n-l(\beta)} + n - l(\beta) = 2^n + n$. Thus,

$C_{2^n+n}(r_{n-1}) = F_{n-1}$, which is the value of the given quantified Boolean formula F. Thus, F is true iff $\mathcal{D}$ is reachable from C, and this completes our proof of Theorem 4.1. □

4.3 Reducing the Number of Symmetry Classes

In this section, we show how the Reachability problem for r-symmetric SyDSs (for any fixed r) can be reduced to the Reachability problem for symmetric SyDSs. Moreover, when we start with a DAG-SyDS, the reduction produces a DAG-SyDS. Our approach uses the idea of *configuration embedding*, which is defined below. In discussing configuration embeddings, for a pair of configurations C and $\mathcal{D}$, the notation $C \longrightarrow \mathcal{D}$ indicates that $\mathcal{D}$ is the successor of C.

Definition 4.2. Given a pair of SyDSs $\mathcal{S} = (G(V, E), \mathcal{F})$ and $\mathcal{S}' = (G'(V', E'), \mathcal{F}')$, let h be an onto function from V' to V. Let ψ_h be the function that maps each configuration C of $\mathcal{S}$ into the configuration C' of $\mathcal{S}'$ such that for each node $v' \in V'$, $C'(v') = C(h(v'))$. We say that ψ_h is a **configuration embedding** of $\mathcal{S}$ into $\mathcal{S}'$ if the following condition holds: for every configuration C of $\mathcal{S}$, letting $\mathcal{D}$ denote the configuration such that $C \longrightarrow \mathcal{D}$ in $\mathcal{S}$, $\psi_h(C) \longrightarrow \psi_h(\mathcal{D})$ in $\mathcal{S}'$.

Example. Let $\mathcal{S} = (G(V, E), \mathcal{F})$, where $V = \{v_1, v_2, v_3\}$, $E = \{(v_2, v_1), (v_3, v_1)\}$ and $\mathcal{F} = ((v_1 \wedge v_2) \vee v_3,\ \overline{v_2},\ v_3)$. Let $\mathcal{S}' = (G'(V', E'), \mathcal{F}')$, where $V' = \{u_1, u_2, u_3, u_4, u_5\}$, $E' = \{(u_2, u_1), (u_3, u_1), (u_4, u_1), (u_5, u_1)\}$ and $\mathcal{F}' = (f_{u_1}, \overline{u_2}, u_3, u_4, u_5)$, where f_{u_1} is 1 iff at least two inputs are 1. Let h be the following function from V' to V: $u_1 \mapsto v_1, u_2 \mapsto v_2, u_3 \mapsto v_3, u_4 \mapsto v_3, u_5 \mapsto v_3$. Consider the configuration $C = (0, 1, 1)$ of $\mathcal{S}$. Note that $(0, 1, 1) \longrightarrow (1, 0, 1)$ in $\mathcal{S}$, $\psi_h(C) = (0, 1, 1, 1, 1)$, $\psi_h((1, 0, 1)) = (1, 0, 1, 1, 1)$, and that indeed $(0, 1, 1, 1, 1) \longrightarrow (1, 0, 1, 1, 1)$ in $\mathcal{S}'$. More generally, let C be an arbitrary configuration of $\mathcal{S}$. In $\psi_h(C)$, nodes u_3, u_4, and u_5 all have the value of $C(v_3)$. Thus, the evaluation of local function f_{u_1} of $\mathcal{S}'$, given configuration $\psi_h(C)$, will produce the same result as the evaluation of local function f_{v_1} of $\mathcal{S}$. Consequently, it can be seen that ψ_h is a configuration embedding.

Suppose that, in the above example, local function f_{v_1} is an arbitrary 2-symmetric function where one symmetry class consists of v_1 and v_2, and the other symmetry class consists of v_3. Note that function ψ_h has the property that every configuration C of $\mathcal{S}$ maps into a configuration $\psi_h(C)$ of $\mathcal{S}'$ where the number of inputs of f_{u_1} that equal 1 indicates how many members of each of the symmetry classes of f_{v_1} equal 1. Thus, f_{u_1} can be designed so that all the local functions of $\mathcal{S}$ are symmetric, and ψ_h is a configuration embedding.

The next lemma shows a key property of configuration embeddings.

Lemma 4.3. *Let $\mathcal{S} = (G(V, E), \mathcal{F})$ and $\mathcal{S}' = (G'(V', E'), \mathcal{F}')$ be two SyDSs. Let h be an onto function from V' to V such that ψ_h is a configuration embedding of $\mathcal{S}$ into $\mathcal{S}'$. For every pair of configurations C and $\mathcal{D}$ of $\mathcal{S}$, $\mathcal{S}$ starting from C reaches $\mathcal{D}$ iff $\mathcal{S}'$ starting from $\psi_h(C)$ reaches $\psi_h(\mathcal{D})$.*

Proof. Since h is an onto function, ψ_h is injective. For every configuration C of $\mathcal{S}$ and every $t \geq 0$, by simple induction on t, it can be seen that $\psi_h(C_t) = (\psi_h(C))_t$. Thus, given any pair of configurations C and $\mathcal{D}$ of $\mathcal{S}$, the SyDS $\mathcal{S}$ starting from C reaches $\mathcal{D}$ iff the SyDS $\mathcal{S}'$ starting from the configuration $\psi_h(C)$ reaches the configuration $\psi_h(\mathcal{D})$. □

Definition 4.4. A **generalized neighbor** of a given node v in a directed graph is a node that is either the source node of an incoming edge to v, or the node v itself.

We now show that given $\mathcal{S}$, a suitable $\mathcal{S}'$ and h can be constructed efficiently.

Theorem 4.5. *For any fixed value of r, there is a polynomial time algorithm that given a directed SyDS*

$\mathcal{S} = (G(V, E), \mathcal{F})$ with r-symmetric local functions, constructs a directed SyDS $\mathcal{S}' = (G'(V', E'), \mathcal{F}')$ with symmetric local functions, and an onto function h from V' to V, such that ψ_h is a configuration embedding of $\mathcal{S}$ into $\mathcal{S}'$. Moreover, if G is acyclic, then so is G'.

Proof. For a given node $v \in V$, suppose that local function f_v contains r_v classes. Note that $1 \leq r_v \leq r$. Let $g_0^v, g_1^v, \ldots, g_{r_v-1}^v$ be these classes, where g_0^v is the class that contains self-variable v. We assume that function f_v is represented as a r_v-dimensional table T_v, accompanied by a list of which nodes are in each class. Table T_v gives the value of f_v for each tuple of the form $(j_0, j_1, \ldots, j_{r_v-1})$, where for each i, $0 \leq j_i \leq |g_i^v|$.

We construct $\mathcal{S}'$ as follows: Let Δ be the cardinality of the largest class in $\mathcal{F}$. Let $q = \Delta + 1$.

Node set V' and function h are constructed as follows: For each node $v \in V$, node set V' contains q^{r-1} nodes, each of which is mapped into v by h.

Edge set E' is constructed as follows: For a given node $v' \in V'$, suppose that $h(v') = v$. Consider each incoming edge $(u, v) \in E$. Suppose that $u \in g_i^v$. Then, q^i nodes in $h^{-1}(u)$ are selected, and an incoming edge to v' from each of these nodes is added to E'.

Note that for each edge (x, y) in E', $(h(x), h(y))$ is an edge of E. Thus, if G is acyclic, then so is G'.

The tables representing the set of symmetric local functions $\mathcal{F}'$ are constructed as follows: Consider a given node $v' \in V'$. Let $v = h(v')$. Let $c_v = \sum_{i=0}^{r_v-1} q^i |g_i^v|$. Since v lies in class g_0^v, the number of incoming edges into v' is $c_v - 1$. Thus, table $T_{v'}$ specifying symmetric function $f_{v'}$ is a one-dimensional table with an entry for each value j such that $0 \leq j \leq c_v$. Table $T_{v'}$ is filled in so that for each tuple $(j_0, j_1, \ldots, j_{r_v-1})$ corresponding to an entry in the f_v table, the value of $T_v[j_0, j_1, \ldots, j_{r_v-1}]$ is copied into the $T_{v'}$ table entry $T_{v'}[\sum_{i=0}^{r_v-1} q^i j_i]$. More precisely, table $T_{v'}$ is constructed as follows:

Consider the $T_{v'}$ entry for a given j, $0 \leq j \leq c_v$. For each i such that $0 \leq i < r_v$, values k_i and j_i are computed as follows: Set $k_0 = j$, and then for $1 \leq i < r_v$, set $k_i = \lfloor k_{i-1}/q \rfloor$. Then, for $0 \leq i < r_v$, set $j_i = k_i \mod q$. If each j_i satisfies the requirement that $j_i \leq |g_i^v|$, we let $\hat{j}$ be the r_v-tuple $(j_0, j_1, \ldots, j_{r_v-1})$, and we set $T_{v'}[j]$ equal to $T_v[\hat{j}]$. Otherwise, we set $T_{v'}[j]$ equal to 0. Note that all nodes v' such that $h(v') = v$ have the same table.

Since r is fixed, the above construction of $\mathcal{S}'$ can be done in polynomial time.

We now argue that ψ_h is indeed a configuration embedding of $\mathcal{S}$ into $\mathcal{S}'$. Consider a given configuration C of $\mathcal{S}$. Let $\mathcal{D}$ be the successor configuration of C. Let $C' = \psi_h(C)$, and let $\mathcal{D}'$ be the successor configuration of C'.

Consider a given node $v' \in V'$. Suppose that $h(v') = v$. For each i, $0 \leq i < r_v$, let j_i be the number of generalized neighbors of v that are in class g_i^v and have value 1 in C. Then $\mathcal{D}(v) = T_v[j_0, j_1, \ldots, j_{r_v-1}]$. Consider a given class g_i^v of f_v. Let u be a generalized neighbor of v such that $u \in g_i^v$. Then, node v' has q^i generalized neighbors u' such that $h(u') = u$. Since $C' = \psi_h(C)$, for each of these q^i nodes u', $C'(u') = C(u)$. Since the number of generalized neighbors of v from class g_i^v with value 1 in C is j_i, the number of generalized neighbors of v' that map into members of g_i^v under h and have value 1 in C' is $q^i j_i$. Considering all the classes of f_v, the number of generalized neighbors of v' that have value 1 in C' is $j = \sum_{i=0}^{r_v-1} q^i j_i$, so $\mathcal{D}'(v') = T_{v'}[j]$. By the construction of $\mathcal{S}'$, $T_{v'}[j] = T_v[j_0, j_1, \ldots, j_{r_v-1}]$. Thus, $\mathcal{D}'(v') = \mathcal{D}(v)$.

Since the above applies to every node $v' \in V'$, we have that $\mathcal{D}' = \psi_h(\mathcal{D})$. Thus, ψ_h is a configuration embedding of $\mathcal{S}$ into $\mathcal{S}'$. □

The following is the main result of Section 4.3.

Corollary 4.6. *For any fixed value of r, the Reachability problem for directed r-symmetric SyDSs is polynomial-time reducible to the Reachability problem for directed symmetric SyDSs, and the*

Reachability problem for r-symmetric DAG-SyDSs is polynomial-time reducible to the Reachability problem for symmetric DAG-SyDSs.

Proof. From Lemma 4.3 and Theorem 4.5. □

Theorem 4.1 and Corollary 4.6 together imply the following main result of Section 4.

Theorem 4.7. *The Reachability problem for DAG-SyDS with symmetric local functions is PSPACE-complete.*

5 CONVERGENCE PROBLEM FOR QUASI-DAG-SYDSS

We define a **quasi-DAG** to be a directed graph that is not a DAG, but can be turned into a DAG by the deletion of *one* edge. We use the term Quasi-DAG-SyDS for a SyDS whose underlying graph is a quasi-DAG. As shown in [13], the Convergence problem for DAG-SyDSs is efficiently solvable. The results in this section show that the complexity of the Convergence problem changes significantly when one considers Quasi-DAG-SyDSs.

For convenience, we define the following computational problem, which we call *Positive Reachability* . An instance of the Positive Reachability problem consists of a SyDS $\mathcal{S} = (G(V, E), \mathcal{F})$, an initial configuration C, and a nonempty set of distinguished nodes $W \subseteq V$. The computational question is the following: Starting from configuration C, does $\mathcal{S}$ reach a configuration in which all the nodes in W equal 1?

Lemma 5.1. *The Positive Reachability problem is PSPACE-hard for DAG-SyDSs where each local function is 6-symmetric.*

Proof. Consider the reduction in the proof of Theorem 4.1, from quantified Boolean formulas to the Reachability problem for DAG-SyDSs where each local function is 6-symmetric. Let n be the number of variables in the given quantified Boolean formula F. Let $\mathcal{S}$, C, and $\mathcal{D}$ be the constructed DAG-SyDS, initial configuration, and final configuration, respectively. Note that $C_t(h) = 1$ iff $t \geq 2^n + n$. Also note that for all $t \geq 2^n + n$, $C_t(r_{n-1})$ equals the value of F. Thus, $F = 1$ iff $\mathcal{S}$, starting in configuration C, reaches a configuration in which h and r_{n-1} both equal 1.

Consider an instance of the Positive Reachability problem with the same SyDS $\mathcal{S}$, the same initial configuration C, and distinguished node set $W = \{h, r_{n-1}\}$. Then, $F = 1$ iff the constructed instance of the Positive Reachability problem has answer "YES". Thus, the Positive Reachability problem for DAG-SyDSs with 6-symmetric local functions is PSPACE-hard. □

Lemma 5.2. *For any fixed value of r, the Positive Reachability problem for r-symmetric DAG-SyDSs is polynomial-time reducible to the Positive Reachability problem for symmetric DAG-SyDSs.*

Proof. Consider an instance of the Positive Reachability problem consisting of a r-symmetric DAG-SyDS $\mathcal{S} = (G(V, E), \mathcal{F})$, initial configuration C, and set W of distinguished nodes. Since r is fixed, from Theorem 4.5, in polynomial time we can construct a DAG SyDS $\mathcal{S}' = (G'(V', E'), \mathcal{F}')$ with symmetric local functions, and a function h from V' onto V such that ψ_h is a configuration embedding of $\mathcal{S}$ into $\mathcal{S}'$.

Let W' be the set of nodes w of $\mathcal{S}'$ such that $h(w)$ is in W. Let C' be $\psi_h(C)$.

For any configuration $\mathcal{D}$ of $\mathcal{S}$, let $\mathcal{D}'$ denote the configuration $\psi_h(\mathcal{D})$. Since h is an onto function, configuration $\mathcal{D}$ of $\mathcal{S}$ has all nodes in W equal to 1 iff configuration $\mathcal{D}'$ of $\mathcal{S}'$ has all nodes in W' equal to 1. Since ψ_h is a configuration embedding, $\mathcal{D}$ is reachable from C iff $\mathcal{D}'$ is reachable from C'. Consider the Positive Reachability problem instance with SyDS $\mathcal{S}'$, initial configuration C', and distinguished node set W'. Then, SyDS $\mathcal{S}$, starting from configuration C, reaches a configuration in which all the nodes in W equal 1, iff $\mathcal{S}'$, starting from configuration C', reaches a configuration in which all the nodes in W' equal 1. □

Lemma 5.3. *The Positive Reachability problem for DAG-SyDSs with symmetric local functions is polynomial-time reducible to the Convergence problem for Quasi-DAG-SyDSs with symmetric local functions.*

Proof. Consider an instance of the Positive Reachability problem for a DAG-SyDS, with given SyDS $\mathcal{S} = (G(V, E), \mathcal{F})$, where every function in $\mathcal{F}$ is symmetric, initial configuration C, and set W of distinguished nodes.

The reduction constructs an instance of the Convergence problem, consisting of a Quasi-DAG-SyDS $\mathcal{S}' = (G'(V', E'), \mathcal{F}')$ and initial configuration C', as follows:

Let Δ be the maximum node degree in G. Let Y be a set of $\Delta + 2$ new nodes. Let v_0, v_a, and v_q be three additional new nodes. Node set $V' = V \cup Y \cup \{v_0, v_a, v_q\}$. Edge set E' includes all the edges from E, an edge from v_0 to v_a, an edge from v_0 to each node in Y, an edge from v_q to v_0, an edge from each node in Y to each node in V, and an edge from each node in W to v_q. Note that deletion of the edge from v_q to v_0 makes G' a DAG, so G' is a quasi-DAG.

The local function for v_0 is v_0 OR v_q.

The local function for v_a is v_0 NOR v_a.

The local function f'_q for v_q is a threshold function with threshold $|W|$; that is, f'_q equals 1 iff at least $|W|$ of its variables equal 1.

The local function for each node y in Y is v_0 OR y.

The local function f'_i for each node v_i in V is a modification of function f_i from $\mathcal{S}$, as follows: Let δ_i denote the degree of v_i in G. If at most $\delta_i + 1$ of the variables of f'_i equal 1, then f'_i has the same value as f_i; if at least $\delta_i + 2$ of the variables of f'_i equal 1, then f'_i equals 1.

The initial configuration C' for $\mathcal{S}'$ is as follows: The nodes in V have the same value as in C. The nodes in $V' - V$ have value 0.

Suppose that the given Positive Reachability problem instance has answer "YES". Let τ be the earliest time when $\mathcal{S}$, starting from configuration C, reaches a configuration C_τ where all the nodes in W have value 1. Then, when $\mathcal{S}'$ starts in configuration C', $C'_\tau[W]$ consists of all 1's. Consequently, $C'_{\tau+1}(v_q) = 1$. Then, $C'_{\tau+2}(v_0) = 1$. Since the local function for v_0 is OR, for all $t \geq \tau + 2$, $C'_t(v_0) = 1$. Since the local function for v_a is NOR, for all $t \geq \tau + 3$, $C'_t(v_a) = 0$. Moreover, since the local function for each node y in Y is OR, for each node y in Y and all $t \geq \tau + 3$, $C'_t(y) = 1$. Consequently, for each node v_i in V, including every node in W, and all $t \geq \tau + 4$, $C'_t(v_i) = 1$. Then, for all $t \geq \tau + 5$, $C'_t(v_q) = 1$. Thus, $C'_{\tau+5}$ is a fixed point of $\mathcal{S}'$, so the constructed instance of the Convergence problem has the answer "YES".

Now suppose that the given Positive Reachability problem instance has answer "NO". Then $\mathcal{S}'$ starting in configuration C' always has v_q and v_0 equal to 0. Thus, node v_a is alternating at time 0, so the constructed instance of the Convergence problem has answer "NO".

Thus, the answer to the given Positive Reachability problem instance is "YES" iff the answer to the constructed Convergence problem instance is "YES". This completes our proof of Lemma 5.3. □

Theorem 5.4. *The Convergence problem for Quasi-DAG-SyDSs is PSPACE-complete, even when each local function is symmetric.*

Proof. It is easy to see that the problem is in PSPACE. PSPACE-harness follows from Lemmas 5.1–5.3. □

6 CONVERGENCE GUARANTEE PROBLEM FOR DAG-SYDSS

6.1 Overview

As mentioned earlier, Chistikov et al. [13] showed that the Convergence Guarantee problem is PSPACE-complete for SyDSs on general directed graphs. However, they did not address the

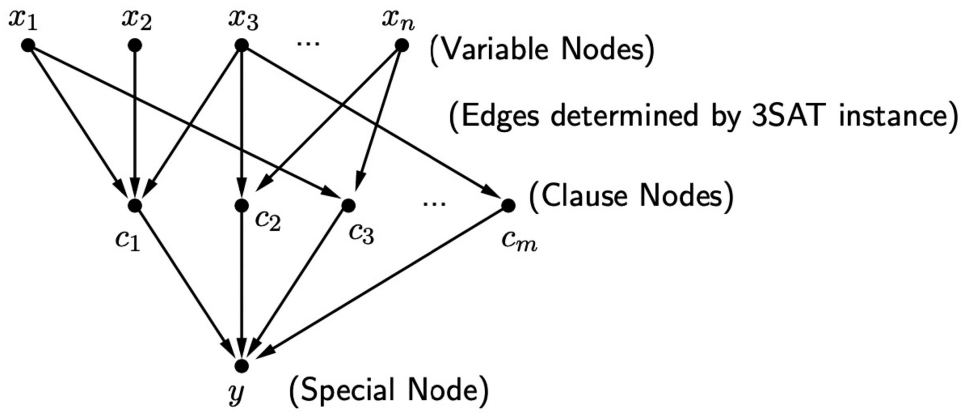


Fig. 4. A schematic of the DAG constructed in the proof of Theorem 6.1.

convergence guarantee problem for DAG-SyDSs. Here, we show that this problem remains computationally intractable for DAG-SyDSs, even when each local function is 2-symmetric. We also show that this result is tight; that is, when each local function is symmetric, the Convergence Guarantee problem for DAG-SyDSs can be solved efficiently.

6.2 Hardness of Convergence Guarantee for DAG-SyDSs with 2-Symmetric Local Functions

Theorem 6.1. *The Convergence Guarantee problem for DAG-SyDSs is Co-NP-complete, even when restricted to DAGs with three levels and 2-symmetric functions.*

Proof. The Convergence Guarantee for DAG-SyDSs is in Co-NP since the "NO" answer can be checked in polynomial time by guessing an initial configuration C and verifying that the given DAG-SyDS does not reach a fixed point by simulating the system for L time steps, where L is the number of levels in the underlying DAG.

To prove NP-hardness, we use a reduction from 1-IN-3 3SAT, where each clause contains only positive literals [23]. The constructed SyDS $\mathcal{S}$ has a level zero node for each variable, a level one node for each clause, and a single level two node. We denote these three sets of nodes as $\{x_1, x_2, \ldots, x_n\}$, $\{c_1, c_2, \ldots, c_m\}$, and $\{y\}$, respectively.

There is a directed edge from each level zero node, say for variable x_i, to each of the level one nodes for the clauses involving variable x_i. There is a directed edge from each level one node to node y. A schematic of the resulting DAG is shown in Figure 4.

The local function for each level zero node x_i ($1 \leq i \leq n$) is the identity function, which is symmetric. The local function for each level one node c_j ($1 \leq j \leq m$) is the function that equals 1 if exactly 1 of the incoming variables equals 1. This function is 2-symmetric with node c_j in one class and the inputs to c_j from level 0 in the other class. The local function for y is the function $\overline{y} \wedge c_1 \wedge c_2 \wedge \cdots \wedge c_m$. This function is 2-symmetric, with the variable y in one class and the remaining variables in the other class.

Note that for every initial configuration, the level zero nodes are all stable at time 0, and the level one nodes are all stable at time 1.

Suppose the given problem instance has a satisfying 1-in-3 assignment. Then any initial configuration of $\mathcal{S}$ where the level zero nodes have values corresponding to a satisfying 1-in-3 assignment results in node y being alternating at time 1. Suppose the given problem instance has no satisfying 1-in-3 assignment. Then, for every initial configuration, node y is stable with value 0 at time 2.

Thus, $\mathcal{S}$ reaches a fixed point from every initial configuration, i.e., convergence is guaranteed for $\mathcal{S}$, iff the given problem instance has no satisfying 1-in-3 assignment. □

6.3 Efficient Solvability of Convergence Guarantee for DAG-SyDSs with Symmetric Local Functions

Here, we show that the Convergence Guarantee problem for DAG-SyDSs in which each local function is symmetric can be solved in polynomial time.

Additional Notation. We assume that the symmetric local function f_v for a node v with Δ_v incoming variables is represented by a table T_v with $\Delta_v + 2$ entries. The value of the entry $T_v[i]$, $0 \leq i \leq \Delta_v + 1$, gives the value of f_v when exactly i of its $\Delta_v + 1$ variables equal 1.

Note that f_v is a constant function iff the entries in T_v are either all 0 or all 1; f_v is a threshold function iff there is no i, $0 \leq i \leq \Delta_v$, such that $T_v[i] = 1$ and $T_v[i+1] = 0$; and f_v is a nonthreshold function iff there is an i, $0 \leq i \leq \Delta_v$, such that $T_v[i] = 1$ and $T_v[i+1] = 0$. In particular, symmetric function f_v is a threshold function iff it is monotone. If f_v is a threshold function, we refer to v as a **threshold node**, and let τ_v denote its threshold. Note that a threshold function f_v is a nonconstant threshold function iff $1 \leq \tau_v \leq \Delta_v + 1$. If f_v is not a threshold function, we refer to v as a **nonthreshold node**.

We will show in Theorem 7.1 that for a DAG-SyDS where every local function is monotone, every configuration leads to a fixed point. So, if every node of a DAG-SyDS is a threshold node, the answer to the Convergence Guarantee problem is "YES".

We refer to a DAG node with no incoming edges as a **source** node of the DAG. Note that for every source node v that is a nonconstant threshold node, f_v is the identity function.

Definition 6.2. Given a DAG-SyDS $\mathcal{S}$ and node y of $\mathcal{S}$, we let $\mathcal{S}^y$ denote the DAG-SyDS consisting of all the nodes that precede y (including node y itself), the edges between these nodes, and their local functions. Given a DAG-SyDS $\mathcal{S}$, a node y such that every node in $\mathcal{S}^y$ is a nonconstant threshold node, source node u in $\mathcal{S}$, and Boolean value b, we say that u b-**forces** y if u is in $\mathcal{S}^y$, and, if so, for every configuration C of $\mathcal{S}^y$ such that $C[u] = b$, the fixed point configuration reached from C has y equal to b.

Suppose that u is in $\mathcal{S}^y$ and every node in $\mathcal{S}^y$ is a nonconstant threshold node. Let $C^{y,u,b}$ denote the configuration of $\mathcal{S}^y$ where source node u has value b, and every other node in $\mathcal{S}^y$ has value $\overline{b}$. Note that since every local function in $\mathcal{S}^y$ is monotone, u b-forces y iff the fixed point configuration reached from $C^{y,u,b}$ has y equal to b.

Suppose that, for a given nonsource node w of a DAG-SyDS, every node v that properly precedes w is a nonconstant threshold node. For source node u and Boolean value b, we let $q_b^{w,u}$ denote the number of nodes y that directly precede w such that u b-forces y.

Lemma 6.3. *Let $\mathcal{S}$ be a DAG-SyDS. Let u be any source node, and w be any nonsource node. Suppose that every node that properly precedes w is a nonconstant threshold node. Then, $q_0^{w,u} + q_1^{w,u} \leq \Delta_w + 1$. Moreover, if w is a nonconstant threshold node, then u does not both 0-force w and 1-force w.*

Proof. The proof is by induction on the level of w. Let L denote the level of w. Since w is a nonsource node, $L \geq 1$.

Basis Step. Suppose $L = 1$. Then the Δ_w nodes that directly precede w are all source nodes, and their local functions are all the identity function. Suppose that u 0-forces w. Then u directly precedes w, f_w is the AND function, and $\tau_w = \Delta_w + 1$. Suppose that u 1-forces w. Then u directly precedes w, f_w is the OR function, and $\tau_w = 1$. Since $\Delta_w \geq 1$, f_u has at least two variables, so u does not both 0-force w and 1-force w.

Induction Step. Suppose the result holds for all levels $\leq L$. Suppose the level of w is $L+1$. If u directly precedes w, then f_u is the identity function, so u both 0-forces u and 1-forces u. For any other source node v that directly precedes w, u neither 0-forces v nor 1-forces v. For any nonsource node v that directly precedes w, from the induction hypothesis, u does not both 0-force v and 1-force v. Thus,

$$q_0^{w,u} + q_1^{w,u} \leq \Delta_w + 1. \tag{1}$$

Now suppose w is a nonconstant threshold node.

Suppose that u 1-forces w. Then, $T_w[q_1^{w,u}] = 1$. Thus, $q_1^{w,u} \geq \tau_w$.

Suppose that u 0-forces w. Then, $T_w[\Delta_w + 1 - q_0^{w,u}] = 0$. Thus, $\Delta_w + 1 - q_0^{w,u} < \tau_w$. I.e., $q_0^{w,u} > \Delta_w + 1 - \tau_w$.

Suppose that u both 0-forces w and 1-forces w. Then ,$q_0^{w,u} > \Delta_w + 1 - \tau_w$ and $q_1^{w,u} \geq \tau_w$, which together imply that $q_0^{w,u} + q_1^{w,u} > \Delta_w + 1$, contradicting Equation (1). Thus, u does not both 0-force w and 1-force w. This completes the induction step, thereby proving the lemma. □

In the remainder of this section, for any pair of SyDSs $\mathcal{S}$ and $\mathcal{S}'$, we use the notation $\mathcal{S} \equiv \mathcal{S}'$ to indicate $\mathcal{S}$ and $\mathcal{S}'$ are identical; that is, they have the same set of nodes, the same underlying graph and each pair of corresponding nodes in the two SyDSs have the same local function.

Definition 6.4. A *critical* DAG-SyDS is a DAG-SyDS where every local function is a nonconstant symmetric function, and there is at least one nonthreshold node. A *simple critical* DAG-SyDS is a critical SyDS $\mathcal{S}$ such that there is exactly one nonthreshold node w and $\mathcal{S} \equiv \mathcal{S}^w$. We call the nonthreshold node of a simple critical DAG-SyDS $\mathcal{S}$ the *critical node* of $\mathcal{S}$.

Lemma 6.5. *Let $\mathcal{S}$ be a simple critical DAG-SyDS. The phase space of $\mathcal{S}$ contains a cycle of length 2.*

Proof. The proof is by induction on the number of nodes in $\mathcal{S}$. Let w denote the critical node of $\mathcal{S}$.

Basis Step. Suppose that $\mathcal{S}$ contains only 1 node. Then, critical node w is the only node in $\mathcal{S}$. Since w has no incoming variables and its local function is a nonthreshold function, f_w is the complement function. Thus, the phase space of $\mathcal{S}$ consists of a cycle of length 2.

Induction Step. Suppose that the result holds for all simple critical SyDSs with at most n nodes, for some $n \geq 1$. Suppose that $\mathcal{S}$ contains $n+1$ nodes.

Let u be any source node of $\mathcal{S}$. Since $\mathcal{S} \equiv \mathcal{S}^w$, u does not have an incoming edge from w, and u is a threshold node. Thus, f_u is a nonconstant threshold function, so f_u is the identity function. For Boolean value b, we define the DAG-SyDS $\mathcal{S}_{u=b}$, as follows. $\mathcal{S}_{u=b}$ has the same set of nodes and edges as $\mathcal{S}$, except that every threshold node v such that u b-forces v is deleted, along with all edges incident on v. For every nondeleted node y, the table T'_y for y in $\mathcal{S}_{u=b}$ is defined as follows: First, note that $q_b^{y,u}$ of the incoming edges to y are deleted, so the number of incoming edges Δ'_y of y in $\mathcal{S}_{u=b}$ is $\Delta'_y = \Delta_y - q_b^{y,u}$. If $b = 0$, then for each i, $0 \leq i \leq \Delta'_y + 1$, $T'_y[i] = T_y[i]$. If $b = 1$, then for each i, $0 \leq i \leq \Delta'_y + 1$, $T'_y[i] = T_y[i + q_1^{y,u}]$. This completes the definition of $\mathcal{S}_{u=b}$.

Let y be a node that is retained in $\mathcal{S}_{u=b}$, such that y is not the critical node of $\mathcal{S}$. So, y is a nonconstant threshold node in $\mathcal{S}$. Since y is retained in $\mathcal{S}_{u=b}$, the local function for y in $\mathcal{S}_{u=b}$ is not a constant function. If $b = 0$, then $\tau_y \leq \Delta_{y'} + 1$, and T'_y represents a threshold function with threshold τ_y. If $b = 1$, then $\tau_y > q_1^{y,u}$, and T'_y represents a threshold function with threshold $\tau_y - q_1^{y,u}$. In either case, every threshold node that is not deleted remains a nonconstant threshold node in $\mathcal{S}_{u=b}$.

Consider critical node w. We define a **critical index** for T_w to be an integer j, $0 \leq j \leq \Delta_w$, such that $T_w[j] = 1$, and $T_w[j+1] = 0$. Since w is a nonthreshold node, there is at least one critical index for T_w.

Suppose that T_w contains a critical index j such that $j \geq q_1^{w,u}$. Then, we let $\mathcal{S}'$ be the DAG-SyDS $[\mathcal{S}_{u=1}]^w$. Otherwise, we note that there is at least one critical index j such that $j < q_1^{w,u}$, and we let $\mathcal{S}'$ be the DAG-SyDS $[\mathcal{S}_{u=0}]^w$.

We claim that $\mathcal{S}'$ is a simple critical DAG-SyDS. First, note that every node in $\mathcal{S}'$ other than w is a nonconstant threshold node. Second, note that $\mathcal{S}' \equiv [\mathcal{S}']^w$. Finally, we claim that w is a nonthreshold node in $\mathcal{S}'$. Consider the two cases of how $\mathcal{S}'$ is obtained from $\mathcal{S}$. Suppose that $\mathcal{S}' \equiv [\mathcal{S}_{u=1}]^w$, so that T_w contains a critical index j such that $j \geq q_1^{w,u}$. Since in table T_w, $T_w[j] = 1$ and $T_w[j+1] = 0$, in table T'_w we have that $T'_w[j - q_1^{w,u}] = 1$ and $T'_w[j + 1 - q_1^{y,u}] = 0$. So, in $\mathcal{S}'$, w is a nonthreshold node. Suppose instead that $\mathcal{S}' \equiv [\mathcal{S}_{u=0}]^w$, so that T_w contains a critical index j such that $j < q_1^{w,u}$. Since every node that properly precedes w is a nonconstant threshold node, from Lemma 6.3, $q_0^{w,u} + q_1^{w,u} \leq \Delta_w + 1$. Thus, $q_1^{w,u} \leq \Delta_w + 1 - q_0^{w,u} = \Delta'_w + 1$. Since $j < q_1^{w,u}$, it follows that $j + 1 \leq \Delta'_w + 1$. Thus, table T'_w contains entries for j and $j + 1$. Since in table T_w, $T_w[j] = 1$ and $T_w[j+1] = 0$, in table T'_w we have that $T'_w[j] = 1$ and $T'_w[j+1] = 0$. So, in $\mathcal{S}'$, w is a nonthreshold node. Thus, $\mathcal{S}'$ is a simple critical DAG-SyDS.

From the induction hypothesis, the phase space of $\mathcal{S}'$ contains a cycle of length 2. Thus, there is a configuration C' of $\mathcal{S}'$ that is part of a phase space cycle of length two in $\mathcal{S}'$. Let b be the Boolean value such that $\mathcal{S}' \equiv [\mathcal{S}_{u=b}]^w$. Let C be the extension of C' to $\mathcal{S}$ where every node that is in $\mathcal{S}$ but not in $\mathcal{S}'$ has value b. Note that in configuration C of $\mathcal{S}$, every node other than w is a stable node at time 0 while w is alternating at time 0. Thus, C is part of a cycle of length 2 in the phase space of $\mathcal{S}$. □

Lemma 6.6. *Let $\mathcal{S}$ be a critical DAG-SyDS. Then. the answer to the Convergence Guarantee Problem for $\mathcal{S}$ is "NO".*

Proof. Let w be a nonthreshold node of $\mathcal{S}$, such that no nonthreshold node of $\mathcal{S}$ occurs at a lower level than w. Then, $\mathcal{S}^w$ is a simple critical DAG-SyDS. From Lemma 6.5, the phase space of $\mathcal{S}^w$ contains a cycle of length 2. Let C' be a configuration of $\mathcal{S}^w$ that leads to a phase space cycle of length 2. Let C be any extension of C' to all the nodes of $\mathcal{S}$. Then C is a configuration of $\mathcal{S}$ that leads to a phase space cycle of length 2, so the answer to the Convergence Guarantee Problem for $\mathcal{S}$ is "NO". □

Theorem 6.7. *The Convergence Guarantee Problem for DAG-SyDSs where each local function is symmetric can be solved in polynomial time.*

Proof. Let $\mathcal{S}$ be a given DAG-SyDSs where each local function is symmetric. An algorithm for the Convergence Guarantee Problem can proceed as follows, by first eliminating all nodes for constant functions, and then checking to see if the local function for any of the remaining nodes is a nonthreshold function.

The algorithm first performs a loop where it removes nodes whose local function is a constant function. Each iteration of the loop proceeds as follows: If every local function is a nonconstant function, the loop is completed. Otherwise, the current iteration selects a node v whose local function is a constant function. Let b be the value of this constant function, i.e., the value in all entries in table T_v is b. The current loop iteration modifies $\mathcal{S}$ to produce SyDS $\mathcal{S}'$, as follows: Node v and all its incident edges are deleted. Consequently, each node y for which $\mathcal{S}$ had an incoming edge from v has this edge deleted. Table T_y is modified as follows: If $b = 0$, the last entry of the table is deleted. If $b = 1$, the first entry of the table is deleted. The current loop iteration then sets S to be S', and the next loop iteration begins.

Consider S after the loop has exited. If S contains any nonthreshold node, then S is a critical DAG-SyDS, so from Lemma 6.6, the answer to the Convergence Guarantee Problem for $\mathcal{S}$ is "NO". So, the algorithm reports "NO", and terminates.

Otherwise, every node in S is a nonconstant threshold function, so the algorithm reports "YES", and terminates. (Note that it is possible that after the loop has exited, $\mathcal{S}$ contains no nodes, in which case all initial configurations for the original given DAG-SyDS lead to the same fixed point configuration.) □

7 REACHABILITY FOR SPECIAL CLASSES OF DAG-SYDSS

7.1 Overview

In this section, we show that the Reachability problem can be solved efficiently for two special classes of DAG-SyDSs. In one class, which we call **monotone DAG-SyDSs**, all local functions are assumed to be monotone (as defined in Section 2.4). In the other class, all the local functions are assumed to be **nested canalyzing functions (NCFs)**. The definition of this class of Boolean functions is given in Section 7.3. For both classes of SyDSs, efficient algorithms are obtained by establishing bounds on the lengths of transients and cycles in the corresponding phase spaces.

7.2 Reachability for Monotone DAG-SyDSs

Theorem 7.1. *For any monotone DAG-SyDS, every cycle is a fixed point, and the length of any transient does not exceed the number of levels.*

Proof. We show by induction on the number of levels that for each level i, every level i node is stable after at most i steps.

From Lemma 3.2, every level 0 node is either stable or alternating at time 1. The complement function is not monotone, so a level 0 node cannot be alternating, and thus is stable at time 1.

Suppose that all the incoming edges to a given node v have stable values at time t. From Lemma 3.1, node v is either stable or alternating at time $t + 1$. Since the local function for node v is monotone, and all the incoming edges have stable values at time t, node v is stable at time $t + 1$. □

We observe that that for every $L \geq 0$, there exists a monotone DAG-SyDS with a transient of length L. Consider the SyDS whose underlying graph is a directed chain of L nodes. The local function of the level 0 node is the constant 1, and of every other node is the *or* of its value and the value from the incoming edge. The configuration of all zeros takes L steps to reach the fixed point of all ones.

Theorem 6.7 shows that for DAG-SyDSs where each local function is symmetric, the Convergence Guarantee problem can be solved efficiently. Theorem 7.1 points out that for DAG-SyDSs where the local functions are monotone, the answer to the Convergence Guarantee problem is always "YES". Moreover, from any initial configuration, such a SYDS reaches a fixed point in a number of steps bounded by the number of levels of the underlying DAG.

The following result is a direct consequence of Theorem 7.1 and Theorem 3.6:

Theorem 7.2. *The Reachability problem for monotone DAG-SyDSs can be solved in polynomial time.*

7.3 Reachability Problem for NCF-DAG-SyDSs

7.3.1 Nested Canalizing Functions and Generalized Nested Canalizing Functions. The class of **Nested Canalizing Functions (NCFs)** was introduced in [26] as a model for the behavior of

certain biological systems. We follow the presentation in [29] in defining NCFs. As usual, for a Boolean value b, the complement is denoted by $\overline{b}$.

Definition 7.3. Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a set of n Boolean variables. Let π be a permutation of $\{1, 2, \ldots, n\}$. A Boolean function $f(x_1, x_2, \ldots, x_n)$ over X is **nested canalizing** in the variable order $x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}$ with **canalizing values** $a_1, a_2, \ldots, a_n$ and **canalized values** $b_1, b_2, \ldots, b_n$ if f can be expressed in the following form:

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

For convenience, we will use a notation introduced in [39] to represent NCFs. For $1 \leq i \leq n$, line i of this representation has the form

$$x_{\pi(i)} : a_i \longrightarrow b_i$$

with $x_{\pi(i)}$ being the **canalizing variable** that is tested in line i, and a_i and b_i being respectively the **canalizing** and **canalized** values in line i, $1 \leq i \leq n$. Each such line is called a **canalization rule**. When none of the conditions "$x_{\pi(i)} = a_i$" is satisfied, we have line $n + 1$ with the "Default" rule for which the canalized value is $\overline{b_n}$:

Default: $\overline{b_n}$

As in [39], 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. Note that the default value is always the complement of the canalized value in the last canalizing rule.

Example. Consider the function $f(x_1, x_2, x_3) = x_1 \vee (\overline{x_2} \wedge x_3)$. This function is nested canalizing using the identity permutation π on $\{1, 2, 3\}$ with canalizing values $1, 1, 0$ and canalized values $1, 0, 0$. A simplified representation of this function is as follows:

$$x_1 : 1 \longrightarrow 1$$

$$x_2 : 1 \longrightarrow 0$$

$$x_3 : 0 \longrightarrow 0$$

Default: 1

NCFs can be seen as a special form of decision lists studied in [36]. We will also consider a generalization of NCFs, as defined in [39].

Definition 7.4. A *generalized NCF* is a function that can be represented as either a constant or an NCF representation of a subset (not necessarily proper) of the function's variables.

Example. Consider a Boolean function $f_1(x_1, x_2, x_3, x_4)$ whose generalized NCF representation is as follows:

$$x_2 : 1 \longrightarrow 0$$

$$x_3 : 0 \longrightarrow 1$$

Default: 0

Note that the function f_1 depends only on the variables x_2 and x_3.

We now show that generalized NCFs are closed under projection, i.e., if some of the variables of a generalized NCF are assigned constant values, the resulting function of the remaining variables is also a generalized NCF.

Lemma 7.5. *Generalized NCFs are closed under projection.*

Proof. We focus on showing that if a single variable of a generalized NCF f is assigned a specified constant value, then the projection onto the remaining variables is a generalized NCF. The lemma follows by induction on the number of variables assigned constant values.

So, consider the projection $f_{x=a}$, where a variable x is assigned value a, and consider a given generalized NCF representation of f.

Case 1: Variable x does not occur in any line of the representation of f. Then, $f_{x=a}$ is a generalized NCF, with the same representation as f.

Case 2: Variable x is the canalizing variable in some line of the representation of f. We call this line **critical line**. Let b denote the canalized value in the critical line.

Case 2A: The canalizing value in the critical line is a.

Case 2A(i): No canalization rule with canalized value $\overline{b}$ precedes the critical line. Then $f_{x=a}$ is the constant b.

Case 2A(ii): There is a canalization rule preceding the critical line for which the canalized value is $\overline{b}$. Let the last line that precedes the critical line and for which the canalized value is $\overline{b}$, be called the **semicritical line**. Then a generalized NCF representation of $f_{x=a}$ can be obtained from the representation of f by deleting all lines after the semicritical line, and making the default value be b.

Case 2B: The canalizing value in the critical line is $\overline{a}$.

Case 2B(i): The critical line contains the last canalization rule in the representation of f (so that the default value is $\overline{b}$), and every canalization rule other than the the one in the critical line has canalized value $\overline{b}$. Then $f_{x=a}$ is the constant $\overline{b}$.

Case 2B(ii): Either there is a canalization rule after the critical line, or the critical line is immediately preceded by a line containing a canalization rule with canalized value b. Then, a generalized NCF representation of $f_{x=a}$ can be obtained from the representation of f by deleting the critical line.

Case 2B(iii): The critical line contains the last canalization rule in the representation of f (so that the default value is $\overline{b}$), is immediately preceded by a line containing a canalization rule with canalized value $\overline{b}$, and there is at least one other line containing a canalization rule with canalized value b. Let the last line that precedes the critical line and for which the canalized value is b, be called the **semicritical line**.

Then a generalized NCF representation of $f_{x=a}$ can be obtained from the representation of f by deleting all lines after the semicritical line, and making the default value be $\overline{b}$. □

7.3.2 The Reachability Problem for Generalized-NCF-DAG-SyDSs. We are now ready to discuss our result for the Reachability problem for DAG-SyDSs where each local function is a generalized NCF. We refer to such SyDSs as Generalized-NCF-DAG-SyDSs. We begin with a lemma that points out a timing property of certain nodes.

Lemma 7.6. *Let $\mathcal{S}$ be a generalized-NCF-DAG-SyDS. For a given initial configuration, let v be a node such that each incoming variable of v is either stable or alternating at time t. Then, for that initial configuration, v is either stable or alternating at time $t+2$.*

Proof. Let V' be the set of nodes with incoming edges to v, such that they are alternating at time t, and let $V'' = V - V'$ be the set of nodes with incoming edges to v, such that they are stable

at time t. Let g_v be the generalized-NCF obtained from f_v by replacing the variables in V'' with their stable values, shortening and deleting lines from the representation of the NCF as appropriate. Note that g_v determines the transitions of node v for all times t' such that $t' \geq t$.

Let α be the assignment of Boolean values to V' where each node is assigned its value at time t, and let $\overline{\alpha}$ be the assignment of Boolean values to V' where each node is assigned the complement of its value in α. Note that for every $q \geq 0$, α corresponds to the values of the nodes in V' at time $t + 2q$, and $\overline{\alpha}$ corresponds to the values of the nodes in V' at time $t + 2q + 1$.

The proof proceeds by a case analysis of g_v. Consider a given generalized NCF representation of g_v.

Case 1: Variable v does not occur in any line in the representation of g_v. If $g_v(\overline{\alpha}) = g_v(\alpha)$, then node v is stable at time $t + 1$. If $g_v(\overline{\alpha}) \neq g_v(\alpha)$, then node v is alternating at time $t + 1$.

Case 2: Variable v occurs in the representation of g_v, but is not the canalizing variable in the first line of the representation. Let u be the canalizing variable in the first line of g_v, with canalizing value a_u, and canalized value b_u. Since u is alternating at time t, and assignments α and $\overline{\alpha}$ are complementary, u equals a_u in either α or $\overline{\alpha}$. If u equals a_u in α, then $C_{t+1}(v)$ and $C_{t+3}(v)$ both equal b_u, so v is either stable or alternating at time $t + 1$. If u equals a_u in $\overline{\alpha}$, then $C_{t+2}(v)$ and $C_{t+4}(v)$ both equal b_u, so v is either stable or alternating at time $t + 2$.

Case 3: Variable v is the canalizing variable in the first line of the representation of g_v.

Case 3A: The first line of g_v is the following, for some canalizing value a:

$$v : a \longrightarrow a$$

If $C_{t+q}(v) = a$ for some $q \geq 0$, then node v is stable with value a at time $t+q$. In particular, if $C_{t+q}(v) = a$, then v is stable with value a at time t. Otherwise, if $C_t(v) = \overline{a}$ but $C_{t+1}(v) = a$, then v is stable with value a at time $t + 1$. Otherwise, if $C_t(v) = \overline{a}$ and $C_{t+1}(v) = \overline{a}$, but $C_{t+2}(v) = a$, then v is stable with value a at time $t + 2$. Otherwise, $C_t(v)$, $C_{t+1}(v)$ and $C_{t+2}(v)$ all equal $\overline{a}$; consequently v is stable with value $\overline{a}$ at time t.

Case 3B: The first line of g_v is the following, for some canalizing value a:

$$v : a \longrightarrow \overline{a}$$

Case 3B1: Suppose $C_t(v) = a$. Then $C_{t+1}(v) = \overline{a}$. If $C_{t+2}(v) = a$, then node v is alternating at time t. So, suppose that $C_{t+2}(v) = \overline{a}$. If $C_{t+3}(v) = \overline{a}$, then node v is stable with value $\overline{a}$ at time $t + 1$. If $C_{t+3}(v) = a$, then $C_{t+4}(v) = \overline{a}$, so node v is alternating at time $t + 2$.

Case 3B2: Suppose $C_t(v) = \overline{a}$. If $C_{t+1}(v) = a$, then $C_{t+2}(v) = \overline{a}$, and v is alternating at time t. If $C_{t+1}(v) = \overline{a}$ and $C_{t+2}(v) = a$, then node v is alternating at time $t + 1$. If $C_{t+1}(v) = \overline{a}$ and $C_{t+2}(v) = \overline{a}$, then node v is stable at time t.

This completes all the cases, and the lemma follows. □

The above lemma enables us to prove bounds on the maximum lengths of transients and cycles in the phase space of a Generalized-NCF-DAG-SyDS.

THEOREM 7.7. *For a generalized-NCF-DAG-SyDS $\mathcal{S}$, the length of every phase space cycle is at most 2. Moreover, if the number of levels of $\mathcal{S}$ is L, no transient is longer than $2L - 1$.*

PROOF. From Lemma 3.2, each level 0 node is either stable or alternating at $t = 1$.

From Lemma 7.6, for each subsequent level, each node at that level is either stable or alternating at most two steps after all the nodes at lower levels have become stable or alternating.

Since each node of $\mathcal{S}$ eventually becomes either stable or alternating, the length of every phase space cycle is at most 2. Moreover, a node at a given level j becomes either stable or alternating after at most $2j + 1$ steps. Since $\mathcal{S}$ contains L levels, any directed path leading to a fixed point or a cycle of length 2 can contain at most $2L - 1$ configurations. □

Since Theorem 7.7 provides a polynomial bounds on the lengths of transients and cycles in the phase space of any Generalized-NCF-DAG-SyDS, the following result is immediate:

Theorem 7.8. *The Reachability problem is efficiently solvable for Generalized-NCF-DAG-SyDSs.*

8 GARDEN OF EDEN EXISTENCE PROBLEM

Recall that a Garden of Eden (GE) configuration of a SyDS is one which has no predecessor. In this section, we present our results for the GE existence problem for DAG-SyDSs. To do this, we begin with an observation and some definitions for *directed* SyDSs, that is, SyDSs whose underlying graph are directed, but not necessarily acylic.

Observation 8.1. *A directed SyDS has a GE configuration iff there exist two distinct configurations with the same successor.*

Proof. Let $\mathcal{S}$ be a directed SyDS with n nodes. Since $\mathcal{S}$ is deterministic, each configuration has a unique successor. Thus, the outdegree of each of the 2^n nodes in the phase of $\mathcal{S}$ is 1 and so the number of directed edges in the phase space is also 2^n. To prove the "if" direction, suppose there are two distinct configurations C_1 and C_2 of $\mathcal{S}$ such that they have the same successor, say C. Thus, indegree of C is ≥ 2. If there is no GE configuration, then each of the $2^n - 1$ nodes of the phase space (i.e., each node except C) has an indegree of at least 1 and C has an indegree of at least 2; that is, the number of directed edges in the phase space is at least $2^n + 1$, a contradiction. Thus, $\mathcal{S}$ must have a GE configuration. For the "only if" part, let C' be a GE configuration. Thus, we have 2^n configurations for which there are only $2^n - 1$ successors. Hence, by the pigeonhole principle, there must be two configurations with the same successor. □

Definition 8.1. For a node v of a directed SyDS, an *incoming assignment* is an assignment of a value to each of the incoming variables of v

For a node v, Boolean value a and incoming assignment α, we use the notation $f_v(a, \alpha)$ to denote the value of local function f_v when $v = a$ and the incoming variables have the values in α.

Definition 8.2. A variable x of a Boolean function f is *fully essential* to f if for every assignment to the other variables of f, the value of f depends on the value of x.

The local function f_v of a node v of a given directed SyDS is *fully self-essential* if self-variable v is fully essential to f_v; that is, for every incoming assignment α, $f_v(0, \alpha) \neq f_v(1, \alpha)$.

Theorem 8.3. *A DAG-SyDS $\mathcal{S}$ has a Garden of Eden configuration iff at least one of its local functions is not fully self-essential.*

Proof. **If:** Suppose that $\mathcal{S}$ has at least one local function that is not fully self-essential. Let v be a node whose local function is not fully self-essential. Let α be an incoming assignment to v, such that $f_v(0, \alpha) = f_v(1, \alpha)$. Let j be the level of v. Let C be any configuration of $\mathcal{L}'_j$ such that that C has the values from α on the incoming variables of v. Let $\mathcal{D}$ be the configuration on $\mathcal{L}'_j$ that is identical to C, except that $\mathcal{D}(v) \neq C(v)$. Since C and $\mathcal{D}$ differ only in the value of v, they have the same successor configuration on $\mathcal{L}'_j$. Since there are two configurations of $\mathcal{L}'_j$ with the same successor, there is at least one configuration of $\mathcal{L}'_j$ with no predecessor configuration. Any extension of this configuration to a full configuration of $\mathcal{S}$ has no predecessor, and so is a GE configuration of $\mathcal{S}$.
Only If: Suppose that all the local functions of $\mathcal{S}$ are fully self-essential. We now argue that every pair of distinct configurations of $\mathcal{S}$ has distinct successors. Let C and $\mathcal{D}$ be distinct configurations of $\mathcal{S}$. Let j be the lowest level number of $\mathcal{S}$ such that $C[\mathcal{L}_j] \neq D[\mathcal{L}_j]$. Thus, $C[\mathcal{L}'_{j-1}] = D[\mathcal{L}'_{j-1}]$. Let v be a level j node such that $C[v] \neq \mathcal{D}[v]$. Since $C[\mathcal{L}'_{j-1}] = D[\mathcal{L}'_{j-1}]$, the incoming assignment to v is the same in C and $\mathcal{D}$. Let C_1 denote the successor of C, and $\mathcal{D}_1$ denote the successor of

$\mathcal{D}$. Since the local function of node v is fully self-essential, $C_1(v) \neq \mathcal{D}_1(v)$. Thus, every pair of distinct configurations of $\mathcal{S}$ has distinct successors. Consequently, by Observation 8.1, $\mathcal{S}$ has no GE configuration. □

We now establish the complexity of the GE Existence problem for DAG-SyDSs.

Proposition 8.4. *The GE existence problem for DAG-SyDSs whose local functions are specified as Boolean formulas is NP-complete, even when restricted to DAGs with two levels.*

Proof. The membership in NP of the GE existence problem for all directed SyDSs follows from Observation 8.1.

The proof of NP-hardness is via a reduction from 3SAT. Let f denote the given 3SAT Boolean formula. Let n be the number of variables in f. For the reduction, we construct a SyDS $\mathcal{S}$ with $n+1$ nodes. Let X denote the first n nodes of $\mathcal{S}$, all of which are in layer zero. The local function of each of these nodes is the identity function. Node x_{n+1} has an incoming edge from each of the other n nodes. Its local function is: $x_{n+1} \vee f$.

Note that the local function for each node in X is fully self-essential. Also note that every assignment to the nodes in X is an incoming assignment to node x_{n+1}. We also observe that the local function for x_{n+1} can change the value of x_{n+1} only if $x_{n+1} = 0$ and the values of the other variables satisfy f.

Suppose that formula f is satisfiable. Let α be a satisfying assignment for f. Then, $f_{x_{n+1}}(0, \alpha) = f_{x_{n+1}}(1, \alpha) = 1$, so $f_{x_{n+1}}$ is not fully self-essential.

Now suppose that formula f is unsatisfiable. Then, the local function for x_{n+1} is the identity function, which is fully self-essential.

Thus, formula f is satisfiable iff the local function for x_{n+1} is not fully self-essential Consequently, from Theorem 8.3, f is satisfiable iff $\mathcal{S}$ has a Garden of Eden configuration. This completes the proof of Proposition 8.4. □

We now consider the GE problem for DAG-SyDSs whose local functions are each represented as an r-symmetric table, accompanied by a list of the nodes in each of the r groups. We denote these groups as $g_0, g_1, \ldots, g_{r-1}$. Such a table gives the value of the function for each tuple of the form $(j_0, j_1, \ldots, j_{r-1})$, where for each i, $0 \leq j_i \leq |g_i|$.

Lemma 8.5. *Suppose an r-symmetric function f is represented as an r-dimensional table T, accompanied by a list of which variables are in each group. The problem of determining whether a given variable x is fully essential to f can be solved in time that is linear in the size of the input. Moreover, if x is not fully essential to f, an assignment α to the variables of f other than x, such that f has the same value when α is extended to a complete assignment by setting x to either 0 or 1, can be constructed in time that is linear in the size of the input.*

Proof. Let g_k be the group containing x. Then x is fully essential to f iff for all tuples $(j_0, j_1, \ldots, j_{r-1})$, where $0 \leq j_k < |g_k|$ and for each $i \neq k$, $0 \leq j_i \leq |g_i|$,

$$T[j_0, j_1, \ldots, j_{k-1}, j_k, j_{k+1}, \ldots, j_{r-1}] \neq T[j_0, j_1, \ldots, j_{k-1}, j_k + 1, j_{k+1}, \ldots, j_{r-1}].$$

This property can be tested for in time linear in the size of the input. Moreover, if there is a tuple such that

$$T[j_0, j_1, \ldots, j_{k-1}, j_k, j_{k+1}, \ldots, j_{r-1}] = T[j_0, j_1, \ldots, j_{k-1}, j_k + 1, j_{k+1}, \ldots, j_{r-1}],$$

such a tuple can be can be found in time that is linear in the size of the input. From this tuple, an assignment to the variables of f such that f has the same value whether or not x is complemented in the assignment, can be constructed in time that is linear in the size of the input. □

Theorem 8.6. *Given a DAG-SyDSs $\mathcal{S}$ whose local functions are specified as r-symmetric tables for some r, the GE existence problem can be solved in time that is linear in the size of the input. Moreover, if a GE configuration exists, one can be constructed in time that is linear in the size of the input.*

Proof. From Theorem 8.3, $\mathcal{S}$ has a GE configuration iff at least one of its local functions is not fully self-essential. From Lemma 8.5, each local function table for $\mathcal{S}$ can be tested for the property of being fully self-essential, in time linear in the size of the table for that node. Thus, the GE existence problem can be solved in time linear in the size of the description of $\mathcal{S}$.

Suppose that at least one local function is not fully self-essential. A GE configuration can be found as follows. Let v be a node whose local function is not fully self-essential, but the local functions of all nodes whose level is below that of v are fully self-essential. Let j denote the level of v. Thus, all nodes in $\mathcal{L}'_{j-1}$ are fully self-essential, so, from Theorem 8.3, $\mathcal{L}'_{j-1}$ has no GE configuration.

From Lemma 8.5, in linear time an incoming assignment α to v can be found such that $f_v(0, \alpha) = f_v(1, \alpha)$. Let α' be any extension of α to all the nodes in $\mathcal{L}'_{j-1}$. Let β be the successor configuration of α' on $\mathcal{L}'_{j-1}$. Since $\mathcal{L}'_{j-1}$ has no GE configuration, α' is the only predecessor configuration of β. Let $b = f_v(0, \alpha)$. Let γ be any configuration of $\mathcal{S}$ such that $\gamma(v) = \bar{b}$ and $\gamma[\mathcal{L}'_{j-1}] = \beta$. Since $\gamma[\mathcal{L}'_{j-1}] = \beta$, any predecessor of γ must equal α' on $\mathcal{L}'_{j-1}$. Since $\gamma(v) = \bar{b}$, any predecessor of γ cannot equal α' on $\mathcal{L}'_{j-1}$. Thus, γ has no predecessor configuration, so γ is a GE configuration of $\mathcal{S}$. □

When an r-symmetric local function f_v with k variables is specified by giving the table representation, the size of the given table is $O(k^r)$. As noted in the following corollary, when r is *fixed*, the size of the representation of SyDS $\mathcal{S}$ and the running time of the Theorem 8.6 algorithm for solving the GE existence problem and constructing a GE configuration when a GE configuration exists are both polynomial functions of n, the number of nodes. However, even when r is not fixed, while the size of the input and the running time of the algorithm are not necessarily polynomial functions of n, the running time remains linear in the input size, i.e., linear in the size of the given representation of $\mathcal{S}$.

Corollary 8.7. *For any fixed value of r, given a DAG-SyDSs $\mathcal{S}$ whose local functions are specified as r-symmetric tables, the GE existence problem can be solved in time that is polynomial in n. Moreover, if a GE configuration exists, one can be constructed in time that is polynomial in n.*

9 COMPUTATIONALLY HARD PROBLEMS FOR TWO-LEVEL DAG-SYDSS

In this section, we consider the Fixed Point Existence and Predecessor Existence problems for DAG-SyDSs. (These problems were defined in Section 2.5.) We show the decision versions of these problems are NP-complete and their counting versions are #P-complete even for SyDSs whose underlying DAGs have only two levels. In proving the hardness results for the counting problems, we use the definitions of #P and #P-completeness from [23]. Specifically, a counting problem Π is in #P if the number of accepting computations of a polynomial-time bounded nondeterministic Turing machine for the corresponding decision problem is equal to the number of solutions to the problem. A counting problem Π is #P-complete if $\Pi \in$ #P and for all $\Pi' \in$ #P, there is a polynomial time Turing reduction from Π' to Π. Such a reduction allows us to efficiently compute the number of solutions to each instance I' of Π' from the number of solutions to the instance I of Π resulting from the reduction. One special form of Turing reduction is a *parsimonious reduction*. A polynomial time parsimonious reduction from a counting problem Π' to a counting problem Π is such that for each instance $I' \in \Pi'$, the number of solutions to I' is equal to that of the instance I of Π produced by the reduction. Our reductions to prove NP-hardness also serve as Turing reductions.

Proposition 9.1. *The Fixed Point Existence problem for DAG-SyDSs is NP-complete, even when restricted to DAGs with two levels and maximum node degree 3. Further, the problem of counting the number of fixed points of such DAG-SyDSs is #P-complete.*

Proof. It can be seen that the Fixed Point Existence problem is in NP and the corresponding counting problem is in #P. To prove NP-hardness, we use a reduction from 3SAT. The constructed SyDS S has a level-zero node for each variable, and a level-one node for each clause. There is a directed edge from a level-zero node for a given variable to each of the nodes for the clauses involving that variable. The local function for each level-zero node is the identity function. The local function for each level-one node is the function that equals 1 if the value of the node is 0, or if the value of at least one of the incoming edges makes the clause true.

Suppose the given 3SAT problem instance is satisfiable. Then the configuration of S where the level-zero nodes have values corresponding to a satisfying assignment, and the level-one nodes all have value 1, is a fixed point.

Suppose that SyDS S has a fixed point. In a fixed point, the level-one nodes all have value 1, and the values of the level-zero nodes in each fixed point constitute a satisfying assignment.

Thus, S has a fixed point iff the given 3SAT problem instance is satisfiable.

We observe that the above reduction is parsimonious; that is, the number of fixed points of the constructed DAG-SyDS is equal to the number of satisfying assignments of the given 3SAT instance. Therefore, by carrying out the above reduction from #3SAT establishes the #P-completeness of counting the number of fixed points of a DAG-SyDS. □

Proposition 9.2. *The predecessor existence problem for DAG-SyDSs is NP-complete, even when restricted to DAGs with two levels and maximum node degree 3. Further, the problem of counting the number of predecessors of such DAG-SyDSs is #P-complete.*

Proof. The Predecessor Existence problem and its counting version are clearly in NP and #P, respectively. To prove NP-hardness, we use a reduction from 3SAT. The underlying graph of the constructed SyDS S is the same as the underlying graph constructed in the proof of Proposition 9.1. The local function for each level-zero node is the constant function 1. The local function for each level-one node is the function that equals 1 iff the value of at least one of the incoming edges makes the clause true. The configuration C for the constructed predecessor existence problem instance is the configuration where all nodes have value 1.

Suppose the given 3SAT problem instance is satisfiable. Then any configuration of S where the level-zero nodes have values corresponding to a satisfying assignment, and the level-one nodes have arbitrary values, is a predecessor of configuration C.

Suppose that C has a predecessor, say configuration C_0. Since all the level-one nodes in C have value 1, the values of the level-zero nodes in C_0 constitute a satisfying assignment.

Thus, C has a predecessor iff the given 3SAT problem instance is satisfiable.

Suppose the the given 3SAT problem instance contains m clauses. Then, for any satisfying assignment for the the given 3SAT problem instance, C contains 2^m predecessors. Thus, the above reduction also serves as a Turing reduction to establish the #P-completeness of the counting version. □

10 SUMMARY AND FUTURE RESEARCH DIRECTIONS

We considered a number of decision problems for DAG-SyDSs and established complexity and efficient solvability results. In particular, while previous work [13] showed that the Convergence problem is efficiently solvable for DAG-SyDSs, we showed that the Reachability problem remains PSPACE-complete even for symmetric DAG-SyDSs. We also showed that the Convergence problem is PSPACE-complete for SyDSs whose underlying graphs are Quasi-DAGs (i.e., directed graphs that

become DAGs by the deletion of a single edge). We also identified some special classes of DAG-SyDSs for which the Reachability problem can be solved efficiently. In the process of proving the above results, we established bounds on the lengths of transients and directed cycles in the phase spaces of DAG-SyDSs.

We close by presenting some directions for future research. It will be of interest to establish bounds on the lengths of transients and phase space cycles for other restricted classes of DAG-SyDSs (e.g., DAG-SyDSs in which each node has bounded indegree, DAG-SyDSs where nodes have positive or negative weights and the local functions are weighted threshold functions). Investigating the complexity of Reachability and other problems for these restricted classes of DAG-SyDSs is also an interesting direction. We limited our focus to deterministic local functions. SyDSs where the local functions are stochastic have also been considered in the literature (e.g., [5]). The hardness results for deterministic DAG-SyDSs readily extend to stochastic SyDSs; it will be interesting to study whether the efficient solvability results for the deterministic case can be extended to the stochastic case.

ACKNOWLEDGMENTS

We thank the referees for a thorough review. Their suggestions helped us to significantly improve the paper.

REFERENCES

[1] A. Adiga, C. J. Kuhlman, M. V. Marathe, H. S. Mortveit, S. S. Ravi, and A. Vullikanti. 2019. Graphical dynamical systems and their applications to bio-social systems. *Springer International Journal of Advances in Engineering Sciences and Applied Mathematics* 11, 2 (2019), 153–171.

[2] 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.

[3] K. F. Arnold, W. J. Harrison, A. J. Heppenstall, and M. S. Gilthorpe. 2019. DAG-informed regression modelling, agent-based modelling and microsimulation modelling: A critical comparison of methods for causal inference. *International Journal of Epidemiology* 48, 1 (2019), 243–253.

[4] V. Auletta, D. Ferraioli, and G. Greco. 2018. Reasoning about consensus when opinions diffuse through majority dynamics. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*. ijcai.org, Online publisher, 49–55.

[5] C. 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.

[6] C. Barrett, H. B. Hunt III, M. V Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and M. Thakur. 2007. Predecessor existence problems for finite discrete dynamical systems. *Theoretical Computer Science* 386, 1 (2007), 3–37.

[7] 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. System Sci.* 72, 8 (2006), 1317–1345.

[8] D. A. Barrington. 1989. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Computer and System Sciences* 38 (1989), 150–164.

[9] M. Blonski. 1999. Anonymous games with binary actions. *Games and Economic Behavior* 28, 2 (1999), 171–180.

[10] S. Botan, U. Grandi, and L. Perrussel. 2019. Multi-issue opinion diffusion under constraints. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '19)*. IFAAMAS, Online publisher, 828–836.

[11] R. Bredereck and E. Elkind. 2017. Manipulating opinion diffusion in social networks. In *Proc. IJCAI*. ijcai.org, Online publisher, 894–900.

[12] J.-Y. Cai and M. Furst. 1991. PSPACE survives constant-width bottlenecks. *Intl. J. Foundations of Computer Science* 2, 1 (1991), 67–76.

[13] D. Chistikov, G. Lisowski, M. Paterson, and P. Turrini. 2020. Convergence of opinion diffusion is PSPACE-complete. In *Proc. AAAI*. AAAI Press, Online publisher, 7103–7110.

[14] O. M. Cliff, M. Prokopenko, and R. Fitch. 2020. Inferring Coupling of Distributed Dynamical Systems via Transfer Entropy. ArXiv Report: 1611.00549v1.

[15] R. Colley and U. Grandi. 2022. Spread of opinions via Boolean networks. In *Proceedings of the 19th European Conference on Multi-Agent Systems (EUMAS '22)*. Dorothea Baumeister and Jörg Rothe (Eds.). Springer, Berlin, Germany, 96–115.
[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2009. *Introduction to Algorithms* (Second ed.). MIT Press and McGraw-Hill, Cambridge, MA.
[17] Y. Crama and P. Hammer. 2011. *Boolean Functions: Theory, Algorithms, and Applications*. Cambridge University Press, New York, NY.
[18] E. Creager, D. Madras, T. Pitassi, and R. Zemel. 2020. Causal modeling for fairness in dynamical systems. In *Proceedings of the International Conference on Machine Learning (PMLR)*, San Francisco, CA, 2185–2195.
[19] C. Daskalakis and C. H. Papadimitriou. 2007. Computing equilibria in anonymous games. In *Proceedings of the 48th Annual IEEE FOCS*. IEEE, Los Vequeros, CA, 83–93.
[20] C. Daskalakis and C. H. Papadimitriou. 2015. Approximate Nash equilibria in anonymous games. *Journal of Economic Theory* 156 (2015), 207–245.
[21] D. Easley and J. Kleinberg. 2010. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY.
[22] P. Floréen and P. Orponen. 1989. On the computational complexity of analyzing Hopfield nets. *Complex Systems* 3, 6 (1989), 577–587.
[23] M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman & Co., San Francisco.
[24] M. Granovetter. 1978. Threshold models of collective behavior. *Amer. J. Sociology* 83, 6 (1978), 1420–1443.
[25] M. O. Jackson. 2010. *Social and Economic Networks*. Princeton University Press, Princeton, NJ.
[26] 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.
[27] S. Kosub and C. M. Homan. 2007. Dichotomy results for fixed point counting in Boolean dynamical systems. In *Proceedings of the 10th Italian Conference on Theoretical Computer Science*. Springer-Verlag, Berlin, 163–174.
[28] 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 *Proceedings of the Workshop on Multi-Agent Interaction Networks (MAIN 2013), held in conjunction with the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, (Minneapolis, MN, May 2013). IFAAMAS, Online publisher, 26–33.
[29] L. Layne. 2011. *Biologically Relevant Classes of Boolean Functions*. Ph. D. Dissertation. Department of Mathematics, Clemson University.
[30] D. Materassi and M. V. Salapaka. 2013. Reconstruction of directed acyclic networks of dynamical systems. In *Proceedings of the 2013 American Control Conference*. IEEE Press, Los Vaqueros, CA, 4687–4692.
[31] H. Mortveit and C. Reidys. 2007. *An Introduction to Sequential Dynamical Systems*. Springer Science & Business Media, New York, NY.
[32] M. Ogihara and K. Uchizawa. 2017. Computational complexity studies of synchronous Boolean finite dynamical systems on directed graphs. *Inf. Comput.* 256 (2017), 226–236.
[33] M. Ogihara and K. Uchizawa. 2020. Synchronous Boolean finite dynamical systems on directed graphs over XOR functions. In *Proceedings of the 45th MFCS*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Online publisher, 76:1–76:13.
[34] P. Orponen. 1993. On the computational power of discrete hopfield nets. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP '93)* (Lecture Notes in Computer Science, Vol. 700). Springer, Heidelberg, Germany, 215–226.
[35] P. Orponen. 1994. Neural networks and complexity theory. *Nord. J. Comput.* 1, 1 (1994), 94–110.
[36] R. L. Rivest. 1987. Learning decision lists. *Machine Learning* 2, 3 (1987), 229–246.
[37] D. J. Rosenkrantz, M. V. Marathe, S. S. Ravi, and R. E. Stearns. 2018. Testing phase space properties of synchronous dynamical systems with nested canalyzing local functions. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)* (Stockholm, Sweden, July 10–15, 2018). IFAAMAS, Online publisher, 1585–1594.
[38] D. J. Rosenkrantz, M. V. Marathe, S. S. Ravi, and R. E. Stearns. 2021. Synchronous dynamical systems on directed acyclic graphs (DAGs): Complexity and algorithms. In *Proceedings of the 35th AAAI Conference (AAAI '21)*. AAAI Press, Online publisher, 11334–11342.
[39] R. E. Stearns, D. J. Rosenkrantz, S. S. Ravi, and Madhav V. Marathe. 2018. A characterization of nested canalyzing functions with maximum average sensitivity. *Discrete Applied Mathematics* 251 (2018), 5–14.

Received 21 February 2023; revised 20 March 2024; accepted 21 March 2024