Analysis Problems for Graphical Dynamical Systems: A Unified Approach Through Graph Predicates

Daniel J. Rosenkrantz
Computer Science Dept.
University at Albany – SUNY
Albany, NY 12222, USA
drosenkrantz@gmail.com

Madhav V. Marathe
VA Bioinformatics Institute
Virginia Tech
Blacksburg, VA 24061, USA
mmarathe@vbi.vt.edu

Harry B. Hunt III
Computer Science Dept.
University at Albany – SUNY
Albany, NY 12222, USA
hunt@cs.albany.edu

S. S. Ravi
Computer Science Dept.
University at Albany – SUNY
Albany, NY 12222, USA
sravi@albany.edu

Richard E. Stearns
Computer Science Dept.
University at Albany – SUNY
Albany, NY 12222, USA
rstearns@albany.edu

ABSTRACT

We present a unified approach for studying the complexity of analysis problems for Synchronous Dynamical Systems (SyDSs), a class of graphical models for networked multiagent systems. Our approach uses predicates based on graph embeddings to capture many phase space properties of SyDSs studied in the literature and additional properties which have not been considered previously. Using this formalism, we develop general results to show that many analysis problems for SyDSs are computationally intractable. However, when the underlying graph of the given SyDS is treewidth bounded and the local functions are r-symmetric (for any fixed r), we show that even counting versions of analysis problems can be solved efficiently.

Categories and Subject Descriptors

F.1 [**Computation by abstract devices**]:

General Terms

Theory; Algorithms

Keywords

Networked multiagent systems; Discrete Dynamical Systems; Phase Space; Complexity; Treewidth; Algorithms

1. INTRODUCTION

1.1 Model and Motivation

We study analysis problems for networked multiagent systems represented as graphical discrete dynamical systems. Our work follows similar efforts in machine learning and games where graphical models were introduced to capture natural structural restrictions that often lead to tractable problems [23, 25]. Discrete dynamical systems were initially proposed as an abstract model for computer simulations [4]; however, they also serve as powerful abstract models for networked multiagent systems [30,40]. Graphical models of multiagent systems have been studied by many researchers (e.g. [21, 29, 36–38, 40]); see Section 1.3 for additional discussion. Here, we focus on one such model, namely *synchronous* discrete dynamical systems (SyDSs). We provide an informal description of a SyDS here; a formal description is given later. A SyDS consists of an undirected graph whose vertices represent entities (agents) and edges represent local interactions among entities. Each vertex has a state value chosen from a finite domain (e.g. {0,1}). In addition, each vertex v also has a local transition function whose inputs are the current state of v and those of its neighbors; the output of this function is the next state of v. The vector consisting of the state values of all the nodes at each time instant is referred to as the **configuration** of the system at that instant. In each time step, all nodes of a SyDS compute and update their states *synchronously*. Starting from a (given) initial configuration, the time evolution of a SyDS consists of a sequence of successive configurations. Models similar to SyDSs have been used in applications such as the propagation of diseases and social phenomena (e.g. [15, 28]).

A central theme in complex systems is to categorize systems based on the notion of *predictability* [8, 20, 39]. In a finite, discrete setting, a certain behavioral pattern of a given complex system is considered predictable if the existence of such a pattern can be detected efficiently (i.e., in time which is a polynomial in the size of the system). Although some progress has been made on this topic (see [19,26,34–36] and references therein), the basic issue remains largely open. Furthermore, most of the results deal with specific behavioral patterns. In this paper, we study the computational complexity of a broad class of behavioral patterns for SyDSs. Such patterns are subgraphs of the **phase space**[1] of a SyDS. Our results are summarized below.

1.2 Our contributions

We present a general framework for studying analysis problems for networked multiagent systems modeled by discrete dynamical systems. Our framework relies on a general formalism (graph predicates) that can capture large classes of phase space properties. This formalism allows us to study analysis problems as appropriate subgraph embedding problems on the phase space (which is a directed

[1]Informally, the phase space of a SyDS specifies a representation of all valid sequences of transitions among the configurations of the SyDS. A formal definition is given in Section 2.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

graph). The formalism includes many phase properties which have been considered before (e.g. existence of fixed points and Garden-of-Eden (GE) configurations[2]) and many additional complex properties (e.g. for a fixed k, does the phase space have two node-disjoint directed paths, each with at least k nodes?) which have not been addressed in the literature. Our results unify and generalize many known results from the literature on testing phase space properties of SyDSs. Specifically, our contributions are as follows.

1. We develop (Section 2) a formalism (namely, strongly and weakly dichotomizing graph predicates) to capture a large class of phase space properties of SyDSs. This general formalism allows us to capture many properties studied in the literature (and many additional properties which have not been considered) as questions concerning graph embeddings.

2. For every weakly dichotomizing predicate P, we prove that it is NP-hard to determine whether the phase space of a given SyDS satisfies P when the local transition functions of the SyDS are specified as Boolean expressions. Since the existence of a Garden–of–Eden (GE) configuration can be expressed as a weakly dichotomizing predicate, this result generalizes the NP-hardness result for GE existence shown in [3]. The general result also implies the NP-hardness of testing properties such as whether for any fixed $k \geq 2$, the phase space of a given SyDS contains a directed path consisting of at least k distinct nodes. (A number of other properties modeled by weakly dichotomizing predicates are given in a complete version of this paper [32].)

3. For every strongly dichotomizing predicate P, we prove that it is NP-hard to determine whether the phase space of a given SyDS satisfies P when the local transition functions of the SyDS are specified as r-symmetric functions (defined in Section 2) for some fixed r. Since the existence of fixed points can be expressed as a strongly dichotomizing predicate, this result generalizes the NP-hardness result for the existence of fixed points shown in [3]. The general result also implies the NP-hardness of testing a property such as whether for any fixed $k \geq 2$, the phase space of a given SyDS contains a transient (defined in Section 2) of length at least k. (Many additional properties modeled by strongly dichotomizing predicates are given in [32].)

4. In contrast to the general hardness results, we show that when the treewidth of the underlying graph of a SyDS is bounded and all the local functions are r-symmetric for a fixed integer r, determining whether any fixed subgraph H is embedded in the phase space of a given SyDS can solved in *polynomial time* (Section 5). This result can also be extended to the counting problem associated with the graph H. For example, the problem of determining whether a given SyDS has a fixed point can be expressed as the embedding problem for the fixed subgraph corresponding to a self loop. Thus, this result generalizes the efficient algorithm in [3] for counting the number of fixed points. Further, this result also points out that many other analysis problems (such as the problem of counting the number of transients with exactly k edges) can be solved efficiently for treewidth bounded SyDSs with r-symmetric local transition functions. (In obtaining efficient algorithms for treewidth-bounded systems, the restriction to r-symmetric functions is necessary; if local functions that are not r-symmetric for some fixed r are permitted, decision versions of embedding problems are in general NP-hard even for treewidth-bounded systems [2].)

[2]Formal definitions of fixed points and GE configurations appear in Section 2.

1.3 Graphical Dynamical Systems as Models of Multi-agent Systems

SyDSs provide a framework for agent-based models (ABM) and to capture interactions among agents in a network. Wellman [38] discusses the relationships between ABM and MAS. SyDSs have the same motivation as other graphical models studied in the literature (graphical games and graphical inference): to study how network structure affects overall behavior. Our results show that when the network is treewidth-bounded and local functions are symmetric, large classes of analysis problems are efficiently solvable. These results provide design criteria for building MAS. Our lower bound results show that further generalizations of network structure and/or local functions lead to computationally intractable analysis problems.

Graphical games are very closely related to SyDSs; this is especially true when the primary focus is on action profiles (states) and not so much on eventual payoff. The notion of pure Nash equilibria in such games [16, 23] corresponds to finding a fixed point in the phase space of the corresponding SyDS. Progressive and non-progressive threshold models discussed in [24] as models for coordination games can be directly represented as SyDSs.

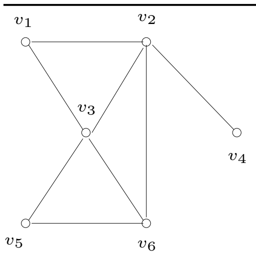
SyDSs can also capture numerous models of social interaction in networks (e.g. Granovetter's threshold model [18] and the complex contagion model of Centola and Macy [11]). Many Ph.D.theses have explored the use of SyDS and similar models to capture social interactions in networked systems (e.g. [10, 22, 27]). Additional discussion on this topic appears in [33].

1.4 Additional Discussion on our Results

Recently, dichotomy style results for constraint satisfaction problems have received a lot of attention in the literature (e.g. [7, 12, 14, 26]). However, with the exception of [26], few rigorous dichotomy results are known in the context of models that represent networked multiagent systems. Reference [26] presents a dichotomy result for the counting problem associated with one specific phase space property, namely fixed points. Our results can be viewed as a step towards developing dichotomy style results for general classes of phase space properties pertaining to such systems. Also, our results share a certain similarity with the dichotomy results for graph homomorphism problems studied in the literature [9]. A significant difference is that we are reasoning about certain kinds of subgraphs in an exponentially sized graph (phase space) specified succinctly by the underlying dynamical system.

Our hardness proofs as well as our efficient algorithm for treewidth bounded graphs are somewhat intricate because of two reasons. First, they must handle the full generality provided by the predicates or subgraphs that capture a large class of phase space properties. Second, we must construct (while proving NP-hardness results) or work with (while developing an efficient algorithm) the static description of a SyDS, whereas the predicate (or subgraph) provides a property to be satisfied by the corresponding phase space, whose size is exponential in the size of the SyDS.

Graphical models for inference problems have been studied extensively (e.g. [25]). The results presented here are an attempt to develop a similar theory of graphical models for dynamical systems; they suggest that obtaining tractability results requires restrictions on the graphical structure as well as on the structure of local functions.



Initial Configuration:	(1, 1, 0, 0, 0, 0)
Configuration at time 1:	(1, 1, 1, 0, 0, 0)
Configuration at time 2:	(1, 1, 1, 0, 0, 1)
Configuration at time 3:	(1, 1, 1, 0, 1, 1)

Note: Each configuration has the form $(s_1, s_2, s_3, s_4, s_5, s_6)$, where s_i is the state of node v_i, $1 \leq i \leq 6$. The configuration at time 3 is a fixed point.

Figure 1: An Example of a SyDS

1.5 Other Related work

Many papers have addressed the computational aspects of testing phase space properties of discrete dynamical systems and multiagent systems. For example, Barrett et al. [3] studied existence questions for fixed points and GE configurations under the sequential update model, where a permutation of the vertices is also given, and state updates are carried out in the order specified by the permutation. Bounds on the lengths of transients and cycles in restricted versions of dynamical systems under the sequential update model are established in [30]. Tosic [36] presented results for fixed point counting problems for systems with restricted forms of local transition functions. Kosub and Homan [26] presented dichotomy results that delineate computationally intractable and efficiently solvable versions of counting fixed points, based on the class of allowable local transition functions. Barrett et al. [2] considered the question of determining whether a given configuration y has a **predecessor** (i.e., a configuration x such that the phase space has the directed edge (x, y)) in deterministic SyDSs. They present hardness results for various restricted graph structures (e.g. grid graphs) and for various restricted families of local transition functions (e.g. k-threshold functions for any $k \geq 2$). They also present polynomial time algorithms when the underlying graph is treewidth-bounded and the local transition functions are r-symmetric for some fixed integer r. Problems similar to predecessor existence have also been considered in the context of cellular automata [13, 19]. The above references considered each phase space property *separately* to establish complexity results.

Wooldridge [40] presents a good discussion of complexity results for multiagent systems. Montali et al. [29] discuss verification of temporal properties in multiagent systems where interactions among the agents vary over time. Tsang and Larson [37] consider opinion dynamics in multiagent systems. In their model, the state value of each node is a real value in the closed interval $[0, 1]$; however, the interactions among agents are expressed in the form of local transition functions which depend on the state values of neighbors. Rabinovich et al. [31] study the complexity of analyzing the behavior of multiagent systems using a different representation (namely, multi-prover interactive protocols) for the interactions among agents.

2. DEFINITIONS AND PROBLEM FORMULATION

2.1 Formal Definition of the SyDS Model

Let $\mathbb{B}$ denote the Boolean domain {0,1}. A **Synchronous Dynamical System** (SyDS) $\mathcal{S}$ over $\mathbb{B}$ is a pair $\mathcal{S} = (G, \mathcal{F})$, where

(a) $G(V, E)$, an undirected graph with $|V| = n$, represents the underlying graph of the SyDS, and

(b) $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ 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 function f_i specifies the local interaction between node v_i and its neighbors in G. The inputs to function f_i are the state of v_i and those of the 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 computed efficiently. In a SyDS, all nodes compute and update their next state *synchronously*. Other update disciplines (e.g. sequential updates) for discrete dynamical systems have also been considered in the literature (e.g. [1, 30]). At any time t, the **configuration** $\mathcal{C}$ of a SyDS is the n-vector $(s_1^t, s_2^t, \ldots, s_n^t)$, where $s_i^t \in \mathbb{B}$ is the state of node v_i at time t ($1 \leq i \leq n$).

Example: Consider the graph shown in Figure 1. Suppose the local interaction function at each node is the **2-threshold function**, that is, the value of the function is 1 if and only if at least two of the inputs are 1. Assume that initially, v_1 and v_2 are in state 1 and all other nodes are in state 0. During the first time step, the state of node v_3 changes to 1 since two of its neighbors (namely v_1 and v_2) are in state 1; the states of other nodes remain the same. The configurations at subsequent time steps are shown in the figure. The configuration $(1, 1, 1, 0, 1, 1)$ reached at time step 3 is a fixed point for this system.

If a SyDS has a one step transition from configuration $\mathcal{C}$ to configuration $\mathcal{C}'$, we say that $\mathcal{C}'$ is the **successor** of $\mathcal{C}'$ and that $\mathcal{C}$ is a **predecessor** of $\mathcal{C}'$. Since our SyDS model is deterministic, each configuration has exactly one successor. However, a configuration may have zero or more predecessors. A configuration $\mathcal{C}$ is called a **fixed point** if the successor of $\mathcal{C}$ is $\mathcal{C}$ itself. A configuration $\mathcal{C}$ is a **Garden–of–Eden** if it has no predecessors.

The **phase space** $\mathcal{P}_\mathcal{S}$ of a SyDS $\mathcal{S}$ is a directed graph defined as follows. There is a node in $\mathcal{P}_\mathcal{S}$ for each configuration of $\mathcal{S}$. There is a directed edge from a node representing configuration $\mathcal{C}$ to that representing configuration $\mathcal{C}'$ if there is a one step transition of the SyDS from $\mathcal{C}$ to $\mathcal{C}'$. For a SyDS with n vertices, the number of vertices in the phase space is 2^n; thus, the size of phase space is *exponential* in the size of the description of the SyDS. Each vertex in the phase space has outdegree equal to 1 (since our SyDS model is deterministic). Also, in the phase space, each fixed point of a SyDS is a self-loop and each GE configuration is a vertex of indegree zero.

2.2 Capturing Classes of Phase Space Properties Through Graph Predicates

We consider problems involving directed graphs that can have self–loops, so that an edge is an ordered pair of nodes, not necessarily distinct. Since our goal is to model phase space properties of deterministic SyDSs, the corresponding directed graph has the special property that the outdegree of each vertex is one. The following definition relies on this property and expresses some phase space properties in terms of subgraphs.

Definition 2.1. *A **nonbranching** graph is a directed graph where each node has at most one outgoing edge. We refer to a nonbranching graph as an **NDAG**. A **full** nonbranching graph is a nonbranching graph where each node has exactly one outgoing edge.*

*A **Garden–of–Eden node** in a directed graph is a node with no entering edges. A **transient** in a directed graph is a path, beginning at a Garden–of–Eden node, none of whose edges is part of a cycle. The **length** of a transient is the number of edges in the path. The **length** of a cycle is the number of edges in the cycle. A cycle is a **pure cycle** if every node in the cycle has only one incoming edge (so that there are no transients entering the cycle). A **fringed cycle** is a graph consisting of a cycle and at least one length 1 transient entering the cycle, and no transients of length greater than 1 entering the cycle. The **predecessors** of a given node in a directed graph are the nodes that are the sources of incoming edges to the given node, and the **successor**s of a given node are the nodes that are the destinations of exiting edges of the given node. A **fixed point** of a nonbranching graph is a node that is its own successor.*

Suppose all the cycles in the phase space of a given SyDS are of the same length, say p. The number of nodes in the phase space for any SyDS is a power of 2. Consequently, if all the cycles are pure cycles, p must be a power of 2. If p is not a power of two, it is possible for all the cycles to be of length p, provided the cycles can be fringed cycles, rather than pure cycles.

DEFINITION 2.2. *We say that a given predicate P on graphs is **strongly dichotomizing** if*

1. *there is a full nonbranching graph G_0 such that P is true for every full nonbranching graph that contains a set of connected components that are isomorphic to G_0 **and***
2. *there exists a $p \geq 1$ such P is false for every full nonbranching graph for which every connected component is a fringed cycle of length p.*

Note that in the above definition, G_0 can consist of more than one connected component.

Examples: We now provide two examples of *strongly dichotomizing* graph predicates on NDAGs. Let $\mathbb{P}_1$ denote the predicate "for any fixed $k \geq 3$, the graph contains a path consisting of at least k distinct nodes" and let $\mathbb{P}_2$ denote the predicate "for any fixed $k \geq 2$, the graph contains a transient of length at least k". Both of these can be seen to be strongly dichotomizing by letting G_0 be a graph consisting of a length k path leading to a fixed point node, and choosing $p = 1$. Many additional examples appear in [32].

DEFINITION 2.3. *We say that a given predicate P on graphs is **weakly dichotomizing** if*

1. *there is a full nonbranching graph G_0 such that P is true for every full nonbranching graph that contains a set of connected components that are isomorphic to G_0 **and***
   - *(a) **either** there exists a $p \geq 1$ such that P is false for every full nonbranching graph for which every connected component is a fringed cycle of length p*
   - *(b) **or** there exists a $p \geq 1$ such that p is a power of 2 and P is false for every full nonbranching graph for which every connected component is a pure cycle of length p,*

As before, G_0 can consist of more than one connected component.

Examples: We now provide two examples of weakly dichotomizing predicates. Let $\mathbb{P}_3$ denote the predicate "for any fixed $k \geq 1$, the graph contains at least k Garden–of–Eden nodes". $\mathbb{P}_3$ can be seen to be weakly dichotomizing by letting G_0 be the graph consisting of k Garden–of–Eden nodes, each having an outgoing edge to a common fixed point node and choosing $p = 1$. Let $\mathbb{P}_4$ be the predicate that is true iff the graph contains an edge that is not a length 1 cycle. This predicate is weakly dichotomizing, since G_0 can be a pure length 2 cycle, and the predicate is false if every connected component is a pure length 1 cycle. Many additional examples appear in [32].

Note also that every predicate that is strongly dichotomizing is also weakly dichotomizing. However, the converse is not true. As an example, the predicate $\mathbb{P}_4$ defined above is weakly dichotomizing; however, it is not strongly dichotomizing.

DEFINITION 2.4. *Graph G_1 is **strongly embedded** in graph G_2 if G_1 is isomorphic to the subgraph of G_2 induced by some subset of nodes of G_2. Graph G_1 is **weakly embedded** in graph G_2 if G_1 is isomorphic to some subgraph of G_2 obtained by deleting some nodes and edges from G_2.*

*For any fixed graph H, the $\mathbf{SE}_H$ **predicate** (the $\mathbf{WE}_H$ **predicate**) is the predicate that is true for a given graph G iff H is strongly (weakly) embedded in G.*

Proofs of the following observations are straightforward and they appear in [32].

OBSERVATION 2.5. *For any nonbranching graph H that contains at least one edge, the SE_H predicate and the WE_H predicate are each weakly dichotomizing.* ■

OBSERVATION 2.6. *For any nonbranching graph H that contains at least one cycle, or at least one path of length at least 2, the SE_H predicate and the WE_H predicate are each strongly dichotomizing.* ■

2.3 Other Definitions

The following definitions of *tree decomposition* and *treewidth* are from [5].

DEFINITION 2.7. *Given an undirected graph $G(V, E)$, a **tree-decomposition** of G is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a family of subsets of V and $T = (I, F)$ is an undirected tree with the following properties:*

- *(a) $\bigcup_{i \in I} X_i = V$.*
- *(b) For every edge $e = \{v, w\} \in E$, there is a subset X_i, $i \in I$, with $v \in X_i$ and $w \in X_i$.*
- *(c) For all $i, j, k \in I$, if j lies on the path from i to k in T, then $X_i \bigcap X_k \subseteq X_j$.*

*The **treewidth** of a tree-decomposition $(\{X_i \mid i \in I\}, T)$ is $\max_{i \in I}\{|X_i| - 1\}$. The treewidth of a graph is the minimum over the treewidths of all its tree decompositions. A class of graphs is **treewidth bounded** if there is a constant k such that the treewidth of every graph in the class is at most k.*

A number of problems that are NP-hard on general graphs can be solved efficiently when restricted to the class of treewidth-bounded graphs (e.g. [5, 6]).

The following definitions (from [2]) allow us to consider restrictions on the local transition functions of SyDSs.

DEFINITION 2.8. *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.*

*A Boolean function f is r-**symmetric** if the inputs to f can be partitioned into at most r classes such that the value of f depends only on how many of the inputs in each of the r classes are 1.*

A SyDS is r-symmetric if each of its local transition functions is r'-symmetric for some $r' \leq r$.

It can be seen that a symmetric Boolean function is 1-symmetric and an r-symmetric Boolean function with t inputs can be specified using a table of size $O(t^r)$.

3. FORMULA–SPECIFIED SYDSS

DEFINITION 3.1. *A* **formula–specified SyDS** *is a SyDS where the node transition functions are specified by Boolean formulas, where the formulas are permitted to use the Boolean operators* and*,* or*, and* not*.*

THEOREM 3.2. *For every weakly dichotomizing predicate P, the problem of given a formula–specified SyDS, determining whether its phase space graph satisfies P is* NP*–hard.*

Proof sketch: The proof is by a reduction from 3SAT which is known to be NP-hard [17]. Let f be a given 3SAT formula.

Let G_0 be a full nonbranching graph such that P is true for every complete nonbranching graph that contains a set of connected component that are isomorphic to G_0. Let $p \geq 1$ be such that either P is false for every full nonbranching graph for which every connected component is a fringed cycle of length p, or p is a power of 2 and P is false for every full nonbranching graph for which every connected component is a pure cycle of length p. Let G_0' be G_0 augmented with a minimum number of fixed point nodes so that the number of nodes in G_0' is even, is a power of 2, and is greater than p. Note that P is true for every full nonbranching graph that contains a set of connected component that are isomorphic to G_0'.

Let the variables in f be $x_1, x_2, \ldots, x_n$. Let the number of nodes in G_0' be 2^m. If P is false for every full nonbranching graph for which every connected component is a fringed cycle of length p, then let q be the smallest integer such that $2^q > p$. Otherwise, in which case p is a power of 2 and P is false for every full nonbranching graph for which every connected component is a pure cycle of length p, let q be $log_2\, p$, so that $2^q = p$.

Note that $m \geq q \geq 0$. The constructed SyDS $\mathcal{S}$ has $n+m$ nodes. We refer to these nodes as $X = x_1, x_2, \ldots, x_n$, and $Y = y_1, y_2, \ldots, y_m$. Let Y' denote the subset of Y consisting of nodes $y_1, y_2, \ldots, y_q$. Note that if q is zero, then Y' is empty.

The undirected graph for $\mathcal{S}$ has an edge between each node in its graph and each node in Y.

Node x_i of $\mathcal{S}$ corresponds to variable x_i of f. We encode each of the nodes of G_0' as an m bit number, with y_1 as the lowest order bit. Let the nodes of G_0' be denoted as $v_0, v_1, \ldots, v_{2^m-1}$, where the subscript of each node corresponds to the encoding of that node. We envision the values of the nodes Y as representing a node from G_0'.

For a given j, where $0 \leq j \leq 2^m - 1$, let g_j denote the Boolean formula consisting of the conjunction of m literals, one for each node in Y. In this conjunction, a given variable is unnegated or negated, depending on whether its value is 1 or 0 in the Boolean representation of number j. Thus, formula g_j evaluates to 1 iff the values of the nodes in Y encode the value j.

Suppose that for a given y_i in Y, graph G_0' contains r edges entering a node whose encoding has value 1 for bit y_i. Let h_i denote the Boolean formula consisting of the disjunction of r disjuncts, as follows. For each edge (v_j, v_k) of G_0' such that the encoding of k has value 1 for bit y_i, h_i contains the conjunct g_j.

For a given assignment of values to the nodes in Y', let $j(Y')$ denote the number encoded by this assignment. For each y_i in Y', let $\hat{h}_i$ denote the Boolean formula of input variables $y_1, y_2, \ldots, y_q$, such that $\hat{h}_i$ evaluates to 1 iff $j(Y') < p$ and number $(j(Y')+1)$ mod p has bit i equal to 1. Note that if $j(Y') \geq p$, then $\hat{h}_i$ evaluates to 0.

For each y_i in $Y - Y'$, let $\hat{h}_i$ denote the Boolean formula y_i.

For each node x_i in X, the transition function formula is simply x_i. These transition functions ensure that in every transition of $\mathcal{S}$, the values of the nodes in X never change.

Let f' be a formula for the complement of f. (Note that since f is expressed using *and*, *or*, and *not*, formula f' can be efficiently constructed from f.)

Now consider a node y_i in Y. The transition function formula for y_i is $(f \wedge h_i) \vee (f' \wedge \hat{h}_i)$, with parentheses added as needed.

Suppose that in a given configuration c of $\mathcal{S}$, the values of nodes X in c make formula f true. Suppose that the values of nodes Y encode node v_j of G_0'. Let v_k denote the unique node in G_0', such that G_0' contains the edge (v_j, v_k). Then the transition functions for nodes Y ensure that the value of the nodes in Y in the successor configuration to c encode v_k.

Suppose the values of nodes X in c make formula f false and the number $j(Y')$ encoded by nodes Y' in c is such that $j < p$. Then the transition functions for nodes Y ensure that the value of the nodes in Y' in the successor configuration to c encode the first q bits of the number $(j(Y')+1)$ mod p, and that the values of nodes $Y - Y'$ will remain unchanged.

Now suppose the values of nodes X in c make formula f false and the number $j(Y')$ encoded by nodes Y' in c is such that $j \geq p$. Then the transition functions for nodes Y ensure that in the successor configuration to c, nodes Y' will encode the number 0, and that the values of nodes $Y - Y'$ remain unchanged.

This completes the construction. It can be shown that the predicate P is true for $\mathcal{S}$ if and only if f is satisfiable; see [32] for details. ■

4. TABLE–SPECIFIED SYDSS

DEFINITION 4.1. *For a fixed $r \geq 1$, an r*–**table–specified SyDS** *is a SyDS whose node transition functions are r–symmetric, and are specified via tables.*

First we observe that there is a *weakly dichotomizing* graph predicate on phase space that is polynomially solvable for r–table–specified SyDSs. Thus, in general, Theorem 3.2 does not hold for table-specified SyDSs.

THEOREM 4.2. *For every fixed $r \geq 1$, the problem of determining whether the phase space for a given r–table–specified SyDS contains an edge that is not a length 1 cycle can be solved in polynomial time.*

Proof: The problem can be solved by inspecting the table for each node to determine for each of the two possible values for the node, if there are values for the neighboring nodes which would cause the given node to change value. If such a node is found, let c be any configuration where the given node has the given value, and the neighbors have values that would cause a change in the value of the given node. Then, in phase space, the outgoing edge for node c goes to a different node. If such a node is not found, then every edge in phase space is a length 1 cycle. ■

In contrast, for strongly dichotomizing predicates, the problem remains NP-hard even for table-specified SyDSs as shown by the following result.

THEOREM 4.3. *For every strongly dichotomizing predicate P, the problem of given a 5–table–specified SyDS, determining whether its phase space graph satisfies P is* NP*–hard.*

Proof idea: Our proof of the above theorem is also by a reduction from 3SAT. The details, which are omitted here for space reasons, are available in [32]. ■

5. EFFICIENT ALGORITHMS FOR TREEWIDTH BOUNDED SYDSS

5.1 Additional Terminology for Treewidth-Bounded Graphs

When a graph G has bounded treewidth, it is well known that a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G can be constructed in time that is a polynomial in the size of G. Moreover, this can be done so that all of the following conditions hold (where some node of T is selected to be a root node) [2, 5]: (a) each node of T has at most two children, (b) the number of nodes of T with fewer than two children is at most n, the number of nodes in G and (c) the number of nodes of T with two children is at most n. Our algorithm relies on this special form of tree decomposition.

The following terminology regarding nodes in tree decompositions is from [2]. Let T be the given tree decomposition of a graph G. For a given node i of T, the nodes of G in X_i are called **explicit nodes** of i. If a given explicit node v of i is also an explicit node of the parent of i, then v is referred to as an **inherited node** of i; and if v does not occur in the parent of i, then v is called an **originating node** of i. We refer to the set of all explicit nodes occurring in the subtree of T rooted at i that are not explicit nodes of i as **hidden nodes** of i. (Thus, the hidden nodes of i are the union of the originating and hidden nodes of the children of i.)

The main result of this section is our dynamic programming algorithm (presented in Section 5.4) for counting the number of embeddings of a fixed graph H into the phase space of a SyDS $\mathcal{S}$. The definitions needed to specify this algorithm are discussed in the next two subsections.

5.2 Homomorphisms and Embeddings

DEFINITION 5.1. *A* homomorphism *of graph H into graph K is a function h that maps each node of H into a node of K such that for every edge (u, v) of H, graph K contains the edge $(h(u), h(v))$.*

Given a graph H, a *uniqueness requirement* is an unordered pair of distinct nodes from H. A given homomorphism h from graph H to graph K *satisfies* a given uniqueness requirement $\{u, v\}$ if $h(u)$ is a different node than $h(v)$. For a nonbranching graph H, a set of uniqueness requirements R is *uniqueness–sufficient* if for every homomorphism h from H to any nonbranching graph K, satisfaction of all the requirements in R ensures that for every pair $\{u, v\}$ of distinct nodes from H, the nodes $h(u)$ and $h(v)$ are distinct.

Note that for a given nonbranching graph H, the set of all pairs of distinct nodes is uniqueness–sufficient. However, smaller sets of uniqueness requirements may also be uniqueness–sufficient. For instance, it is unnecessary to include a uniqueness requirement for a pair of nodes that occur in connected components of H that contain cycles of different lengths. Also, it is unnecessary to include a uniqueness requirement for a pair of nodes that that have different distances from a cycle.

Given a graph H, an *inducement requirement* is an ordered pair of nodes (u, v), not necessarily distinct, from H, such that node u has no exiting edge in H. A given homomorphism h from graph H to graph K *satisfies* a given inducement requirement (u, v) if K does not contain the edge $(h(u), h(v))$. For a nonbranching graph H, the *complete* set of inducement requirements for H is the set of all inducement requirements for H. The following is a direct consequence of the above definitions.

OBSERVATION 5.2. *A homomorphism of nonbranching graph H into nonbranching graph K is a weak embedding iff it satisfies all the uniqueness requirements in a set of uniqueness-sufficient requirements for H.*

A homomorphism of nonbranching graph H into nonbranching graph K is a strong embedding iff it satisfies all the uniqueness requirements in a set of uniqueness-sufficient requirements for H, and satisfies the complete set of inducement requirements for H.

5.3 Configuration and Signature Ensembles

Given a graph $G(V, E)$, a **configuration** c is an assignment of a Boolean value to each node in V. Given a set of nodes $Y \subseteq V$, a Y–**configuration** c_Y assigns a Boolean value to each node in Y. Given a configuration c, we use $c(y)$ to denote the value of node y in that configuration. Similarly, given a Y–configuration c_Y and a node y in Y, we use $c_Y(y)$ to denote the value of node y in c_Y. We extend this notation to subsets of variables as follows. Given a configuration c and a set of nodes $W \subseteq V$, $c(W)$ denotes the combination of values assigned to the nodes in W. Similarly, given a Y–configuration c_Y and a subset of nodes $W \subseteq Y$, $c_Y(W)$ denotes the combination of values assigned to the nodes in W.

Let H be a fixed graph. Given a set of nodes $Y \subseteq V$, a Y–**configuration ensemble** is a function h_Y that maps each node u of H into a Y–configuration. We let $\mathcal{H}_Y$ denote the set of all Y–configuration ensembles. Note that the cardinality of $\mathcal{H}_Y$ is 2^{ms}, where m is the number of nodes in H, and s is the number of nodes in Y. If H is fixed, and s is bounded by a constant, then the cardinality of $\mathcal{H}_Y$ is also bounded by a constant.

Consider a given Y–configuration c_Y and a given Z–configuration c_Z. We say that c_Y and c_Z are **consistent** if for every node x of $Y \cap Z$, it is the case that $c_Y(x) = c_Z(x)$. We say that a given Y–configuration ensemble h_Y and a given Z–configuration ensemble h_Z are **consistent** if for every node u of H, configurations $h_Y(u)$ and $h_Z(u)$ are consistent.

For disjoint subsets Y and Z of V, suppose that h_Y is a Y–configuration ensemble and h_Z is a Z–configuration ensemble. We use the notation $h_Y \cup h_Z$ to denote the $(Y \cup Z)$–configuration ensemble that maps each node u of H into the $(Y \cup Z)$–configuration $h_Y(u) \cup h_Z(u)$.

DEFINITION 5.3. *A **generalized neighbor** of a node v of a graph is either a neighbor of node v, or node v itself.*

Let G be the underlying graph of an r–symmetric SyDS $\mathcal{S}$. For a given node w of G, let $cl(w)$ denote the set of at most r classes of inputs to the r-symmetric transition function for w. Note that each class in $cl(w)$ is a set of generalized neighbors of node w. For a given set of nodes W of G, let $cl(W)$ denote a set containing an element for each class of each node w in W. Note that each element of $cl(W)$ is a distinct set of nodes of G.

For not necessarily disjoint sets of nodes Y and W of G, a (Y, W)–**signature** is a function $g : cl(W) \longrightarrow \mathbb{N}$, such that for each class ν in $cl(W)$, the value of $g(\nu)$ does not exceed the number of members of ν that are in Y.

Given subsets Y and W of V, a (Y, W)–**signature ensemble** is a function that maps each node u of H into a (Y, W)–*signature*. We let $\Gamma_{Y,W}$ denote the set of all possible (Y, W)–signatures, and let $\Gamma^H_{Y,W}$ denote the set of all possible (Y, W)–signature ensembles.

Let m denote the number of nodes in H. Note that $\Gamma^H_{Y,W}$ is isomorphic to $\Gamma^m_{Y,W}$. Also note that if H is a fixed graph, the cardinality of W is bounded, and the maximum number of classes for any node in W is bounded, then the cardinality of $\Gamma^H_{Y,W}$ is bounded by a polynomial function of the the number of nodes in G.

For a given a Y–configuration η, and a given set of nodes W of G, we define an associated (Y, W)–signature, which we denote as $sig_W(\eta)$, and which is defined as follows. For $\nu \in cl(W)$, $(sig_W(\eta))(\nu)$ is the number of nodes y in Y such that y is in class ν, and $\eta(y) = 1$.

For a given Y–configuration ensemble h , and a given set of nodes W of G, we define an associated (Y, W)–signature ensemble, which we denote as $sigen_W(h)$, and which is defined as follows. For node u of H, $(sigen_W(h))(u)$ is the signature $sig_W(h(u))$.

DEFINITION 5.4. *Consider sets of nodes $W \subseteq Y \subseteq V$, such that every generalized neighbor of every node w in W is in Y. Given a Y–configuration ensemble h, we say that h is W–**viable** iff for every node w in W, and every edge (u, v) of H, the evaluation of the local transition function f_w gives the value $(h(v))(w)$, using the value of $(h(u))(x)$ for every generalized neighbor x of w.*

DEFINITION 5.5. *For disjoint sets of nodes Y and Z of G, suppose σ_1 is a (Y, W)–signature ensemble and σ_2 is a (Z, W)–signature ensemble. We define $\sigma_1 \oplus \sigma_2$ to be the $(Y \cup Z, W)$–signature ensemble such that for each ν in $cl(W)$, $(\sigma_1 \oplus \sigma_2)(\nu) = \sigma_1(\nu) + \sigma_2(\nu)$.*

Suppose that σ is a (Y, W)–signature ensemble. Suppose that Z is a set of nodes of G such that $W \subseteq Z$ and no member of Y is a generalized neighbor of any member of $Z - W$. Then, the Z–**extension** of σ is the (Y, Z)–signature ensemble σ', defined as follows. Consider each ν in $cl(Z)$. If ν is in $cl(W)$, then for each node u of H, $(\sigma'(u))(\nu) = (\sigma(u))(\nu)$. If ν is in $cl(Z) - cl(W)$, then for each node u of H, $(\sigma'(u))(\nu) = 0$.

5.4 Counting Embeddings for Treewidth-Bounded SyDSs

DEFINITION 5.6. *For any fixed graph H, the SE_H (WE_H) counting problem is the problem of determining for a given SyDS $\mathcal{S}$ how many strong (weak) embeddings there are of H into the phase space of $\mathcal{S}$.*

THEOREM 5.7. *For any fixed graph H, fixed r, and fixed t, there is a polynomial time algorithm for solving the SE_H counting problem and the WE_H counting problem for an r–symmetric SyDS whose underlying graph has a treewidth of at most t.*

Proof: Let graph $G(V, E)$ be the underlying graph for the given SyDS $\mathcal{S}$. Let T be the given tree decomposition for G. For any node i in T, we use Y^i_{inh}, Y^i_{org} and Y^i_{hid} to denote respectively the set of inherited nodes, the set of originating nodes and the set of hidden nodes of i.

If the problem to be solved is a WE_H counting problem, let R be a set of uniqueness-sufficient requirements for H. If the problem to be solved is a SE_H counting problem, let R be the union of a set of uniqueness-sufficient requirements for H, and the complete set of inducement requirements for H. Let $\mathcal{R}$ denote 2^R, the power set of R. Note that since H is fixed, set $\mathcal{R}$ is also fixed.

To count the number of strong embeddings or weak embeddings of H into the phase space of $\mathcal{S}$, we use bottom-up dynamic programming on the decomposition tree. For each node i of T, we compute a table, which we denote as J^i. This table contains an entry for each triple in $\mathcal{H}_{Y^i_{inh}} \times \Gamma^H_{(Y^i_{org} \cup Y^i_{hid}), Y^i_{inh}} \times \mathcal{R}$. The value of each entry is a non–negative integer. Since H is a fixed graph and the treewidth t is a constant, the number of entries in each table J^i is a polynomial in n, the number of nodes of the underlying graph $G(V, E)$.

Consider a given element of table J^i, say $J^i[h_{inh}, \sigma, R']$, where h_{inh} is a Y^i_{inh}–configuration ensemble, σ is a $(Y^i_{org} \cup Y^i_{hid}, Y^i_{inh})$–signature ensemble, and R' is a subset of R. Table J^i will be computed so that the value of this element will equal the number of pairs (h_{org}, h_{hid}) consisting of a Y^i_{org}–configuration ensemble h_{org} and a Y^i_{hid}–configuration ensemble h_{hid}, such that $h_{inh} \cup h_{org} \cup h_{hid}$ is $(Y^i_{hid} \cup Y^i_{org})$–viable, σ is the signature ensemble $sigen_{Y^i_{inh}}(h_{org} \cup h_{hid})$ and R' is the set of requirements that are satisfied by $h_{org} \cup h_{hid}$. Note that the definition of a tree decomposition ensures that every generalized neighbor of a hidden or originating node of i is either an explicit node or a hidden node of i. Also note that Y^i_{hid} can contain up to n nodes, so the values of the table entries can be exponential in n. However, the number of bits needed to represent the table entries is proportional to n.

To facilitate the computation of table J^i, we also compute for each node i of T, a table which we denote as K^i. This table contains a non–negative integer for each four-tuple in $\mathcal{H}_{Y^i_{inh}} \times \mathcal{H}_{Y^i_{org}} \times \Gamma^H_{Y^i_{hid},(Y^i_{inh} \cup Y^i_{org})} \times \mathcal{R}$. Since H is a fixed graph and the treewidth t is a constant, the number of entries in this table is a polynomial in n, the number of nodes of the underlying graph $G(V, E)$.

Let $K^i[h_{inh}, h_{org}, \sigma_{hid}, R_{hid}]$ denote a given element of table K^i, where h_{inh} is a Y^i_{inh}–configuration ensemble, h_{org} is a Y^i_{org}–configuration ensemble, σ_{hid} is a $(Y^i_{hid}, Y^i_{inh} \cup Y^i_{org})$–signature ensemble, and R_{hid} is a subset of R. Table K^i will be computed so that this element will equal the number of Y^i_{hid}–configuration ensemble h_{hid} such that $h_{inh} \cup h_{org} \cup h_{hid}$ is Y^i_{hid}–viable, σ_{hid} is the signature ensemble $sigen_{Y^i_{inh} \cup Y^i_{org}}(h_{hid})$, and R_{hid} is the set of requirements that are satisfied by h_{hid}. Note that the definition of a tree decomposition ensures that every generalized neighbor of a hidden node of i is either an explicit node or a hidden node of i.

We now describe the bottom-up construction of the J and K tables defined above. The description is presented in four parts in the following order.

(1) Computation of table K^i for a leaf node i of the decomposition tree.

(2) Computation of table J^i for an arbitrary node i of the decomposition tree, given the table K^i.

(3) Computation of table K^i for a nonleaf node i of the decomposition tree, given the J tables for the children of node i in the decomposition tree.

(4) Determining the solution to the SE_H counting problem or the WE_H counting problem, given the J^i table for the root node of the decomposition tree.

Part 1: Consider a leaf node i of the decomposition tree. Since a leaf node has no hidden nodes, set $\Gamma_{Y^i_{hid},(Y^i_{inh} \cup Y^i_{org})}$ contains only one member: a function that maps each member of $cl(Y_{inh} \cup Y_{org})$ to zero. Thus $\Gamma^H_{Y^i_{hid},(Y^i_{inh} \cup Y^i_{org})}$ has only one member. An element of K^i, say $K^i[h_{inh}, h_{org}, \sigma, R']$, is set to 1 if R' is the empty set, and is set to 0 otherwise.

Part 2: Consider an arbitrary node i of the decomposition tree. Suppose that table K^i has already been computed, and table J^i is to be computed next.

Initially, all the elements of table J^i are set to zero. Then, table K^i is scanned, and each element of table K^i that is nonzero and that represents a Y^i_{org}–viable configuration ensemble, will result in having its value added to some element of table J^i. (A given element of table J^i might be incremented multiple times because of multiple elements of K^i.)

Suppose a given element, say $K^i[h_{inh}, h_{org}, \sigma_{hid}, R_{hid}]$, of table K^i has a nonzero value. Let σ_{org} be the $(Y^i_{inh} \cup Y^i_{org}, Y^i_{org})$–signature ensemble $sigen_{Y^i_{org}}(h_{inh} \cup h_{org})$. Let σ'_{hid} be the (Y^i_{hid}, Y^i_{org})–signature ensemble obtained by restricting σ_{hid} to $cl(Y^i_{org})$. Let σ'_{org} be $\sigma_{org} \oplus \sigma'_{hid}$. Note that σ'_{org} is a $(Y^i_{hid} \cup Y^i_{inh} \cup Y^i_{org}, Y^i_{org})$–signature ensemble. We can test for Y^i_{org}–viability by checking that for every node w in Y^i_{org}, and every edge (u, v) of H, the evaluation of the local transition function f_w gives the value $(h_{org}(v))(w)$, using the value of $(\sigma'_{org}(u))(\nu)$ for every class ν in $cl(w)$. If the given element of K^i does not represent a Y^i_{org}–viable configuration ensemble, then processing of the given element is finished. Otherwise, an element of table J^i is found as follows.

Let σ_{inh} be $sigen_{Y^i_{inh}}(h_{org})$. Note that σ_{inh} is a (Y^i_{org}, Y^i_{inh})–signature ensemble. Let σ'_{inh} be the (Y^i_{hid}, Y^i_{inh})–signature ensemble obtained by restricting σ_{hid} to $cl(Y^i_{inh})$. Let σ''_{inh} be $\sigma_{inh} \oplus \sigma'_{inh}$. Note that σ''_{inh} is a $(Y^i_{org} \cup Y^i_{hid}, Y^i_{inh})$–signature ensemble.

Let R_{org} is the set of requirements that are satisfied by h_{org}. Let R' be $R_{hid} \cup R_{org}$. Table element $J^i[h_{inh}, \sigma''_{inh}, R']$ is incremented by the value of $K^i[h_{inh}, h_{org}, \sigma_{hid}, R_{hid}]$.

Part 3, Case 1: Consider an arbitrary nonleaf node i that has only one child. Let i_1 denote the child of i in the tree decomposition. Suppose that table J^{i_1} has already been computed, and table K^i is to be computed next.

Initially, all the elements of table K^i are set to zero. Then, table J^{i_1} is scanned, and each element of table J^{i_1} that is nonzero will result in having its value added to some elements of table K^i.

Suppose a given element of table J^{i_1}, say $J^{i_1}[h^{i_1}_{inh}, \sigma^{i_1}, R^{i_1}]$, has a nonzero value. Note that σ^{i_1} is a $(Y^{i_1}_{org} \cup Y^{i_1}_{hid}, Y^{i_1}_{inh})$–signature ensemble. Since i has only one child, $Y^i_{hid} = Y^{i_1}_{org} \cup Y^{i_1}_{hid}$. Let σ^i be the $(Y^i_{inh} \cup Y^i_{org})$–extension of σ^{i_1}. Let R^i equal R^{i_1}.

For every Y^i_{inh}–configuration ensemble h^i_{inh} and Y^i_{org}–configuration ensemble h^i_{org} such that h^i_{inh} is consistent with $h^{i_1}_{inh}$, and h^i_{org} is consistent with $h^{i_1}_{inh}$, table element $K^i[h^i_{inh}, h^i_{org}, \sigma^i, R^i]$ is incremented by the value of $J^{i_1}[h^{i_1}_{inh}, \sigma^{i_1}, R^{i_1}]$.

Part 3, Case 2: Consider an arbitrary nonleaf node i that has two children.. Let i_1 and i_2 denote the children of i in the tree decomposition. Suppose that tables J^{i_1} and J^{i_2} have already been computed, and table K^i is to be computed next.

Initially, all the elements of table K^i are set to zero. Then, tables J^{i_1} and J^{i_2} are jointly scanned. Each pair consisting of an element of table J^{i_1} that is nonzero and an element of table J^{i_2} that is nonzero, such that the first components of these two elements are consistent, will result in having the product of their two values added to some of the elements of table K^i.

Suppose a given element of table J^{i_1}, say $J^{i_1}[h^{i_1}_{inh}, \sigma^{i_1}, R^{i_1}]$, has a nonzero value. Further, suppose a given element of table J^{i_2}, say $J^{i_1}[h^{i_1}_{inh}, \sigma^{i_1}, R^{i_1}]$, has a nonzero value, and configuration ensembles $h^{i_1}_{inh}$ and $h^{i_2}_{inh}$ are consistent. Note that Y^i_{hid} is the disjoint union $Y^{i_1}_{org} \cup Y^{i_1}_{hid} \cup Y^{i_2}_{org} \cup Y^{i_2}_{hid}$. Let $\sigma^{i'_1}$ be the $(Y^i_{inh} \cup Y^i_{org})$–extension of σ^{i_1}. Let $\sigma^{i'_2}$ be the $(Y^i_{inh} \cup Y^i_{org})$–extension of σ^{i_2}. Let σ^i be $(\sigma^{i'_1} \oplus \sigma^{i'_2})$. Let R^i be $(R^{i_1} \cup R^{i_2})$.

For every Y^i_{inh}–configuration ensemble h^i_{inh} and Y^i_{org}–configuration ensemble h^i_{org}, such that h^i_{inh} is consistent with both $h^{i_1}_{inh}$ and $h^{i_2}_{inh}$, and h^i_{org} is consistent with both $h^{i_1}_{inh}$ and $h^{i_2}_{inh}$, table element $K^i[h^i_{inh}, h^i_{org}, \sigma^i, R^i]$ is incremented by the product of the values of $J^{i_1}[h^{i_1}_{inh}, \sigma^{i_1}, R^{i_1}]$ and $J^{i_1}[h^{i_1}_{inh}, \sigma^{i_1}, R^{i_1}]$.

Part 4: Let r be the root node of the tree decomposition. The root node has no inherited nodes, so Y^r_{inh} is empty, and $(Y^r_{hid} \cup Y^r_{org})$ is V, the set of all nodes of G. Thus, there is only one Y^r_{inh}–configuration, an empty configuration. Also, there is only one Y^i_{inh}–configuration ensemble, in which each node u of H is mapped to the empty configuration. We denote this configuration ensemble as h_ϕ. Since Y^r_{inh} is empty, $cl(Y^r_{inh})$ is also empty. Thus, there is only one $(Y^i_{org} \cup Y^i_{hid}, Y^i_{inh})$–signature, an empty signature. Also, there is only one $(Y^i_{org} \cup Y^i_{hid}, Y^i_{inh})$–signature ensemble, in which each node u of H is mapped to the empty signature. We denote this configuration signature as σ_ϕ.

Table J^r contains an element $J^r[h_\phi, \sigma_\phi, R']$, for each subset $R' \subseteq R$. The value of such an element of table J^r is the number of V–configuration ensembles h_V such that h_V is V–viable, and R' is the set of requirements that are satisfied by h_V.

The solution to the counting problem is the value of $J^r[h_\phi, \sigma_\phi, R]$. ■

As consequences of the above theorem, we also obtain efficient algorithms for the corresponding decision problems. A statement of these results is provided below.

COROLLARY 5.8. *For any fixed graph H, fixed r, and fixed t, there is a polynomial time algorithm for determining for a r–symmetric SyDS whose underlying graph has a treewidth of at most t, whether the phase space satisfies the* SE_H *predicate or whether the phase space satisfies the* WE_H *predicate.* ■

6. SUMMARY AND FUTURE RESEARCH

We established a general result showing that for large classes of phase space properties of SyDSs, the problem of testing those properties is computationally intractable. We also showed that for SyDSs whose underlying graph are treewidth-bounded, the testing problem is efficiently solvable for large classes of properties when the local functions are r-symmetric, for some fixed integer r.

We conclude by mentioning some directions for future research. One direction is to identify other structural restrictions on SyDSs for which phase space properties expressed as graph predicates can be efficiently tested. Researchers (e.g. [1]) have studied various *configuration reachability* problems in discrete dynamical systems. A typical reachability problem is the following: given a dynamical system $\mathcal{S}$ and two configurations $\mathcal{C}$ and $\mathcal{C}'$, determine whether the system starting from $\mathcal{C}$ will reach $\mathcal{C}'$. In general, such problems are PSPACE-complete even when the underlying graph has bounded treewidth and the local transition functions are symmetric; polynomial time algorithms are known for some restricted classes of local transition functions (see e.g. [1]). Although reachability is a phase space property, our graph predicate formalism cannot capture such properties since the length of the corresponding directed path in the phase space may be exponential in the input size. Thus, developing a general framework for analyzing reachability-like properties is an interesting topic for future work. Probabilistic versions of SyDS (where the local transition functions are stochastic) are useful in studying diffusion phenomena in social networks [15]. Thus, another research direction is to develop a framework for studying phase space properties of probabilistic discrete dynamical systems.

Acknowledgments: We thank the reviewers for carefully reading the manuscript and providing valuable suggestions. This work has been partially supported by DTRA Grant HDTRA1-11-1-0016 and DTRA CNIMS Contract HDTRA1-11-D-0016-0010, NSF NetSE Grant CNS-1011769, NSF SDCI Grant OCI-1032677 and NIH MIDAS Grant 5U01GM070694-11.

REFERENCES

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

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

[3] C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and P. T. Tosic. Gardens of eden and fixed points in sequential dynamical systems. In *DM-CCG*, pages 95–110, 2001.

[4] C. L. Barrett and C. M. Reidys. Elements of a theory of simulation i: Sequential CA over random graphs. *Applied Mathematics and Computation*, 98(3):241–259, 1999.

[5] H. Bodlaender. Treewidth: Algorithmic techniques and results. *Proc. 22nd Symposium on Mathematical Foundations of Computer Science*, pages 29–36, 1997.

[6] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11(1-2):1–22, 1993.

[7] A. Bulatov, M. Dyer, L. Goldberg, M. Jerrum, and C. McQuillan. The expressibility of functions on the boolean domain, with applications to counting CSPs. arXiv:1108.5288v4, 2012.

[8] S. Buss, Christos H. Papadimitriou, and J.N. Tsitsiklis. On the predictability of coupled automata: an allegory about chaos. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 788–793 vol.2, Oct 1990.

[9] J. Cai, X. Chen, and P. Lu. Graph homomorphisms with complex values: A dichotomy theorem. *SIAM J. Computing*, 42(3):924–1029, 2013.

[10] D. Centola. *Simple Models of Collective Behavior*. PhD thesis, Cornell University, Ithaca, NY, 2006.

[11] D. Centola and M. Macy. Complex Contagions and the Weakness of Long Ties. *The American Journal of Sociology*, 113(3):702–734, 2007.

[12] H. Chen. A rendezvous of logic, complexity and algebra. *ACM Computing Surveys*, 42(1):2:1–2:32, 2009.

[13] B. Durand. A random NP-complete problem for inversion of 2d cellular automata. *Theor. Comput. Sci.*, 148(1):19–32, 1995.

[14] M. Dyer and D. Richerby. An effective dichotomy for the counting constraint satisfaction problem. Technical Report, School of Computing, University of Leeds, UK, 2013.

[15] D. Easley and J. Kleinberg. *Networks, Crowds and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, 2010.

[16] A. Galeotti, S. Goyal, M. Jackson, and F. Vega-Redondo. Network games. *Review of Economic Studies*, 77:218–244, 2010.

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

[18] M. Granovetter. Threshold Models of Collective Behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[19] F. Green. NP-complete problems in cellular automata. *Complex Systems*, 1(3):453–474, 1987.

[20] H. Gutowitz. *Cellular Automata: Theory and Experiment*. North Holland, 1989.

[21] D. Helbing and S. Balietti. Agent-based modeling. In D. Helbing, editor, *Social Self-Organization: Understanding Complex Systems*, pages 25–71. Springer-Verlag, Berlin, Germany, 2012.

[22] Z. Jin. *Coordinated Control for Networked Multiagent Systems*. PhD thesis, California Institute of Technology, Pasadena, CA, 2007.

[23] M. Kearns. Graphical games. In N. Nissan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Graph Theory*, chapter 7. Cambridge University Press, New York, NY, 2008.

[24] J. Kleinberg. Cascading Behavior in Networks: Algorithmic and Economic Issues. In N. Nissan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 24, pages 613–632. Cambridge University Press, New York, NY, 2007.

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

[26] S. Kosub and C. M. Homan. Dichotomy Results for Fixed Point Counting in Boolean Dynamical Systems. In *Proc. ICTCS*, pages 163–174, 2007.

[27] C. Kuhlman. *High Performance Computational Social Science modeling of Networked Populations*. PhD thesis, Computer Science Department, Virginia Tech, 2013.

[28] M. Macy and R. Willer. From factors to actors: Computational sociology and agent-based modeling. *Annual Reviews in Sociology*, 28:143–166, 2002.

[29] M. Montali, D. Calvenese, and G. De Giacomo. Verification of data-aware commitment-based multiagent system. In *Proc. AAMAS 2014*, pages 157–164, 2014.

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

[31] Z. Rabinovich, C. Goldman, and J. Rosenschein. The complexity of multiagent systems: The price of silence. In *Proc. AAMAS 2003*, pages 1102–1103, 2003.

[32] D. Rosenkrantz, M. V. Marathe, H. B. Hunt III, S. S. Ravi, and R. E. Stearns. Analysis problems for synchronous discrete dynamical systems: A unified approach. Technical Report 14-094, NDSSL, Virginia Bioinformatics Institute, Blacksburg, VA, 2014.

[33] M. Rovatsos. Multiagent Systems for Social Computation. In *Proc. 13th AAMAS*, pages 1165–1168, 2014.

[34] Klaus Sutner. On the computational complexity of finite cellular automata. *J. Comput. Syst. Sci.*, 50(1):87, 1995.

[35] Klaus Sutner. Computational classification of cellular automata. *Int. J. General Systems*, 41(6):595–607, 2012.

[36] P. T. Tosic. On the complexity of enumerating possible dynamics of sparsely connected boolean network automata with simple update rules. In *Automata 2010 - 16th Intl. Workshop on CA and DCS*, pages 125–144, 2010.

[37] A. Tsang and K. Larson. Opinion dynamics of skeptical agents. In *Proc. AAMAS 2014*, pages 277–284, 2014.

[38] M. P. Wellman. Putting the *Agent* in agent-based modeling. Invited talk at *AAMAS* 2014, 2014.

[39] S. Wolfram. *Theory and Applications of Cellular Automata*. World Scientific, 1987.

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