Electronic Colloquium on Computational Complexity, Report No. 51 (2005)

On Complexity of Counting Fixed Point Configurations in Certain Classes of Graph Automata

PREDRAG T. TOŠIĆ
Open Systems Laboratory (http://www-osl.cs.uiuc.edu/),
Department of Computer Science, University of Illinois at Urbana-Champaign
Mailing address: Thomas Siebel Center for Computer Science,
201 N. Goodwin Ave., Urbana, IL 61801, USA
p-tosic@cs.uiuc.edu

Abstract

We study computational complexity of counting the *fixed point configurations (FPs)* in certain discrete dynamical systems. We prove that both exact and approximate counting of FPs in *Sequential* and *Synchronous Dynamical Systems* (SDSs and SyDSs, respectively) are computationally intractable, even when each node is required to update according to a symmetric Boolean function. We also show that counting exactly the *garden of Eden configurations (GEs),* as well as all *transient configurations,* is just as hard in this setting. Moreover, the hardness of enumerating FPs holds even in some severely restricted cases, such as when the nodes of an SDS or SyDS use only two different symmetric Boolean update rules, and when each node has a neighborhood size bounded by a small constant.

Keywords: Cellular and graph automata, sequential and synchronous dynamical systems, configuration space properties, computational complexity of enumeration problems, **#P**-completeness

1 Introduction and Motivation

We study in this work certain classes of *graph automata* that can be used as an abstract idealization of the classical networked distributed systems, as well as of various multi-agent systems and *ad hoc* networks, and as a theoretical model for the computer simulation of a broad variety of computational, physical, social, and socio-technical distributed infrastructures. In this and several related papers (see, e.g., [2, 3, 4, 5, 6, 7, 9, 10, 32, 45, 46]), the general approach has been to study mathematical and computational *configuration space properties* of such graph automata: what are the possible *global behavior patterns* of the entire system, given the simple local behaviors of its components, and the interaction pattern among these components.

We specifically focus on determining *how many* fixed point configurations such graph automata have, and *how hard* is the computational problem of *counting* (or enumerating) these configurations. In a nutshell, the contributions of this paper are as follows. We prove that both exact and approximate counting of the number of *fixed point* configurations in *Sequential* and *Synchronous Dynamical Systems* is computationally intractable, even when each node is required to update according to a symmetric Boolean function. We also show that the exact counting of the "*Garden of Eden*" configurations, as well as of all *transient configurations,* is just as hard in this setting.

The rest of the paper is organized as follows. Section 2 is devoted to the necessary preliminaries about the models studied in this paper, namely, the sequential and synchronous dynamical systems. Section 3

ISSN 1433-8092

summarizes our technical results, and reviews some recent research that is closely related to our work. The original results are presented in Section 4. Finally, we conclude and outline some possible extensions in Section 5.

2 Preliminaries

In this section, we define and briefly discuss the discrete dynamical system models studied in this paper, and their configuration space properties. *Sequential Dynamical Systems* (henceforth referred to as SDSs) are proposed in [8, 9, 10] as an abstract model for computer simulations. This model has been successfully applied in the development of large-scale socio-economic simulation systems such as the *TRANSIMS* project at the Los Alamos National Laboratory [11].

A *Sequential Dynamical System (SDS)* $\mathcal{S}$ is a triple (G, F, Π), whose components are as follows:

1. $G(V, E)$ is an undirected graph without multi-edges or self-loops. G is referred to as the *underlying graph* of $\mathcal{S}$. We often use n to denote $|V|$ and m to denote $|E|$. The nodes of G are enumerated 1, 2, ..., n.

2. Each node is characterized by its *state.* The state of node i, denoted by s_i, takes on a value from some finite domain, $\mathcal{D}$. In this paper, we shall restrict $\mathcal{D}$ to $\{0, 1\}$. We use d_i to denote the degree of node i. Each node i is associated with a *node update rule* $f_i : \mathcal{D}^{d_i+1} \rightarrow \mathcal{D}$, for $1 \leq i \leq n$. We also refer to f_i as the *local transition function.* The inputs to f_i are the state of node i itself and the states of the neighbors of i. We use F to denote $\{f_1, f_2, \ldots, f_n\}$.

3. Finally, Π is a permutation of $V = \{1, 2, \ldots, n\}$ specifying the order in which the nodes update their states using their local transition functions.

The nodes are processed in the *sequential* order specified by the permutation Π. The processing associated with a node consists of computing the new value of its state according to the node's update function, and changing its state to this new value.

Most of the early work on sequential dynamical systems has focused primarily on the SDSs with *symmetric Boolean functions* as the node update rules [2, 3, 4, 5, 7, 9, 10]. By "symmetric" is meant that the future state of a node does not depend on the order in which the input values of this node's neighbors are specified. Instead, the future state depends only on $\Sigma_{j \in N(i)}\, x_i$ (where $N(i)$ stands for the *extended neighborhood* of a given node, i, that includes the node i itself), i.e., on how many of the node's neighbors are currently in the state 1. Thus symmetric Boolean SDSs correspond to *totalistic (Boolean) cellular automata* of Wolfram [52, 53].

The assumption about *symmetric* Boolean functions can be easily relaxed to yield more general SDSs. We give special attention to the symmetry condition for two reasons. First, our computational complexity theoretic lower bounds for such SDSs imply stronger lower bounds for determining the corresponding configuration space properties[1] of the more general classes of graph automata and communicating finite state machines (CFSMs). Second, symmetry provides one possible way to model the "mean field effects" used in statistical physics and studies of other large-scale systems. A similar assumption is made in [12].

A *Synchronous Dynamical System* (SyDS) is an SDS *without* the node permutation. In an SyDS, at each discrete time step, all the nodes perfectly synchronously in parallel compute and update their state values. Thus, SyDSs are similar to the finite classical cellular automata (CA), except that in an SyDS the nodes may be interconnected in an arbitrary fashion, whereas in a classical cellular automaton the nodes are interconnected in a regular fashion (such as, e.g., a line, a rectangular grid, or a hypercube). Another

[1] Configuration spaces of sequential and synchronous dynamical systems will be defined in subsection 2.1.

difference is that, while in the classical CA all the nodes update according to the same rule, in an SyDS different nodes, in general, use different local update rules.

2.1 SDS and SyDS Configuration Space Properties

A configuration of an SDS or SyDS $\mathcal{S} = (G, F, \Pi)$ is a vector $(b_1, b_2, \ldots, b_n) \in \mathcal{D}^n$. A configuration $\mathcal{C}$ can also be thought of as a function $\mathcal{C} : V \to \mathcal{D}$.

The function computed by SDS $\mathcal{S}$, denoted by $F_{\mathcal{S}}$, specifies for each configuration $\mathcal{C}$ the next configuration $\mathcal{C}'$ reached by $\mathcal{S}$ after carrying out the updates of the node states in the order given by Π. Thus, the function $F_{\mathcal{S}} : \mathcal{D}^n \to \mathcal{D}^n$ is a total function on the set of global configurations. This function therefore defines the dynamics of the SDS $\mathcal{S}$. We say that $\mathcal{S}$ moves from a configuration $\mathcal{C}$ to a configuration $F_{\mathcal{S}}(\mathcal{C})$ in a single transition step. Alternatively, we say that SDS $\mathcal{S}$ moves from a configuration $\mathcal{C}$ at time t to a configuration $F_{\mathcal{S}}(\mathcal{C})$ at time $t+1$. Assuming that each node update function f_i is computable in time polynomial in the size of the description of $\mathcal{S}$, clearly each transition step will also take polynomial time (in the size of the SDS's description).

The *configuration space* (also called *phase space*) $\mathcal{P}_{\mathcal{S}}$ of an SDS or SyDS $\mathcal{S}$ is a directed graph defined as follows. There is a vertex in $\mathcal{P}_{\mathcal{S}}$ for each global configuration of $\mathcal{S}$. There is a directed edge from a vertex representing configuration $\mathcal{C}$ to that representing configuration $\mathcal{C}'$ if $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}'$. Since an SDS is deterministic, each vertex in its phase space has the out-degree of 1. Since the domain $\mathcal{D}$ of state values is assumed finite, and the number of nodes in the SDS is finite, the number of configurations in the phase space is also finite. If the size of the domain (that is, the number of possible states of each node) is $|\mathcal{D}|$, then the number of global configurations in $\mathcal{P}_{\mathcal{S}}$ is $|\mathcal{D}|^n$.

Definition 2.1 *Given two configurations $\mathcal{C}$ and $\mathcal{C}$ of an SDS or SyDS $\mathcal{S}$, configuration $\mathcal{C}$ is a* **predecessor** *of $\mathcal{C}$ if $F_{\mathcal{S}}(\mathcal{C}') = \mathcal{C}$, that is, if $\mathcal{S}$ moves from $\mathcal{C}'$ to $\mathcal{C}$ in one global transition step.*

Definition 2.2 *Given two configurations $\mathcal{C}$ and $\mathcal{C}$ of an S(y)DS $\mathcal{S}$, $\mathcal{C}'$ is an* **ancestor** *of $\mathcal{C}$ if there is a positive integer* t *such that $F_{\mathcal{S}}{}^t(\mathcal{C}') = \mathcal{C}$, that is, if $\mathcal{S}$ evolves from $\mathcal{C}'$ to $\mathcal{C}$ in one or more transitions.*

In particular, a predecessor of a given state $\mathcal{C}$ is trivially also its ancestor.

Definition 2.3 *A configuration $\mathcal{C}$ of an S(y)DS $\mathcal{S}$ is a* **Garden of Eden (GE)** *configuration if $\mathcal{C}$ has no predecessor.*

Definition 2.4 *A configuration $\mathcal{C}$ of an S(y)DS $\mathcal{S}$ is a* **fixed point (FP)** *configuration if $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}$, that is, if the transition out of $\mathcal{C}$ is to $\mathcal{C}$ itself.*

Note that a *fixed point* is a configuration that is its own predecessor.

Definition 2.5 *A configuration $\mathcal{C}$ of an S(y)DS is a* **cycle configuration (CC)** *if there exists an integer $t \geq 2$ such that*
(i) $F_{\mathcal{S}}{}^t(\mathcal{C}) = \mathcal{C}$; and
(ii) $F_{\mathcal{S}}{}^q(\mathcal{C}) \neq \mathcal{C}$, for any integer q, $0 < q < t$.
Integer t *above is called the* **period** *or* **length** *of the temporal cycle.*

In other words, $\mathcal{C}$ is a *cycle configuration* if it is reachable from itself in two or more transitions, but not in a single transition.

Definition 2.6 *A configuration $\mathcal{C}$ of an S(y)DS is a* **transient configuration (TC)** *if $\mathcal{C}$ is neither a fixed point nor a cycle configuration.*

As the name suggests, transient configurations, unlike fixed points or cycle configurations, are never revisited. We note that a GE configuration is a special case of a transient configuration; a GE configuration is not reachable from *any* configuration including itself. We also remark that a node in the phase space may have multiple predecessors. This means that the time evolution map F of an SDS or SyDS is in general *not invertible* but is *contractive*. The existence of configurations with multiple predecessors also implies that certain configurations have no predecessors. A configuration with no predecessors is called a *garden of Eden* configuration (see Definition 2.3). Such configurations can occur only as the initial states and can never be generated during the time evolution of an SDS or SyDS.

3 Summary of Results and Related Work

Given an SDS or SyDS $\mathcal{S}$, let $|\mathcal{S}|$ denote the size of the representation of $\mathcal{S}$. In general, this includes the number of nodes and edges, and the description of the local transition functions. When $\mathcal{D} = \{0,1\}$ and the local transition functions are given as the truth tables, $|\mathcal{S}| = O(m + |T|n)$, where $|T|$ denotes the maximum size of a table, n is the number of nodes and m is the number of edges in the underlying graph. Thus, for a node v_i with degree d_i, the size of the table specifying an arbitrary Boolean function is $O(2^{d_i})$, while the size of the table specifying a *symmetric* Boolean function is $O(d_i)$. Another, more common way of specifying the local transition functions is via *Boolean formulae;* we shall assume that f_i of SDSs and SyDSs considered in the sequel are indeed given as reasonably succinctly encoded Boolean formulae of appropriate kinds. We shall also assume throughout that evaluating any local transition function given its input values can be done in polynomial time.

We study herewith the problem of *counting* the fixed point (FP) configurations of Boolean SDSs and SyDSs. In particular, we prove the following results:

- counting FPs in the general Boolean (and, consequently, also in any other finite domain) SDSs and SyDSs is **#P**-complete;
- this hardness result still holds when the node update rules of these S(y)DSs are restricted to *symmetric Boolean functions;*
- moreover, the result remains valid even when only two different symmetric update rules are used, and when the maximum degree of each node in the underlying graph is a small constant.

3.1 Related work

Various computational aspects of *cellular automata (CA)* have been studied by a number of researchers; see for example [13, 14, 22, 23, 31, 41, 50, 52, 53]. Much of this work addresses decidability of various properties for infinite CA. Insofar as the computational complexity of fundamental problems about *finite CA* are concerned, we single out the following. The first **NP**-complete problems for CA are shown by Green in [22]; these problems are of a general *reachability* flavor, i.e., they address the properties of the *forward dynamics* of CA. Sutner addresses the *backward dynamics* problems, such as the problem of an arbitrary configuration's *predecessor existence,* and their computational complexity in [41]. In the same paper, Sutner also establishes the efficient solvability of the predecessor existence problem for any CA with a *fixed neighborhood radius.* In [15], Durand solves the injectivity problem for arbitrary 2-D CA but restricted to the finite configurations only; that paper contains one of the first results on **coNP**-completeness of a natural and important problem about CA. Durand also addresses the *reversibility problem* in the same, two-dimensional CA setting in [16].

SDSs and SyDSs investigated in this paper are closely related to the *graph automata (GA)* models studied in [29, 34] and the one-way cellular automata studied by Roka in [37]. In fact, the general finite-domain SyDSs exactly correspond to the graph automata of Nichitiu and Remila as defined in [34].

Barrett, Mortveit and Reidys [9, 10, 32, 36] and Laubenbacher and Pareigis [28] investigate the mathematical properties of sequential dynamical systems. Barrett et al. study the computational complexity of several phase space questions for SDSs. These include the REACHABILITY, PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE problems [5, 6]. Problems related to the existence of *garden of Eden* and *fixed point* configurations are studied in [7]. In particular, the basic **NP**-completeness results for the problems of FP, GE and non-unique predecessor existence in various restricted classes of Boolean S(y)DSs are proven in that paper. Algorithms for efficiently finding an FP in certain other restricted classes of S(y)DSs can be also found in [7]. Our results in Section 4 of this paper can be viewed as a natural partial extension of the work in [7]: instead of the appropriate *decision problems* about the fixed points and gardens of Eden in SDSs and SyDSs, we focus herein on studying the related *counting problems.*

Among various restricted classes of Boolean SDSs and SyDSs, those with the local update rules restricted to *symmetric functions* have received particular attention (e.g., [10, 28, 32]). Computational complexity of the reachability-related problems in the context of, among other restricted types, symmetric Boolean SDSs is investigated in [6]. We show in this paper that, in contrast to the computational feasibility of the problem of their reachability [6], the problem of *counting* stable configurations (FPs) in symmetric SDSs and SyDSs, under the usual assumptions in computational complexity theory, is intractable.

4 Counting Fixed Points in Boolean SDSs and SyDSs

The results in this section constitute an extension of the work presented in [7] and [6]. In [7], the computational complexity of *decision problems* related to the fixed point and the garden of Eden configurations in SDSs and SyDSs is studied. Once **NP**-completeness of these decision problems has been established [7], a natural further course of inquiry about the fundamental SDS phase space properties is to determine how hard it is to count *how many* FPs, GEs, and/or other configurations of interest an SDS of a given type may have.

Intuitively, one would expect, for example, that counting the fixed points of an arbitrary Boolean SDS or SyDS is no easier than counting the satisfying truth assignments of an arbitrary instance of the SATISFIABILITY problem [17, 35]. The intuitive notion of computational hardness of counting problems is formalized via the definition of the class **#P** (read: "sharp-P" or "number-P"). A counting problem Ψ belongs to the class **#P** if there exists a nondeterministic algorithm such that for each instance I of Ψ, the number of nondeterministic "guesses" this algorithm makes that lead to acceptance equals the number of solutions of Ψ, and, in addition, it is required that the longest of the nondeterministic computations of this algorithm on any input as specified by I be polynomially bounded in the size of the description of I. For an alternative but equivalent definition of class **#P** in terms of *polynomially balanced relations,* we refer the reader to [35].

The hardest problems in class **#P** are the **#P**-complete problems. A counting problem Ψ is **#P**-complete if and only if (i) it is in class **#P**, and (ii) any other problem in **#P** is efficiently reducible to Ψ. Thus, if we could solve any **#P**-complete problem in polynomial time, then all the problems in class **#P** would be solvable in polynomial time, and the entire class **#P** would collapse to **P**. For more on class **#P** we refer the interested reader to Chapter 18 of [35], and references therein.

As one would expect, counting versions of the "standard" decision **NP**-complete problems, such as SATISFIABILITY or HAMILTON CIRCUIT, are **#P**-complete [35]. What is curious, however, is that the counting versions of some tractable decision problems, such as BIPARTITE MATCHING or MONOTONE 2CNF SATISFIABILITY, are also **#P**-complete [49, 48].

If we could reduce the problem of counting the satisfying truth assignments of an instance of, say, Boolean *3CNF-SAT* or *PE3SAT* formulae [17], to counting the fixed points of a corresponding SDS, this would establish the **#P**-completeness of the latter. However, the reduction from *ODD-PE3SAT* to *FPE* that is used in [7] to establish the **NP**-completeness of the Fixed Point Existence problem for SDSs would not suffice, since it does not map the satisfying assignments of an instance of *ODD-PE3SAT* to the fixed points of the corresponding SDS in a one-to-one fashion. That is, in order to prove the intractability of counting FPs of Boolean SDSs and SyDSs, not any *polynomial time* reduction from the known **#P**-complete problems suffices. What is required is a kind of an efficient reduction that *preserves the number of solutions.* That is, we need a construction whereby each satisfying truth assignment of an instance of, e.g., *3CNF-SAT* or *PE3SAT* translates into a *distinct* fixed point of the corresponding SDS or SyDS. We define this special kind of efficient reductions next:

Definition 4.1 *Given two decision problems* Π *and* Π', *a* PARSIMONIOUS REDUCTION *from* Π *to* Π' *is a polynomial-time transformation* g *that preserves the number of solutions; that is, if an instance* I *of* Π *has* n_I *solutions, then the corresponding instance* $g(I)$ *of* Π' *also has* $n_{g(I)} = n_I$ *solutions.*

In practice, one often resorts to reductions that are "almost parsimonious", in a sense that, while they do not exactly preserve the number of solutions, $n_{g(I)}$ in the previous definition can be efficiently computed from n_I.

Definition 4.2 *Given two decision problems* Π *and* Π', *a* WEAKLY PARSIMONIOUS REDUCTION *from* Π *to* Π' *is a polynomial-time transformation* g *such that, if an instance* I *of* Π *has* n_I *solutions, then the corresponding instance* $g(I)$ *of* Π' *has the number of solutions* $n_{g(I)}$ *that can be computed from* n_I *in polynomial time.*

Our fundamental result on the hardness of counting the fixed point configurations of an arbitrary Boolean S(y)DS, as well as similar hardness results in the next two subsections, will follow from

Proposition 4.1 *(e.g., [35]) Given two decision problems* Π *and* Π', *if the corresponding counting problem* $\#\Pi$ *is* **#P**-*hard and if there exists a weakly parsimonious reduction from* Π *to* Π', *then the counting problem* $\#\Pi'$ *is* **#P**-*hard, as well.*

4.1 Counting Fixed Points of General Boolean SDSs and SyDSs

We shall use reductions from the known **#P**-complete problems, such as the counting version of *POSITIVE-EXACTLY-1-IN-3SAT (PE3SAT),* to the problems of counting FPs in certain classes of the SDS and SyDS automata. These reductions will formally establish the **#P**-completeness of those counting problems about SDSs and SyDSs. We now define the variants of *Satisfi ability* [17, 35] that we shall use in the sequel:

Definition 4.3 Exactly-one-in-three-satisfiability *(or* **E3SAT** *for short), is a version of* 3CNF-SAT *[17] such that, fi rst, each clause in a given* 3CNF *formula contains exactly three literals, and, second, where a truth assignment is considered to satisfy the given* 3CNF *formula if and only if* exactly one of the three literals *is* true *in each clause.* Positive-exactly-one-in-three-satisfiability *(***PE3SAT***) is further restricted: no clause in the* 3CNF *formula is allowed to contain a negated literal.*

Consider the following reduction from PE3SAT to #FP-SDS, where #FP-SDS denotes the problem of counting the fixed point configurations of an arbitrary Boolean SDS.

Let an arbitrary instance I of PE3SAT be given. We construct the corresponding instance of an SDS $\mathcal{S} = \mathcal{S}(I)$ as follows. We remark that $\mathcal{S}$ in this subsection will be "nearly symmetric"; we will modify our construction to a fully symmetric Boolean SDS and SyDS in the next subsection.

Assume that I has n variables and m clauses. The underlying graph of $\mathcal{S}$ has a distinct node for each variable x_i, $1 \le i \le n$, and for each clause C_j, $1 \le j \le m$. The node labeled x_i is connected to the node labeled C_j if and only if, in the Boolean formula I, variable x_i appears in clause C_j. In addition, our graph has one additional node, labeled y, that is adjacent to nodes C_j for all indices $j = 1, ..., m$. Hence, each C_j has exactly four neighbors, and node y has m neighbors.

The node update functions of our SDS $\mathcal{S}$ are as follows:

- Each node C_j evaluates the logical AND of the current value of node y, the value evaluated by the PE3SAT function of the three variables $\{x_{j_1}, x_{j_2}, x_{j_3}\}$ that appear in the corresponding clause C_j of I, and the current value of itself; that is, the node update function C_j evaluates to 1 if and only if:

(i) *exactly* one out of the three neighboring nodes $x_{j_1}, x_{j_2}, x_{j_3}$ currently holds the value 1; and

(ii) the node y currently holds the value 1; and

(iii) the current value of C_j itself is 1.

- The "special" node y evaluates the *AND* of its own current value and the entire set of current values held in the clause nodes C_j, $1 \le j \le m$. This will enable us to argue that the node y, in effect, evaluates the Boolean formula for the specified truth assignment $\{x_1, ..., x_n\}$, provided that the initial value stored in node y is $y^{t=0} = 1$, and, likewise, that $C_j^{t=0} = 1$, for all j, $1 \le j \le m$.

- Each node x_i evaluates the logical AND of itself and the current values stored in the clause nodes $C_{j(i)}$ such that, in the original formula I, variable x_i appears in clause $C_{j(i)}$.

The order of the node updates is $(C_1, ..., C_m, y, x_1, ..., x_n)$.

Since $\mathcal{S}$ has $n + m + 1$ nodes, the corresponding phase space will have 2^{n+m+1} configurations.

We now claim that the reduction from #PE3SAT to #FP-SDS based on the above SDS construction from an instance I of PE3SAT is weakly parsimonious; it will then immediately follow that

Theorem 4.1 *The problem of counting the fixed points of an arbitrary Boolean SDS (and therefore also of any more general finite domain SDS), is* **#P***-complete.*

Remark: Similarly, by a straight-forward modification of the given SDS construction, one can establish that the #FP-SyDS problem for the general Boolean (and therefore any finite domain) SyDSs is **#P**-complete, as well.

Proof: That #FP-SDS is a member of the class **#P** is immediate from the definition of SDS and the assumptions stated in Section 3. The **#P**-hardness will follow from the **#P**-hardness of the corresponding counting version of PE3SAT, once we establish that the reduction from #PE3SAT to #FP-SDS is, indeed, (weakly) parsimonious.

First, assume we pick an initial configuration $\mathcal{C}$ such that its sub-configuration $(x_1^0, ..., x_n^0)$ is an unsatisfying truth assignments for the variables $(x_1, ..., x_n)$ in the corresponding instance of PE3SAT. Then, at the first step, at least one of the C_j's nodes will evaluate to 0, and hence the node y will subsequently evaluate to 0. Once the node y holds the value 0, at the next step *all* clause nodes C_j will evaluate to 0, and subsequently they will force all the variable nodes x_i to evaluate to 0, as well[2]. Thus, it follows that, if initially the sub-configuration $(x_1^0, ..., x_n^0)$ corresponds to a falsifying truth assignment for I, then the fixed point configuration 0^{n+m+1} is reached in (at most) two global transition steps.

Let us assume now that the initial configuration $\mathcal{C}^{t=0}$ of $\mathcal{S}$ has a sub-configuration $(x_1^0, ..., x_n^0)$ that corresponds to a satisfying truth assignment to the corresponding Boolean variables in the instance I of

[2]We shall assume in this and all other constructions in this paper that each Boolean variable in any given formula I appears in at least one clause.

PE3SAT and, in addition, that $y^{t=0} = 1$ and $C_j^{t=0} = 1$. Then each C_j will evaluate to 1, thereby causing the node y to remain evaluated to 1, as well. Since all $C_j = 1$, each node x_i will keep its original value: $x_i^1 = x_i^0$. Since these values form a satisfying truth assignment, at the next step of the dynamic evolution of $\mathcal{S}$, again each C_j will evaluate to 1, causing y to re-evaluate to 1, and all of x_i to remain the same; in other words, a fixed point configuration has been reached. Hence, if the initial configuration $\mathcal{C}^0$ has $y^0 = 1$ and $C^0 = 1^m$, and it encodes a satisfying truth assignment $(x_1^0, ..., x_n^0)$ of I, then $\mathcal{C} = \mathcal{C}^0$ already is a fixed point, given by $(C_1, ..., C_m, y, x_1, ..., x_n) = (1, ..., 1, 1, x_1^0, ..., x_n^0)$. Thus, it follows that each satisfing truth assignment $(x_1, ..., x_n)$ of I gets mapped into a *distinct* fixed point $(1, ..., 1, 1, x_1, ..., x_n)$ of the corresponding SDS $\mathcal{S} = \mathcal{S}(I)$.

Finally, it is easy to see that, if $y^0 = 0$, then $\mathcal{S}$ reaches the fixed point 0^{n+m+1} in a single step, and if there exists at least one index j such that initially $C_j^0 = 0$, then the sink 0^{n+m+1} is reached in at most two steps. Since each initial configuration that encodes a falsifying truth assignment $(x_1, ..., x_n)$ to I yields the fixed point configuration 0^{n+m+1} in at most two steps, we conclude that there cannot be any fixed points of $\mathcal{S}$ except for 0^{n+m+1} and those fixed points that correspond to the satisfying assignments to I. Therefore, if I has L satisfying assignments, where $0 \leq L \leq 2^n$, then the SDS $\mathcal{S}$ as constructed above will have exactly $L + 1$ fixed points.

This reduction establishes that, in general, counting fixed points of an arbitrary Boolean SDS is no easier than counting satisfying truth assignments of instances of PE3SAT formulae, and the **#P**-hardness of #FP-SDS follows, thereby establishing the claim of the theorem.

■

4.2 Counting Fixed Points of Symmetric Boolean SDSs and SyDSs

The hardness results for symmetric Boolean SDSs and SyDSs will be based on an appropriate reduction from the *PE2-IN-3SAT* problem. We define PE2-IN-3SAT similarly to how we defined PE3SAT, only this time we require each clause to have *exactly two* true variables (rather than *exactly one* as was the case in PE3SAT). We observe that, since PE3SAT is **NP**-complete, so is PE2-IN-3SAT, and moreover the **#P**-completeness of the counting version of the former, let's denote it #PE3SAT, also implies the **#P**-completeness of the counting version of the latter, #PE2-IN-3SAT.

Let an instance I of PE2-IN-3SAT be given. Assume that there are n Boolean variables, denoted $x_1, ..., x_n$, and m clauses, $C_1, ..., C_m$, in I. We recall that each clause C_j contains *exactly* three unnegated variables, $x_{j_1}, x_{j_2}, x_{j_3}$. An instance I is a *positive* or *satisfying* instance of PE2-IN-3SAT if and only if there exists a truth assignment to $x_1, ..., x_n$ such that *exactly* two variables in each clause are true.

We now prove that counting FPs of a symmetric Boolean SyDS or SDS is **#P**-complete. We recall that fixed points are invariant under the node update ordering; that is, regardless of whether the nodes update synchronously in parallel, or sequentially according to an arbitrary ordering Π, the fixed points of the underlying dynamical system as specified by its graph and the local node update functions remain the same (see [32] for a proof).

Theorem 4.2 *The problem of counting fi xed points of a symmetric Boolean Synchronous Dynamical System, abbreviated as* #FP-SYM-SYDS*, is* **#P**-*complete.*

Proof: To show **#P**-hardness, we reduce the problem of counting the satisfying truth assignments of an instance of PE2-IN-3SAT to counting the fixed points of a symmetric Boolean SyDS. We construct an SyDS, $\mathcal{S}$, from an instance of PE2-IN-3SAT as follows. We let the underlying graph of $\mathcal{S}$ have $m+n+1$ vertices: one for each variable, one for each clause, and one additional vertex, denoted by y. The edges of the underlying SyDS graph are as follows: each vertex node x_i is adjacent to those and only those clause nodes $C_{j(i)}$ such that the corresponding variable x_i appears in the corresponding clause $C_{j(i)}$ of formula I. Let each clause node C_j be adjacent to all other clause nodes C_k (for all k, $1 \leq k \leq m, \;\; k \neq j$), to the special node y, and to the three nodes $x_{j_1}, x_{j_2}, x_{j_3}$ corresponding to the Boolean variables that appear in the clause C_j in the formula; and, finally, by symmetry, let the node y be adjacent to all the clause nodes C_k.

We define the node update functions as follows:

$x_i^{t+1} = x_i^t \wedge (\wedge_{j(i)} C_{j(i)}^t)$;
$C_j^{t+1} =$ *ALL-BUT-ONE* $\{x_{j_1}^t, x_{j_2}^t, x_{j_3}^t, C_1^t, ..., C_m^t, y^t\}$;
$y^{t+1} = y^t \wedge (\wedge_{j=1}^m C_j^t)$,

where the Boolean function *ALL-BUT-ONE* $\{z_1, ..., z_q\} = 1$ if and only if *exactly* one of its inputs z_l is 0, and all the rest are $1s$.

We now claim that the constructed synchronous dynamical system has $|T|+2$ fixed points if and only if the corresponding instance of PE2-IN-3SAT has $|T|$ satisfying truth assignments.

To prove the claim, we will carefully analyze all possible scenarios of the dynamic behavior of $\mathcal{S}$, based on its initial configuration. We shall adopt the notation that x and C without any subscripts denote Boolean n- and m-vectors, respectively, the former being a shorthand for $(x_1, ..., x_n)$ and the latter for $(C_1, ..., C_m)$. Hence, using this abridged notation, we can now write arbitrary configurations of $\mathcal{S}$ as ordered triples (x, C, y).

We start with a simple observation that, since the node update functions at the variable nodes x_i, as well as the special node y, are conjunctions of inputs that include the old value of the node in question itself, once any x_i or the node y evaluates to 0, it remains 0 thereafter. We split the analysis of the dynamic behavior of $\mathcal{S}$ into two parts.

Case 1: $y^{t=0} = 0$. First consider the case when, initially, $x_i^{t=0} = 1$ for all i, $1 \leq i \leq n$, and also $C_j^{t=0} = 1$, for all j, $1 \leq j \leq m$. At time $t = 1$, all the variable nodes x_i will remain in the state 1. Also, since each clause node update function C_j at time $t = 1$ will have all inputs equal to 1 except for a single one (namely, the input $y^0 = 0$), $C_j^1 = 1$. On the other hand, clearly $y^t = 0$ for $t = 1, 2, ...$, irrespective of the remaining inputs C_j^{t-1}. Hence, we conclude that the configuration $(x, C, y) = (1^n, 1^m, 0)$ is a fixed point of $\mathcal{S}$. Notice, however, that this configuration does not correspond to a satisfying truth assignment of the corresponding instance I of PE2-IN-3SAT, since, if all $x_i = 1$, then no clause C_j of I will be satisfied, as each clause requires exactly two inputs equal to 1 and one input equal to 0.

Now let's consider a starting configuration where there exists an index $j_\star$ such that $C_{j_\star}^0 = 0$. Then, at time $t = 1$, all the clause nodes C_j will have at least two 0 inputs (namely, y^0 and $C_{j_\star}^0$), and, since they evaluate the *ALL-BUT-ONE* function of their inputs, they will all evaluate to 0: $C_j^1 = 0$, for all j, $1 \leq j \leq m$. Hence, at the next step, $x_i^2 = x_i^1 \wedge (\wedge_{j(i)} C_{j(i)}^1) = 0$ for all i, $1 \leq i \leq n$, and it is easy to see that, for $t \geq 2$, $(x^t, C^t, 0) = 0^{n+m+1}$, i.e. the fixed point 0^{n+m+1} is swiftly reached - in at most two transition steps. Similar analysis, and the same conclusion, hold if we assume that there is at time $t = 0$ at least one index $i_\star$ such that $x_{i_\star}^0 = 0$. We observe that, just like the fixed point $(1^n, 1^m, 0)$, the fixed point $(0^n, 0^m, 0) = 0^{n+m+1}$ does not correspond to a satisfying truth assignment $(x_1, ..., x_n)$ of formula I. This completes the analysis of all possible scenarios when $y^0 = 0$.

Case 2: $y^{t=0} = 1$. There are two sub-cases to consider. The first sub-case is when there exists an index $j_\star$ such that $C_{j_\star}^{t=0} = 0$. The second sub-case is when, initially, $C_j^{t=0} = 1$, for all $1 \leq j \leq m$.

Let's first assume that there exists $j_\star$ such that $C_{j_\star}^{t=0} = 0$. Then $y^t = 0$ for all $t \geq 1$, and, furthermore, the three variable nodes $\{x_{j_\star,1}, x_{j_\star,2}, x_{j_\star,3}\}$, corresponding to the variables that appear in the clause $C_{j_\star}$, will also evaluate to 0 at time $t = 1$, and remain 0 thereafter. At time $t = 2$, all C_j will have more than one input equal to 0. Consequently, all $C_j^{t=2} = 0, \ \ 1 \le j \le n$. Thus, a single $C_{j_\star}^{t=0} = 0$ assures the quick collapse to the "sink" stable configuration 0^{n+m+1}.

Now we examine the most interesting scenario, when the initial configuration $(x^{t=0}, C^{t=0}, y^{t=0})$ is of the form $(x^{t=0}, 1^m, 1)$; that is, we assume that, initially, all $C_j^{t=0} = 1$ as well as $y^{t=0} = 1$. There are two possibilities: either $x^{t=0}$ is a satisfying truth assignment of the PE2-IN-3SAT instance I, or it is not a solution of I. If $I(x^{t=0}) =$ *false*, then there must be at least one index j such that the clause $C_j = 0$. If so, then the corresponding clause node C_j of our SyDS will evaluate to zero, as well: $C_j^{t=1} = 0$. Hence, at time $t = 2$, $y^2 = 0$, and also $x_{j,1}^{t=2} = x_{j,2}^{t=2} = x_{j,3}^{t=2} = 0$. Thus, the resulting SyDS dynamics is the same as in case of an initial configuration with $C_j^{t=0} = 0$, only beginning one time step later. In particular, after three time steps, $(x^3, C^3, y^3) = 0^{n+m+1}$, and, of course, $(x^t, C^t, y^t) = 0^{n+m+1}$ for all $t \ge 3$.

Finally, let us now assume that $x^{t=0} = (x_1^0, ..., x_n^0)$ is a *satisfying* truth assignment of the PE2-IN-3SAT formula. Then, at time $t = 1$, all the clause nodes $C_j^{t=1}$ will re-evaluate to 1, since each clause C_j in the Boolean formula will have exactly two *true* inputs if and only if each clause node C_j of the corresponding SyDS has exactly $m + 1 + (3 - 1) \ = \ m + 3$ (i.e., *all but one*) of its inputs equal to 1. Similarly, $y^{t=2} = y^{t=1} = y^{t=0} = 1$. Since all the nodes C_j satisfy $C_j^{t=1} = C_j^{t=0} = 1$, it follows that each variable node x_i will retain its old value: $x_i^1 = x_i^0 \wedge (\wedge_{j(i)} C_{j(i)}^0) = x_i^0 \wedge 1 = x_i^0$, and, similarly, also $x_i^2 = x_i^1 \wedge 1 = x_i^1 = x_i^0$. It is now immediate that any starting configuration of the form $(x^0, 1^m, 1)$, where the Boolean n-vector x^0 is a satisfying truth assignment of the given PE-2-IN-3SAT instance I, is a fixed point of $\mathcal{S}$.

By the above analysis, we see that the phase space of $\mathcal{S}$ has a rather simple structure: no cycles whatsoever, only short transients (the longest chains of transient states are of length 3), and the fixed points of $\mathcal{S}$ are precisely the "sink" 0^{n+m+1}, the configuration $(x, C, y) = (1^n, 1^m, 0)$, and those configurations (x, C, y) such that $C = 1^m$, $y = 1$, and the Boolean n-vector $x = (x_0, ..., x_n)$ is a satisfying truth assignment of I. In particular, if I has k solutions (for some integer k, $0 \le k \le 2^n$), then $\mathcal{S}$ has exactly $k + 2$ fixed points, and the claim of the theorem follows.

■

By the aforementioned invariance of fixed points with respect to the node update ordering, the next result on the hardness of counting FPs in symmetric Boolean SDSs is not at all surprising.

Theorem 4.3 *The problem of counting fixed point configurations of symmetric Boolean SDSs (abbreviated as* #FP-SYM-SDS*) is* **#P***-complete.*

Proof: In order to prove the theorem explicitly, as well as establish several other complexity-theoretic counting results for symmetric Boolean SDSs, we consider the following construction of an SDS $\mathcal{S}$' from the SyDS $\mathcal{S}$ used in the proof of the previous theorem.

- The underlying graph and the local node updating functions are as in the SyDS construction in the previous theorem.
- Let the node ordering be given by $\Pi = (y, C_1, ..., C_m, x_1, ..., x_n)$. Thus,
  $y^{t+1} = y^t \wedge (\wedge_{j=1}^m C_j^t)$,
  $C_j^{t+1} =$ *ALL-BUT-ONE* $\{y^{t+1}, C_1^{t+1}, ..., C_{j-1}^{t+1}, C_j^t, C_{j+1}^t, ..., C_m^t, x_{j_1}^t, x_{j_2}^t, x_{j_3}^t\}$,
  and, for any i such that $1 \le i \le n$,
  $x_i^{t+1} = x_i^t \wedge (\wedge_{j(i)} C_{j(i)}^{t+1})$,
  where, as before, $C_{j(i)}$ denotes precisely those clause nodes that correspond to the clauses in the original Boolean formula in which the variable x_i appears.

We will only sketch the analysis of what the phase space of $\mathcal{S}$' looks like, since much of the case analysis coincides with that for SyDS $\mathcal{S}$ in the previous Theorem.

Case 1: $C^{t=0} \neq 1^m$. Since $y^1 = y^0 \wedge (\wedge_{j=1}^{m} C_j^0)$, and at least one of C_j^0 (say, $C_{j_\star}^0$) is 0, the node vertex y will evaluate to $y^1 = 0$. Hence, each C_j^1 will have at least two zero inputs, namely y and $C_{j_\star}$, and hence the *ALL-BUT-ONE* function at node C_j will evaluate to zero at time $t = 1$, for all j. Hence, if not all C_j^0 are initially equal to 1, $\mathcal{S}$' will collapse to the sink 0^{n+m+1} in a single step. We observe that there are exactly $(2^m - 1) \times 2^{n+1} = 2^{m+n+1} - 2^{n+1}$ configurations $\mathcal{C}$ such that $C^0 \neq 1^m$, all of which except for the sink 0^{n+m+1} are transient configurations and, moreover, each of these TCs is also a garden of Eden.

Case 2: $C^{t=0} = 1^m$. This is the more interesting case with several sub-cases to consider. First, if $(y^0, C^0, x^0) = (0, 1^m, 1^n)$, then it is straight-forward to verify that this configuration is a fixed point that DOES NOT correspond to a solution of the corresponding instance I of PE2-IN-3SAT. If, on the other hand, $y^0 = 0$ and $x^0 \neq 1^n$, then at time $t = 1$, there are at least two nodes holding the value 0; in particular, there exists $C_{j_\diamond}$ such that $C_{j_\diamond}^{t=1} = 0$ since the node update function at $C_{j_\diamond}$ has at least two zero inputs at time $t = 1$. Consequently, at time $t = 2$, *every* clause node C_j will have at least two zero inputs, namely, $y^{t=2}$ and either $C_{j_\diamond}^{t=1}$ (if $j \leq j_\diamond$), or $C_{j_\diamond}^{t=2}$ (if $j > j_\diamond$). Therefore, $C_j^{t=2} = 0$ for all $j = 1, ..., m$, and subsequently $x_i^2 = 0$, for all $i = 1, ..., n$. Thus, in this case, the collapse to the sink 0^{n+m+1} takes (at most) two steps. Furthermore, the convergence from an initial state $(0, 1^m, x^0 \neq 1^n)$ to 0^{n+m+1} takes two steps *if and only if* $C_1^{t=1} = 1$ (and only one step, otherwise).

Finally, the remaining sub-cases to consider correspond to the initial configurations of $\mathcal{S}$' of the form $(y^0, C^0, x^0) = (1, 1^m, x^0)$. In this case, if $x^{t=0} = x_{true}$ is a satisfying truth assignment for the PE2-IN-3SAT formula I, then the configuration $(1, 1^m, x^0)$ will be a fixed point of $\mathcal{S}$'. If, however, $x^0 = x_{false}$ is a falsifying truth assignment for I, then, at time $t = 1$, at least one of the $C_j^{t=1}$ will evaluate to 0, and consequently, at time $t = 2$, first the node y will update to $y^{t=2} = 0$, and, since each $C_j^{t=2}$ will have at least two zero inputs, all the clause nodes will then evaluate to 0, and subsequently so will all the variable nodes $x_i^{t=2}$; i.e., $\mathcal{S}$' will converge to the sink 0^{n+m+1} in at most two steps. We observe that, in the case of an initial global configuration of the form $(1, 1^m, x_{false}^0)$, the convergence to 0^{n+m+1} will always take *exactly* two steps: that it cannot take more than two steps follows from the discussion above, whereas that it cannot take only one step stems from the observation that $y^1 = y^0 \wedge (\wedge_{j=1}^{m} C_j^0) = 1$, implying that $(y^1, C^1, x^1) \neq 0^{n+m+1}$.

Given the above analysis, it is immediate that $\mathcal{S}$' will have $|T| + 2$ fixed points if and only if the corresponding PE2-IN-3SAT formula has $|T|$ satisfying truth assignments. Hence, the **#P**-hardness of counting the fixed points of this restricted class of symmetric Boolean SDSs follows from the **#P**-hardness of counting the satisfying truth assignments of instances of PE2-IN-3SAT formulae. Since the membership of #FP-SYM-SDS in the class **#P** is easy to show, the claim of the theorem follows. ■

To summarize, the phase space of SDS $\mathcal{S}$' constructed in the proof above looks as follows. Since there are $n + m + 1$ nodes, there are 2^{n+m+1} configurations in total. Among these, there are precisely $|T| + 2$ fixed points, where $|T|$ is the number of solutions of the corresponding PE2-IN-3SAT formula I. The number of these solutions is in the range $\{0, 1, ..., 2^n\}$. All of the $|T|$ fixed points corresponding to the solutions of I, as well as the fixed point $(y = 0, C = 1^m, x = 1^n)$, are "isolated" fixed points, in a sense that they do not have any in-coming transients. In other words, each such configuration has the unique predecessor, namely, itself. The state 0^{n+m+1} is the "sink" for $\mathcal{S}$', in that all transient chains eventually end in 0^{n+m+1}. All the remaining configurations are transient states, and, in particular, $\mathcal{S}$' does not have any temporal cycles. Furthermore, all the transient chains are very short, since every transient configuration is either a garden of Eden, or its predecessor is a garden of Eden; this is immediate from the fact that every convergence to the sink 0^{n+m+1} takes at most two steps.

How many transient states, then, does SDS $\mathcal{S}$' have? Let $|F| = 2^n - |T|$ denote the number of *falsifying* truth assignments for the PE2-IN-3CNF formula I. Since there are $|T| + 2$ fixed points and no temporal cycles[3], it is immediate that there are exactly $2^{m+n+1} - |T| - 2 = 2^{m+n+1} + |F| - 2^n - 2$ transient states; we denote the number of transient configurations by $|\#TC|$. Since $0 \leq |T| \leq 2^n$, it follows that $2^{m+n+1} - 2 \geq |\#TC| \geq 2^{m+n+1} - 2^n - 2$. Therefore, in order to determine the *exact* number of transient states of $\mathcal{S}$', one has to determine the number of satisfying truth assignments of the corresponding PE2-IN-3SAT formula I; but, even without knowing anything about the number of solutions of I, one can always readily estimate $|\#TC|$, since the fraction of all global configurations of $\mathcal{S}$' that are TCs lies, roughly, between $1 - \Theta(2^{-m})$ and 1. Hence, determining $|\#TC|$ for this class of symmetric Boolean SDSs *exactly* is hard, but *approximately estimating* this number is relatively easy, and it gets easier as the number of clauses m grows with respect to the number of variables n.

It is possible to make the given bounds on $|\#TC|$ and $|\#FP|$ sharper, if we notice that the nontrivial instances of the CNF-type Boolean formulae in general, and our PE2-IN-3SAT in particular, are *never* tautologies, and furthermore one can use combinatorial arguments to come up with lower bounds for the number of falsifying truth assignments. We shall not dwell upon a detailed combinatorial analysis based on various features of the underlying instance of PE2-IN-3SAT. Instead, we will only establish a crude lower bound for the number of falsifying assignments, $|F|$. First, we observe that both 0^{n+m+1} and 1^{n+m+1} are *always* falsifying truth assignment, for any *nonempty* instance of PE2-IN-3SAT. Second, consider any satisfying truth assignment, $x_{true} \in \{0,1\}^n$. By definition of PE2-IN-3SAT, if we assign Boolean values to $x_1, ..., x_n$ according to x_{true}, then each clause of the given instance will contain exactly two variables equal to 1. Hence, the component-wise negation of this Boolean vector will yield exactly one out of three variables being *true* in each clause, and therefore it will be a falsifying truth assignment of this PE2-IN-3SAT formula. These two facts that hold for any nontrivial PE2-IN-3SAT formula together imply that the number of falsifying truth assignments for any instance of PE2-IN-3SAT must satisfy $|F| \geq 2^{n-1} + 1$, or, equivalently, $0 \leq |T| \leq 2^{n-1} - 1$. This enables us to sharpen the previously given bounds on the number of fixed points, transient states and gardens of Eden of an SDS constructed from a PE2-IN-3SAT formula the way we constructed $\mathcal{S}$'. Concretely,

$2 \leq |\#FP| = |T| + 2 \leq 2^{n-1} + 1$;
$2^{m+n+1} - 2^{n-1} - 1 \leq |\#TC| = 2^{m+n+1} + |F| - 2^n - 2 \leq 2^{m+n+1} - 2$; and
$2^{m+n+1} - 2^{n+1} - 1 \leq |\#GE| \leq |\#TC| - |F| \leq 2^{m+n+1} - 2^{n-1} - 1$.

Thus, for the restricted class of symmetric Boolean SDSs constructed from the PE2-IN-3SAT instances, *approximating* the number of fixed points is as hard as approximating the number of satisfying truth assignments of the corresponding instances of PE2-IN-3SAT, but estimating the number of transient states and gardens of Eden, i.e., the fraction of all configurations that happen to be TC (GE), is relatively easy, and gets easier as the number of clauses m in the corresponding PE2-IN-3SAT formula grows.

In summary, enumerating the fixed points of Symmetric Boolean SDSs and SyDSs *exactly* is **#P**-complete, and approximating the number of FPs to within, say, $2^{|V|^{1-\epsilon}}$ is **NP**-hard, for any $\epsilon > 0$. Similarly, counting *exactly* all TCs or all GEs of a Symmetric Boolean S(y)DS is **#P**-complete, as well. The complexity of counting GEs and TCs in symmetric S(y)DSs *approximately,* however, cannot be deduced from our constructions herewith and, to the best of our knowledge, is still open.

4.3 Counting in Symmetric Boolean SDSs and SyDSs with Bounded Node Degrees

The constructions of symmetric Boolean SDSs and SyDSs in the previous subsection include a "central control" node, y, that has an unbounded degree. Also, the clause nodes C_j in Theorems 4.2 and 4.3 are

[3]For some deeper reasons behind cycle-freeness of the sequential graph automata similar to our SDS $\mathcal{S}$', see, e.g., [20, 45].

forming a clique, thus also being of unbounded degree. We now transform the SyDS and SDS constructions from the previous subsection so that the node y is eliminated altogether, and so that each clause node C_j has only $O(1)$ neighbors. This reduction in the maximum node degree allowed will be done at the expense of doubling the number of the clause nodes, so that the resulting symmetric Boolean S(y)DS has $n + 2m$ nodes in total, where, as before, n is the number of variables and m is the number of clauses in the original 3CNF Boolean formula.

Indeed, let's eliminate the node y in the constructions in Theorems 4.2 and 4.3, and, instead, for each clause node C_j, introduce its "clone" clause node, C^c_j. Let's now connect each node C_j to its clone C^c_j and also to the clone of the successor clause node, $C^c_{j+1\ (mod\ m)}$. We also delete all the edges among the original clause nodes C_j. Thus, each original clause node C_j will now have *exactly five* neighbors: the three variable nodes, x_{j_1}, x_{j_2} and x_{j_3}, and the two clause clone nodes, C^c_j and $C^c_{j+1\ (mod\ m)}$.

We will also assume that the 3CNF SAT instance is from a restricted class of *monotone* 3CNF formulae where each variable x_i appears in at most five clauses. This restriction does not affect the **#P**-completeness of the underlying counting problem. In fact, counting satisfying truth assignments of the *positive* (also called *monotone*) 2CNF formulae, abbreviated as *MON-2CNF-SAT,* is **#P**-complete even when each variable appears in at most five clauses [47]. Each of these MON-2CNF formulae can be converted into a special case of the *MAJORITY-MON-3CNF* formulae, in which a clause is satisfied if and only if *at least two out of three* unnegated variables (that is, their *majority*) appearing in this clause are true. Namely, let's introduce a fresh Boolean variable z, and expand each monotone clause $(x_{j_1} + x_{j_2})$ in the MON-2CNF formula into $(x_{j_1} + x_{j_2} + z)$, as well as add a new clause, $(z + z + z)$. Clearly, the satisfying assignments of the original MON-2CNF formula are mapped in a one-to-one manner to the satisfying truth assignments of the resulting MAJORITY-MON-3CNF formula, while the number of appearances of each of the "old" variables x_i has remained the same. Now, since only the new variable z occurs in a number of clauses that is not bounded by $O(1)$, this can be "fixed" by replacing a single variable z with a sequence of distinct new variables $z_1, z_2, ..., z_m$, by modifying each $C_j = (x_{j_1} + x_{j_2})$ from the original MON-2CNF into $C_j = (x_{j_1} + x_{j_2} + z_j)$, and by adding m new clauses, $C'_j = (z_j + z_j + z_j)$, to the resulting MAJORITY-MON-3CNF formula.

Since this, restricted type of the counting problem #MAJORITY-MON-3CNF is **#P**-complete, even when no variable occurs in more than five different clauses, and since the general #MAJORITY-MON-3CNF is clearly in the class **#P**, we conclude that the general problem of counting the satisfying assignments to a monotone 3CNF formula according to the *MAJORITY* rule is **#P**-complete *even when no variable appears in more than five different clauses,* as well.

We now turn to the construction of a *bounded-degree symmetric Boolean SDS or SyDS* from an instance of the MAJORITY-MON-3CNF SAT.

Let the variable nodes in the S(y)DS constructed from such a 3CNF formula with restricted number of appearances of each variable update according to the Boolean *AND* rule on (at most) six inputs. Each variable node, as before, is connected to those, and only those, clause nodes such that the corresponding variable in the MAJORITY-MON-3CNF formula appears in the corresponding clause. Hence, each of these variable nodes will have at most five neighbors.

Recall that each clause node C_j is connected to the two clause "clone" nodes C^c_j and $C^c_{j+1\ (mod\ m)}$. Since each of the "original" clause nodes has exactly five neighbors in total, the local update rule at each such node needs to be a symmetric Boolean function of six inputs. So, we let each node C_j update its state according to the "AT LEAST FIVE OUT OF SIX" rule.

Furthermore, let's also connect all the clone nodes C^c_j into a ring, so that the only neighbors of C^c_j (beside C_j and $C_{j-1\ (mod\ m)}$) are $C^c_{j-1\ (mod\ m)}$ and $C^c_{j+1\ (mod\ m)}$. Finally, let each of the clone clause nodes C^c_j update according to the Boolean AND function of its five inputs (the states of its four neighbors plus the current state of itself).

If a single clone node $C^c_{j_\star}$ at any point updates to 0, this node will eventually force all the remaining clause clone nodes C^c_j, and consequently also all the original clause nodes C_j, to become 0s, as well. Similarly, if any of the original clause nodes $C_{j_\star}$ ever evaluates to 0, this will first cause its clone, $C^c_{j_\star}$, to evaluate to 0 (and stay at 0 thereafter), and that will, in turn, subsequently force all the other clause clone nodes to become 0s. Since each of the original clause nodes C_j will then have at least two neighbors stuck in the state 0, that will also ensure that eventually $C_j = 0$ for all $j = 1, ..., m$. Therefore, if any of the clauses in the original formula is not satisfied, the corresponding S(y)DS will converge to the sink fixed point 0^{n+m+1}.

In contrast, if initially all $C^c_j = C_j = 1$, and the original Boolean formula is satisfied, then all the clause clone nodes will remain at 1, and the corresponding global S(y)DS configuration is a fixed point corresponding to a satisfying truth assignment of the original Boolean formula.

To summarize, the following strengthening of the results in the previous subsection holds:

Theorem 4.4 *The problem of counting the fixed points of a Symmetric Boolean SDS or SyDS is* **#P***-complete, even when each node in the underlying graph of the S(y)DS is of a degree* $d_i \leq 5$, *and the nodes of the S(y)DS use only two different symmetric update rules.*

In fact, the upper bound on the maximum degree of any node in a symmetric Boolean S(y)DS can be further reduced: the problem of exactly counting FPs in such SDSs and SyDSs remains **#P**-complete even when each node degree is required not to exceed 4 (instead of 5 as in the theorem above). A weakly parsimonious reduction directly from MON-2CNF SAT can be used to establish that result. We leave out the details due to space constraints. Insofar as the symmetric SDSs with the maximum node degrees no greater than 3 are concerned, counting FPs in their configuration spaces remains an open problem; our attempts to obtain the **#P**-completeness by an appropriate modification of the constructions used in this paper have turned out to be futile. We still suspect that this hardness nonetheless holds, but perhaps a different approach is needed. We leave resolving this conjecture about the symmetric SDSs whose underlying graphs have the maximum node degrees not exceeding 3 for the future work.

5 Conclusions and Future Directions

Large-scale distributed computational and communication systems are often characterized by the property that, while the individual components may be relatively simple and their behavior well-understood, due to the interaction among these components and the interdependencies among different processes taking place at different components, the overall system behavior can become extremely complex and hard to predict. This, in particular, makes the design of reliable such systems challenging. Equally importantly, the verification of various properties of interest, as well as the forecast of the likely future behavior patterns, become difficult task.

As a step towards understanding the kind of emerging complexity in such large-scale decentralized infrastructures, as well as towards developing a general theory of their computer simulation, we have adopted a discrete dynamical systems approach to abstracting and then formally analyzing these distributed infrastructures. The primary methodological approach to studying properties of a dynamical system is to study its behavior, i.e., its *configuration space.* In this paper, we consider certain types of graph automata as appropriate abstract discrete-time, discrete-state dynamical systems. We specifically focus on the problem of counting how many "fixed point" configurations such dynamical systems have in their configuration spaces, when each of their nodes has only two distinct states, and updates according to some simple Boolean function of the states of its neighboring nodes. Concretely, we establish that counting these fixed points in Sequential and Synchronous Dynamical Systems is **#P**-complete, even when the following constraints on the structure of an SDS or SyDS *simultaneously* hold:

- each local update rule is required to be a *symmetric* Boolean function;
- the underlying graph of this SDS or SyDS is *sparse* in a very strong sense: all the node degrees are *uniformly bounded* by a small constant; and
- the nodes of this SDS or SyDS use only two different symmetric Boolean update rules.

The counting problems and their complexity addressed herewith are, by themselves, perhaps of only a limited practical interest. However, when the results in this paper are considered together with what has been recently shown about the computational complexity of the existence of fixed points and gardens of Eden [7], as well as of the reachability of these fixed points [6], a much more complete picture about the complexity of various restricted models of Boolean SDSs and SyDSs, and their fundamental configuration space properties, is obtained. For example, combining together the hardness of *Fixed Point Reachability* with the hardness of *Counting Fixed Points* implies that, for the actual decentralized systems that can be abstracted via an appropriate type of SDS or SyDS or a similar graph automaton, global long-term prediction is, in general, intractable. More specifically, given a starting global configuration of such a system, we in general cannot efficiently predict either whether the actual system's behavior is going to eventually stabilize and reach a steady state, or how long is it going to take before it settles into this steady state, or what exactly steady state (among possibly exponentially many) is the system going to settle in.

As for our ongoing and future work, there are several directions along which we can strengthen the results presented in this paper, and extend them to similar results about counting other types of configurations and other emerging structures in discrete dynamical systems such as SDSs, Hopfield Networks or classical Cellular Automata. One concrete open problem is the complexity of counting FPs in symmetric Boolean SDSs when no node degree exceeds 3. We have been also studying the complexity of counting in various restricted types of Boolean SDSs when it comes to the *backward dynamics* problems, such as those related to the number of predecessors or the number of all ancestors of an arbitrary configuration. We hope to report some new results in that context very soon.

Another important issue, not directly addressed in this work, is that of *approximately counting* GEs and all transient configurations in symmetric Boolean SDSs. The issue certainly cannot be resolved based on the constructions we used to establish the computational intractability of counting FPs (and *exactly* counting GEs), since approximately estimating the number of GEs and TCs in the constructed SDSs and SyDSs is shown to be easy. In general, the SDSs and CA with the simple threshold rules, such as *Boolean AND* or *Boolean OR* or *MAJORITY,* tend to have a relatively large number of GEs, and also most of their configurations are typically TCs. However, this need not hold for arbitrary symmetric SDSs and SyDSs, nor does it need imply that approximating #GE and #TC is always necessarily tractable. We leave further discussion about approximately counting GEs, TCs and other types of structures for the future work.

In summary, the formal discrete dynamical systems concepts, paradigms and methodology provide a rich arsenal with which to tackle, in an abstract yet mathematically elegant setting, many fundamental problems in large-scale distributed computational and communication infrastructures and multi-agent systems. Our results in this paper are an example of how the paradigms from nonlinear complex dynamics, coupled with the computational complexity tools, can provide insights into which aspects of the large-scale distributed systems' global behaviors can be reasonably expected to be feasible to predict in practice, and which ones cannot. In particular, it then follows that, in case of the latter, and under the usual assumptions in computational complexity theory, there is no "short-cut" to the step-by-step computer simulation.

Acknowledgments: The author expresses his sincere gratitude to Gul Agha (University of Illinois), Harry Hunt (SUNY-Albany) and Madhav Marathe (Los Alamos National Laboratory) for many useful discussions and suggestions on various matters related to the problems studied in this paper.

References

[1] S. Amoroso and Y. Patt. "Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures", *J. of Computer and System Sciences (JCSS),* vol. 6, pp. 448-464, 1972

[2] C. Barrett, B. Bush, S. Kopp, H. Mortveit and C. Reidys. "Sequential Dynamical Systems and Applications to Simulations", Technical Report, Los Alamos National Laboratory, Sept. 1999

[3] C. Barrett, M. Marathe, H. Mortveit, C. Reidys, J. Smith, S.S. Ravi. "AdhopNET: Advanced Simulation-based Analysis of Ad-Hoc Networks", Los Alamos Unclassifi ed Internal Report, 2000.

[4] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns. "Dichotomy Results for Sequential Dynamical Systems", Los Alamos National Laboratory Report, LA-UR-00-5984, 2000

[5] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns. "Predecessor and Permutation Existence Problems for Sequential Dynamical Systems", Los Alamos National Laboratory Report, LA-UR-01-668, 2001

[6] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns. "Reachability problems for sequential dynamical systems with threshold functions", *Theoretical Comp. Sci.,* vol. 295, issues 1-3, pp. 41-64, Feb. 2003

[7] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, P. T. Tosic. "Gardens of Eden and Fixed Points in Sequential Dynamical Systems", DMTCS Proc. AA (DM-CCG), pp. 95-110, 2001

[8] C. Barrett and C. Reidys. "Elements of a theory of computer simulation I: sequential CA over random graphs" *Applied Mathematics and Computation*, vol. 98, pp. 241-259, 1999

[9] C. Barrett, H. Mortveit, and C. Reidys. "Elements of a theory of simulation II: sequential dynamical systems" *Applied Math. and Comput.*, vol 107/2-3, pp. 121-136, 2000

[10] C. Barrett, H. Mortveit and C. Reidys. "Elements of a theory of computer simulation III: equivalence of SDS", *Applied Math. and Comput.,* vol. 122, pp.325-340, 2001

[11] R.J. Beckman, et. al. "TRANSIMS: Case Study", Dallas Ft-Worth. Los Alamos National Laboratory, LA UR 97-4502, 1999

[12] S. Buss, C. Papadimitriou and J. Tsitsiklis. "On the predictability of coupled automata: An allegory about Chaos" *Complex Systems*, 1(5), pp. 525-539, 1991. Preliminary version appeared in *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Oct. 1990

[13] K. Cullik, J. Pachl and S. Yu. "On the limit sets of cellular automata", *SIAM J. Computing*, 18(4), pp. 831-842, 1989.

[14] K. Culik, L. P. Hurd, S. Yu. "Computation theoretic aspects of cellular automata", *Physica D*, vol. 45 (1-3), pp. 357-378, 1990

[15] B. Durand. "Inversion of 2D cellular automata: some complexity results" *Theoretical Computer Science*, 134 (2) , pp. 387-401, November 1994

[16] B. Durand. "A random NP-complete problem for inversion of 2D cellular automata", *Theoretical Computer Science*, 148 (1) , pp. 19-32, August 1995

[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. Garzon. ”Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks”, Springer, 1995

[19] M. Gouda, C. Chang. “Proving Liveness for Networks of Communicating Finite State Machines.” *ACM Transactions on Programming Languages and Systems (TOPLAS)* vol. 8 (1), pp. 154-182, 1986

[20] E. Goles, S. Martinez. “Neural and Automata Networks: Dynamical behavior and Applications”, Math. and Its Applications series (vol. 58), Kluwer, 1990

[21] E. Goles, S. Martinez (eds.) “Cellular Automata and Complex Systems”, Nonlinear Phenomena and Complex Systems series, Kluwer, 1999

[22] F. Green. “NP-Complete Problems in Cellular Automata”, *Complex Systems*, vol. 1, No. 3, pp. 453-474, 1987

[23] H. Gutowitz (Editor). *Cellular Automata: Theory and Experiment* North Holland, 1989

[24] B. Huberman and N. Glance. “Evolutionary games and computer simulations” *Proc. National Academy of Sciences*, 1999

[25] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns. “Designing Approximation Schemes using L-reductions” *Proc. 14th Conference on Foundations of Software Technology and Theoretical Computer Science* (FST&TCS'94), Madras, India, Dec. 1994, pp. 342-353

[26] L.P. Hurd. “On Invertible cellular automata” *Complex Systems*, vol. 1(1), pp. 69-80, 1987

[27] T. E. Ingerson and R. L. Buvel. “Structure in asynchronous cellular automata”, *Physica D: Nonlinear Phenomena*, vol. 10 (1-2), pp. 59-68, Jan. 1984

[28] R. Laubenbacher and B. Pareigis. “Finite Dynamical Systems” Technical report, Department of Mathematical Sciences, New Mexico State University, Las Cruces, 2000.

[29] B. Martin. A Geometrical Hierarchy of Graphs via Cellular Automata. Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, Aug. 1998.

[30] M.V. Marathe, H.B. Hunt III, D.J. Rosenkrantz and R.E. Stearns. “Theory of periodically specified problems: Complexity and Approximability” *Proc. 13th IEEE Conference on Computational Complexity*, Buffalo, NY, June, 1998.

[31] C. Moore. “Generalized shifts: unpredictability and undecidability in Dynamical Systems” *Nonlinearity*, 4, pp. 199-230, 1991.

[32] H. Mortveit, C. Reidys. “Discrete, sequential dynamical systems”, *Discrete Mathematics*, vol. 226, pp. 281-295, 2001

[33] John von Neumann. “Theory of Self-Reproducing Automata”, edited and completed by A. W. Burks, Univ. of Illinois Press, Urbana, 1966

[34] C. Nichitiu and E. Remila. “Simulations of Graph Automata” Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, Aug. 1998.

[35] C. Papadimitriou. *“Computational Complexity”* , Addison-Wesley, Reading, Massachusetts, 1994.

[36] C.M. Reidys. “On Acyclic Orientations & Sequential Dynamical Systems”, Los Alamos National Laboratory Report, LA-UR-01-598, 2001

[37] Z. Roka. “One-way cellular automata on Cayley graphs”, *Theoretical Computer Science*, 132(1-2), pp. 259-290, September 1994

[38] D. Roth. "On the Hardness of Approximate Reasoning", *Artificial Intelligence*, vol. 82, pp. 273-302, 1996

[39] P. Sarkar. "A Brief History of Cellular Automata", *ACM Comp. Surveys*, vol. 32 (1), March 2000

[40] T. Schaefer. "The Complexity of Satisfiability Problems" *Proc. 10th ACM Symposium on Theory of Computing* (STOC'78), 1978, pp. 216-226.

[41] K. Sutner. "On the computational complexity of finite cellular automata" *Journal of Computer and System Sciences*, 50(1), pp. 87-97, February 1995

[42] K. Sutner. "Computation theory of cellular automata", *MFCS'98 Satellite Workshop on Cellular Automata* Bruno, Czech Republic, Aug. 1998

[43] C. Schittenkopf, G. Deco and W. Brauer. "Finite automata-models for the investigation of dynamical systems" *Information Processing Letters*, 63(3), pp. 137-141, August 1997

[44] S. Toda. "PP is as Hard as the Polynomial-Time Hierarchy", *SIAM J. Computing*, vol. 20 (5), pp. 865-877, 1991

[45] P. Tosic, G. Agha. "Concurrency vs. Sequential Interleavings in 1-D Threshold Cellular Automata" APDCM Workshop within IPDPS'04, Santa Fe, New Mexico, USA, April 2004 (in Proc. IEEE-IPDPS '04)

[46] P. Tosic, G. Agha. "Characterizing Configuration Spaces of Simple Threshold Cellular Automata", ACRI'04, Amsterdam, The Netherlands, Oct. 25-28, 2004, Springer-Verlag LNCS series, vol. 3305

[47] S. Vadhan. "The Complexity of Counting in Sparse, Regular and Planar Graphs", *SIAM J. Computing*, vol. 31, # 2, pp. 398-427, 2001

[48] L. Valiant. "The Complexity of Computing the Permanent", *Theoretical Comp. Sci.*, vol. 8, pp. 189-201, 1979

[49] L. Valiant. "The Complexity of Enumeration and Reliability Problems", *SIAM J. Computing*, vol. 8 (3), 1979

[50] S. Wolfram. "Computation theory of cellular automata", *Commun. Math. Physics*, vol. 96, 1984

[51] S. Wolfram. "Twenty problems in the theory of cellular automata", *Physica Scripta*, T9, pp. 170-183, 1985

[52] S. Wolfram. "Theory and applications of cellular automata", World Scientific, 1986

[53] S. Wolfram (ed.). "Cellular Automata and Complexity (collected papers)", Addison-Wesley, 1994

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'