On Computational Complexity of Counting Fixed Points in Symmetric Boolean Graph Automata

PREDRAG T. TOŠIĆ and GUL A. AGHA
Open Systems Laboratory (http://www-osl.cs.uiuc.edu/),
Department of Computer Science, University of Illinois at Urbana-Champaign
Thomas Siebel Center for Computer Science,
201 N. Goodwin Avenue, Urbana, IL 61801, USA
{p-tosic, agha}@cs.uiuc.edu

Abstract. We study computational complexity of *counting* the *fixed point configurations (FPs)* in certain classes of graph automata viewed as 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 the problems of counting exactly the *garden of Eden configurations (GEs)*, as well as all *transient configurations*, are in general intractable, as well. Moreover, exactly enumerating FPs or GEs remains hard 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 every 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, **#P**-completeness

1 Introduction and Motivation

We study certain classes of *graph automata* that can be used as an abstraction 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, 8, 9, 22, 31, 32]), the general approach has been to study mathematical and computational *configuration space properties* of such graph automata: what are the possible *global behaviors* of the entire system, given the simple local behaviors of its components, and the interaction pattern among these components.

We specifically focus in this paper 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 intractable, as well.

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 summarizes our technical results, and reviews some literature 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 $F_{\mathcal{S}}$ to denote the *global map* of $\mathcal{S}$, obtained by appropriately composing together all the local update rules f_i, $i = 1, ..., n$.
3. Finally, Π is a permutation of $V = \{1, 2, \ldots, n\}$ specifying the sequential ordering in which the nodes update their states using their local transition functions. Alternatively, Π can be envisioned as a total order on the set of nodes. In particular, $F_{\mathcal{S}} = (f_{\Pi^{-1}(1)}, f_{\Pi^{-1}(2)}, \ldots, f_{\Pi^{-1}(n)})$.

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, 8, 9]. 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 $\Sigma_{j \in N(i)} x_j$ (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.

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 difference is that, while in the classical CA all nodes update according to the same rule, in an SyDS different nodes, in general, may 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 $\mathcal{C} : V \rightarrow \mathcal{D}$.

The function computed by an S(y)DS $\mathcal{S}$, 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 $F_{\mathcal{S}} : \mathcal{D}^n \rightarrow \mathcal{D}^n$ is a total function on the set of global configurations. This function therefore defines the dynamics of $\mathcal{S}$. 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 S(y)DS $\mathcal{S}$ 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 $\mathcal{S}$, each transition step will also take polynomial time in the size of the S(y)DS'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 or SyDS 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 S(y)DS 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 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. Similarly, $\mathcal{C}'$ 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 configuration is a special case of an ancestor.

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

Definition 2. 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 3. 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.

Definition 4. 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.

Definition 5. 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.

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. By *the size of a truth table* we shall throughout the paper mean, for simplicity, just the number of rows in this table. Thus, for a node v_i of degree d_i, the size of a truth table specifying an arbitrary Boolean function is $O(2^{d_i})$, and actually, for any (sufficiently big) positive integer d_i, most Boolean functions on $d_i + 1$ inputs cannot be encoded substantially more succinctly than via a truth table of size $\Theta(2^{d_i})$. In contrast, the size of an optimally succinct truth table fully specifying an arbitrary *symmetric* Boolean function is only $O(d_i)$.

Another, more common way of specifying the local transition functions is via *Boolean formulae*. We shall assume that f_i of *non-symmetric* SDSs and SyDSs considered in the sequel are indeed given as (reasonably succinct[2]) Boolean formulae of appropriately restricted kinds. It follows from the discussion above that, for symmetric Boolean update rules, the exact way these update rules are encoded in an S(y)DS is inconsequential, as long as this encoding is reasonably succinct. We shall also assume throughout

[2] By *reasonably succinct* we mean, formulae whose sizes are not artificially blown up by, e.g., repeating the same clause(s) over and over again.

that evaluating any local transition function f_i, 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 any other finite domain) SDSs and SyDSs is **#P**-complete;
- this hardness result still holds when the node update rules of these SDSs and SyDSs 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 node degree in the underlying graph is a small constant.

3.1 Related work

SDSs and SyDSs investigated in this paper are closely related to the *graph automata (GA)* models studied in [21, 23] and the one-way cellular automata studied by Roka in [25]. In fact, the general finite-domain SyDSs exactly correspond to the graph automata of Nichitiu and Remila as defined in [23].

Barrett, Mortveit and Reidys [8, 9, 22] and Laubenbacher and Pareigis [20] 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., [9, 20, 22]). Computational complexity of the reachability-related problems in the context of, among other restricted types, symmetric Boolean SDSs and SyDSs is investigated in [6]. We show in this paper that the problems of exact and approximate *counting* of stable configurations (FPs) in symmetric Boolean SDSs and SyDSs, under the usual assumptions in computational complexity theory, are both intractable.

4 Counting Fixed Points of Boolean SDSs and SyDSs

The results in this section constitute an extension of the work presented in [7] and [6]. In [7], 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, a natural further course of inquiry

about the fundamental SDS and SyDS configuration space properties is to determine how hard it is to count *how many* FPs, GEs, and/or other configurations of interest an S(y)DS 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 [13, 24]. 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* that this algorithm makes that lead to acceptance equals the number of solutions of I_{Ψ}, and, in addition, it is required that the longest of these nondeterministic computations of the algorithm be polynomially bounded in the size of the description of I_{Ψ}. For an alternative but equivalent definition of class **#P** in terms of *polynomially balanced relations*, we refer the reader to [24].

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) *every* other problem in **#P** is efficiently reducible to Ψ. Thus, if we could solve *any* particular **#P**-complete problem in deterministic polynomial time, then *all* problems in class **#P** would be solvable in deterministic polynomial time. For more on the class **#P** we refer the interested reader to Chapter 18 of [24], and references therein.

In order to prove the intractability of counting FPs of Boolean SDSs and SyDSs, not any *polynomial time* reduction from a known **#P**-complete problem suffices. What is required is a kind of efficient reduction that *preserves the number of solutions.* We define this special kind of efficient reductions next:

Definition 6. Given two decision problems Π and $\Pi^{'}$, a PARSIMONIOUS REDUCTION from Π to $\Pi^{'}$ 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 $\Pi^{'}$ 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_I in the previous definition can be efficiently recovered from $n_{g(I)}$.

Definition 7. Given two decision problems Π and $\Pi^{'}$, a WEAKLY PARSIMONIOUS REDUCTION from Π to $\Pi^{'}$ is a polynomial-time transformation g such that, if an instance I of Π has n_I solutions, and the corresponding instance $g(I)$ of $\Pi^{'}$ has $n_{g(I)}$ solutions, then n_I can be computed from $n_{g(I)}$ in polynomial time.

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

Proposition 8. [24] *Given two decision problems Π and $\Pi^{'}$, if the corresponding counting problem $\#\Pi$ is* **#P***-hard and if there exists a weakly parsimonious reduction from Π to $\Pi^{'}$, 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-ONE-IN-THREE-SAT (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 *Satisfiability* [13, 24] that we shall use in the sequel:

Definition 9. EXACTLY-ONE-IN-THREE-SATISFIABILITY (or **E3SAT** for short), is a version of 3CNF-SAT [13] such that, first, 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 the counting problem #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 SDS $\mathcal{S}$ in this subsection will be "nearly symmetric"; we will modify our construction to a fully symmetric Boolean SDS (or 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, $1 \leq i \leq n$, and for each clause C_j, $1 \leq j \leq m$. 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 of 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, $1 \leq j \leq m$. 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, $1 \leq j \leq m$.

- 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 configuration 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 10. *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.

Detailed proof that establishes that the given reduction from #PE3SAT to #FP-SDS is, indeed, weakly parsimonious can be found in our electronic technical report [30].

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, denoted #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 [22] for a proof).

Theorem 11. *The problem of counting fixed points of a symmetric Boolean Synchronous Dynamical System, abbreviated as* #FP-SYM-SYDS*, is* **#P***-complete.*

Proof sketch: 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:
$x_i^{t+1} = x_i^t \wedge (\wedge_{j(i)} C_{j(i)}^t)$;
$C_j^{t+1} = \text{ALL-BUT-ONE}\ \{x_{j_1}^t, x_{j_2}^t, x_{j_3}^t, C_1^t, ..., C_m^t, y^t\}$;
$y^{t+1} = y^t \wedge (\wedge_{j=1}^m C_j^t)$,
where the symmetric Boolean function ALL-BUT-ONE satisfies $f(z_1, ..., z_q) = 1$ if and only if *exactly* one of the inputs z_l $(1 \leq l \leq q)$ is 0, and all the rest are 1.

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. That is, the given reduction from #PE2-IN-3SAT to #FP-SYM-SYDS is, indeed, weakly parsimonious. We omit details due to space constraints, and refer the interested reader to [30].

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

Proof sketch: 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,
  $y^{t+1} = y^t \wedge (\wedge_{j=1}^m C_j^t)$,
  $C_j^{t+1} = \text{ALL-BUT-ONE}\ \{y^{t+1}, C_1^{t+1}, ..., C_{j-1}^{t+1}, C_j^t, C_{j+1}^t, ..., C_m^t, x_{j_1}^t, x_{j_2}^t, x_{j_3}^t\}$,
  and, for any i such that $1 \leq i \leq n$,
  $x_i^{t+1} = x_i^t \wedge (\wedge_{j(i)} C_{j(i)}^{t+1})$,
  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.

Due to the space constraints, we will only summarize what the configuration space of SDS $\mathcal{S}$' looks like. Since there are $n + m + 1$ nodes, there are 2^{n+m+1} global configurations in total. Among these, by virtue of Theorem 11 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. The configuration 0^{n+m+1} is *the sink* for $\mathcal{S}$', since all transient chains eventually converge to 0^{n+m+1}. All the remaining configurations are transient states, and, in particular, $\mathcal{S}$' does not have any temporal cycles. Furthermore, it can be readily shown that all 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[3].

In summary, enumerating the fixed points of Symmetric Boolean SDSs and SyDSs *exactly* is **#P**-complete; moreover, it follows from the results in [26] that *approximating* the number of FPs to within $2^{|V|^{1-\epsilon}}$ is **NP**-hard, for any $\epsilon > 0$. Similarly, it can be shown from the given construction of $\mathcal{S}$' that 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 in this subsection and, to the best of our knowledge, is still open.

4.3 Counting in Symmetric Boolean S(y)DSs 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 11 and 12 are forming a clique, thus also being of unbounded degrees. 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 11 and 12, and, instead, for each clause node C_j, introduce its *clone,* C_j^c. Let's now connect each node C_j to its clone C_j^c 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 "cloned" clause nodes, C_j^c 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 *monotone* 2CNF formulae, abbreviated as MON-2CNF-SAT, is **#P**-complete even when each variable appears in at most five clauses [33]. Each of these MON-2CNF formulae can be readily 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.

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

[3] For a much more detailed argument, and a rigorous justification of all other assertions that we make in this proof sketch, we refer the reader to the online publication [30].

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 formula.

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 *cloned clause nodes* C_j^c and $C_{j+1 \ (mod \ m)}^c$. 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 cloned clause nodes C_j^c into a ring, so that the only neighbors of C_j^c (beside C_j and $C_{j-1 \ (mod \ m)}$) are $C_{j-1 \ (mod \ m)}^c$ and $C_{j+1 \ (mod \ m)}^c$. Finally, let each of the cloned clause nodes C_j^c 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 cloned node $C_{j_\star}^c$ at any time step updates to 0, this node will eventually force all the remaining cloned clause nodes C_j^c, 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, $C_{j_\star}^c$, to evaluate to 0 (and stay at 0 thereafter), and that will, in turn, subsequently force all the other cloned clause 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 quickly converge to the sink fixed point 0^{n+2m}.

In contrast, if initially all $C_j^c = C_j = 1$, and the original Boolean formula is satisfied, then all the cloned clause 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 13. *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 such an S(y)DS is of a degree $d_i \leq 5$, and the nodes of that 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's 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 S(y)DSs with the maximum node degree not 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 and SyDSs whose underlying graphs have node degrees not exceeding 3 for the future work.

5 Conclusions and Future Directions

We study in this paper certain types of graph automata viewed as 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; and
- 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.

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 and SyDSs when no node degree exceeds 3. We have been also studying the complexity of counting in various restricted types of Boolean S(y)DSs 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 will report new results in that context elsewhere.

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 about 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 a step-by-step computer simulation.

Acknowledgments: The first author expresses his sincere gratitude to Harry Hunt (SUNY-Albany), Michael Loui (University of Illinois) and Madhav Marathe (Los Alamos National Laboratory) for many useful discussions and suggestions on various matters related to this paper. This work was supported in part by the ONR MURI Grant, contract number N00014-02-1-0715.

References

1. S. Arora, Y. Rabani and U. Vazirani. "Simulating quadratic dynamical systems is **PSPACE**-complete," *Proc 26th. Annual ACM Symposium on the Theory of Computing* (STOC), pp. 459-467, Montreal, Canada, May 1994
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, September 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 Unclassified 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, February 2003
7. C. L. Barrett, H. B. Hunt, 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", Proc. AA DM-CCG, *Discrete Math. & Theoretical Comp. Sci.*, pp. 95-110, 2001
8. C. Barrett, H. Mortveit, and C. Reidys. "Elements of a theory of simulation II: sequential dynamical systems" *Applied Math. & Comput.*, vol 107/2-3, pp. 121-136, 2000
9. C. Barrett, H. Mortveit and C. Reidys. "Elements of a theory of computer simulation III: equivalence of SDS", *Applied Math. & Comput.*, vol. 122, pp. 325-340, 2001
10. C. Barrett and C. Reidys. "Elements of a theory of computer simulation I: sequential CA over random graphs" *Applied Math. & Comput.*, vol. 98, pp. 241-259, 1999
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*, vol. 1(5), pp. 525-539, 1991
13. 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, California, 1979
14. M. Garzon. *"Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks"*, Springer, 1995
15. E. Goles, S. Martinez. *"Neural and Automata Networks: Dynamical behavior and Applications"*, Math. and Its Applications series (vol. 58), Kluwer, 1990

16. E. Goles, S. Martinez (eds.) *"Cellular Automata and Complex Systems"*, Nonlinear Phenomena and Complex Systems series, Kluwer, 1999
17. F. Green. "NP-Complete Problems in Cellular Automata", *Complex Systems*, vol. 1, No. 3, pp. 453-474, 1987
18. B. Huberman, N. Glance. "Evolutionary games and computer simulations", *Proc. Nat'l Academy of Sciences*, 1999
19. 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
20. R. Laubenbacher and B. Pareigis. "Finite Dynamical Systems", Technical report, Dept. of Mathematical Sciences, N. Mexico State Univ., Las Cruces, 2000
21. B. Martin. "A Geometrical Hierarchy of Graphs via Cellular Automata", Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, August 1998
22. H. Mortveit, C. Reidys. "Discrete, sequential dynamical systems", *Discrete Mathematics*, vol. 226, pp. 281-295, 2001
23. C. Nichitiu and E. Remila. "Simulations of Graph Automata", Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, August 1998
24. C. Papadimitriou. *"Computational Complexity"* , Addison-Wesley, Reading, Massachusetts, 1994
25. Z. Roka. "One-way cellular automata on Cayley graphs", *Theoretical Computer Science*, 132(1-2), pp. 259-290, September 1994
26. D. Roth. "On the Hardness of Approximate Reasoning", *Artificial Intelligence*, vol. 82, pp. 273-302, 1996
27. K. Sutner. "On the computational complexity of finite cellular automata", *Journal of Computer and System Sciences*, vol. 50(1), pp. 87-97, February 1995
28. K. Sutner. "Computation theory of cellular automata", *MFCS'98 Satellite Workshop on Cellular Automata* Bruno, Czech Republic, August 1998
29. C. Schittenkopf, G. Deco and W. Brauer. "Finite automata-models for the investigation of dynamical systems" *Information Processing Letters*, vol. 63(3), pp. 137-141, August 1997
30. P. Tosic. "On Complexity of Counting Fixed Point Configurations in Certain Classes of Graph Automata", *Electronic Colloquium on Computational Complexity*, ECCC-TR05-051 (revision 1), April 2005
31. 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)
32. P. Tosic, G. Agha. "Characterizing Configuration Spaces of Simple Threshold Cellular Automata", *Proc. 6th 6th Int'l Conference on Cellular Automata for Research and Industry (ACRI'04)*, Amsterdam, The Netherlands, Oct. 25-28, 2004, Springer-Verlag LNCS series, vol. 3305
33. S. Vadhan. "The Complexity of Counting in Sparse, Regular and Planar Graphs", *SIAM J. Computing*, vol. 31 (2), pp. 398-427, 2001
34. L. Valiant. "The Complexity of Computing the Permanent", *Theoretical Comp. Sci.*, vol. 8, pp. 189-201, 1979
35. L. Valiant. "The Complexity of Enumeration and Reliability Problems", *SIAM J. Computing*, vol. 8 (3), pp. 410 - 421, 1979