Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 91 (2005)

Counting Fixed Points and Gardens of Eden of Sequential Dynamical Systems on Planar Bipartite Graphs

Predrag T. Tošić

Open Systems Laboratory (http://www-osl.cs.uiuc.edu/)
Department of Computer Science, University of Illinois at Urbana-Champaign
1334 Thomas Siebel Center for Computer Science,
201 N. Goodwin Avenue, Urbana, IL 61801, USA
p-tosic@cs.uiuc.edu

ABSTRACT

We study counting various types of configurations in certain classes of graph automata viewed as discrete dynamical systems. The graph automata models of our interest are *Sequential* and *Synchronous Dynamical Systems* (SDSs and SyDSs, respectively). These models have been proposed as a mathematical foundation for a theory of large-scale simulations of complex multi-agent systems. Our emphasis in this paper is on the computational complexity of counting the fixed point and the garden of Eden configurations in Boolean SDSs and SyDSs. We have shown in [47] that counting fixed points is, in general, computationally intractable. We show in the present report that this intractability still holds when both the underlying graphs and the node update rules of these SDSs and SyDSs are severely restricted. In particular, we prove that the problems of exactly counting fixed points, gardens of Eden and two other types of S(y)DS configurations are all **#P**-complete, even if the SDSs and SyDSs are defined over *planar bipartite graphs,* and each of their nodes updates its state according to a *monotone* update rule given as a Boolean formula. We thus add these formal discrete dynamical systems to the list of those problem domains for which counting the combinatorial structures of interest is intractable even when the related decision problems are known to be efficiently solvable.

Keywords: discrete dynamical systems, cellular and graph automata, computational complexity, enumeration problems, **#P**-completeness

1. Introduction

We study in this work 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 agent-based simulation of a broad variety of physical, computational, and socio-technical distributed infrastructures. In this and several other loosely related papers (e.g., [3, 4, 5, 6, 7, 8, 9, 10, 34, 45, 46]), the general approach has been to study mathematical and computational *configuration space properties* of such graph automata.

We specifically focus in this paper on determining *how many* configurations of certain kinds such graph automata have, and *how hard* are the computational problems of *counting* (or *enumerating*) those configurations. The main contributions of our work are as follows. We prove that *counting* the number of *fixed point* configurations or *"garden of Eden"* configurations in two related classes of graph automata, called *Sequential* and *Synchronous*

ISSN 1433-8092

Dynamical Systems, is computationally intractable, even when (i) such a discrete dynamical system is defined over a graph that is both *planar* and *bipartite,* and (ii) each node of a given graph automaton is required to update its state according to a *monotone* Boolean function.

The rest of the paper is organized as follows. Section 2 is devoted to the necessary preliminaries about the models that we study, namely, the sequential and synchronous dynamical systems. Section 3 outlines the main contributions of this paper, and briefly reviews the research literature related to our work. The original results are presented in Sections 4 and 5. We show in Section 4 that the problems of counting fixed points, gardens of Eden, and all transient configurations[a] in general Boolean Sequential and Synchronous Dynamical Systems are all **#P**-complete. We also state several other consequences of the weakly parsimonious reductions used to establish the intractability of these counting problems. In Section 5, we show that counting fixed points, gardens of Eden and all transient configurations remains hard, even when the underlying graph of an SDS or SyDS is required to be both planar and bipartite, and the node update rules are restricted to monotone Boolean functions given either as oracles or as reasonably succinct formulae. The paper summary is given in Section 6.

2. Preliminaries

In this section, we define the graph automata models studied in this paper, and their configuration space properties. *Sequential Dynamical Systems* (henceforth referred to as SDSs) are proposed in [9, 10, 11] as an abstract model for computer simulations. This model has been successfully applied in the development of large-scale socio-technical simulation systems such as the *TRANSIMS* project at the Los Alamos National Laboratory [12].

Definition 1 *A* Sequential Dynamical System (SDS) $\mathcal{S}$ *is a triple* (G, F, Π)*, whose components are as follows:*

1. $G(V, E)$ *is a connected undirected graph without multi-edges or self-loops.* $G = G_{\mathcal{S}}$ *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* $G(V, E) = G_{\mathcal{S}}$ *are enumerated 1, 2, ..., n.*

2. *Each node is characterized by its* state*. The state of a node* v_i*, denoted by* s_i*, takes on a value from some finite domain,* $\mathcal{D}$*. In this paper, we shall focus on* $\mathcal{D} = \{0, 1\}$*. We use* d_i *to denote the degree of the node* v_i*. Each node* v_i *has an associated* node update rule $f_i : \mathcal{D}^{d_i+1} \to \mathcal{D}$*, for* $1 \leq i \leq n$*. We also refer to* f_i *as the* local transition function*. The inputs to* f_i *are* s_i *and the current states of the neighbors of* v_i*. We use* $F = F_{\mathcal{S}}$ *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 order in which the nodes update their states using their local transition functions. Alternatively,* Π *can be envisioned as a total ordering on the set of nodes* V*. In particular, we can view the global map as a sequential composition of the local actions of each* f_i *on the respective state* s_i*, where the node states are updated according to the order* Π*; that is,* $F_{\mathcal{S}} = (f_{\Pi^{-1}(1)}, f_{\Pi^{-1}(2)}, \ldots, f_{\Pi^{-1}(n)})$*.*

[a] All these types of configurations will be defined in Section 2.

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.

Definition 2 *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)* [16, 18, 19, 21, 51, 52], 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 classical CA all nodes update according to the same rule, in an SyDS different nodes, in general, may use different update rules.

In the sequel, we shall often slightly abuse the notation, and not explicitly distinguish between an SDS's or SyDS's node itself, v_i, and its state, s_i. The intended meaning will be clear from the context. We shall discuss in more detail several possible interpretations of the global map $F = F_{\mathcal{S}}$, and how this map acts on the (global) configurations of an S(y)DS $\mathcal{S}$, in subsection 2.1.

Insofar as the restricted classes of Boolean update rules that we consider in this paper, the strongest result in Section 5 is in the context of Boolean S(y)DSs defined on the star graphs and with the node update rules restricted to *monotone functions.*

Definition 3 *Given two Boolean vectors, $X = \langle x_1, \ldots, x_n \rangle$ and $Y = \langle y_1, \ldots, y_n \rangle$, define a binary relation "$\preceq$" as follows: $X \preceq Y$ if $x_i \leq y_i$ for all i, $1 \leq i \leq n$. A Boolean function of n variables $f = f(x_1, ..., x_n)$ is* monotone *if $X \preceq Y$ implies that $f(X) \leq f(Y)$.*

S(y)DSs with finite domains are a generalization of S(y)DSs with Boolean domains. In the most general class of the S(y)DSs with arbitrary finite domains, there are no restrictions on the local transition functions. All the hardness results in this paper, explicitly shown for the Boolean S(y)DSs, clearly also hold for the more general, non-Boolean finite domains - as long as those domains' sizes are $O(1)$, which we will assume throughout.

2.1. SDS and SyDS Configuration Space Properties

Much of the previous work on SDSs and SyDSs has focused on computational complexity of determining their various configuration space properties. We briefly survey the relevant literature in subsection 3.1. This paper expands on that prior work by addressing computational complexity of some basic *counting* or *enumeration problems* about S(y)DSs and their configuration spaces.

Definition 4 *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 $\mathcal{C}: V \to \mathcal{D}$. Given a configuration $\mathcal{C}$, the state of a node $v \in V$ in $\mathcal{C}$ is denoted by $v(\mathcal{C})$; similarly, for a nonempty subset $W \subset V$ of nodes, $W(\mathcal{C})$ denotes the tuple of states of the nodes $v \in W$. We refer to $W(\mathcal{C})$ as a* **sub-configuration** *of $\mathcal{C}$.*

The global map $F_{\mathcal{S}}$ of an SDS or SyDS $\mathcal{S}$ specifies for each configuration $\mathcal{C}$ the next configuration $\mathcal{C}'$ reached by $\mathcal{S}$ after carrying out the updates of the node states

(where, in case of an SDS, these updates are carried out 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 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 $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}'$ 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 initial configuration of an SDS $\mathcal{S}$ will be often denoted by $\mathcal{C}^0$ in the sequel. Given an SDS $\mathcal{S}$ with an initial configuration $\mathcal{C}^0$, the configuration of $\mathcal{S}$ after t time steps is denoted by $\mathcal{C}^t$.

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 a *deterministic* dynamical system, each vertex in its configuration space has the out-degree of 1. Since the domain $\mathcal{D}$ of state values is assumed finite, and the number of nodes in any S(y)DS is always finite, the number of configurations in the configuration 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 5 *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. We use the notation $pred(\mathcal{C}') = \mathcal{C}$ to denote that $\mathcal{C}$ is a predecessor of $\mathcal{C}'$.*

Definition 6 *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. For any $t \geq 2$, a configuration $\mathcal{C}$ is a t-ancestor of a configuration $\mathcal{C}'$ if ${F_{\mathcal{S}}}^t(\mathcal{C}) = \mathcal{C}'$, i.e., if $\mathcal{S}$ starting from $\mathcal{C}$ reaches $\mathcal{C}'$ in* exactly *t transition steps.*

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

Definition 7 *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 8 *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 9 *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. Both CCs and FPs are *recurrent* configurations; in particular, every CC or FP is its own ancestor.

Definition 10 *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 their name suggests, transient configurations, unlike fixed points and 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.

Finally, we define the basic computational problems about the SDSs and SyDSs - more precisely, about their configuration space properties - that will be addressed in the sequel. The Fixed Point Existence problem (abbreviated as FPE) is the problem of determining, given an S(y)DS (or a class of S(y)DSs of a particular kind), does this S(y)DS (alternatively, does every S(y)DS in this class) have at least one fixed point configuration? The Garden of Eden Existence problem (or GEE for short) is defined analogously. The counting versions of these two decision problems ask, *how many* fixed points or gardens of Eden does a given S(y)DS have? These two counting problems will be abbreviated as #FP and #GE, respectively. The main focus of this paper is the computational complexity of #FP and #GE for the general, as well as certain restricted types of, Boolean (and other finite domain) SDSs and SyDSs.

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, the number of 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 $\Theta(2^{d_i})$.

Another, more common way of specifying the local transition functions is via *Boolean formulae*. Unless stated otherwise, 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 will discuss issues related to the choice of the local update rules' representation, and some implications of that choice, in much more detail in Section 5.

As mentioned earlier, we study in this paper the problems of *counting* the fixed point (FP) and the garden of Eden (GE) configurations of Boolean and finite domain SDSs and SyDSs. In particular, we prove the following results:

- the problems of counting FPs and GEs in the general Boolean (and, consequently, also in any other finite domain) SDSs and SyDSs are **#P**-complete;
- these two, as well as some other, related counting problems remain **#P**-complete even when the underlying graphs of Boolean S(y)DSs are required to be both *planar* and *bipartite;*
- these hardness results still hold when the node update rules of these S(y)DSs are restricted to *monotone Boolean functions;*

- moreover, the results remain valid even when *only two* different monotone update rules are used, and when the average node degree in the underlying graph is less than two.

3.1. Related Work

SDSs and SyDSs investigated in this paper are closely related to the *graph automata (GA)* models studied in [32, 36] and the one-way cellular automata studied by Roka in [39]. In fact, the general finite-domain SyDSs exactly correspond to the graph automata of Nichitiu and Remila as defined in [36]. The main difference between the graph automata in [32, 36] and the SDSs studied herein is the sequential ordering aspect. In particular, SDSs generalize the finite *sequential CA* [45, 46] in that they allow *arbitrary (finite) underlying graphs,* as opposed to restricting cellular spaces to the regular Cayley graphs only [16].

Over the recent years, several researchers have considered the issue of the *communication model* in discrete dynamical systems [15, 18, 24, 39, 45]. For instance, Huberman and Glance in [24] discuss and demonstrate experimentally that certain (simulations of) n-person evolutionary games exhibit very different, but probably more realistic, dynamics when the cells are updated sequentially as opposed to when they are updated synchronously in parallel. Comparison and contrast of the resulting dynamics in cellular automata and automata networks when the nodes update perfectly synchronously vs. when they update sequentially can be found, e.g., in [18, 45].

Barrett, Mortveit and Reidys [9, 10, 34, 38] and Laubenbacher and Pareigis [31] investigate the mathematical properties of sequential dynamical systems. Barrett et al. study the computational complexity of several configuration space problems for SDSs. These include REACHABILITY, PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE [6, 7]. Problems related to the existence of GARDEN OF EDEN and FIXED POINT configurations are studied in [8]. In particular, the basic computational complexity results for the problems of FP, GE and non-unique predecessor existence in various restricted classes of Boolean S(y)DSs, including those with *monotone Boolean* update rules, are proven in that paper. Our results in Section 4 of this paper can be viewed as a natural partial extension of the work in [8]: instead of the appropriate *decision problems* about fixed points and gardens of Eden in SDSs and SyDSs, we focus on studying the related *counting problems.*

Counting or *enumeration* problems naturally arise in many contexts, from approximate reasoning and Bayesian belief networks in AI (e.g., [40]), to network reliability and fault-tolerance (e.g., [48]), to statistical physics (e.g., [28, 30]). Counting problems often arise in the context of discrete dynamical systems and theoretical models for parallel and distributed computing, as well. Indeed, being able to efficiently solve certain counting problems is essential for the full understanding of the underlying dynamical system's qualitative behavior. Perhaps the most fundamental counting problem about any dynamical system is to determine (or estimate) the number of *attractors* of that system [2]; these attractors correspond to our *fixed points.* Counting the non-FP temporal cycles and cycle configurations, or the *garden of Eden (GE)* configurations, or the sizes of basins of attraction for different attractors, may also be of interest. For example, in *deterministic* discrete dynamical systems that are temporal cycle-free, such as the *asynchronous* (that is, sequential) discrete Hopfield networks with *linear threshold* update rules [22, 23], or the sequential CA with *monotone* threshold update rules [18, 45, 46], the number of fixed points equals the number of possible dynamical

evolutions of the system. In the context of Hopfield networks, the interpretation of the FP count is that it tells us *how many distinct patterns* a given Hopfield net can *memorize* [2, 22, 23]. In other, more general deterministic dynamical systems, the total number of possible evolutions equals the sum of the number of fixed points and the number of non-FP temporal cycles. Similarly, the number of GE configurations indicates what fraction of all possible configurations are *unreachable* [8].

4. Complexity of Counting in General Boolean SDSs and SyDSs

The results in this section constitute an extension of the work presented in [7] and [8]. In particular, computational complexity of *decision problems* about the fixed point and the garden of Eden configurations in S(y)DSs is studied in [8]. Once **NP**-completeness of these decision problems has been established, a natural next step is to determine the computational complexity of the related *counting problems* : how many FPs, GEs, or other configurations of interest an SDS or SyDS of a given type may have.

One would intuitively expect that, for instance, counting the FPs of an arbitrary Boolean SDS or SyDS is no easier than counting the satisfying truth assignments of an arbitrary instance of the SATISFIABILITY problem [14, 37]. The intuitive notion of computational hardness of counting problems is formalized via the definition of the class **#P** (read: "sharp-P" or "number-P").

Definition 11 *A counting problem* Ψ *belongs to the class* **#P** *if there exists a polynomial time bounded* nondeterministic Turing machine (NTM) *such that, for each instance* I *of* Ψ*, the number of nondeterministic computational paths this NTM takes that lead to acceptance of this problem instance equals the number of solutions of* $\mathrm{I}(\Psi)$.

For an alternative but equivalent definition of the class **#P** in terms of *polynomially balanced relations,* we refer the reader to [37].

The hardest problems in the class **#P** are the **#P**-complete problems.

Definition 12 *A counting problem* Ψ *is* **#P***-complete if and only if (i) it belongs to the class* **#P***, and (ii) it is* hard *for this class, i.e.,* 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 the problems in class **#P** would be solvable in (deterministic) polynomial time, and the entire class **#P** would collapse to **P**.[b] For more on the class **#P**, we refer the interested reader to Chapter 18 of [37] and references therein.

As one would expect, the counting versions of the standard decision **NP**-complete problems, such as SATISFIABILITY or HAMILTON CIRCUIT, are **#P**-complete [37]. 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, 50].

[b] Strictly speaking, since **#P** is a class of *function problems* (as opposed to classes of *decision problems* and hence the formal languages associated with those decision problems, such as the familiar language classes **P**, **NP** or **PSPACE**), if any **#P**-complete problem turns out to be solvable in deterministic polynomial time, this would imply that $\mathbf{P}^{\#\mathbf{P}} = \mathbf{P}$.

Definition 13 *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_I in the previous definition can be efficiently recovered from $n_{g(I)}$.

Definition 14 *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, and the corresponding instance* g(I) *of* Π' *has* $n_{g(I)}$ *solutions, then* n_I *can be computed from* $n_{g(I)}$ *in polynomial time.*

All of our results on the computational complexity of counting various kinds of configurations in SDSs and SyDSs will be obtained by reducing the counting problems about certain types of Boolean formulae that are *known to be* **#P**-complete to the problems about S(y)DSs. That this suffices follows from the well-known property of any problem that is *hard* for a given complexity class; for the record, we state that property in the Proposition below.

Proposition 1 [37] *Given two decision problems* Π *and* Π', *if the corresponding counting problem* $\#\Pi$ *is known to be* **#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-3-SAT (PE3SAT), to the problems of counting FPs and other types of configurations in certain classes of SDSs and SyDSs. These reductions will establish the **#P**-completeness of those counting problems about SDSs and SyDSs. We now define the variants of SATISFIABILITY [14, 37] that we shall use in the sequel:

Definition 15 EXACTLY-ONE-IN-THREE-SATISFIABILITY *(or* **E3SAT** *for short), is a version of* 3CNF-SAT *[14] 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.*

Our **#P**-hardness results in this section will be consequences of Proposition 1 and the following result by Hunt et al. on the hardness of counting satisfying truth assignments of the PE3SAT Boolean formulae:

Proposition 2 [25] *The problems of exact enumeration of the satisfying assignments of (i)* E3SAT, *(ii)* PE3SAT *and (iii)* PLANAR-PE3SAT *are all* **#P***-hard.*

We recall from [47] that counting fixed points in an arbitrary Boolean SDS or SyDS is, in general, **#P**-complete. We sketch the construction from [47] below, as we shall use this construction in the sequel in order to establish computational complexity results about various other configuration space properties of the general Boolean S(y)DSs.

Let an arbitrary instance I of PE3SAT be given, where I has n variables and m clauses. The only assumption we make about I is that it does not have any redundant variables; that is, we assume that each of the n variables appears in at least one clause. We construct the corresponding instance of an SDS $\mathcal{S} = \mathcal{S}(I)$ from Boolean formula I as follows. 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 the nodes C_j for all indices $j = 1, ..., m$. Hence, each C_j has exactly four neighbors, and the 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 the 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 of the three neighboring nodes $x_{j_1}, x_{j_2}, x_{j_3}$ is in the state 1;

(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 Boolean AND of its own current value and the current values of 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 of 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 each of the clauses $C_{j(i)}$ (and in no other clauses of I).

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

We observe that each of the clause nodes C_j in $\mathcal{S}$ has only $O(1)$ neighbors, and, consequently, the description of its update rule, whether given as a (reasonably succinct) Boolean formula or a truth table, will also be of size $O(1)$. In contrast, node y has $\Theta(|V|)$ neighbors. However, y updates according to a very simple rule, namely, the Boolean AND rule, that can certainly be encoded much more succinctly than via a truth table with $\Theta(2^n)$ rows. Similarly, the variable nodes, x_i, also update according to the Boolean AND function, which can be encoded, if need be, in a succinct tabular form even for those nodes that correspond to the variables in the original PE3SAT formula that appear in more than $O(1)$ clauses. In particular, the description of $\mathcal{S} = (G_{\mathcal{S}}, F_{\mathcal{S}}, \Pi_{\mathcal{S}})$ can certainly be encoded so that the size of that encoding is *linear* in the size of the representation of the underlying graph $G_{\mathcal{S}}$ *alone,* and is also *polynomially related* to the size of the original PE3SAT formula I.

We claim that the reduction from #PE3SAT to #FP-SDS based on the above SDS construction from an instance I of PE3SAT is weakly parsimonious. Both problems clearly belong to the class **#P**. Since #PE3SAT is also hard for that class [25], the result below immediately follows:

Theorem 1 *The problem of* exactly counting *the fixed points of an arbitrary Boolean SDS (and therefore also of any more general finite domain SDS) is* **#P***-complete.*

A formal proof that the given SDS construction is, indeed, a weakly parsimonious reduction from #PE3SAT to #FP-SDS, as well as a detailed analysis of what the resulting configuration

space looks like, can be found in [47]. By a straight-forward modification of the given SDS construction and a similar argument to the one in the proof of Theorem 1 (see [47]), the #FP-SyDS problem for the general Boolean and other finite domain SyDSs is intractable, as well.

Corollary 1 *The problem #FP-SyDS for the general Boolean and other finite domain SyDSs is* $\#\mathbf{P}$*-complete.*

We remark that the underlying graph in the construction above is *bipartite,* as there are no lateral edges among the variable nodes or among the clause nodes, and hence only even-length closed paths[c] are possible. Moreover, by the results of Hunt et al. in [25] on the complexity of counting problems for the planar graphs (see Proposition 2), the underlying graph $G_{\mathcal{S}}$ in our construction can be also made *planar* while preserving the hardness of the counting problem #FP-S(y)DS. Likewise, as a straight-forward corollary to the complexity results by Vadhan on counting in sparse graphs and sparse Boolean formulae [48], a $O(1)$ bound on the maximum node degree for the variable nodes can be imposed, while preserving the $\#\mathbf{P}$-completeness of #FP-S(y)DS.

Therefore, when all these restrictions on $G_{\mathcal{S}}$ are imposed simultaneously, the conclusion is that the #FP-S(y)DS problem is $\#\mathbf{P}$-complete even when the underlying graph is (i) planar, (ii) bipartite, (iii) with only one node of degree greater than $O(1)$ (and hence, in particular, sparse on average). We will elaborate more on the restrictions on an SDS's or SyDS's underlying graph that still maintain the hardness of counting FPs in Section 5. Before doing so, however, we first explore several other consequences of the basic SDS construction preceding the statement of Theorem 1.

4.2. Some Implications for Complexity of Other Configuration Space Properties

We now briefly reflect on the weakly parsimonious reduction in the previous subsection. The construction of an SDS from a Boolean formula there resembles of how one constructs a Boolean circuit from a Boolean expression. That is, not only is the construction "preserving the number of solutions", but, furthermore, one can view it as a *very literal translation* from a Boolean formula into an SDS. Additionally, the resulting SDS has a rather simple configuration space: every initial configuration that encodes a satisfying truth assignment of I, and initially has $y^0 = 1$ and all $C_j^0 = 1$, is already a fixed point, whereas all initial configurations that encode falsifying truth assignments of I, as well as those that have $y^0 = 0$ or at least one $C_j^0 = 0$, reach the "sink" fixed point 0^{n+m+1} within two steps. These scenarios are, indeed, the only possible evolutions of $\mathcal{S}$: its configuration space has no temporal cycles whatsoever, no chains of transient configurations longer than two, and no fixed points except for 0^{n+m+1} and those that are of the form $(1, ..., 1, 1, x_1, ..., x_n)$, where $(x_1, ..., x_n) \in \{0, 1\}^n$ is a truth assignment that satisfies the original Boolean formula I.

We summarize the implications of the construction in Theorem 1 in the Lemma below:

[c]We purposefully avoid using the word "cycle" here, even though it is the more common term in the graph theory literature, in order to avoid possible confusion between closed paths, or cycles, in the underlying graph of an SDS on one hand, and the temporal cycles characterizing this dynamical system's behavior (that is, the directed cycles in the resulting configuration space), on the other.

Lemma 1 *Let an arbitrary instance* I *of* PE3SAT *be given, and let the corresponding SDS* $\mathcal{S} = \mathcal{S}(I)$ *be constructed from it as in* Theorem 1. *Then the following statements are equivalent:*

(i) I *is satisfiable, i.e. there exists a truth assignment to the variables appearing in* I *so that the underlying PE3SAT formula evaluates to* **true**.

(ii) SDS $\mathcal{S}$ *has a fixed point configuration* $\mathcal{C} \neq 0^{n+m+1}$.

(ii') SDS $\mathcal{S}$ *has two or more fixed points.*

(iii) $\mathcal{S}$ *has a configuration* $\mathcal{C}$ *such that* $\mathcal{C} \neq 0^{n+m+1}$ *and* $\mathcal{C}$ *has a 2-ancestor, that is,* $pred(pred(\mathcal{C}))$ *exists.*

(iii') $\mathcal{S}$ *has a configuration* $\mathcal{C}$ *such that* $\mathcal{C} \neq 0^{n+m+1}$ *and, for any* $k \geq 1$, $\mathcal{C}$ *has a* k*-ancestor.*

Proofs of these claims are rather straight-forward and will be omitted.

We recall the definition of AMBIGUOUS-SAT, sometimes also called DOUBLE-SAT (see, e.g., [14]): *Given an instance of a Boolean CNF formula, does it have two or more solutions,* i.e., are there two or more satisfying truth assignments? One of the consequences of Lemma 1 is that determining whether an SDS has more than one fixed point is no easier than determining whether an instance of PE3SAT has a satisfying truth assignment. We notice the similarity between the problem of non-uniqueness of FPs in an SDS or SyDS, and the AMBIGUOUS-SAT problem, which can be re-phrased as the problem of non-uniqueness of satisfying truth assignments to a Boolean formula. We recall that AMBIGUOUS-SAT is **NP**-complete in general [14].

Definition 16 *The* AMBIGUOUS-FPE *problem: Given an arbitrary Boolean-valued SDS or SyDS* $\mathcal{S}$*, does it have more than one fixed point?*

It is immediate from Lemma 1 that the following result holds:

Corollary 2 *For general Boolean and other finite domain SDSs and SyDSs, the* AMBIGUOUS-FPE *problem is* **NP***-complete.*

We conclude this section by establishing two more results on the hardness of *exact enumeration* in the context of arbitrary Boolean SDSs and SyDSs. These two results follow from the construction used to establish Theorem 1, in conjunction with the discussion at the beginning of this subsection. Since the S(y)DSs constructed from the PE3SAT CNF formulae in Theorem 1 do not have any cycle configurations, it is immediate that $|\#TC| = 2^{m+n+1} - |\#FP|$. Since the class of function problems **#P** is closed under taking the complement, it follows that *exactly counting* all transient configurations of Boolean and other finite domain S(y)DSs is **#P**-complete, as well.

The sequel of lemmata that follows has the purpose of establishing two facts. One, SDSs and SyDSs constructed as in the SDS construction of Theorem 1 tend to have a great deal of GE configurations. Consequently, approximating the number of those gardens of Eden, denoted $|\#GE|$, to within a constant multiplicative factor is trivial for such S(y)DSs. (This fact, however, by no means implies the easiness of the problem of approximate enumeration of GEs for Boolean S(y)DSs in general.) Two, and more importantly for our purposes, determining $|\#GE|$ *exactly* is intractable.

Lemma 2 *Let an SDS* $\mathcal{S}$ *be constructed from an instance* I *of* PE3SAT *as in the construction in Theorem 1. Assume that* I *contains n Boolean variables and m clauses, where each clause is made of three unnegated variables. Let* $\mathcal{C} = (C, y, x)$ *denote a generic global configuration, where* $C \in \{0,1\}^m$, $y \in \{0,1\}$ *and* $x \in \{0,1\}^n$. *Then:*

(i) Any configuration $\mathcal{C}$ *of the form* $(C, y, x) = (C, 1, x)$ *with* $C \neq 1^m$ *and any* $x \in \{0,1\}^n$ *is a GE.*

(ii) Any configuration $\mathcal{C}$ *of the form* $(C, y, x) = (1^m, 0, x)$ *with* $x \in \{0,1\}^n$ *is a GE.*

Proof. Recall that the node update ordering is $\Pi = (C_1, ..., C_m, y, x_1, ..., x_n)$.

Assume there exists a configuration $\mathcal{C} = (C, y, x) = (C, 1, x)$ with $C \neq 1^m$ that actually is not a GE. Then this configuration must have a predecessor. Let $\mathcal{C}' = pred(\mathcal{C})$. Since the node y updates according to Boolean *AND* that includes its own old value, $y(\mathcal{C}') = 1$. Assume there exists $j_\star \in \{1, ..., m\}$ such that $C_{j_\star}(\mathcal{C}') = 0$. Then, at the next time step, the node $C_{j_\star}$ has to reevaluate to 0: $C_{j_\star}(\mathcal{C}) = 0$. Consequently, since $y^{t+1} \leftarrow y^t \cdot \Pi_{j=1}^m C_j^{t+1}$, it follows that $y(\mathcal{C}) = 0$, a contradiction. Therefore, it must be that $\forall j \in \{1, ..., m\} : C_j(\mathcal{C}') = 1$, i.e., $\mathcal{C}'$ is of the form $(C', y', x') = (1^m, 1, x')$ for some $x' \in \{0,1\}^n$.

Let $TRUE \subset \{0,1\}^n$ denote the set of those Boolean n-vectors that, if used as a truth assignment to $x = (x_1, ..., x_n)$, make the given PE3SAT formula I evaluate to *true* : $\mathrm{I}(x) = 1$. Let the set *FALSE* be defined analogously. For any of the 2^n $(m + n + 1)$-vectors of the form $(C, y, x) = (1^m, 1, x')$, there are two exhaustive and mutually exclusive possibilities: either $x' \in TRUE$, or else $x' \in FALSE$. If $x' \in TRUE$, then $\mathcal{C}' = (1^m, 1, x^{true})$ is a fixed point; that is, $F(\mathcal{C}') = \mathcal{C}' \neq \mathcal{C}$, contradicting our assumption that $pred(\mathcal{C}) = \mathcal{C}'$. On the other hand, if $x' \in FALSE$, then there exists $j_\star$ such that the clause $C_{j_\star}$ of formula I is falsified for this truth assignment $x' = (b_1, ..., b_n)$. Hence, the corresponding clause node in the SDS $\mathcal{S}$ constructed from I will evaluate to zero at the next time step, thereby coercing the node y (that gets its turn only once all the nodes C_j have already updated their respective states) to update to zero, as well. This, however, contradicts our assumption that $y(\mathcal{C}) = 1$. Hence, no configuration of the form $\mathcal{C} = (C, y, x) = (C, 1, x)$ can have a predecessor, i.e., all such configurations are gardens of Eden.

To show that any configuration of the form $\mathcal{C} = (C, y, x) = (1^m, 0, x)$ also must be a GE, we observe that, since each node C_j updates according to Boolean *AND* on several inputs including this node's own current state, a candidate predecessor $\mathcal{C}$' of $\mathcal{C}$ must satisfy $\mathcal{C}' = (1^m, y, x')$ for some Boolean n-vector x'. If $y(\mathcal{C}') = 1$, then, since also $C(\mathcal{C}) = 1^m$, at the next time step the node y would update to 1 again; hence, $F(\mathcal{C}')$ cannot be of the form $(C, y, x) = (1^m, 0, x)$. But if $y(\mathcal{C}') = 0$, that would force $C(\mathcal{C}) \leftarrow 0^m$, thereby violating the assumption that $\mathcal{C}$ is of the form $(C = 1^m, y, x)$. Hence, each configuration of the form $(C, y, x) = (1^m, 0, x)$ is a GE. □

An immediate consequence of the above result is that the GE configurations are, indeed, abundant in those SDSs that are constructed from the PE3SAT formulae as in Theorem 1:

Lemma 3 *Let an SDS* $\mathcal{S}$ *be constructed from an instance of a* PE3SAT *formula with n variables and m clauses, as in the construction of Theorem 1. Then* $\mathcal{S}$ *has a number of garden of Eden configurations that is* exponential *in both n and m.*

Proof. There are $(2^m - 1) \cdot 2^n$ configurations of $\mathcal{S}$ that are of the form $(C, y, x) = (C, 1, x)$, where $x \in \{0,1\}^n$ is arbitrary, and $C \in \{0,1\}^m - \{1^m\}$. By Lemma 2, each of these $\Theta(2^{m+n})$ configurations is a GE. □

Since there are also 2^n GE configurations that are of the form $(C, y, x) = (1^m, 0, x)$, the next result is immediate.

Corollary 3 *Let an SDS $\mathcal{S}$ be constructed from an instance of a* PE3SAT *formula with n variables and m clauses, as in Theorem 1. Then at least a half of all global configurations in the phase space of $\mathcal{S}$ are gardens of Eden.*

As already mentioned, we would like to argue that the problem #GE-SDS of *exactly* counting the gardens of Eden in an SDS constructed from an instance of PE3SAT is, in general, intractable. To that end, we establish a one-to-one correspondence between the *falsifying* truth assignments of a PE3SAT formula, and the GE configurations in the corresponding SDS of the general form $(C, y, x) = (1^m, 1, x^{false})$, where $x^{false} \in FALSE$. It will follow from that correspondence that, in order to determine $|\#GE|$ exactly, we need to be able to exactly enumerate all falsifying truth assignments to the underlying PE3SAT Boolean formula; the latter computational task, however, is known to be **#P**-complete.

Lemma 4 *Let an SDS $\mathcal{S}$ be defined as in the construction of Theorem 1. Then* every *configuration that is of the form $(C, y, x) = (1^m, 1, x^{false})$, with $x^{false} \in \{0,1\}^n$ corresponding to a* falsifying *truth assignment of the underlying* PE3CNF *Boolean formula, is a garden of Eden of $\mathcal{S}$.*

Proof. Let $\mathcal{C} = (C, y, x) = (1^m, 1, x^{false})$ and let's assume there exists a configuration $\mathcal{C}$' such that $pred(\mathcal{C}) = \mathcal{C}'$. Since the node y updates according to Boolean *AND* that includes its own current value, $y(\mathcal{C}') = 1$ must hold. Similarly, for all $j \in \{1, ..., m\}$, $C_j(\mathcal{C}') = 1$ must also hold. Hence, the assumed predecessor configuration itself must be of the form $\mathcal{C}' = (C, y, x') = (1^m, 1, x')$, for some $x' \in \{0,1\}^n$. Now, just like in the proof of Lemma 2, there are only two possibilities: either $x' \in TRUE$, or else $x' \in FALSE$. Either possibility clearly leads to a contradiction. Hence, $\mathcal{C}$ cannot have a predecessor, i.e., it must be a garden of Eden. □

The stage has now been set for the analog of Theorem 1 that pertains to the complexity of *exact enumeration* of GEs and TCs in Boolean and other finite domain SDSs and SyDSs:

Theorem 2 *Given an arbitrary finite domain SDS or SyDS, counting* exactly *all of its transient configurations or all of its garden of Eden configurations is* **#P***-complete.*

5. Counting Configurations of SDSs Defined on Planar Bipartite Graphs

We show in this section that the problems of counting FPs and GEs in Boolean SDSs and SyDSs remain intractable, when these discrete dynamical systems are defined over very restricted underlying graphs - in particular, when these underlying graphs are required to be *both planar and bipartite.* While the hardness of these counting problems can be obtained from Theorem 1 by imposing appropriate restrictions on the instances of PE3SAT formulae and then using work of other researchers on the complexity of counting in planar, bipartite and/or sparse graphs (see discussion at the end of subsection 4.1), we prefer to prove the desired properties in an alternative manner, that is both simpler and independent of the prior work. To that end, we now focus on the *star graphs.* Notice that just about any interesting graph-theoretic problem, such as, e.g., MINIMAL VERTEX COVER, HAMILTON CIRCUIT, MINIMAL DOMINATING SET, or k-COLORING, is trivial to solve on a star graph. Hence, the

computational hardness results on various problems about general planar, bipartite and/or sparse graphs, such as those in [25, 48], cannot be conveniently translated into the context of computational complexity of determining various configuration space properties of SDSs that are defined on the star graphs.

We will show that counting FPs and GEs of a Boolean SDS or SyDS defined on a star graph is, in general, **#P**-complete - as long as the local update rule of the central node is *not given as a truth table.* Moreover, counting these configurations in star graphs remains intractable even when the central node's update rule is given as a *monotone Boolean formula* of a modest size. This is in contrast to the proven tractability of both resolving the fixed point existence (the FPE problem), and actually finding a fixed point, in every Boolean SDS or SyDS all of whose nodes update according to the *monotone Boolean functions* [8].

5.1. Counting FPs and GEs in SDSs and SyDSs Defined on Star Graphs

We now consider SDSs and SyDSs with arbitrary Boolean update rules, but such that the underlying graphs of these S(y)DSs are required to be the star graphs.

We first argue that, if the node update functions are encoded as the truth tables, then, for general Boolean S(y)DSs defined on the star graphs, essentially any computational problem about the configuration space properties is solvable in time polynomial in the size of the S(y)DS's description. However, if we allow the node update rules either to be considered "black boxes" (i.e., oracles), or else be given as sufficiently succinct *Boolean formulae,* then the counting problems of interest become intractable. That is, under the usual complexity-theoretic assumptions, the problems of determining $|\#FP|$ and $|\#GE|$ in the latter two cases cannot be done in the time polynomial in the size of the SDS's or SyDS's representation.

Let us first consider SDSs and SDSs whose node update functions are given as the truth tables. Assume node y is the center of the star, and that it is adjacent to n periphery nodes $x_1, ..., x_n$. Since each node x_i has only one neighbor and therefore depends on only two inputs (i.e., the current values of y and itself), the table size for each x_i is only $O(1)$, and any local update rule computation via a table search will only take $O(1)$ time. However, the central node y has n neighbors; therefore, assuming an arbitrary Boolean function $f_y = g(x_1, ..., x_n, y)$ as the node update rule of y, the truth table for f_y will have $\Theta(2^n)$ rows. Hence, for an S(y)DS defined on a star graph, and assuming an arbitrary Boolean function at the central node, the size of the overall description is $\Theta(2^n)$.

When the node update functions are given as the truth-tables, then, for arbitrary Boolean SDSs and SyDSs defined on the star graphs, counting fixed points and gardens of Eden can be easily accomplished in time polynomial in the size of the S(y)DS representation. This is due to the exponential (in the number of nodes) amount of storage that the central node needs for representing its update rule as a truth table. It is fairly straight-forward to convince oneself that, in that case, virtually *every* configuration space property of interest, including the problems of exactly determining $|\#FP|$, $|\#GE|$ or $|\#TC|$, can be done in a number of steps that is *polynomial* in the size of the S(y)DS's encoding, $\Theta(2^n)$.

These observations on arbitrary Boolean SDSs and SyDSs defined on the star graphs that include the truth tables for each node as a part of their description can now be contrasted with the scenario where the truth tables are *not* a part of the description of an SDS or SyDS. In this latter case, the S(y)DSs are represented much more succinctly, and it turns out that

(unless $\mathbf{P} = \mathbf{P}^{\#\mathbf{P}}$) the fundamental counting problems for such S(y)DSs, defined on the star graphs, become intractable. Throughout the rest of this subsection, therefore, we assume the size of the S(y)DS's encoding is $O(n)$, where n is the number of nodes, i.e. that the central node does not need to store (and then compute by searching) the truth table.

Clearly, if the central node has its update rule given as a "black box" (i.e., an oracle), the overall size of the automaton's encoding is indeed going to be $O(n)$, as desired. We will first establish the hardness of counting results under that assumption; subsequently, we will show that counting remains intractable in the worst case if the central node's rule is given as a Boolean formula, where the size of the formula is considered a part of the overall SDS's or SyDS's encoding.

Let f_y denote the update rule of the central node. We assume that f_y is an entirely arbitrary Boolean function on $n+1$ inputs. We also assume (for now) that the central node, y, has an oracle for evaluating f_y in a single unit of time. We are now ready to state and prove the first of the two main counting results in this Section:

Theorem 3 *When the truth tables are* not *a part of the S(y)DS description, the following counting problems about Boolean SDSs and SyDSs: given an S(y)DS $\mathcal{S}$,*

(i) the problem #FP *of counting the fixed points of $\mathcal{S}$;*

(ii) given an arbitrary configuration $\mathcal{C}$, the problem of determining the number of predecessors of $\mathcal{C}$ (abbreviated as #PRED);

(iii) the problem #TC *of counting the transient configurations of $\mathcal{S}$;*

(iv) the problem #GE *of determining the number of garden of Eden configurations of $\mathcal{S}$*

are all $\#\mathbf{P}$-complete, even when the underlying graph of $\mathcal{S}$ is required to be a star.

Proof. We construct an SyDS $\mathcal{S}$' that, in essence, emulates an *unbounded fan-in* Boolean circuit that evaluates an arbitrary Boolean-valued function f of n Boolean variables.

- The underlying graph $G_{\mathcal{S}} = G(V, E)$ has $n+1$ nodes, that is, one node for each input variable x_i, and one special node, y, that will, under appropriate circumstances, store the value of $f(x_1, ..., x_n)$;

- Node y is adjacent to all nodes x_i (that is, the graph G is a star), and it updates its state according to the rule $y^{t+1} = y^t \cdot f(x_1^t, ..., x_n^t)$;

- Each node x_i updates its value according to $x_i^{t+1} = x_i^t \cdot y^t$.

If $y^0 = 0$, then the node y will remain holding the value 0 after all future updates, regardless of whether the truth assignment $(x_1^0, ..., x_n^0)$ satisfies the Boolean formula f or not. Since each $x_i^1 = x_i^0 \cdot y^0 = x_i^0 \cdot 0$, it follows that all x_i values will be updated to 0, regardless of their initial values. Hence, if $y^0 = 0$, then $\mathcal{S}$' collapses to the "sink" 0^{n+1} in a single step of its evolution.

On the other hand, if $y^0 = 1$, then $y^1 = y^0 \cdot f(x_1^0, ..., x_n^0) = f(x_1^0, ..., x_n^0)$, i.e., the node y will update to 1 iff $(x_1^0, ..., x_n^0)$ is a satisfying truth assignment for f. Insofar as the nodes x_i are concerned, they will evaluate to their previous values at time $t = 1$, whereas at time $t = 2$ they will reevaluate to the same value iff y has evaluated to 1 at $t = 1$, and to 0, otherwise. Thus, after two steps, the configuration reached will either be 0^{n+1}, if the initial choice of $(x_1^0, ..., x_n^0)$ corresponds to a falsifying truth assignment of f, or it will be of the form $(y^2, x_1^2, ..., x_n^2) = (1, x_1^0, ..., x_n^0)$, if $(x_1^0, ..., x_n^0)$ is a satisfying truth assignment of f. Either way, $\mathcal{S}$' will stay at the configuration it has reached after two transitions.

To summarize, the configuration space of $\mathcal{S}$' has no cycles or long transients, and its fixed points are precisely 0^{n+1} and those configurations that are of the form $(y^0 = 1, x_1^0, ..., x_n^0)$, where $(x_1^0, ..., x_n^0)$ is a satisfying truth assignment of the corresponding Boolean function f.

It is almost immediate that, if the node values are updated sequentially instead of synchronously in parallel, i.e., if we consider an SDS corresponding to the SyDS described above, our analysis of the fixed points remains valid. Namely, the fixed points are invariant with respect to the choice of a node update ordering. For the sake of definiteness, but also in order to be able to show the hardness of counting configurations other than fixed points for SDSs defined on the star graphs, we convert the above SyDS $\mathcal{S}$' into a concrete SDS $\mathcal{S}$ by specifying the node update ordering $\Pi = (y, x_1, ..., x_n)$. Thus the node update functions of $\mathcal{S}$ are

$y^{t+1} \leftarrow y^t \cdot f(x_0^t, ..., x_n^t)$;
$for \;\; i = 1, ..., n$
$\quad x_i^{t+1} \leftarrow x_i^t \cdot y^{t+1}$;
$end \;\; for$

We omit a detailed analysis of this SDS's behavior (the analysis is rather similar to that for the corresponding SyDS), and summarize what the configuration space of $\mathcal{S}$ looks like. The fixed points of $\mathcal{S}$ are precisely the "sink" 0^{n+1} and the configurations of the form $(y^0 = 1, x_1^0, ..., x_n^0)$ where $f(x_1^0, ..., x_n^0) = 1$. Thus $\mathcal{S}$ has $T + 1$ fixed points *if and only if* the corresponding Boolean function f has T satisfying truth assignments. We also observe that the fixed point configurations of the form $(1, x_1^0, ..., x_n^0)$ have no incoming transients, simply because all transients lead to the sink 0^{n+1}. Furthermore, the configuration space of $\mathcal{S}$ has no cycles and no transient chains longer than one. Since the convergence from any transient configuration to the sink 0^{n+1} takes exactly one step, every TC is necessarily also a garden of Eden. We recall that every configuration $\mathcal{C}$ with $y(\mathcal{C}) = 0$ and different from the sink 0^{n+1} is a transient state. Thus, if the function f has T solutions, then the corresponding SDS $\mathcal{S}$ has exactly $|\#FP| = T+1$ fixed points and exactly $|\#TC| = |\#GE| = (2^n - 1) + (2^n - T) = 2^{n+1} - T - 1$ transient configurations, each of which is also a garden of Eden.

We also observe that, since the transition from *every* transient configuration leads to the sink FP 0^{n+1}, this fixed point configuration has the number of predecessors equal to $|\#TC| + 1 = |\#GE| + 1 = 2^{n+1} - T$.

We remark that the construction described above establishes only the hardness part for the counting problems (i) - (iv) in the theorem. It is easy to see, however, that all these counting problems belong to the class **#P**, since the corresponding decision problems can be readily seen to be in the class **NP**. □

While our primary goal in this section is to establish the hardness of (exactly) enumerating the fixed point and the garden of Eden configurations for the Boolean and other finite domain SDSs and SyDSs whose underlying graphs are severely restricted, the given constructions also have some implications for the *decision problems* about certain S(y)DS configuration space properties of interest. We next list a few of those properties. The hardness of the configuration space properties below follows directly from the hardness of appropriate satisfiability problems for Boolean functions, and the constructions in the proof of Theorem 3.

Lemma 5 *When the truth tables are* not *a part of an SDS's or SyDS's description, the following decision problems about Boolean SDSs and SyDSs are* **NP***-complete, even when the underlying graph is required to be a star:*

(i) The AMBIGUOUS-FPE PROBLEM: *Given a Boolean S(y)DS* $\mathcal{S}$*', does it have more than one fixed point?*

(ii) Given a Boolean S(y)DS $\mathcal{S}$*', a subset of nodes* $W \subset V$ *such that* $|W| = k$ *with* $k \geq 1$*, and an arbitrary Boolean vector* $b = (b_1, ..., b_k)$*, does* $\mathcal{S}$*' have a fixed point configuration* $\mathcal{C}'$ *such that* $W(\mathcal{C}') = b$ *?*

(iii) Given a Boolean S(y)DS $\mathcal{S}$*', a subset of nodes* $W \subset V$ *such that* $|W| = k$ *where* $k \geq 1$*, a Boolean vector* $b = (b_1, ..., b_k)$*, and a configuration* $\mathcal{C}$*, does* $\mathcal{C}$ *have any predecessors that satisfy* $W(pred(\mathcal{C})) = b$ *?*

We recall that the TAUTOLOGY PROBLEM for Boolean functions ("Given an arbitrary Boolean function f, does f evaluate to *true* for *all* truth assignments to its variables ?") is a paradigmatic **coNP**-complete problem. The SDS $\mathcal{S}$ in the proof of Theorem 3 will have no more than $2^n - 1$ transient configurations or gardens of Eden (out of 2^{n+1} configurations in total) *if and only if* $\mathcal{S}$ *does not have* any TC (GE) configurations with $y = 1$ *if and only if* f is a tautology. Hence, another corollary to our construction in the proof of Theorem 3 is that determining whether an SDS defined over a star graph will have any *nontrivial* general transient or garden of Eden configurations is **NP**-hard (where, in our example, *nontrivial configurations* would be those configurations $\mathcal{C}$ such that $y(\mathcal{C}) = 1$). One can view this corollary as an extension of the already known result on **NP**-completeness of the GARDEN OF EDEN EXISTENCE problem for Boolean and finite range SDSs [8]. Likewise, complexity results about the existence of GEs and TCs analogous to the claims in parts (ii) and (iii) of Lemma 5 also hold:

Corollary 4 *When the truth tables are* not *a part of an S(y)DS's description, the following decision problems are* **NP***-complete, even when the underlying graph is required to be a star: given a Boolean S(y)DS* $\mathcal{S}$*', a subset of nodes* $W \subset V$ *such that* $|W| = k$*, and an arbitrary Boolean vector* $b = (b_1, ..., b_k)$*, does* $\mathcal{S}$*' have (i) a GE configuration, or (ii) a transient configuration* $\mathcal{C}'$ *such that* $W(\mathcal{C}') = b$ *?*

5.2. Counting FPs and GEs of Monotone *SDSs Defined on Star Graphs*

All results in the previous subsection have been shown under the unrealistic assumption that the central node updates its state by using an oracle. Moreover, all the hardness of counting results in this paper thus far have been shown for the SDSs and SyDSs for which the nontrivial corresponding decision problems, such as AMBIGUOUS-FPE, are intractable in general.

We shall now drop the oracle assumption, and assume that the central node is given its update rule f as a Boolean formula that is a part of the S(y)DS's description. Furthermore, we will also choose f to be from a class of Boolean formulae for which the corresponding satisfiability problem is tractable - thereby assuring that the related decision problems about the resulting SDS or SyDS, such as FPE and AMBIGUOUS-FPE, are also tractable.

The purpose of this exercise is twofold. One, we show that counting configurations of S(y)DSs over the star graphs is, indeed, intractable - as long as the central node's update

rule is encoded *reasonably succinctly.* Two, just like what has been known for the Boolean formulae and many other combinatorial problem domains, we show in the context of discrete dynamical systems of our interest that there is an intrinsic aspect of computational hardness that is *peculiar to the counting problems,* and that does not have an analog among the corresponding decision problems.

Consider the class of MON-2CNF Boolean formulae: each clause has exactly two literals, and no negated variables are allowed. These formulae are a classical example where the problem of existence of a solution (i.e., a satisfying truth assignment) is trivial, yet enumerating all satisfying assignments is, in general, **#P**-complete [50]. Moreover, it has been shown much more recently that the counting problem #MON-2CNF remains **#P**-complete even if no variable appears in more than four clauses [48]. In particular, if no variable appears in more than $O(1)$ clauses, it immediately follows that, given such a MON-2CNF formula that contains n Boolean variables and some number of 2CNF clauses with those variables, the length of the entire formula is $O(n)$.

Now consider an SyDS or SDS as constructed in the proof of Theorem 3, except that, instead of the central node's update rule being treated as an oracle, this node, y, is given its update rule f_y as a MON-2CNF formula with each variable x_i appearing in only $O(1)$ clauses (say, at most four). Similarly, for the sake of consistency, assume that the update rules for the peripheral nodes x_i (which are just the Boolean AND functions of two variables) are also given as formulae, and considered a part of the problem instance's description. Now the size of this S(y)DS's encoding is $O(|E| + |f_{max}| \cdot |V|)$, where $|f_{max}|$ stands for the maximum size of any update rule formula. Clearly, under the stated assumptions, and since $|E| = O(|V|)$ for the star graphs, this encoding is of a size only $O(n^2)$; in fact, a more careful analysis shows that it is $O(n)$. That is, it is *polynomial, and not exponential, in* n. Consequently, incorporating the encodings of the local update rules into the description of such an S(y)DS does not affect much the size of that dynamical system's overall description.

We can now re-state Theorem 3, only without the artificial "black box" assumption about the nodes' update rules:

Theorem 4 Exactly enumerating *each of the following types of configurations:*
(i) all fixed points;
(ii) all predecessors of an arbitrary configuration;
(iii) all transient configurations; and
(iv) precisely those transient configurations that are gardens of Eden
of Boolean and other finite domain S(y)DSs is, in general, **#P***-complete, even when the underlying graph is restricted to a* star graph, *and each node updates according to a* monotone *Boolean-valued function, where the local update rules are considered a part of the S(y)DS description, and are given as negation-free Boolean* 2CNF *formulae.*

We summarize the main results of this section in the following

Corollary 5 *Counting each of (i) the fixed points, (ii) the gardens of Eden, (iii) the predecessors, and (iv) the transient configurations of finite domain SDSs and SyDSs is, in the worst case, computationally intractable, even when all of the following restrictions on the underlying graphs and the local update rules* simultaneously *hold:*

- *the underlying graph is* planar, bipartite, *and with* $|E| \leq |V|$*;*

- *all the local update rules are* monotone Boolean functions;
- *these update rules are considered a part of the SDS's or SyDS's description, and are given as (monotone) Boolean formulae; and*
- *SDS or SyDS uses only two different update rules from the given class of functions.*

6. 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 those components and the interdependencies among different processes taking place at different components, the overall system behavior can become extremely complex. This, in particular, makes the design of reliable such systems challenging. Equally importantly, formal verification of various properties of such systems, as well as the forecast of their likely future behavior patterns, become difficult tasks.

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 perspective. 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 herein on the problems of counting how many stable configurations (FPs) and unreachable configurations (GEs) such dynamical systems have in their configuration spaces, when each of their nodes has only two distinct states, and updates according to a simple Boolean function of the states of its neighboring nodes. We establish that these counting problems in Sequential and Synchronous Dynamical Systems are **#P**-complete, even when the following constraints on the structure of an SDS or SyDS *simultaneously* hold:

- the underlying graph of this SDS or SyDS is *planar, bipartite,* cycle-free and very sparse - in particular, this graph can be restricted to be a *star;*
- each local update rule is required to be a *monotone* Boolean function; and
- the nodes of the S(y)DS use *only two* different update rules (e.g., for the star graphs, one rule for the central node, and the other rule for everyone else).

In particular, we have shown in this work that important counting problems about distributed discrete dynamical systems are intractable when two important restrictions simultaneously hold. One, insofar as the *inter-agent local interactions* are concerned, we restricted the "communication topology", that is, the underlying graph of an SDS, to a star graph. Two, insofar as each agent's *individual behavior* is concerned, we limited the node update rules to the monotone Boolean-valued functions encoded as (sparse) Boolean formulae. The importance of our results for the SDSs defined on the star graphs stems from the fact that almost any computational graph-theoretic problem of interest is trivial when the graph instances are restricted to the (ordinary, static) star graphs. The importance of the results for

the restricted local update rules lies in the fact that the corresponding decision problems for the monotone Boolean SDSs (as well as for the monotone Boolean formulae in general) are tractable.

That there are important configuration space properties of Boolean SDSs that remain intractable even when those SDSs are severely restricted both in terms of the allowable underlying graphs and the allowable local update rules is an indication that a very complex and, in general, unpredictable global dynamics can be obtained by coupling together only rather simple, monotonically behaving local interactions. Indeed, we have formally shown that this general observation holds true as long as there exists a *single agent* in such a dynamical system that is allowed to interact with a large number of other agents.

Acknowledgments: The author expresses his sincere gratitude to Gul Agha, Alfred Hubler and Michael Loui (all from University of Illinois), Harry Hunt (SUNY-Albany) and Madhav Marathe (Los Alamos National Laboratory) for useful discussions, suggestions and/or feedback on various matters related to this paper.

References

1. S. Amoroso and Y. Patt. "Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures", *Journal of Computer and System Sciences (JCSS)*, vol. 6, pp. 448 - 464, 1972
2. R. J. Bagley and L. Glass. "Counting and Classifying Attractors in High Dimensional Dynamical Systems", *Journal of Theoretical Biology,* vol. 183, pp. 269 - 284, 1996
3. C. Barrett, B. Bush, S. Kopp, H. Mortveit, C. Reidys. "Sequential Dynamical Systems and Applications to Simulations", Technical Report, Los Alamos National Laboratory, September 1999
4. 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
5. 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
6. 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
7. 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 Computer Science,* vol. 295, issues 1-3, pp. 41 - 64, February 2003
8. 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", Proc. AA (DM-CCG), spec. ed. of *Discrete Mathematics and Theoretical Computer Science (DMTCS),* pp. 95 - 110, 2001
9. C. Barrett, H. Mortveit, and C. Reidys. "Elements of a theory of simulation II: sequential dynamical systems" *Applied Mathematics and Compututation,* 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 Mathematics and Computation,* vol. 122, pp. 325 - 340, 2001

11. 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

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

13. 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 (Preliminary version appeared in *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, October 1990)

14. 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

15. P. Gacs. "Deterministic computations whose history is independent of the order of asynchronous updating", Tech. Report, Computer Science Dept, Boston University, 1997

16. M. Garzon. *"Models of Massive Parallelism: Analysis of Cellular Automata and Neural Networks",* Springer, 1995

17. 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

18. E. Goles, S. Martinez. *"Neural and Automata Networks: Dynamical Behavior and Applications",* Math. and Its Applications series (vol. 58), Kluwer, 1990

19. E. Goles, S. Martinez (editors). *"Cellular Automata and Complex Systems",* Nonlinear Phenomena and Complex Systems series, Kluwer, 1999

20. F. Green. "NP-Complete Problems in Cellular Automata", *Complex Systems,* vol. 1 (3), pp. 453 - 474, 1987

21. H. Gutowitz (Editor). *"Cellular Automata: Theory and Experiment",* North Holland, 1989

22. J. J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities", *Proc. Nat'l Academy Sci. USA,* vol. 79, pp. 2554 - 2558, 1982

23. J. J. Hopfield, D. W. Tank. "Neural computation of decisions in optimization problems", *Biological Cybernetics,* vol. 52, pp. 141 - 152, 1985

24. B. Huberman and N. Glance. "Evolutionary games and computer simulations" *Proc. National Academy of Sciences*, 1999

25. H. B. Hunt, M. V. Marathe, V. Radhakrishnan, R. E. Stearns. "The Complexity of Planar Counting Problems", *SIAM J. Computing,* vol. 27, pp. 1142 - 1167, 1998

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, January 1984

28. M. Jerrum. "Two-dimensional monomer-dimer systems are computationally intractable", *J. Statist. Physics,* vol. 48, pp. 121 - 134, 1987 (Erratum in vol. 59, pp. 1087 - 1088, 1990)

29. M. Jerrum, A. Sinclair. "Approximating the permanent", *SIAM J. Computing* vol. 18, pp. 1149 - 1178, 1989

30. M. Jerrum, A. Sinclair. "Polynomial-time approximation algorithms for the Ising model", *SIAM J. Computing,* vol. 22, pp. 1087 - 1116, 1993

31. R. Laubenbacher, B. Pareigis. "Finite Dynamical Systems", Tech. report, Dept. of Mathematical Sciences, New Mexico State University, Las Cruces, N. Mexico, 2000

32. B. Martin. "A Geometrical Hierarchy of Graphs via Cellular Automata", Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, August 1998

33. C. Moore. "Unpredictability and undecidability in dynamical systems", *Physical Review Letters,* vol. 64 (20), pp. 2354 - 2357, 1990

34. H. Mortveit, C. Reidys. "Discrete, sequential dynamical systems", *Discrete Mathematics*, vol. 226, pp. 281 - 295, 2001

35. J. Myhill. "The converse of Moore's Garden-of-Eden theorem", *Proc. Amer. Math. Society,* vol. 14, pp. 685 - 686, 1963

36. C. Nichitiu and E. Remila. "Simulations of Graph Automata", Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, August 1998

37. C. Papadimitriou. *"Computational Complexity" ,* Addison-Wesley, Reading, Massachusetts, 1994

38. C.M. Reidys. "On Acyclic Orientations & Sequential Dynamical Systems", Los Alamos National Laboratory Report, LA-UR-01-598, 2001

39. Z. Roka. "One-way cellular automata on Cayley graphs", *Theoretical Computer Science,* vol. 132 (1-2), pp. 259 - 290, September 1994

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

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

42. K. Sutner. "Computation theory of cellular automata", Proc. MFCS'98 Satellite Workshop on Cellular Automata, Bruno, Czech Republic, August 1998

43. 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

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, in Proc. IEEE IPDPS'04, Santa Fe, New Mexico, USA, April 26-30, 2004

46. P. Tosic, G. Agha. "Characterizing Configuration Spaces of Simple Threshold Cellular Automata", Proc. 6th Int'l Conf. on Cellular Automata for Research and Industry (ACRI'04), Amsterdam, The Netherlands, October 25-28, 2004; in Springer-Verlag LNCS series, vol. 3305, pp. 861 - 870

47. P. Tosic. "On Complexity of Counting Fixed Point Configurations in Certain Classes of Graph Automata", *Electronic Colloquium on Computational Complexity,* ECCC TR05-051, 2005

48. S. Vadhan. "The Complexity of Counting in Sparse, Regular and Planar Graphs", *SIAM J. Computing,* vol. 31 (2), pp. 398 - 427, 2001

49. L. Valiant. "The Complexity of Computing the Permanent", *Theoretical Computer Science,* vol. 8, pp. 189 - 201, 1979

50. L. Valiant. "The Complexity of Enumeration and Reliability Problems", *SIAM J. Computing,* vol. 8 (3), pp. 410 - 421, 1979

51. S. Wolfram. *"Theory and applications of cellular automata",* World Scientific, 1986

52. 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'