On the Computational Complexity of Predicting Dynamical Evolution of Large Agent Ensembles

Predrag T. Tošić* and Gul A. Agha

Open Systems Laboratory, Department of Computer Science,
University of Illinois at Urbana-Champaign (UIUC)
1334 T. Siebel Center for Computer Science,
201 N. Goodwin Avenue, Urbana, IL 61801, U.S.A.
E-mail: {p-tosic, agha}@cs.uiuc.edu

Abstract

We study global behavior of large ensembles of simple reactive agents. We do so by applying computational complexity tools to the analysis of formal complex systems and their dynamics. Since we are interested in the global dynamics and emerging behavior of large agent ensembles, rather than in an individual agent's deliberation, learning or other cognitive abilities, the discrete complex systems we study are based on the communicating finite state machine (CFSM) abstraction. In particular, we show that counting the number of possible evolutions of a particular class of CFSMs is computationally intractable, even when those CFSMs are very severely restricted both in terms of an individual agent's behavior (that is, the local update rules), and the inter-agent interaction pattern (that is, the underlying communication network topology). We use this abstract framework to formally prove the well-known intuition about multi-agent systems (MAS) that a complex and, in general, unpredictable global behavior may arise from coupling of rather simple local behaviors and interactions.

Keywords: Agents and complex systems, self-organizing systems, large-scale MAS, cellular and graph automata, discrete dynamical systems, computational complexity, **#P**-completeness

1 Introduction and Motivation

Multi-Agent Systems (MAS) are a research area where artificial intelligence and distributed computing overlap [50]. Hence, research in MAS heavily draws on the existing theories, tools and methodologies from both AI and distributed computing. What we would like to contribute to the more thorough understanding and better design of large-scale MAS are some ideas, paradigms and tools from another scientific discipline, namely, *complex dynamical systems.* Among many abstract mathematical models of discrete dynamical systems, the one class that we find particularly simple yet useful for addressing many fundamental issues in distributed computing in general, and in large-scale multi-agent systems in particular, are the classical *cellular automata* and some of their *graph automata* extensions.

Cellular automata (CA) [16, 17, 18, 21, 34, 52, 53] are discrete dynamical systems whose individual components are rather simple, yet that can exhibit highly complex and unpredictable behavior due to these simple components' mutual interaction and synergy. A CA is made of a finite or infinite regular 1-D, 2-D or higher-dimensional grid of nodes, where each node behaves like a finite state machine with a fixed, usually small, number of distinct states. The nodes are interconnected together and can affect each other: the future state of a given node, in addition to its own current state, also depends on the current states of some of its near-by nodes. The "program" that tells a node to what value it should update its state, based

---

*Contact author. Office phone: +1-217-244-1976. Fax: +1-217-333-9386.

on the states of these neighboring nodes, is deterministic and fixed; it is called the *local update rule.* All the CA nodes update (i) synchronously in parallel with each other, and (ii) according to the same update rule.

What are the important properties of the large-scale distributed computational and communication systems in general, and MAS in particular, that can be adequately captured by the CA-like models? Let's consider a cellular automaton from the distributed computing and multi-agent system perspectives. Studying global dynamics of a CA then translates into an exploration of the global behavior of a multi-agent system when (i) the individual agent behaviors are fixed, (ii) the pattern of multi-agent interaction ("network topology") is fixed, and (iii) both the individual agent behaviors and the interaction patterns among the agents are uniform (that is, *homogeneous*) across the entire system. In particular, CA and other related models capture the important distributed and multi-agent systems' properties of *locality* of interaction among the agents, and of the bounded speed of information and impact propagation. However, CA-like models do not allow for an individual agent's deliberation or adaptation of any sort; the individual agents are strictly reactive, and are also characterized by a fixed, nonadaptive behavior. Instead, the focus of such models is on various *emergent properties* at the level of *agent ensembles.*

Several modifications of the basic CA model along different dimensions can be argued to provide appropriate abstractions for the large-scale distributed computing and, in particular, for the *large-scale MAS* made of simple reactive, autonomously executing agents. We identify the following four as the most important:

- *heterogeneity* of the cellular/graph automata in terms of (i) the individual agent behaviors and (ii) the inter-agent interaction pattern, in contrast to the strict *homogeneity* of the classical CA in both these respects;

- *model of inter-agent communication* insofar as whether the agents locally compute synchronously or asynchronously, and whether they interact with one another synchronously or asynchronously;

- *adaptability* of the individual agents, i.e., are these agents capable of *dynamically changing their behavior* via, e.g., reinforcement learning, or are their individual behaviors *fixed* once the state of their environment is specified;

- *dynamic changes* of the MAS network topology, that would be captured by allowing the underlying cellular space of a cellular or graph automaton to change as a function of time.

Of the outlined four dimensions along which the classical CA can be readily generalized, this paper will focus on the first. In particular, we will study a particular class of models of *communicating finite state machines (CFSMs)* as an abstraction of a MAS made of simplistic, reactive agents, in which (i) different agents are allowed to behave differently from one another, and (ii) the interaction (i.e., communication) pattern among the agents is allowed to be irregular.

1.1   Paper Outline

We study in this work certain classes of CFSM-based *graph automata* that we find to be a useful abstraction of various multi-agent systems made of *reactive agents,* and a sufficiently general theoretical model for the agent-based simulation of a broad variety of physical, computational, and socio-technical distributed infrastructures [3, 12]. In this work, as well as several other related papers (e.g., [4, 5, 6, 7, 8, 9, 10, 33, 44, 45, 46]), the general approach is to study various *configuration space properties* of such graph automata.

We specifically focus herewith on determining *how many* configurations of certain kinds such graph automata have, and *how hard* are the computational problems of *enumerating* those configurations. The main contributions of our work are as follows. We prove that *exactly enumerating* the number of *fixed point* configurations in two related classes of graph automata, called *Sequential* and *Synchronous Dynamical Systems,* is computationally intractable, even when (i) such a discrete dynamical system is defined over a graph that is *regular, bipartite,* and *uniformly sparse,* and (ii) each node of a given graph automaton is required to update its state according to a *symmetric* 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. Some related work

is briefly reviewed in Section 3. Our results on the hardness of enumerating fixed points,[1] and therefore different possible dynamical evolutions, of the CFSM models of interest are presented in Section 4. The paper summary is given in Section 5.

2 Preliminaries

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

Definition 2.1. *A* Sequential Dynamical System (SDS) $\mathcal{S}$ *is an ordered triple* (G, F, Π), *with the following components:*

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* $\mathcal{S}$.

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* $1 \le i \le n$. *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* $V = \{v_1, v_2, \ldots, v_n\}$ *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,* $F_{\mathcal{S}} = (f_{\Pi^{-1}(v_1)}, f_{\Pi^{-1}(v_2)}, \ldots, f_{\Pi^{-1}(v_n)})$.

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 [3, 4, 5, 6, 8, 9, 10]. By *symmetric* is meant that the future state of a node does not depend on the order in which the input values of this node's neighbors are specified. Instead, the future state depends only on $\Sigma_{j \in N(i)} x_j$ (where $N(i)$ stands for the *extended neighborhood* of a given node, v_i, that includes the node v_i itself), i.e., on how many of the node's neighbors are currently in the state 1. Thus symmetric Boolean SDSs correspond to the *totalistic (Boolean) cellular automata* of Wolfram (see, e.g., [52, 53]).

The assumption about *symmetric* Boolean functions can be easily relaxed to yield more general SDSs. We give special attention to the symmetry condition for two reasons. First, our computational complexity theoretic lower bounds for such SDSs imply stronger lower bounds for determining the corresponding configuration space properties[2] 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 [13].

Definition 2.2. *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.*

---

[1]Fixed point configurations of a sequential or a synchronous dynamical system will be defined in Section 2.

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

Thus, the SyDSs are similar to the finite classical *cellular automata,* 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. Another difference is that, whereas in the 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.

2.1 SDS and SyDS Configuration Space Properties

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

The function computed by SDS $\mathcal{S}$, denoted by $F_{\mathcal{S}}$, specifies for each configuration $\mathcal{C}$ the next configuration $\mathcal{C}'$ reached by $\mathcal{S}$ after carrying out the updates of the node states in the order given by Π. Thus, the function $F_{\mathcal{S}} : \mathcal{D}^n \to \mathcal{D}^n$ is a total function on the set of global configurations. This function therefore defines the dynamics of the SDS $\mathcal{S}$. We say that $\mathcal{S}$ moves from a configuration $\mathcal{C}$ to a configuration $F_{\mathcal{S}}(\mathcal{C})$ in a single transition step. Alternatively, we say that SDS $\mathcal{S}$ moves from a configuration $\mathcal{C}$ at time t to a configuration $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}'$ 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 the 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 any 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 the SDS is finite, the number of configurations in the phase space is also finite. If the size of the domain (that is, the number of possible states of each node) is $|\mathcal{D}|$, then the number of global configurations in $\mathcal{P}_{\mathcal{S}}$ is $|\mathcal{D}|^n$.

Definition 2.3. *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.*

3 Related Work

SDSs and SyDSs investigated in this paper are closely related to the *graph automata* (GA) models studied in [32, 35] 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 [35]. The main difference between the graph automata in [32, 35] and the SDSs studied herein is the sequential ordering aspect. In particular, SDSs generalize the finite *sequential CA* [44, 45] 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, 17, 24, 39, 44]. 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 [17, 44].

Barrett, Mortveit and Reidys in [9, 10, 33], and Laubenbacher and Pareigis in [31], investigate the mathematical properties of sequential dynamical systems. Barrett et al. study the computational com-

plexity 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]. The first results on the related counting problems, some of which are revisited (in much less detail) in this paper, can be found in [46].

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., [47]), 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 [17, 44, 45], 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.

4 Results on Hardness of Enumerating SDS and SyDS Fixed Points

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 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, 38] that we shall use in the sequel:

Definition 4.1. 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 *(abbreviated as* **PE3SAT***) is further restricted: no clause in the* 3CNF *formula is allowed to contain a negated literal.*

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

Proposition 4.1. [25] *The problems of exact enumeration of the satisfying assignments of (i)* E3SAT, *(ii)* PE3SAT *and (iii)* PLANAR-PE3SAT *are all* **#P**-*hard.*

We have shown in [46] that counting fixed points in an arbitrary Boolean SDS or SyDS is, in general, **#P**-complete. We sketch the construction from [46] below.

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, $1 \le j \le m$. This will enable us to argue that the node y, in effect, evaluates the Boolean formula for the specified truth assignment $\{x_1, ..., x_n\}$, provided that the initial value of y is $y^0 = 1$, and, likewise, that $C_j^0 = 1$, for all j, $1 \le j \le m$.

- Each node x_i evaluates the logical AND of itself and the current values stored in the clause nodes $C_{j(i)}$ such that, in the original formula I, variable x_i appears in 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 $\Theta(2^m)$ 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 4.1. *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 above 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 our online technical report [46].

By a straight-forward modification of the given SDS construction, and a similar argument to the one in the proof of Theorem 4.1, the #FP-SyDS problem for the general Boolean, as well as other finite domain SyDSs, is hard for the class #**P**, as well.

Corollary 4.1. *The problem #FP-SyDS for Boolean and other finite domain SyDSs is* #**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[3] are possible. Moreover, by the results of Hunt et al. in [25] on the complexity of counting problems for the planar graphs (see Proposition 4.1), 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 [47], a $O(1)$ bound on the maximum node degree for the variable nodes can be imposed, while preserving the #**P**-completeness of #FP-S(y)DS.

---

[3]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 the 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.

Therefore, when all these restrictions on $G_{\mathcal{S}}$ are imposed simultaneously, the conclusion is that the #FP-S(y)DS problem is #**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, very sparse on average). We show in the next subsection that the #**P**-completeness of exactly counting FPs still holds when the local update rules are restricted to *symmetric Boolean functions*. Finally, in subsection 4.2, we establish the intractability of counting FPs in symmetric Boolean SDSs and SyDSs even when the underlying graph, in addition to being bipartite, is also *uniformly sparse* and *regular*.

4.1 Counting Fixed Points of Symmetric Boolean SDSs

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

Let an instance I of PE2-IN-3SAT be given. Assume that there are n Boolean variables, denoted $x_1, ..., x_n$, and m clauses, $C_1, ..., C_m$, in I. We recall that each clause C_j contains *exactly* three unnegated variables, $x_{j_1}, x_{j_2}, x_{j_3}$. A monotone 3CNF Boolean formula 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 [33] for a proof).

Theorem 4.2. *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. Next, we define the edges of the underlying SyDS graph. 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. Each clause node C_j is 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. By symmetry, the node y is adjacent to all the clause nodes C_j, $1 \leq j \leq m$.

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 Boolean-valued function ALL-BUT-ONE of q Boolean variables (for any integer $q \geq 1$) is defined as ALL-BUT-ONE$\{z_1, ..., z_q\} = 1$ if and only if *exactly* one of its inputs z_l is 0, and all the rest are 1s.

We 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. A detailed proof of this claim can be found in our electronic technical report [46].

By the 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 surprising.

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

Again, the full proof can be found in [46].

4.2 Counting FPs of Symmetric SDSs over Uniformly Sparse Graphs

In subsection 4.1, we have established that counting FPs of *symmetric Boolean* SDSs and SyDSs is, under the usual assumptions in computational complexity, intractable. The constructions there are only *weakly* parsimonious; see [46] for more details. Perhaps more importantly, the underlying graphs in those constructions have nodes with an unbounded number of neighbors; those nodes would correspond to agents that can *directly communicate* with many other agents. In most large-scale MAS, however, an agent can directly communicate with only a handful of other agents, i.e., the underlying communication topology is *sparse.*

In this subsection, we show that enumerating FPs of symmetric Boolean S(y)DSs remains #**P**-complete, even when the underlying graph, that is, the communication network topology of the agents, is *uniformly sparse.* That is, the result holds even when each node of an SDS or SyDS has only $O(1)$ neighbors. In particular, the intractability of the FP enumeration problem holds even when the underlying graph is 3-*regular* and also *bipartite.* Moreover, our construction that will establish the desired result will be (strongly) parsimonious.

Theorem 4.4. *The problem of enumerating fixed point configurations of symmetric Boolean SDSs and SyDSs is* #**P***-complete even when the underlying graph is simultaneously* 3-regular *and* bipartite.

Proof: We first establish the #**P**-hardness of the problem of enumerating FPs of a symmetric Boolean SDS (Sym-Bool-SDS) when the underlying graph is bipartite and has all node degrees bounded by 3. A simple modification will then yield the desired result for 3-regular graphs.

Let an instance of *monotone* 2CNF Boolean formula be given, such that no variable appears in more than three clauses. We assume that each clause contains *exactly two distinct* unnegated Boolean variables. For the ease of the subsequent SDS construction, we do not allow any redundant clauses or variables in this monotone 2CNF instance; in particular, without loss of generality, we assume that each variable appears in at least one clause. None of these restrictions affects the computational complexity of the underlying enumeration problem.

By the results of Vadhan [47] and Greenhill [20], enumerating satisfying assignments of such a Boolean formula is still #**P**-complete. From this monotone 2CNF formula we parsimoniously construct a *symmetric* Boolean SDS as follows. As before, we denote the number of variables by n and the number of clauses by m. There is a node for each variable x_i in the 2CNF formula, and a node for each clause C_j in the formula. As before, a *variable node* x_i is adjacent to a *clause node* C_j if and only if, in the 2CNF formula, the corresponding variable appears in the corresponding clause. Unlike our prior constructions, no auxiliary nodes or edges are needed. The node update rules are as follows. Each variable node, x_i, updates according to the Boolean *AND* of its own current state, and the current states of (up to three) clause nodes that this variable node is adjacent to. Each clause node, C_j, updates according to the following symmetric (but non-monotone) update rule:

$$C_j \quad \leftarrow \quad \begin{cases} 1, & \text{if } C_j + x_{j_1} + x_{j_2} \neq 1 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where, in the 2CNF formula, the clause C_j is given by $C_j = (x_{j_1} \lor x_{j_2})$.

The node update ordering is any permutation Π such that first all the variable nodes update (in an arbitrary sequential order), and then all the clause nodes update (again, in an arbitrary order).

We claim that the given construction from Mon-2CNF Satisfiability to Sym-Bool-SDS is parsimonious. We will outline the case analysis. There are two main cases to consider. Let's first assume that the SDS starts from an initial configuration where there exists a clause node, $C_{j_\star}$, such that initially

$C_{j\star}^0 = 0.$ Consider the subconfiguration $(C_{j\star}, x_{j\star_1}, x_{j\star_2})$. If initially $(x_{j\star_1}^0, x_{j\star_2}^0) = (0, 0)$, then this subconfiguration is already a part of a (local) temporal two-cycle, i.e., $(C_{j\star}, x_{j\star_1}, x_{j\star_2})$ will alternate between $(0, 0, 0)$ and $(1, 0, 0)$ *ad infinitum,* irrespective of the states of other notes. If initially $(x_{j\star_1}^0, x_{j\star_2}^0) \neq (0, 0)$, then, at time $t = 1$, $(C_{j\star}^1, x_{j\star_1}^1, x_{j\star_2}^1) = (1, 0, 0)$, and the same local temporal two-cycle $\{(1, 0, 0), (0, 0, 0)\}$ is reached. Thus, if initially there exists a clause node that is in the state 0, such starting configuration may be either transient or cyclic, but it cannot be a fixed point.

The second scenario to consider is when, initially, $C_j^0 = 1$, for all j. There are two subcases: one is when the Boolean vector $(x_1, ..., x_n)$ corresponds to a satisfying assignment of the underlying 2CNF formula, and the other is when this Boolean vector falsifies the formula. In the former case, each clause node will have at least two out of three of its inputs equal to 1 at time $t = 1$, so it will evaluate to 1. Similarly, each variable node, that updates according to Boolean *AND,* will have all its neighboring nodes in the state 1, so it will itself stay at its current value. Hence, it is immediate that each configuration of the form $(C_1, ..., C_m, x_1, ..., x_n) = (1, ..., 1, x_1, ..., x_n)$, where $(x_1, ..., x_n)$ constitutes a satisfying assignment to the monotone 2CNF formula, is a fixed point.

In contrast, if $(x_1, ..., x_n)$ falsifies the 2CNF formula, then there exists a clause node that has both its neighboring variable nodes in the state 0, and, since this clause node is currently in the state 1, it will have exactly one of its three inputs equal to 1. Hence, this clause node will change its state to 0, i.e., the corresponding starting configuration cannot be a FP.

Therefore, we conclude that the satisfying truth assignments of the MON-2CNF formula are in a one-to-one correspondence with the fixed point configurations of the symmetric SDS constructed from this formula. The claim of the theorem for the SDSs whose underlying graphs have the maximum node degree of 3 now follows immediately.

Moreover, two straight-forward modifications can ensure that the Theorem holds for 3-regular graphs, as well. First, we can construct an SDS from a monotone 3CNF (instead of a monotone 2CNF) formula; this modification certainly does not make the corresponding problem of enumerating the satisfying truth assignments any easier. Secondly, the monotone 3CNF formula can be required to be such that each variable x_i appears in *exactly* three clauses, as opposed to *at most* three clauses (see, e.g., [20]). These two modifications maintain the **#P**-completeness of the enumeration version of SAT. Thus, the resulting underlying graph in our SDS construction, in addition to being uniformly sparse and bipartite, can be made to be 3-regular, as well.

The claim of the theorem has now been established for the symmetric Boolean SDSs. Essentially the same construction works for the corresponding BOOL-SYM-SYDSs, where all the nodes update synchronously in parallel. The corresponding case analysis that establishes that the resulting reduction is, indeed, parsimonious, is similar to, but slightly different from, the analysis for the BOOL-SYM-SDSs. We leave out the details due to space constraints.

To summarize, determining the number of stable configurations, and therefore of different possible dynamical evolutions that eventually stabilize and reach one of the stable configurations, is **#P**-complete even for some very simplistic CFSM-based discrete dynamical systems, such as the symmetric Boolean SDSs and SyDSs studied in this paper. Moreover, as our final theorem above shows, this intractability holds even when the underlying network topology is regular, bipartite and very sparse. In particular, insofar as the sparseness aspect is concerned, our results for S(y)DSs on 3-regular graphs (and, more generally, all sparse graphs whose maximum node degree does not exceed 3) are, to the best of our knowledge, the sharpest of their kind yet: all previous results on the computational hardness of counting stable configurations for S(y)DSs (e.g., [46]), as well as for parallel and asynchronous Hopfield networks (e.g., [36, 37]), are for the underlying graphs with the maximum node degree of 4 or higher. We refer the reader to [46] for a more detailed discussion of some of the related work.

5 Conclusions and Future Directions

Large-scale distributed computational and communication systems are often characterized by the property that, while the individual components may be relatively simple and their behavior well-understood, due to the interaction among those components and the interdependencies among different processes taking place at different components, the overall system behavior can become extremely complex.

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, based on the general model of communicating finite state machines, as an appropriate class of discrete-time, discrete-state dynamical systems. We specifically focus on the problem of enumerating how many stable configurations (FPs) 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 enumeration problems for 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 such an SDS or SyDS is *bipartite* and *uniformly sparse* - and can also be made *regular*;

- each local update rule is required to be a *symmetric* Boolean function; and

- the nodes of the S(y)DS use *only two* different update rules.

In particular, we have shown in this work that important enumeration 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 graphs of SDSs and SyDSs, to uniformly sparse bipartite graphs. Two, insofar as each agent's *individual behavior* is concerned, we limited the node update rules to symmetric Boolean-valued functions.

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 in multi-agent systems can be obtained via coupling of rather simple and strictly local inter-agent interactions of a homogeneous, *mean field* kind. Further exploring the implications of our results herewith in the context of large-scale multi-agent systems will be the subject of our future work.

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

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

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

[20] C. Greenhill. "The Complexity of Counting Colourings and Independent Sets in Sparse Graphs and Hypergraphs", *Computational Complexity,* vol. 9, pp. 52 - 72, 2000

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

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

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

[36] P. Orponen. "On the computational complexity of discrete Hopfield nets", *Proc. 20th Int'l Colloqium on Automata, Languages and Programming (ICALP '93),* Springer LNCS series, vol. 700, pp. 215 - 226, 1993

[37] P. Orponen. "Computational complexity of neural networks: a survey", *Nordic J. Comp.,* vol. 1 (1), pp. 94 - 110, 1994

[38] C. Papadimitriou. *"Computational Complexity"* , Addison-Wesley, Reading, Massachusetts, 1994

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

[40] D. Roth. "On the Hardness of Approximate Reasoning", *Artificial Intelligence,* vol. 82, pp. 27-302, 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] P. Tosic, G. Agha. "Concurrency vs. Sequential Interleavings in 1-D Threshold Cellular Automata" APDCM Workshop, in Proc. IEEE Int'l Parallel & Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mexico, USA, April 26-30, 2004

[45] 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

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

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

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

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

[50] G. Weiss (ed.), *"Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence",* The MIT Press, Cambridge, Massachusetts, 1999

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

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

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