Discrete Mathematics and Theoretical Computer Science Proceedings **AA (DM-CCG)**, 2001, 95–110

Gardens of Eden and Fixed Points in Sequential Dynamical Systems

Christopher L. Barrett [1], Harry B. Hunt III [2,3], Madhav V. Marathe [1], S. S. Ravi [2,3], Daniel J. Rosenkrantz [2], Richard E. Stearns[2], and Predrag T. Tosic[1,4]

[1]*Los Alamos National Laboratory, P.O. Box 1663, MS M997, Los Alamos, NM 87545. Email: {barrett, marathe, p-tosic}@lanl.gov. The work is supported by the U.S. Department of Energy under contract W-7405-ENG-36.*

[2]*Department of Computer Science, University at Albany - SUNY, Albany, NY 12222. Email: {hunt, ravi, djr, res}@cs.albany.edu. Supported by a grant from Los Alamos National Laboratory and by NSF Grant CCR-97-34936.*

[3]*Part of the work was done while the authors were visiting the Basic and Applied Simulation Sciences Group (TSA-2) at the Los Alamos National Laboratory.*

[4]*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA*

received February 4, 2001, *revised April 30, 2001*, *accepted May 4, 2001.*

A class of finite discrete dynamical systems, called **Sequential Dynamical Systems** (SDSs), was proposed in [BMR99, BR99] as an abstract model of computer simulations. Here, we address some questions concerning two special types of the SDS configurations, namely Garden of Eden and Fixed Point configurations. A configuration $\mathcal{C}$ of an SDS is a Garden of Eden (GE) configuration if it cannot be reached from any configuration. A necessary and sufficient condition for the non-existence of GE configurations in SDSs whose state values are from a finite domain was provided in [MR00]. We show this condition is sufficient but not necessary for SDSs whose state values are drawn from an infinite domain. We also present results that relate the existence of GE configurations to other properties of an SDS. A configuration $\mathcal{C}$ of an SDS is a fixed point if the transition out of $\mathcal{C}$ is to $\mathcal{C}$ itself. The FIXED POINT EXISTENCE (or FPE) problem is to determine whether a given SDS has a fixed point. We show that the FPE problem is **NP**-complete even for some simple classes of SDSs (e.g., SDSs in which each local transition function is from the set {NAND, XNOR}). We also identify several classes of SDSs (e.g., SDSs with linear or monotone local transition functions) for which the FPE problem can be solved efficiently.

Keywords: Discrete Dynamical Systems, Cellular Automata, Computational Complexity

---

1 Introduction and Motivation

Sequential Dynamical Systems (henceforth referred to as SDSs) were proposed in [BR99, BMR99, BMR00] 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 [Be+99]. A precise definition of an SDS is given in Section 2. In simple terms, an SDS $\mathcal{S} = (G, \mathcal{F}, \pi)$ consists of three components. $G(V,E)$ is an undirected graph with n nodes with each node having a 1-bit state[†]. $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, with f_i denoting a symmetric Boolean function associated with node v_i. π is a permutation of (or a total order on) the nodes in V. A **configuration** of an SDS is an n-bit vector $(b_1, b_2, \ldots, b_n)$, where b_i is the value of the state of node v_i $(1 \leq i \leq n)$. A single SDS transition from one configuration to another is obtained by updating the state of each node using the corresponding Boolean function. These updates are carried out in the order specified by π.

SDSs are closely related to classical Cellular Automata (CA), a widely studied class of dynamical systems in physics and complex systems. They are also closely related to a recently proposed extension of CA called **graph automata** [NR98, Ma98] and to one-way cellular automata studied by Roka [Rk94]. The main difference between the graph automata and SDSs is the sequential ordering aspect. Recently other authors [HG99, Ga97, Rk94] have also considered this particular aspect. In particular, Huberman and Glance [HG99] discuss experimentally how certain simulations of n-person games exhibit very different (but probably more realistic) dynamics when the cells are updated sequentially as opposed to when they are updated in parallel. The issue of sequential ordering has been also discussed in our earlier work [MR00, BMR99, BH+00] in the context of developing a theory of large scale simulations. Examples of such systems include various national infrastructures such as transportation, power and communication. See `http://transims.tsasa.lanl.gov` for additional details on this program. To illustrate the applicability of the SDS-like formalizations, we give a simplified yet realistic example that arose in our work.

Example Our example is from the large scale transportation simulation project at the Los Alamos National Laboratory called **TRANSIMS**.[‡] In this project, an SDS-based approach was used to micro-simulate every vehicle in an urban transportation network. For ease of exposition, we assume a single lane road which can be modeled as a one dimensional array of cells, with each cell representing a certain segment of the road. The state of each car (driver) may assume one of $v_{max} + 1$ possible values; these values correspond to discrete speeds from 0 to v_{max}. The state of each cell may assume one of $v_{max} + 2$ different values, the additional value being used to represent an empty cell. In the **TRANSIMS** system implementation, v_{max} was usually a small integer (such as 5). At each instant, the behavior of a car (e.g. whether the speed increases, decreases or remains the same) is a function of its state and the state of the car that is immediately ahead. By associating a variable with each grid cell, the time evolution of the system can be cast as the time evolution of the corresponding SDS. An important point to note is that unlike CA (which are synchronous), different choices of the order of updating the cells may yield completely different dynamics in case of SDSs. For instance, updating the states from front to back acts like a perfect predictor and thus never yields clusters of vehicles. On the other hand, updating downstream yields more realistic traffic dynamics [BWO95].

Given the importance of SDS in developing large scale simulations, we focus on theoretical questions aimed at understanding the phase space structure of our simulations. We focus on the computational complexity of the problems concerning two special types of the SDS configurations, namely the Garden of Eden and the Fixed Point configurations. Both problems have important counterparts in the context

† The restriction to binary states is a mathematical convenience, and allows us to present stronger lower bound results.

‡ TRANSIMS is an acronym for the "TRansportation ANalysis and SIMulation System". See `http://transims.tsasa.lanl.gov` for details.

of understanding large scale simulations. For example, the Garden of Eden question is directly related to liveness properties of certain network protocols [GC86]. Our conclusion is that these questions are, in general, computationally intractable. However, we identify a number of special classes of SDSs for which the questions can be answered efficiently. Several of our results are also applicable to cellular automata (CA) and graph automata (GA).

The remainder of the paper is organized as follows. In Section 2 we provide the necessary definitions. Section 3 defines the problems considered in this paper, summarizes our results and some related results from the literature. Sections 4 and 5 present our results for GARDEN OF EDEN EXISTENCE and FIXED POINT EXISTENCE problems respectively.

2 Definitions

We begin with a formal definition of sequential dynamical systems. Our definition closely follows the original definition of SDS in [BMR99, BMR00, MR00] We also recall basic definitions of the phase space parameters considered in this paper.

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

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

2. Each node has one bit of memory, called its **state**. The state of node i, denoted by s_i, takes on a value from $\mathbb{F}_2 = \{0,1\}$. We use δ_i to denote the degree of node i. Each node i is associated with a *symmetric Boolean function* $f_i : \mathbb{F}_2^{\delta_i+1} \to \mathbb{F}_2$, for $1 \le i \le n$. We refer to f_i as the **local transition function**. The inputs to f_i are the state of i and the states of the neighbors of i. By "symmetric" we mean that the function value does not depend on the order in which the input bits are specified; that is, the function value depends only on how many of its inputs are 1. We use $\mathcal{F}$ to denote $\{f_1, f_2, \ldots, f_n\}$.

3. Finally, π is a permutation of $\{1,2,\ldots,n\}$ specifying the order in which nodes update their states using their local transition functions. Alternatively, π can be envisioned as a total order on the set of nodes.

Computationally, the transition of an SDS from one configuration to another involves the following steps:

---

for $i = 1$ **to** n **do**
(1) Node $\pi(i)$ evaluates $f_{\pi(i)}$. (This computation uses the *current* values of the state of $\pi(i)$ and those of the neighbors of $\pi(i)$.)
(2) Node $\pi(i)$ sets its state $s_{\pi(i)}$ to the Boolean value computed in Step (1).
end-for

Stated another way, the nodes are processed in the *sequential* order specified by the permutation π. The "processing" associated with a node consists of computing the value of the node's Boolean function and changing its state to the computed value.

We point out that the assumption of 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 lower bounds for such SDSs imply stronger lower bounds for computing phase space properties of CA and communicating finite state machines (CFSMs). Second, symmetry provides one possible way to model "mean field effects" used in statistical physics and studies of other large-scale systems. A similar assumption has been made in [BPT91].

Recall that a configuration of an SDS is a bit vector $(b_1, b_2, \ldots, b_n)$. A configuration $\mathcal{C}$ of an SDS $\mathcal{S} = (G, \mathcal{F}, \pi)$ can also be thought of as a function $\mathcal{C}\colon V \to \mathbb{F}_2$. Given a configuration $\mathcal{C}$, the state of a node v in $\mathcal{C}$ is denoted by $\mathcal{C}(v)$; for a subset W of nodes, $\mathcal{C}(W)$ denotes the states of the nodes in W. We refer to $\mathcal{C}(W)$ as a **subconfiguration** of $\mathcal{C}$. 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 update of node states in the order given by π. Thus, $F_{\mathcal{S}} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a global function on the set of configurations.

The function $F_{\mathcal{S}}$ can therefore be considered as defining the dynamic behavior of SDS $\mathcal{S}$. We also say that SDS $\mathcal{S}$ moves from a configuration $\mathcal{C}$ to a configuration $F_{\mathcal{S}}(\mathcal{C})$ in a single transition step. Assuming each node function f_i is computable in time polynomial in the size of the description of $\mathcal{S}$, clearly each transition step will also take polynomial time (in the size of the SDS's description).

The configuration reached by applying the global transition function $F_{\mathcal{S}}$ to configuration $\mathcal{C}$ for t transition steps is denoted by ${F_{\mathcal{S}}}^t(\mathcal{C})$.

The **phase space** of SDS $\mathcal{S}$, denoted by $\mathcal{P}_{\mathcal{S}}$, is a directed graph with one node for each of the 2^n possible configurations; there is a directed edge from the node representing configuration $\mathcal{C}'$ to that representing configuration $\mathcal{C}$ if $\mathcal{S}$ moves from $\mathcal{C}'$ to $\mathcal{C}$ in one transition.

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

Definition 2.2 *Given two configurations $\mathcal{C}'$ and $\mathcal{C}$ of an SDS $\mathcal{S}$, $\mathcal{C}'$ 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, $\mathcal{S}$ evolves from $\mathcal{C}'$ to $\mathcal{C}$ in one or more transitions.*

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

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

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

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

Definition 2.5 *A configuration $\mathcal{C}$ of an SDS is a* **cycle configuration** *if $\mathcal{C}$ is on a cycle of length 2 or more in the phase space or, equivalently, if there is an integer $t \geq 2$ such that ${F_{\mathcal{S}}}^t(\mathcal{C}) = \mathcal{C}$.*

Alternatively, $\mathcal{C}$ is a **cycle configuration** if it is reachable from itself in two or more transitions (or, equivalently, if it is its own ancestor, but not a predecessor).

Definition 2.6 *A configuration $\mathcal{C}$ of an SDS is a* **transient configuration** *if $\mathcal{C}$ is neither a fixed point nor a cycle configuration.*

As the name suggests, transient configurations, unlike fixed points and cycle configurations, are never revisited. Notice that a GE configuration is a special case of a transient configuration; a GE configuration is not reachable in one or more transitions from *any* configuration including itself.

Definition 2.7 *An SDS $\mathcal{S}$ is* **invertible** *if the function $F_{\mathcal{S}}$ is a bijection.*

As will be shown in Section 4, for SDSs where the domain of state values is finite, there is a close relationship between invertibility and the existence of transient and GE configurations.

Let F be any set of symmetric Boolean functions. We use F-SDS to denote an SDS where each local transition function is from the set F.

The definition of an SDS can be extended to obtain several variants. A brief description of these SDS variants is given below.

As defined, the state of an SDS is a Boolean value and the functions associated with the nodes are symmetric and Boolean. When we allow the state of each node to take on values from a finite domain and the node functions to produce values from the domain, we obtain a **Finite Range SDS** (FR-SDS). If the states may store unbounded values and the node functions may also produce unbounded values, we obtain a **Generalized SDS** (Gen-SDS).

A **Linear SDS** is one in which each local transition function is a linear combination of its inputs. To be more precise, consider each node v_i, and let $N(i) = \{v_{i_1}, v_{i_2}, \ldots, v_{i_r}\}$ denote the neighbors of v_i. Let $N'(i) = N(i) \cup \{v_i\}$. In a linear SDS, each local transition function f_i has the following form:

$$f_i(s_i, s_{i_1}, \ldots s_{i_r}) = \alpha_i + \sum_{v_j \in N'(i)} a_{ij} s_j. \tag{1}$$

Here, α_i and a_{ij} ($1 \leq i \leq n$ and $1 \leq j \leq r$) are (scalar) constants, s_j is the state value of node v_j and the arithmetic operations (addition and scalar multiplication) are assumed to be carried out over a *field*. We assume that the field operations can be carried out efficiently. Under this assumption, it is well known that solving a set linear equations over the field can be done in polynomial time. We use this fact in Section 5. When the state of each node is Boolean, each linear local transition function is either XOR (exclusive or) or XNOR (the complement of exclusive or).

A **Synchronous Dynamical System** (SyDS) is an SDS *without* the node permutation. In an SyDS, at each time step, all the nodes synchronously compute and update their state values. Thus, SyDSs are similar to finite CA except that in SyDSs, nodes may be interconnected in an arbitrary fashion while in a cellular automaton, nodes are interconnected in a regular fashion (e.g. line, grid). We can extend the definition of an SyDS to obtain an FR-SyDS and a Gen-SyDS in a manner similar to that of SDS.

One class of Boolean functions considered in this paper is that of monotone functions. A definition of this class is given below.

Definition 2.8 *Given two Boolean vectors $X = \langle x_1, x_2, \ldots, x_n \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$, define the relation "$\preceq$" as follows: $X \preceq Y$ if $x_i \leq y_i$ for all i, $1 \leq i \leq n$. An n-input Boolean function f is* **monotone** *if $X \preceq Y$ implies that $f(X) \leq f(Y)$.*

The above definition of monotonicity can be extended to any domain for which there is a partial or total order on the elements. This extension is indicated below.

Definition 2.9 *Let $\mathcal{D}$ be a set with a partial or total order $\leq$ on its elements. Given two vectors in $\mathcal{D}^n$, $X = \langle x_1, x_2, \ldots, x_n \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$, define the relation "$\preceq$" as follows: $X \preceq Y$ if $x_i \leq y_i$, $1 \leq i \leq n$. An n-input function $f : \mathcal{D}^n \to \mathcal{D}$ is* **monotone** *if $X \preceq Y$ implies that $f(X) \leq f(Y)$.*

We observe that even if the elements of $\mathcal{D}$ are totally ordered by $\leq$, the induced order $\preceq$ on the set of n-tuples of elements of $\mathcal{D}$ (that is, the order on the set of configurations in the corresponding phase space) will be only a *partial* order. Our results on monotone SDSs do not depend on the exact nature of order $\leq$. For clarity, we assume that the order $\leq$ is total; this, indeed, is the case for Boolean SDSs as well as FR-SDSs with the local transition functions defined over the usual domains (such as Z_k).

3 Problems Considered, Summary of Results and Previous Work

Given an SDS $\mathcal{S}$, let $|\mathcal{S}|$ denote the size of the representation of $\mathcal{S}$. In general, this includes the number of nodes and edges, and the description of the local transition functions. When Boolean local transition functions are given as tables, $|\mathcal{S}| = O(m+|T|n)$, where $|T|$ denotes the maximum size of the table, n is the number of nodes and m is the number of edges in the underlying graph. For a node v with degree δ_v, the size of the table specifying an arbitrary Boolean function is $O(2^{\delta_v})$, while the size of the table specifying a symmetric Boolean function is $O(\delta_v)$. We assume that evaluating any local transition function given values for its inputs can be done in polynomial time.

In this paper, we study computational aspects of the following two problems concerning SDSs:

1. Given an SDS $\mathcal{S} = (G, \mathcal{F}, \pi)$, the GARDEN OF EDEN EXISTENCE problem (abbreviated as GEE) is to determine whether $\mathcal{S}$ has a GE configuration.

2. Given an SDS $\mathcal{S} = (G, \mathcal{F}, \pi)$, the FIXED POINT EXISTENCE problem (abbreviated as FPE) is to determine whether $\mathcal{S}$ has a fixed point.

We now summarize our results for the above problems. A necessary and sufficient condition for the non-existence of GE configurations in case of SDSs where each state value is from a finite set was given in [MR00]. We prove that the condition is sufficient but not necessary for the non-existence of GE configurations in SDSs whose domain of state values may be infinite. We also obtain results that relate the existence of GE configurations in FR-SDSs to other properties (e.g., existence of transient states, invertibility) of such SDSs.

We show that, in general, the FPE problem is **NP**-complete for SDSs such that each of their local transition functions is symmetric and Boolean. We present polynomial time algorithms for the FPE problem for several restricted classes of SDSs. We give an algorithm that uses only $O(n)$ evaluations of local transition functions (where n is the number of nodes) to decide whether an SDS such that each of its local transition functions is from the set $\{$AND, OR, NAND, NOR$\}$ has a fixed point. For SDSs where each local transition function is monotone, the answer to the FPE problem is always "yes"; we present an algorithm that uses only $O(m)$ evaluations of local transition functions (where m is the number of edges in the underlying graph) to find a fixed point for such SDSs. We also extend this result to FR-SDSs.

As mentioned earlier, SDSs are closely related to the graph automata model studied in [Ma98, NR98] and the one-way cellular automata studied by Roka [Rk94]. In fact, FR-SyDS exactly correspond to graph automata as defined in [NR98]. Computational aspects of CA have been studied by a number of researchers; see for example [Wo86, Gu89, Gr87, Su95]. Much of this work addresses decidability of properties for infinite CA. Barrett, Mortveit and Reidys [BMR99, BMR00, MR00] and Laubenbacher and Pareigis [LP00] investigate the mathematical properties of sequential dynamical systems. The complexity of the PREDECESSOR EXISTENCE problem ("Given an SDS and a configuration $\mathcal{C}$, does $\mathcal{C}$ have a predecessor?") and its generalizations (e.g., the ancestor existence) were studied by Sutner [Su95] and Green [Gr87]. Sutner also established the efficient solvability of the PREDECESSOR EXISTENCE problem for CA with a fixed neighborhood radius. In our earlier papers, we studied the computational complexity of several phase space questions for SDSs. These include REACHABILITY, PREDECESSOR EXISTENCE and the PERMUTATION EXISTENCE problems.

Invertibility of CA has been extensively studied starting with the work of Richardson [Ri72] and Amoroso and Patt [AP72]. See [MM98, Su98, TM90] for additional details on this topic. Note that a Garden of Eden configuration exists iff the global map of the SDS fails to be surjective (onto). Sutner [Su95] shows that, given a linear CA, deciding if its global map is surjective can be accomplished in quadratic time. The result holds for finite domains but with potentially infinite number of cells. The results in the above cited papers are not directly applicable to SDSs due to the nature of node update. Indeed, we can show that the characterization of Garden of Eden existence for finite SDSs in terms of local transition functions is not applicable to CA (or to GA).

4 Results for GARDEN OF EDEN EXISTENCE

A necessary and sufficient condition for the non-existence of GE configurations in SDSs where the state values are from a finite domain was developed in [MR00]. To express the condition using the notation developed here, we need the following definitions.

Definition 4.1 *Let $\mathcal{D}$ be a set and let $\phi(x_1, x_2, \ldots, x_k) \ : \ \mathcal{D}^k \longrightarrow \mathcal{D}$ be a function of k variables. Let $X = \{x_1, x_2, \ldots, x_k\}$. Variable x_i is* **essential** *for ϕ if for each combination $\alpha = (q_1, q_2, \ldots, q_{i-1}, q_{i+1}, \ldots, q_k)$ of values of the $k-1$ variables in $X - \{x_i\}$, where $q_j \in \mathcal{D}$, $1 \leq j \leq k$ and $j \neq i$, the single variable function $\phi^\alpha(x_i) = \phi(q_1, q_2, \ldots, q_{i-1}, x_i, q_{i+1}, \ldots, q_k)$ is a surjection (i.e., the range of $\phi^\alpha(x_i)$ is $\mathcal{D}$).*

Definition 4.2 *A local transition function f_i for node v_i is* **self-essential** *if variable s_i is essential for f_i.*

Using the above definitions, the characterization of [MR00] (done in the context of invertibility but applicable here) can be stated as follows:

> *A finite domain SDS has no GE configurations if and only if each of its local transition functions is self-essential.*

We now prove that the sufficiency part of this result holds for SDSs with infinite domains as well.

Theorem 4.1 *Suppose every local transition function of a Gen-SDS $\mathcal{S} = (G(V,E), \mathcal{F}, \pi)$ is self-essential. Then $\mathcal{S}$ does not have a GE configuration.*

Proof: When each local transition function is self-essential, we show that every configuration has a predecessor.

Let $\mathcal{C} = (b_1, b_2, \ldots, b_n)$ be a given configuration. We show the existence of a predecessor configuration $\mathcal{C}' = (b'_1, b'_2, \ldots, b'_n)$ as follows. Without loss of generality, assume that the node permutation π is $\langle v_1, v_2, \ldots, v_{n-1}, v_n \rangle$. Consider the nodes of the SDS in the reverse order of π. Thus, the first node considered is v_n with local transition function f_n. Let $N(n) = \{v_{n_1}, v_{n_2}, \ldots, v_{n_r}\}$ denote the set of neighbors of v_n. Since we want the configuration $\mathcal{C}'$ to be a predecessor of $\mathcal{C}$, the following condition must be satisfied:

$$f_n(s_n, b_{n_1}, b_{n_2}, \ldots, b_{n_r}) = b_n. \tag{2}$$

Here, s_n is the unknown value of the state of v_n before f_n is evaluated. Since f_n is self-essential, for the combination $\alpha = (b_{n_1}, b_{n_2}, \ldots, b_{n_r})$ of values, the single-variable function $f_n^\alpha(s_n)$ is a surjection. Thus, there is at least one value for the variable s_n which satisfies Equation (2). We set b'_n to an arbitrarily chosen solution to the equation.

As we proceed in the reverse order of π, when we consider the node f_i $(1 \le i < n)$, we obtain an equation similar to the one for node v_n shown above. In the equation, the only unknown is s_i; for each neighbor v_j of v_i, where $j > i$, we use the value b'_j which has already been computed; for each neighbor v_j of v_i, where $j < i$, we use the given value b_j from $\mathcal{C}$. Since f_i is self-essential, the resulting single-variable equation has a solution. The value of b'_i is set to one of these solution values. By repeating this process, the configuration $\mathcal{C}'$ is obtained. ∎

The following proposition points out the condition of Theorem 4.1 is not necessary in the case of SDSs with infinite domains.

Proposition 4.1 *When the domain of state values is infinite, there exists an SDS without GE configurations, even though one of the local transition functions is not self-essential.*

Proof: Consider an SDS $\mathcal{S}$ with two nodes $\{v_1, v_2\}$ and with a single edge joining the two nodes. The domain of state values for both nodes is the set of nonnegative integers and the permutation $\pi = \langle v_1, v_2 \rangle$. The local transition functions $f_1(s_1, s_2)$ and $f_2(s_1, s_2)$ are defined as follows.

$$\begin{array}{rcll} f_1(s_1,s_2) & = & s_1, & \text{if } s_2 \neq 0 \\ & = & s_1+1, & \text{otherwise.} \end{array}$$

$$f_2(s_1,s_2) = \lfloor s_2/2 \rfloor$$

The function f_1 is not self-essential, since when $s_2 = 0$, the equation $f_1(s_1, 0) = 0$ does not have a solution.

We now argue that every configuration of $\mathcal{S}$ has a predecessor, thus showing that $\mathcal{S}$ does not have a GE configuration. To see this, note that any configuration of the form (x, y), where $y > 0$, has $(x, 2y)$ as a predecessor and that any configuration of the form $(x, 0)$ has $(x, 1)$ as a predecessor. ∎

When the local transition functions are symmetric and Boolean, the necessary and sufficient condition for the non-existence of GE configurations can be expressed in a simple form as indicated in [MR00]: An SDS with symmetric Boolean local transition functions has no GE configurations if and only if each local transition function is either XOR or XNOR. When local transition functions are arbitrary Boolean functions given as Boolean formulas, it is easy to see that determining whether a function is self-essential is **Co-NP**-complete. A characterization of GE existence for SDSs with infinite domains is open.

For FR-SDSs (i.e., SDSs with finite phase spaces), existence of GE configurations can also be related to other properties such as invertibility and existence of transient configurations. The following theorem shows this relationship.

Theorem 4.2 *Let $\mathcal{S}$ be a FR-SDS. Then the following statements are equivalent:*
(i) $\mathcal{S}$ has a transient configuration.
(ii) $\mathcal{S}$ has a GE configuration.
(iii) $\mathcal{S}$ has a configuration with two or more predecessors.
(iv) $\mathcal{S}$ is not invertible.

Proof: By definition, each configuration $\mathcal{C}$ has exactly one successor, namely $F_{\mathcal{S}}(\mathcal{C})$. That is, in the phase space $\mathcal{P}_{\mathcal{S}}$, each node has outdegree equal to 1. Thus, if the phase space has N nodes, then the number of directed edges is also N. These facts will be used throughout the proof.

(i) ⇒ (ii): Suppose $\mathcal{C}$ is a transient configuration. If $\mathcal{C}$ has no predecessor, then it is a garden of eden, and we are done. Else, let Pred($\mathcal{C}$) denote a predecessor of $\mathcal{C}$. (When a configuration has two or more predecessors, we choose one arbitrarily.) Consider the sequence Pred($\mathcal{C}$), Pred(Pred($\mathcal{C}$)), Pred(Pred(Pred($\mathcal{C}$))), ... In this sequence, no configuration can repeat, since a repeating configuration would have to have outdegree of at least 2. Now, the finiteness of the phase space implies that the sequence ends in a configuration with no predecessor, that is, a GE configuration.

(ii) ⇒ (iii): Suppose $\mathcal{C}$ is a GE configuration. Thus, in the phase space, the indegree of $\mathcal{C}$ is zero. If all the other nodes in the phase space have indegree at most 1, then the total number of directed edges will be less than the number of nodes. This is a contradiction and hence there is a node in the phase space with indegree at least 2. In other words, there is a configuration with two or more predecessors.

(iii) ⇒ (iv): Suppose configuration $\mathcal{C}$ has two or more predecessors. Then the function $F_{\mathcal{S}}$ of the SDS is not one-to-one, and therefore not a bijection. That is, $\mathcal{S}$ is not invertible.

(iv) ⇒ (i): Suppose $\mathcal{S}$ is not invertible; that is, $F_{\mathcal{S}}$ is not a bijection. Since the domain of $\mathcal{S}$ is finite, the global map $F_{\mathcal{S}}$ of $\mathcal{S}$ is not bijective iff it is not surjective iff it is not injective. In particular, if $F_{\mathcal{S}}$ is not a surjection (i.e., not an onto function), then there exists a configuration $\mathcal{C}'$ in the phase space whose indegree is zero. Then $\mathcal{C}'$ is a garden of eden, and, in particular, it is a transient configuration. ■

So far, we considered the problem of determining whether a given SDS has a GE configuration. One can also consider a different problem in this context: Given an SDS $\mathcal{S}$ and a configuration $\mathcal{C}$, determine whether $\mathcal{C}$ is a GE configuration. We observe that this problem is the complement of determining whether a given configuration has a predecessor. The latter problem (PREDECESSOR EXISTENCE) is known to be **NP**-complete (see [BH+01]). This reference also identifies a number of restricted classes of SDSs (e.g. linear-SDSs, SDSs whose underlying graphs are treewidth bounded) for which the PREDECESSOR EXISTENCE problem can be solved efficiently. From these results, it follows that testing whether a given configuration is a GE configuration is **Co-NP**-complete in general; also, the problem is efficiently solvable for any class of SDSs for which the PREDECESSOR EXISTENCE problem is efficiently solvable.

Let a configuration $\mathcal{C}$ of an SDS be called a **Strong Garden of Eden** (SGE) configuration if it is a Garden of Eden configuration under *every* node permutation. Analogous to the GARDEN OF EDEN EXISTENCE problem, the following problem (called the STRONG GARDEN OF EDEN EXISTENCE problem, abbreviated as SGEE) can be formulated: Given the underlying graph $G(V,E)$ and the set $\mathcal{F}$ of local transition functions of an SDS $\mathcal{S}$, determine whether $\mathcal{S}$ has an SGE configuration. A characterization of hardness of determining the existence of an SGE configuration is open. The following observation provides some classes of SDSs which have SGE configurations. We omit the straightforward proof.

Observation 4.1 *Let F denote any of the sets* $\{OR\}$, $\{AND\}$, $\{NOR\}$ *and* $\{NAND\}$. *Every F-SDS whose underlying graph has at least one edge has an SGE configuration.* ■

5 Results for FIXED POINT EXISTENCE

In this section, we consider the complexity of the FIXED POINT EXISTENCE problem for several classes of SDSs. In particular, we show that the FIXED POINT EXISTENCE problem is **NP**-complete for SDSs with symmetric Boolean local transition functions. We also identify several restricted classes of SDSs for which the FPE problem can be solved efficiently. We begin with some simple observations about fixed points.

Observation 5.1 *[MR00] A fixed point configuration for an SDS remains a fixed point under all permutations of the nodes. Furthermore, such a configuration is also a fixed point of the corresponding SyDS (i.e., it remains a fixed point if we update all the node values "in parallel").* ∎

Thus, to establish that a given configuration is a fixed point, we can consider the nodes in an arbitrary order. The next observation indicates a property of NAND and NOR functions in any fixed point.

Observation 5.2 *Let $\mathcal{S}$ be an SDS and let $\mathcal{C}$ be any fixed point configuration for $\mathcal{S}$.*

1. *For every node y whose local transition function is NAND, $\mathcal{C}(y) = 1$.*
2. *For every node y whose local transition function is NOR, $\mathcal{C}(y) = 0$.* ∎

Our **NP**-hardness proofs for the FPE problem use appropriate reductions from the following problem.

POSITIVE EXACTLY 1-IN-3 3SAT (PE3SAT)

Instance: A set $X = \{x_1, x_2, \ldots, x_n\}$ of n Boolean variables and a collection $C = \{c_1, c_2, \ldots, c_m\}$ of m clauses, where each clause contains exactly three positive (unnegated) literals.

Question: Is there a truth assignment to the variables such that each clause contains exactly one true literal?

PE3SAT is known to be **NP**-complete [GJ79]. We also need a restricted version of PE3SAT where each variable occurs in an odd number of clauses. We will refer to this restricted version as ODD-PE3SAT. The following proposition establishes the **NP**-completeness of ODD-PE3SAT.

Proposition 5.1 ODD-PE3SAT *is* **NP***-complete.*

Proof: ODD-PE3SAT is obviously in **NP**. We prove its **NP**-hardness through a reduction from PE3SAT. Consider an instance of PE3SAT. For each variable x_i that occurs in an even number of clauses, we introduce two new variables x_i^1 and x_i^2, and add the clause $\{x_i, x_i^1, x_i^2\}$ to the existing set of clauses. The variables x_i^1 and x_i^2 don't occur in any other clause. After this construction, each variable x_i occurs in an odd number of clauses and the new variables x_i^1 and x_i^2 occur in exactly one clause. So, the construction produces an instance of ODD-PE3SAT. It is easily verified that the resulting instance of ODD-PE3SAT has a solution if and only if the given instance of PE3SAT has a solution. ∎

We are now ready to prove the **NP**-completeness of FPE for SDSs with symmetric and Boolean local transition functions.

Theorem 5.1 *The* FIXED POINT EXISTENCE *problem is* **NP***-complete for the following restricted classes of SDSs: (a) $\{$NAND, XNOR$\}$-SDSs, (b) $\{$NAND, XOR$\}$-SDSs, (c) $\{$NOR, XNOR$\}$-SDSs and (d) $\{$NOR, XOR$\}$-SDSs.*

Proof: The FIXED POINT EXISTENCE problem is in **NP** since one can guess a configuration $\mathcal{C}$ and verify in polynomial time that the transition out of $\mathcal{C}$ is to $\mathcal{C}$ itself. For the sake of brevity, we will present the proof of **NP**-hardness for $\{$NAND,XNOR$\}$-SDSs. The proofs for the other cases are similar.

We use a reduction from ODD-PE3SAT. Let x_1, x_2, …, x_n denote the variables and let c_1, c_2, …, c_m denote the clauses in the given instance of ODD-PE3SAT. For each variable x_i, we create a node x_i, $1 \le i \le n$. For each clause c_j, we create two nodes c_j^1 and c_j^2, $1 \le j \le m$. When a variable x_i occurs in clause c_j, we add the two edges $\{x_i, c_j^1\}$ and $\{x_i, c_j^2\}$. The local transition function associated with each variable node x_i is XNOR (or even parity). The local transition function associated with each node c_j^1 is XNOR and that associated with each node c_j^2 is NAND. Thus, we obtain an instance of an {NAND, XNOR}-SDS.

Suppose the given instance of ODD-PE3SAT has a solution. Consider the following configuration. Set each node c_j^1 of the SDS (whose associated function is XNOR) to 0 and each node c_j^2 (whose associated function is NAND) to 1, $1 \leq j \leq n$. Set each variable node x_i to the truth value given by the solution to ODD-PE3SAT. To see that this configuration is a fixed point, first consider the variable nodes. (By Observation 5.1, we can consider the nodes of the SDS in an arbitrary order.) Suppose node x_i has value 1. (A similar proof holds for a node x_i with value 0.) The inputs to the even parity function at x_i are the nodes c_j^1 and c_j^2 for each j such that x_i occurs in clause c_j. In the chosen configuration, the value of each node c_j^1 is 0 and that of c_j^2 is 1. Since x_i appears in an odd number of clauses, the number of 1's in the input to the function at x_i, including the value of x_i itself, is even. Thus, the output of the function at x_i remains as 1. For each node c_j^1 which also computes the even parity function, exactly one of its inputs is 1 because of the property of the solution to ODD-PE3SAT and the fact that c_j^1 was set to 0 in the chosen configuration. Thus, the function value at node c_j^1 remains 0, $1 \leq j \leq m$. For each node c_j^2 which computes the NAND function, two of its inputs are 0 (because of the property of the solution to ODD-PE3SAT). So, function value at that node also remains as 1. Thus, the chosen configuration is indeed a fixed point.

Now, suppose the SDS has a fixed point $\mathcal{C}$. In $\mathcal{C}$, the value for each node c_j^2 ($1 \leq j \leq m$) that computes the NAND function must be 1 by Observation 5.2. Thus, not all three variables adjacent to the node c_j^2 can be set to 1 in $\mathcal{C}$. Now, consider the node c_j^1. Suppose $\mathcal{C}$ assigns the value 0 to c_j^1. (An identical argument holds if $\mathcal{C}$ assigns the value 1 to c_j^1.) Then, the number of 1's among the variable nodes that are adjacent to c_j^1 must be odd; that is, the number must be either 1 or 3. Since we just argued that the number of 1's cannot be 3, it follows that exactly one of the variable nodes to which c_j^1 is adjacent is 1. In other words, the values chosen by $\mathcal{C}$ for the variable nodes constitute a solution to the ODD-PE3SAT instance. This completes the proof of **NP**-hardness for {NAND, XNOR}-SDSs. ■

Next, we use techniques developed in [BH+01] to identify several restricted classes of SDSs for which the FPE problem can be solved in polynomial time. We begin with a straight-forward result concerning linear Gen-SDSs and linear Gen-SyDSs:

Theorem 5.2 *The* FPE *problem can be solved efficiently for the class of linear Gen-SDSs and linear Gen-SyDSs.*

Our next theorem identifies several restricted classes of SDSs (where the domain of state values is Boolean) for which the FPE problem can be solved in polynomial time.

Theorem 5.3 *Let F = {AND, OR, NAND, NOR}. The* FPE *problem for any F-SDS can be solved in linear time. Moreover, when the* FPE *problem has a solution, a fixed point can be found using $O(n)$ evaluations of local transition functions.*

Proof: Given $\mathcal{S}$, construct the following configuration $\mathcal{C}$. For every node v where the local transition function f_v is OR or NAND, set $\mathcal{C}(v) = 1$. For every node v where the local transition function f_v is AND or NOR, set $\mathcal{C}(v) = 0$. We claim that $\mathcal{S}$ has a fixed point if and only if $\mathcal{C}$ is a fixed point.

The "if" part of the proof is trivial. So, we will consider the "only if" part. Suppose $\mathcal{C}$ is not a fixed point. Consider finding the successor $\mathcal{C}'$ of $\mathcal{C}$. Since $\mathcal{C}$ is not a fixed point, there is a node u for which $\mathcal{C}(u) \neq \mathcal{C}'(u)$. Let y be the first such node in the permutation π and let f_y denote the local transition function at node y. The function f_y cannot be OR since $\mathcal{C}(y) = 1$ implies that $\mathcal{C}'(y) = 1$. Similarly, f_y cannot be AND since $\mathcal{C}(y) = 0$ implies that $\mathcal{C}'(y) = 0$. So, $f_y \in \{\text{NOR, NAND}\}$. We will show that f_y cannot be the NOR function. A dual argument can be given to show that f_y cannot be the NAND function.

Suppose the f_y is the NOR function. Then, $\mathcal{C}(y) = 0$ and $\mathcal{C}'(y) = 1$. Since y is the first node in π whose value in $\mathcal{C}'$ differs from its value in $\mathcal{C}$, at the time y is evaluated in the transition from $\mathcal{C}$ to $\mathcal{C}'$, all the neighbors of y have the same value they had in $\mathcal{C}$. Since $\mathcal{C}'(y) = 1$, all these neighbors had value 0 at this time. Therefore, the local transition function of each neighbor of y is either AND or NOR. Suppose there exists a fixed point $\mathcal{B}$ for $\mathcal{S}$. From Observation 5.2, $\mathcal{B}(y) = 0$. Also from Observation 5.2, for every neighbor z of y with local transition function NOR, $\mathcal{B}(z) = 0$. For every neighbor z of y with local transition function AND, since z is adjacent to y which has value 0 in $\mathcal{B}$, $\mathcal{B}(z) = 0$. But the conditions $\mathcal{B}(y) = 0$ and $\mathcal{B}(z) = 0$ for each neighbor z of y imply that $\mathcal{B}$ is not a fixed point. Therefore, $\mathcal{S}$ does not have a fixed point.

In view of the above result, to find a fixed point, we only need to verify whether the configuration $\mathcal{C}$ constructed above is a fixed point. Clearly, this involves only $O(n)$ local function evaluations, where n is the number of nodes in the underlying graph. ■

A monotone SDS (SyDS) is an SDS (SyDS) in which each local transition function is monotone. Our next theorem shows that every monotone SDS (or SyDS) has a fixed point. Moreover, a fixed point can be found efficiently. We begin with a straight forward, yet far-reaching observation:

Lemma 5.1 *Let $\mathcal{S}$ be a Gen-SDS or Gen-SyDS where each local transition function is monotone. Let $\mathcal{C}$ and $\mathcal{C}'$ be configurations of $\mathcal{S}$. If $\mathcal{C} \preceq \mathcal{C}'$, then $F_{\mathcal{S}}(\mathcal{C}) \preceq F_{\mathcal{S}}(\mathcal{C}')$.* ■

The Lemma turns out to be a very useful result for establishing various phase space properties of the *monotone* SDSs and SyDSs over various domains, as the next result on fixed points for the SDSs/SyDSs over finite domains shows.

Theorem 5.4 *Every monotone FR-SDS (or FR-SyDS) over domain $\mathcal{D}$ has a fixed point. Moreover, a fixed point can be found using $O(m|\mathcal{D}|)$ evaluations of local transition functions, where m is the number of edges in the graph.*

Proof: Let 0 denote the minimum element of $\mathcal{D}$ under the total order $\leq$. Let I_0 denote the configuration consisting of all 0's; notice that this configuration is a minimal configuration in the phase space under the induced partial order. From Lemma 5.1, it follows that the sequence of configurations starting from I_0 reaches a fixed point. Moreover, since each step in this sequence that changes the configuration increases at least one element of the configuration, the number of SDS transitions before the sequence reaches a fixed point is bounded by $n(|\mathcal{D}| - 1)$.

If the above process is carried out directly, the number of function evaluations would be $\Theta(n^2|\mathcal{D}|)$, since each transition uses n function evaluations. To improve the number of function evaluations, we start with the configuration I_0, and evaluate each node once. When the value of a node v changes, we schedule all neighbors of v for another evaluation. When the value of a node reaches the maximal value in $\mathcal{D}$ under the total order, the node is not re-evaluated. In this manner, the number of function evaluations for a node

v over the course of the algorithm is at most $|\mathcal{D}|$ times the degree of v. Therefore, the total number of function evaluations is at most $O(m|\mathcal{D}|)$. ■

When the local transition functions are monotone, symmetric and Boolean, the approach presented in the proof of Theorem 5.4 can be modified to obtain a fixed point in $O(n+m)$ time. This improvement exploits the fact that Boolean functions that are symmetric and monotone are simple **threshold functions**; that is, for each such function, there is an integer k such that the value of the function is 1 if the number of 1's in the input is at least k and 0 otherwise. (We permit nodes whose threshold is zero as well as nodes whose threshold may exceed their number of inputs. Thus, the configuration consisting of all 0's or the one consisting of all 1's need not be a fixed point for such SDSs.)

The improved algorithm for SDSs (and SyDSs) where each node computes a simple threshold function is obtained by recording for each node, the number of its neighbors whose value is 1. When this count reaches the threshold value, the value of the node changes to 1 and the count is incremented for all its neighbors. One can think of this process as if the value 1 is being propagated along the edges as follows. First, all nodes are 0 (since we start at the configuration 0^n), and all the edges are labeled by 0. Once a node changes its value from 0 to 1, all the edges incident to this node change their labels to 1; then we look at the neighbors of the updated node - has the number of incoming edges labeled by 1 reached (or surpassed) its threshold? We proceed propagating the node value jumps from 0 to 1 along their incident edges until no new nodes (and therefore no new edges) become 1; at this stage, the fixed point has been reached. It is now easy to see that the algorithm runs in $O(n+m)$ time.

Corollary 5.1 *For SDSs whose local transition functions are Boolean, symmetric and monotone, a fixed point can be obtained in $O(n+m)$ time.* ■

6 Conclusions And Future Work

We consider the computational aspects of Gardens of Eden and fixed points for SDSs. Our results point out that, in general, these problems are computationally intractable. We identify some special classes of SDSs for which these problems can be solved efficiently. We extend a sufficient condition for the non-existence of GE configurations to Gen-SDS over infinite domains. We also relate the existence of GE configurations to other phase space properties of SDSs (such as invertibility).

We close by mentioning some directions for further research. An important open problem is the characterization of Gen-SDSs (SDSs over infinite domains) with GE configurations. Recall that a strong GE configuration is one which remains a GE configuration under all node permutations. Characterizing SDSs with strong GE configurations is also open. Recently, we have made some progress on the problems of *counting* fixed points and gardens of eden in SDSs and SyDSs. Finally, identifying other restricted classes of SDSs for which the fixed point existence problem can be solved efficiently would also further our understanding of the complex behavior of SDSs.

Acknowledgements

This research was also funded by the LDRD-DR project *Foundations of Simulation Science* and the LDRD-ER project *Extremal Optimization* at the Los Alamos National Laboratory. We thank Christian Reidys and Paul Wollan (both from Los Alamos National Laboratory) for several discussions on topics related to this paper.

References

[AP72] S. Amoroso and Y. Patt. Decision procedures for surjectivity and injectivity of parallel maps for tessallation structures. *J. of Computer and System Sciences (JCSS)*, 6, pp. 448-464, 1972.

[BWO95] C. Barrett, M. Wolinsky and M. Olesen. Emergent Local Properties in Particle Hopping Traffic Simulations. in *Proc. Traffic and Granular Flow* Los Alamos Unclassified Internal Report, LAUR-95-4368, 1995.

[BH+00] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz and R. E. Stearns. Dichotomy Results for Sequential Dynamical Systems. Submitted for publication, Dec. 2000.

[BH+01] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz and R. E. Stearns. Predecessor and Permutation Existence Problems for Sequential Dynamical Systems. Submitted for publication, Jan. 2001.

[BR99] C. Barrett and C. Reidys. Elements of a theory of computer simulation I: sequential CA over random graphs. *Applied Mathematics and Computation*, 98, pp. 241–259, 1999.

[BMR99] C. Barrett, H. Mortveit, and C. Reidys. Elements of a theory of simulation II: sequential dynamical systems. *Applied Mathematics and Computation*, 1999, vol 107/2-3, pp. 121-136.

[BMR00] C. Barrett, H. Mortveit and C. Reidys. Elements of a theory of computer simulation III: equivalence of SDS. to appear in *Applied Mathematics and Computation*, 2000.

[Be+99] R.J. Beckman, et. al. TRANSIMS: Case Study, Dallas Ft-Worth. Los Alamos National Laboratory, LA UR 97-4502, 1999.

[BPT91] S. Buss, C. Papadimitriou and J. Tsitsiklis. On the predictability of coupled automata: An allegory about Chaos. *Complex Systems,* 1(5), pp. 525-539, 1991. Preliminary version in *Proc. 31st Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, Oct 1990.

[Du94] B. Durand. Inversion of 2D cellular automata: some complexity results. *Theoretical Computer Science,* 134(2), pp. 387-401, November 1994.

[GJ79] 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.

[Ga97] P. Gacs. Deterministic computations whose history is independent of the order of asynchronous updating. Tech. Report, Computer Science Dept, Boston University, 1997.

[GC86] M. Gouda, C. Chang. Proving Liveness for Networks of Communicating Finite State Machines. *ACM Transactions on Prog. Lang. and Systems (TOPLAS)* 8(1): 154-182, pp. 1986.

[Gr87] F. Green. NP-Complete Problems in Cellular Automata. *Complex Systems*, Vol. 1, No. 3, 1987, pp. 453–474.

[Gu89] H. Gutowitz (Editor). *Cellular Automata: Theory and Experiment* North Holland, 1989.

[HG99] B. Huberman and N. Glance. Evolutionary games and computer simulations. *Proc. National Academy of Sciences*, 1999.

[Hu87] L.P. Hurd, On Invertible cellular automata. *Complex Systems,* 1(1), pp. 69-80, 1987.

[LP00] R. Laubenbacher and B. Pareigis. Finite Dynamical Systems. Technical report, Department of Mathematical Sciences, New Mexico State University, Las Cruces, 2000.

[MM98] G. Manzini and L. Margara. Invertible Linear Cellular Automata over $\mathbb{Z}_m$ *J. of Computer and System Sciences (JCSS)*, 56, pp. 60-67, 1998.

[Ma98] B. Martin. A Geometrical Hierarchy of Graphs via Cellular Automata. Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, Aug. 1998.

[MR00] H. Mortveit, C. Reidys: Discrete sequential dynamical systems. in *Discrete Mathematics*, 2000.

[NR98] C. Nichitiu and E. Remila. Simulations of Graph Automata. Proc. MFCS'98 Satellite Workshop on Cellular Automata, Brno, Czech Republic, Aug. 1998.

[Rk94] Z. Roka. One-way cellular automata on Cayley graphs. *Theoretical Computer Science,* 132(1-2), pp. 259-290, September 1994.

[Ri72] D. Richardson, Tessellations with local transformations. *J. of Computer and System Sciences (JCSS)*, 6, pp. 373-388, 1972.

[Re00a] C. Reidys. Sequential dynamical systems: phase space properties. Advances in Applied Mathematics, to appear.

[Ro99] C. Robinson. *Dynamical systems: stability, symbolic dynamics and chaos*. CRC Press, New York, 1999.

[Su95] K. Sutner. On the computational complexity of finite cellular automata. *Journal of Computer and System Sciences,* 50(1), pp. 87-97, February 1995.

[Su98] K. Sutner. Computation theory of cellular automata. *MFCS'98 Satellite Workshop on Cellular Automata* Bruno, Czech Republic, Aug. 1998.

[SDB97] C. Schittenkopf, G. Deco and W. Brauer. Finite automata-models for the investigation of dynamical systems. *Information Processing Letters,* 63(3), pp. 137-141, August 1997.

[TM90] T. Toffoli, H. Margolus: Invertible cellular automata: A review. *Physica* **D**. 45, 1990.

[Wo86] S. Wolfram, Ed. *Theory and applications of cellular automata,* World Scientific, 1987.