Analysis Problems for Sequential Dynamical Systems and Communicating State Machines

CHRIS BARRETT[1], HARRY B. HUNT III[2,3], MADHAV V. MARATHE[1], S. S. RAVI[2,3], DANIEL J. ROSENKRANTZ[2], and RICHARD E. STEARNS[2]

[1] Los Alamos National Laboratory, MS M997, P.O. Box 1663, Los Alamos, NM 87545. Email: {barrett, marathe}@lanl.gov. The work is supported by the 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) of the Los Alamos National Laboratory.

Abstract. Informally, a sequential dynamical system (SDS) consists of an undirected graph where each node v is associated with a state s_v and a transition function f_v. Given the state value s_v and those of the neighbors of v, the function f_v computes the next value of s_v. The node transition functions are evaluated according to a specified total order. Such a computing device is a mathematical abstraction of a simulation system. We address the complexity of some state reachability problems for SDSs. Our main result is a dichotomy between classes of SDSs for which the state reachability problems are computationally intractable and those for which the problems are efficiently solvable. These results also allow us to obtain stronger lower bounds on the complexity of reachability problems for cellular automata and communicating state machines.

1 Introduction and motivation

We study the computational complexity of some state reachability problems associated with a new class of finite discrete dynamical systems, called **Sequential Dynamical Systems** (SDSs), proposed in [BR99,BMR99]. Informally, an SDS consists of an undirected graph where each node v is associated with a state s_v and a transition function f_v. Given the state value s_v and those of the neighbors of v, the function f_v computes the next value of s_v. The node transition functions are evaluated according to a specified total order. A formal definition of an SDS is given in Section 2.

SDSs are closely related to classical Cellular Automata (CA), a widely studied class of finite discrete dynamical systems used to model problems in physics and complex systems. Computability aspects of dynamical systems in general and cellular automata in particular have been widely studied in the literature (see for example, [Wo86,Gu89]). Dynamical systems are closely related to networks of communicating automata, transition systems and sequential digital circuits (see for example, [Ra92,AKY99,RH93,Hu73]).

In this paper, we will restrict ourselves to *Simple-SDSs*, that is, SDSs with the following additional restrictions: (i) the state of each node is Boolean and (ii) each local transition function is Boolean and symmetric. Our *hardness results* hold even for such simple-SDSs and thus imply analogous hardness results for more general models.

We study the computational complexity of determining some properties of SDSs. The properties studied include classical questions such as reachability ("Does a given SDS starting from configuration $\mathcal{C}$ ever reach configuration $\mathcal{C}'$?") and fixed points ("Does a given SDS have a configuration $\mathcal{C}$ such that once $\mathcal{C}$ is reached, the SDS stays in $\mathcal{C}$ forever?") that are commonly studied by the dynamical systems community. Specifically, we investigate whether such properties can be determined efficiently using computational resources that are polynomial in the size of the SDS representation.

The original motivation to develop a mathematical and computational theory of SDSs was to provide a formal basis for the design and analysis of *large-scale computer simulations*. Because of the widespread use of computer simulations, it is difficult to give a formal definition of a computer simulation that is applicable to all the various settings where it is used. An important characteristic of any computer simulation is the generation of global dynamics by iterated composition of local mappings. Thus, we view simulations as comprised of the following: (i) a collection of entities with state values and local rules for state transitions, (ii) an interaction graph capturing the local dependency of an entity on its neighboring entities and (iii) an update sequence or schedule such that the causality in the system is represented by the composition of local mappings. References [BWO95,BB+99] show how simulations of large-scale transportation systems and biological systems can be modeled using appropriate SDSs.

Following [BPT91], we say that a system is predictable if basic properties such as reachability and fixed point existence can be determined in time that is polynomial in the size of the system specification. Our **PSPACE**-completeness results for predicting the behavior of "very simple" systems essentially imply that the systems are not easily predictable; in fact, our results imply that no prediction method is likely to be more efficient than running the simulation itself. The results here can also be used to show that even simple SDSs are "universal" in that any reasonable model of simulation can be "efficiently locally simulated" by appropriate SDSs that can be constructed in polynomial time. The models investigated include cellular automata, communicating finite state machines, multi-variate difference equations, etc.

We undertake the computational study of SDSs in an attempt to increase our understanding of SDSs in particular and the complex behavior of dynamical systems in general. SDSs are discrete finite analogs of classical dynamical systems, and we aim to obtain a better understanding of "finite discrete computational analogs of chaos". As pointed out in [BPT91,Mo90,Mo91], computational intractability or unpredictability is the closest form of chaotic behavior that such systems can exhibit. Extending the work of [BPT91], we prove a dichotomy result between classes of SDSs whose global behavior is easy to predict and others for which the global behavior is hard to predict. In [Wo86], Wolfram posed the following three general questions in the chapter entitled "Twenty Problems in the Theory of Cellular Automata": (i) **Problem 16:** *How common are computational universality and undecidability* in CA? (ii) **Problem 18:** *How common is computational irreducibility in CA*? (iii) **Problem 19:** *How common are*

computationally intractable problems about CA? The results obtained here and in the companion papers [BH+01a,BH+01b,BH+01c] for SDSs (and for CA as direct corollaries) show that the answer to all of the above questions is "*quite common*". In other words, it is quite common for synchronous as well as sequential dynamical systems to exhibit intractability. In fact, our results show that such intractability is exhibited by *extremely simple* SDSs and CA.

2 Definitions and problem formulation

2.1 Formal definition of an SDS

As stated earlier, we will restrict our selves to *Simple*-SDSs. (Unless otherwise stated, we use "SDS" to mean a simple SDS.) Our definition closely follows the original definition of SDS in [BMR99].

A **Simple 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. Further, we denote by $N(i)$ the neighbors of node i in G, plus node i itself. Each node i is associated with a *symmetric Boolean function* $f_i : \mathbb{F}_2^{\delta_i + 1} \rightarrow \mathbb{F}_2$, $(1 \leq i \leq n)$. We refer to f_i as a **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**

(i) 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 $\pi(i)$.)

(ii) Node $\pi(i)$ sets its state $s_{\pi(i)}$ to the Boolean value computed in Step (i).

end-for

Stated another way, the nodes are processed in the *sequential* order specified by 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.

A **configuration** of an SDS is a bit vector $(b_1, b_2, \ldots, b_n)$, where b_i is the state value of node v_i. A configuration $\mathcal{C}$ of an SDS $\mathcal{S} = (G, \mathcal{F}, \pi)$ can also be thought of

as a function $\mathcal{C} : V \to \mathbb{F}_2$. 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 $\mathcal{C}$ at time t to a configuration $F_{\mathcal{S}}(\mathcal{C})$ at time $(t+1)$. The initial configuration (i.e., the configuration at time $t = 0$) of an SDS $\mathcal{S}$ is denoted by $\mathcal{I}$. Given an SDS $\mathcal{S}$ with initial configuration $\mathcal{I}$, the configuration of $\mathcal{S}$ after t time steps is denoted by $\xi(\mathcal{S}, t)$. We define $\xi(\mathcal{S}, 0) = \mathcal{I}$. For a configuration $\mathcal{C}$, we use $\mathcal{C}(W)$ to denote the states of the nodes in $W \subseteq V$ and $\mathcal{C}(v)$ to denote the state of a particular node $v \in V$.

A **fixed point** of an SDS $\mathcal{S}$ is a configuration $\mathcal{C}$ such that $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}$. An SDS $\mathcal{S}$ is said to **cycle** through a sequence of configurations $\langle \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_r \rangle$ if $F_{\mathcal{S}}(\mathcal{C}_1) = \mathcal{C}_2$, $F_{\mathcal{S}}(\mathcal{C}_2) = \mathcal{C}_3, \ldots, F_{\mathcal{S}}(\mathcal{C}_{r-1}) = \mathcal{C}_r$ and $F_{\mathcal{S}}(\mathcal{C}_r) = \mathcal{C}_1$. A fixed point is a cycle involving only one configuration. Any cycle involving two or more configurations is called a **limit cycle**. The **phase space** $\mathcal{P}_{\mathcal{S}}$ of an SDS $\mathcal{S}$ is a directed graph with one node for each configuration of $\mathcal{S}$. For any pair of configurations $\mathcal{C}$ and $\mathcal{C}'$, there is a directed edge from the node representing $\mathcal{C}$ to that representing $\mathcal{C}'$ if $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}'$.

2.2 Problems considered

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, edges and the description of the local transition functions. We assume that evaluating any local transition function given values for its inputs can be done in polynomial time.

The main problems studied in this paper deal with the *analysis* of a given SDS, that is, determining whether a given SDS has a certain property. These problems are formulated below.

Given an SDS $\mathcal{S}$, two configurations $\mathcal{I}$, $\mathcal{B}$, and a positive integer t, the t-REACHABILITY problem is to determine whether $\mathcal{S}$ starting in configuration $\mathcal{I}$ can reach configuration $\mathcal{B}$ in t or fewer time steps. If t is specified in unary, it is easy to solve this problem in polynomial time since we can execute the SDS for t steps and check whether configuration B is reached at some step. So, we assume that t is specified in binary.

Given an SDS $\mathcal{S}$ and two configurations $\mathcal{I}$, $\mathcal{B}$, the REACHABILITY problem is to determine whether $\mathcal{S}$ starting in configuration $\mathcal{I}$ ever reaches the configuration $\mathcal{B}$. (Note that, for an SDS with n nodes, when $t \geq 2^n$, t-REACHABILITY is equivalent to REACHABILITY.) Given an SDS $\mathcal{S}$ and a configuration $\mathcal{I}$, the FIXED POINT REACHABILITY problem is to determine whether $\mathcal{S}$ starting in state $\mathcal{I}$ reaches a fixed point.

2.3 Extensions of the basic SDS model

As defined, the state of each node of an SDS stores a Boolean value and the local transition functions are symmetric Boolean functions. When we allow the state of each node to assume values from a domain $\mathcal{D}$ of a fixed size and allow the local transition functions to have $\mathcal{D}$ as their range, we obtain a **Finite Range SDS** (FR-SDS). If the

states may store unbounded values and the local transition functions may also produce unbounded values, we obtain a **Generalized SDS** (Gen-SDS).

Another useful variant is a **Synchronous Dynamical System** (SyDS), an SDS *without* the node permutation. In a SyDS, during each time step, all the nodes *synchronously* compute and update their state values. Thus, SyDSs are similar to classical CA with the difference that the connectivity between cells is specified by an arbitrary graph.

The notion of symmetry can be suitably extended to functions with non-Boolean domains as well. In defining the above models, we did not attempt to relax the symmetry property of local transition functions. Dynamical systems in which the local transition functions are not necessarily symmetric are considered in the companion papers [BH+01a,BH+01b,BH+01c].

3 Summary and significance of results

In this paper, we characterize the computational complexity of determining several phase space properties for SDSs and CA. The results obtained are the first such results for SDSs and directly imply corresponding lower bounds on the complexity of similar problems for various classes of CA and communicating finite state machines.

Our main result is a *dichotomy* between easy and hard to predict classes of SDSs. Specifically, we show that t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems for FR-SDSs are **PSPACE**-complete. Moreover, these results hold even if the local transition functions are identical and the underlying graph is a simple path. We further extend these results to show that the above three problems remain **PSPACE**-complete for SDSs, even when the underlying graph is simultaneously k-regular for some fixed k, bandwidth bounded (and hence pathwidth and treewidth bounded) and the local transition functions are *symmetric*.

In contrast to the above intractability results, we show that these problems are efficiently solvable for SDSs in which each local transition function is *symmetric and monotone*. Specifically, we prove that when each local transition function is a k-simple-threshold function[1] for some $k \geq 0$, these problems can be solved in polynomial time.

As a part of our methodology, we also obtain a number of "simulation" results that show how to simulate one type of SDS (or CA) by another typically more restricted type of SDS (or CA). These simulation results may be of independent interest. For instance, we show (i) how a given FR-SyDS with local transition functions that are not necessarily symmetric can be efficiently simulated by a SyDS, and (ii) how a SyDS can be simulated by an SDS.

The results presented here extend a number of earlier results on the complexity of problems for CA and also have other applications. We briefly discuss these extensions and their significance below.

Reference [RH93] shows the **EXPSPACE**-hardness of local STATE REACHABILITY problems for hierarchically-specified linearly inter-connected copies of a single finite automaton. The constructions presented here imply that various local STATE

[1] The k-simple-threshold function has the value 1 iff at least k of its inputs are 1. Conventional definition of threshold functions associates a weight with each input [Ko70]. We use the simplified form where all inputs have the same weight.

REACHABILITY problems are also **EXPSPACE**-hard, for hierarchically-specified and bandwidth-bounded networks of simple SDSs. Using ideas from [SH+96], this last result implies that determining any simulation equivalence relation or pre-order in the Linear-time/Branching-time hierarchies of [vG90,vG93] is **EXPSPACE**-hard for such hierarchically-specified networks of simple SDSs.

Our reductions are carried out starting from the acceptance problem for deterministic linear space bounded automata (LBAs) and are extremely efficient in terms of time and space requirements. Specifically, these reductions require $O(n)$ space and $O(n \log n)$ time. Thus these results imply tight lower bounds on the deterministic time and space required to solve these problems.

The results in [Su95,Su90] prove the **PSPACE**-completeness of REACHABILITY and FIXED POINT REACHABILITY problems for CA. These papers do not address the effect of restricting the class of local transition functions or restricting the structure of the underlying graph on the complexity of these problems. Our results extend these hardness results to much simpler instances and also provide the first step in proving results that delineate polynomial time solvable and computationally intractable instances.

The results presented here can be contrasted with the work of Buss, Papadimitriou and Tsitsiklis [BPT91] on the complexity of t-REACHABILITY problem for coupled automata. Their identity-independence assumption is similar to our symmetric function assumption, except that they consider first order formulas. In contrast to the polynomial time solvability of the reachability problem for globally controlled systems of independent automata [BPT91], our results show that a small amount of local interaction suffices to make the reachability problem computationally intractable. Our reduction leads to an interaction graph that is of constant degree, bandwidth bounded and regular.

Other than [BPT91], papers that are most relevant to our work are the following. References [BMR99,BMR00,MR99,Re00,Re00a,LP00] investigate mathematical properties of sequential dynamical systems. Sutner [Su89,Su90,Su95] and Green [Gr87] characterize the complexity of reachability and predecessor existence problems for finite CA. Moore [Mo90,Mo91] makes an important connection between unpredictability of dynamical systems and undecidability of some of their properties. The models considered here are also related to discrete Hopfield networks [FO00,GFP85,BG88].

Alur et al (see for example [AKY99]) consider the complexity of several problems for hierarchically specified communicating finite state machines. SDSs can be viewed as very simple kinds of concurrent state machines. Moreover, the hardness proof obtained here can be extended to obtain **EXPSPACE**-hardness, when we have exponentially many simple automata (vertices in our case) joined to form a bandwidth bounded graph. This result significantly extends a number of known results in the literature concerning concurrent finite state machines by showing that the hardness results hold even for simple classes of individual machines.

Quadratic dynamical systems are a variant of discrete dynamical systems that aim at modeling genetic algorithms. In [ARV94] it is shown that simulating quadratic dynamical systems is **PSPACE**-hard; specifically, it is shown that the t-reachability problem for such systems is **PSPACE**-complete even when t is specified in unary.

4 Complexity of reachability problems

4.1 Road map for the reductions

In this section we prove our main hardness theorem concerning the t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems for SDSs.

Theorem 1. *(**Main hardness theorem**) The t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems for SDSs with symmetric Boolean functions are **PSPACE**-hard, even when (i) each node is of constant degree and the graph is regular (i.e. all nodes have the same degree), (ii) the pathwidth and hence the treewidth of the graph is bounded by a constant, and (iii) all the nodes have exactly the same symmetric Boolean function associated with them.*

Overall proof idea: The proof of the above theorem is obtained through a series of local replacement type reductions (steps). The reductions involve building general gadgets that may be of independent interest.

Step 1: First, by a direct reduction from the acceptance problem for a LINEAR BOUNDED AUTOMATON (LBA) we can show that the t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems for FR-SyDS (finite CA) and FR-SDS are **PSPACE**-hard even under the following restrictions applied simultaneously. (i) The graph G is a line (which has pathwidth and treewidth of 1). (ii) The number of distinct local transition functions is at most three. (iii) The domain of each function is a small constant, depending only on the size of the LBA encoding. This result is stated as Theorem 2 below; the proof is omitted.

Theorem 2. *(**Step 1:**) The t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems for FR-SyDS (Cellular Automata) and FR-SDS are **PSPACE**-hard, even when restricted to instances such that: (i) The graph G is a line graph (and thus has pathwidth and treewidth of 1), and (ii) The number of distinct local transition functions is at most three, and (iii) The size of the domain of each function is a small constant.* ■

Step 2: Next, we show how to transform these problems for FR-SyDS into the corresponding problems for SyDS in which the maximum node degree is bounded. See Section 4.2.

Step 3: Next, we show how a SyDS can be simulated by an SDS where the maximum node degree is bounded. (The underlying graph of the SDS may not be regular and the local transition functions may not be identical.)

Step 4: Finally, we show how to transform the SDS obtained in Step 3 into another SDS whose underlying graph is regular and whose node functions are all identical (same function and same degree).

For reasons of space, we omit the constructions alluded to in Steps 3 and 4.

4.2 SyDS with symmetric Boolean functions: Step 2

Definition 1. *Given $k \geq 1$, a **distance-k coloring** of a graph $G(V, E)$ is an assignment of colors $h : V \to \{0, 1, 2, \ldots, |V| - 1\}$ to the nodes of G such that for all $u, v \in V$ for which the distance between u and v is at most k, $h(u) \neq h(v)$.*

Proposition 1. *A graph $G(V, E)$ with maximum degree Δ can be distance-2 colored using at most $\Delta^2 + 1$ colors, and such a coloring can be obtained in polynomial time. Thus, for a graph whose node degrees are bounded by a constant, and, in particular, for regular graphs of constant degree, the number of colors used for distance-2 coloring is a constant.* ■

Theorem 3. *For a given μ and Δ, consider the class of FR-SyDSs where the size of the state domain of each node is at most μ and the degree of each node is at most Δ. There is a polynomial time reduction from a FR-SyDS $\mathcal{S} = (G, \mathcal{F})$ in this class and configurations $\mathcal{I}$ and $\mathcal{B}$ for $\mathcal{S}$ to a usual (having symmetric Boolean functions) SyDS $\mathcal{S}_1 = (G_1, \mathcal{F}_1)$ and configurations $\mathcal{I}_1$ and $\mathcal{B}_1$ for $\mathcal{S}_1$ such that*

1. *$\mathcal{S}$ starting in configuration $\mathcal{I}$ reaches $\mathcal{B}$ iff $\mathcal{S}_1$ starting in configuration $\mathcal{I}_1$ reaches $\mathcal{B}_1$. Moreover, for each t, $\mathcal{S}$ reaches $\mathcal{B}$ in t steps iff $\mathcal{S}_1$ reaches $\mathcal{B}_1$ in t steps.*
2. *$\mathcal{S}$ starting in configuration $\mathcal{I}$ reaches a fixed point iff $\mathcal{S}_1$ starting in $\mathcal{I}_1$ reaches a fixed point.*

Proof sketch: Given $\mathcal{S}$, the reduction first constructs a distance-2 coloring h of G, using at most $\Delta^2 + 1$ colors, where the colors are consecutive integers, beginning with zero. The fact that h is a distance-2 coloring is not used in the construction of $\mathcal{S}_1$ from $\mathcal{S}$, but is crucial to the correctness of the reduction.

Next, given graph $G(V, E)$ and coloring h, graph $G_1(V_1, E_1)$ is constructed, as follows. For each node $x_k \in V$, there are $(\mu - 1)\mu^{h(x_k)}$ nodes in V_1. We refer to these nodes as x_{ij}^k, $1 \leq i < \mu$ and $1 \leq j \leq \mu^{h(x_k)}$. Informally, corresponding to a node x_k of $\mathcal{S}$, V_1 contains $\mu - 1$ sets of nodes (which we call *clumps*), each of cardinality $\mu^{h(x_k)}$. For a given node $x_k \in V$, clump $\mathcal{X}_j^k$ refers to the nodes $x_{j,r}^k$ $1 \leq r \leq \mu^{h(x_k)}$. Additionally, we will use $\mathcal{X}^k = \mathcal{X}_1^k \cup \mathcal{X}_2^k \ldots \cup \mathcal{X}_{\mu-1}^k$ to denote the set of all nodes in V_1 corresponding to x_k. E_1 consists of the following two kinds of edges: For each node $x_k \in V$, the nodes in $\mathcal{X}^k$ form a complete graph. Each edge $\{x_k, x_r\} \in E$ is replaced by a complete bipartite graph between the sets of nodes used to replace the nodes x_k and x_r.

Before specifying the construction of the local transition functions of $\mathcal{S}_1$, we define the following mapping ψ_k, for each node $x_k \in V$. Suppose that node $x_k \in V$ has neighbors $y_1, \ldots, y_d$ in G. We define function $\psi_k : \mathbb{N} \to \mu^{d+1}$ (where $\mathbb{N}$ is the set of nonnegative integers) as follows: $\psi_k(w) = \langle c_0, c_1, \ldots, c_d \rangle$, where in the base μ representation of w, c_0 is the coefficient of $\mu^{h(x_k)}$, and c_r, $1 \leq r \leq d$, is the coefficient of $\mu^{h(y_r)}$.

The functions in the set $\mathcal{F}_1$ are defined as follows. We envision the state domain of $\mathcal{S}$ to be the integers $0, 1, \ldots, \mu - 1$. Consider a node x_{ij}^k in $\mathcal{X}^k$ and a vector α of Boolean input values for $f_{x_{ij}^k}$. Suppose that in α, exactly w of the input parameters to $f_{x_{ij}^k}$ are equal to 1. Suppose f_{x_k} from $\mathcal{F}$ is the local transition function at node x_k. Then $f_{x_{ij}^k}(\alpha) = 1$ iff $f_{x_k}(\psi_k(w)) \geq i$. In the reduction of $\mathcal{S}$ to $\mathcal{S}_1$, function $f_{x_{ij}^k}$ is represented by specifying the subset of counts of input parameters taking value 1 for which the output equals 1.

We now define the following mapping g from the configurations of $\mathcal{S}$ to the configurations of $\mathcal{S}_1$, $g : \mu^V \to 2^{V_1}$. Consider a configuration $\mathcal{A}$ of $\mathcal{S}$ and node $x_k \in V$. In

configuration $g(\mathcal{A})$ of $\mathcal{S}_1$, the nodes in the first $\mathcal{A}(x_k)$ clumps of $\mathcal{X}^k$ have state value 1, and the nodes in the other clumps of $\mathcal{X}^k$ have state value 0. More precisely, for $x_k \in V$, $1 \leq i < \mu,\ \ 1 \leq j \leq \mu^{h(x_k)}$, $g(\mathcal{A})(x_{ij}^k) = 1$ iff $\mathcal{A}(x_k) \geq i$.

The reduction constructs $\mathcal{I}_1$ as $g(\mathcal{I})$, and $\mathcal{B}_1$ as $g(\mathcal{B})$. This completes the construction involved in the reduction. The correctness of this construction is based on showing that the phase space of $\mathcal{S}$ is embedded as a subspace of the phase space of $\mathcal{S}_1$, so that $\mathcal{S}_1$ can be used to simulate $\mathcal{S}$. First, we specify which configurations of $\mathcal{S}_1$ are in this subspace.

Define a configuration $\mathcal{A}$ of $\mathcal{S}_1$ to be *proper* if for all k, i, j, p, q,

$$(\mathcal{A}(x_{ij}^k) = 1 \text{ and } i \geq p) \qquad \Rightarrow \qquad \mathcal{A}(x_{pq}^k) = 1.$$

In other words, a configuration $\mathcal{A}$ of $\mathcal{S}_1$ is *proper* if the value at any node in clump $\mathcal{X}_j^k$ equal to 1 implies that *all* nodes in clumps $\mathcal{X}_1^k, \mathcal{X}_2^k, \ldots, \mathcal{X}_j^k$ are also 1. The following claim provides an important property of the mapping g.

Claim 4.1: Mapping g is a bijection between the configurations of $\mathcal{S}$ and the proper configurations of $\mathcal{S}_1$.

For a mapping $\mathcal{D}$ of a set of states into Boolean values, we let $|\mathcal{D}|$ denote the number of 1's in $\mathcal{D}$. We also take the notational liberty of identifying an assignment of state values to a set of nodes with the vector of values representing the assignment. Also, recall that $N(x)$ denotes the neighbors of node x, plus node x itself. For a configuration $\mathcal{C}$ and set of nodes W, $\mathcal{C}(W)$ denotes the restriction of $\mathcal{C}$ to W.

Claim 4.2: For a configuration $\mathcal{C}$ of $\mathcal{S}$, node x_k of V, and node x_{ij}^k of $\mathcal{X}^k$, $\mathcal{C}(N(x_k)) = \psi_k(|g(\mathcal{C})(N(x_{ij}^k))|)$.

We now state our key claim, which says that $\mathcal{S}_1$ properly simulates $\mathcal{S}$.

Claim 4.3: For every configuration $\mathcal{A}$ of $\mathcal{S}$, $F_{\mathcal{S}_1}(g(\mathcal{A})) = g(F_{\mathcal{S}}(\mathcal{A}))$.

Now, the following claim completes the proof of Theorem 3. The claim can be proven using the above claims and induction on t.

Claim 4.4: Let $\mathcal{S}$ and $\mathcal{S}_1$ be as defined above. Consider $\mathcal{S}$ starting in configuration $\mathcal{I}$ and $\mathcal{S}_1$ starting in configuration $g(\mathcal{I})$. Then (1) $\forall t \geq 0$, $\xi(\mathcal{S}_1, t)$ is proper. (2) $\forall t \geq 0$, $\xi(\mathcal{S}_1, t) = g(\xi(\mathcal{S}, t))$. ■

5 Polynomial time solvable cases

In this section we prove that t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems are polynomial time solvable for k-simple-threshold-SDSs, that is, SDSs in which each local transition function is a k-simple-threshold function for some $k \geq 1$. Since each symmetric monotone function is a k-simple-threshold function for some k, these polynomial time results provide the dichotomy between two classes of SDSs: one with symmetric local transition functions and the other with symmetric monotone local transition functions. Some remarks regarding the generality of these polynomial algorithms are provided at the end of this section.

Definition 2. *A* k-**simple-threshold**-*SDS is an SDS in which the local transition function at each node* v_i *is a* k_i*-simple-threshold function, where* $1 \leq k_i \leq \min\{k, \delta_i + 1\}$. *Here,* δ_i *is the degree of node* v_i.

Theorem 4. *For any k-simple-threshold-SDS, $1 \leq k \leq n$, the problems* t-REACHABILITY, REACHABILITY *and* FIXED POINT REACHABILITY *can be solved by executing at most $3m/2$ steps of the given SDS, where m is the number of edges in the underlying graph.*

Proof sketch: The proof of the theorem is based on a *potential function* argument. A similar approach is used in [GFP85] to establish the convergence rate and to bound the transient length of discrete Hopfield networks with threshold gates.

Given an SDS with underlying graph $G(V, E)$, we assign a potential to each node and each edge in G. For the remainder of the proof, we use k_v to denote the threshold value required for a node v to become 1. For each node v define $T_1(v) = k_v$ and $T(v) = \delta_v + 1$. Recall that s_v denotes the state of node v. Thus $s_v = 1$ iff at least $T_1(v)$ of its inputs are 1; s_v is 0 otherwise. Another interpretation of $T_1(v)$ is that it is the smallest integer such that s_v must be assigned 1 if $T_1(v)$ of v's inputs have value 1. Using this analogy, define $T_0(v)$ to be the smallest integer such that s_v must be assigned 0 if $T_0(v)$ of the inputs to v have value 0. The following observation is an easy consequence of the definitions of k_v-simple-threshold, $T_0(v)$ and $T_1(v)$: For any node $v \in V$, $T_1(v) + T_0(v) = T(v) + 1$. Define the potential $P(v)$ at a node v as follows:

$$\begin{aligned} P(v) &= T_1(v) \text{ if } s_v = 1 \\ &= T_0(v) \text{ if } s_v = 0 \end{aligned}$$

The following is an easy consequence of the definitions of $T_0(v)$, $T_1(v)$ and the fact that $k_v \geq 1$: For any node $v \in V$, $1 \leq P(v) \leq \delta_v + 1$.

Define the potential $P(e)$ of an edge $e = \{u, v\}$ as follows:

$$\begin{aligned} P(e) &= 1 \quad \text{if } e = \{u, v\} \text{ and } s_u \neq s_v \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

The potential of the entire SDS is given by $P(G) = \sum_{v \in V} P(v) + \sum_{e \in E} P(e)$. It can be seen that the initial potential $P(G)$ (regardless of the initial configuration) is bounded by $3m + n$. Further, for each node $v \in V$, $P(v) \geq 1$; thus, the potential of the SDS at any time is at least n.

If the system has not reached a fixed point, then at least one node v undergoes a state change. Now, fix a global step in the dynamic evolution of the SDS and consider a particular substep in which the state of node v changes from a to b. This state change may modify the potential of v and the potentials of the edges incident on v. It can be shown that each time there is a change in the state of a node, $P(G)$ decreases by at least 2. As argued above, the initial value of $P(G)$ is at most $3m + n$, and the value of $P(G)$ can never be less than n. Thus, the total number of configuration changes is bounded by $[(3m + n) - n]/2 = 3m/2$.

In other words, any k-simple-threshold-SDS reaches a fixed point after at most $3m/2$ steps. Theorem 4 follows. ■

The above theorem points out an interesting contrast between CA and SDSs. It is easy to construct instances of k-simple-threshold-CA with limit cycles. In contrast, by Theorem 4, k-simple-threshold-SDSs have fixed points but not limit cycles.

Theorem 4 holds even when each node has a different value of the threshold k. As observed earlier, every symmetric and monotone Boolean function is a k-simple-threshold function for some k. Thus, Theorem 4 implies the polynomial time solvability

of reachability problems for SDSs with symmetric monotone local transition functions. Theorem 4 also shows that for k-simple-threshold-SDSs, the length of any transient is at most $3m/2$.

The result of Theorem 4 can also be extended to the case when the local transition function at node v_i is the zero-threshold function (i.e., a function whose output is 1 for all inputs) or the $(\delta_i + 2)$-threshold function (i.e., a function whose output is 0 for all inputs). All nodes with such local transition functions will reach their final values during the first step of the SDS. Therefore, in this case, the number of SDS steps needed to reach a fixed point is at most $3m/2 + 1$.

Acknowledgements: This research has been funded in part by the LDRD-DR project *Foundations of Simulation Science* and LDRD-ER project *Extremal Optimization*. We thank Paul Wollan and Predrag Tosic for reading the manuscript carefully and suggesting changes that substantially improved the readability of the paper. We also thank Gabriel Istrate, Stephen Kopp, Henning Mortveit, Allon Percus and Christian Reidys for fruitful discussions.

References

[AKY99] R. Alur, S. Kannan, and M. Yannakakis. Communicating hierarchical state machines. *Proc. 26th International Colloquium on Automata, Languages and Programming* (ICALP'99), Springer Verlag, 1999.

[ARV94] S. Arora, Y. Rabani and U. Vazirani. Simulating quadratic dynamical systems is **PSPACE**-complete. *Proc. 26th Annual ACM Symposium on the Theory of Computing* (STOC'94), Montreal, Canada, May 1994, pp. 459-467.

[BB+99] C. Barrett, B. Bush, S. Kopp, H. Mortveit and C. Reidys. Sequential Dynamical Systems and Applications to Simulations. Technical Report, Los Alamos National Laboratory, Sept. 1999.

[BG88] J. Bruck and J. Goodman. A generalized convergence theorem for neural networks. *IEEE Trans. on Information Theory*, Vol. 34, 1988, pp. 1089–1092.

[BH+01a] C. Barrett, H. Hunt III, M. Marathe, S. Ravi, D. Rosenkrantz, R. Stearns and P. Tosic. Gardens of Eden and fixed points in sequential dynamical systems. To appear in *Proc. of the International Conference on Discrete Models - Combinatorics, Computation and Geometry* (DM-CCG), Paris, July 2001.

[BH+01b] C. Barrett, H. Hunt III, M. Marathe, S. Ravi, D. Rosenkrantz and R. Stearns. Predecessor and permutation existence problems for sequential dynamical systems. Under preparation, May 2001.

[BH+01c] C. Barrett, H. Hunt III, M. Marathe, S. Ravi, D. Rosenkrantz and R. Stearns. Elements of a theory of computer simulation V: computational complexity and universality. Under preparation, May 2001.

[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.

[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.

[BR99] 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.

[BWO95] C. Barrett, M. Wolinsky and M. Olesen. Emergent local control properties in particle hopping traffic simulations. *Proc. Traffic and Granular Flow*, Julich, Germany, 1995.

[FO00] P. Floréen and P. Orponen. Complexity issues in discrete Hopfield networks. *The Computational and Learning Complexity of Neural Networks: Advanced Topics*. Ian Parberry (Editor), 2000.

[GFP85] E. Goles F. Fogelman and D. Pellegrin. Decreasing energy functions as a tool for studying threshold networks. *Discrete Applied Mathematics*, vol. 12, 1985, pp. 261–277.

[Gr87] F. Green. NP-complete problems in cellular automata. *Complex Systems*, 1(3), pp. 453–474, 1987.

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

[Hu73] H. Hunt III. On the Time and Tape Complexity of Languages. Ph.D. Thesis, Cornell University, Ithaca, NY, 1973.

[Ko70] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill Book Company, New York, 1970.

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

[Mo90] C. Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20), pp. 2354–2357, 1990.

[Mo91] C. Moore. Generalized shifts: unpredictability and undecidability in dynamical systems. *Nonlinearity*, 4, pp. 199–230, 1991.

[MR99] H. Mortveit and C. Reidys. Discrete sequential dynamical systems. *Discrete Mathematics*. Accepted for publication, 2000.

[Ra92] A. Rabinovich. Checking equivalences between concurrent systems of finite state processes. *International Colloquium on Automata Programming and languages* (ICALP'92), LNCS Vol. 623, Springer, 1992, pp. 696–707.

[Re00] C. Reidys. On acyclic orientations and SDS. *Advances in Applied Mathematics*, to appear.

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

[RH93] D. Rosenkrantz and H. Hunt III. The complexity of processing hierarchical specifications. *SIAM Journal on Computing*, 22(3), pp. 627–649, 1993.

[SH+96] S. Shukla, H. Hunt III, D. Rosenkrantz and R. Stearns. On the Complexity of Relational Problems for Finite State Processes. *Proc. International Colloquium on Automata, Languages and Programming* (ICALP'96), pp. 466–477.

[Su89] K. Sutner. Classifying circular cellular automata. *Physica D*, 45(1-3), pp. 386-395, 1989.

[Su90] K. Sutner. De Bruijn graphs and linear cellular automata. *Complex Systems*, 5(1), pp. 19-30, 1990.

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

[vG90] R. van Glabbeek. The linear time-branching time spectrum. Technical Report CS-R9029, Computer Science Department, CWI, Centre for Mathematics and Computer Science, Netherlands, 1990.

[vG93] R. van Glabbeek. The linear time-branching time spectrum II (the semantics of sequential systems with silent moves). *LNCS* 715, 1993.

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