ᔕ1

LA-UR- 01-1940

LA-UR- ~~01-1643~~

c.1

Approved for public release; distribution is unlimited.

Title:

ANALYSIS PROBLEMS FOR SEQUENTIAL DYNAMICAL SYSTEMS AND COMMUNICATING STATE MACHINES

Author(s):

Christopher L. Barrett, D-2
Harry B. Hunt III, D-2
Madhav V. Marathe, D-2
S. S. Ravi, SUNY, Albany
Daniel J. Rosenkrantz, SUNY, Albany
Richard E. Stearns, SUNY, Albany

Submitted to:

26th International Symposium on Mathematical Foundations of Computer Science (MFCS 01)
Marianske Lazne Czech Republic
August 27-31, 2001

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] RICHARD E. STEARNS [2]

April 3, 2001

Abstract

A simple sequential dynamical system (SDS) is a triple $(G, \mathcal{F}, \pi)$, where (i) $G(V, E)$ is an undirected graph with n nodes with each node having a 1-bit state, (ii) $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is a set of local transition functions with f_i denoting a Boolean function associated with node v_i and (iii) π is a fixed permutation of (i.e., a total order on) the nodes in V. A single SDS transition is obtained by updating the states of the nodes in V by evaluating the function associated with each of them in the order given by π. Such a (finite) SDS is a mathematical abstraction of simulation systems [BMR99, BR99]. In this paper, we characterize the computational complexity of determining several phase space properties of SDSs. The properties considered are t-REACHABILITY ("Can a given SDS starting from configuration $\mathcal{I}$ reach configuration $\mathcal{B}$ in t or fewer transitions?"), REACHABILITY ("Can a given SDS starting from configuration $\mathcal{I}$ ever reach configuration $\mathcal{B}$?") and FIXED POINT REACHABILITY ("Can a given SDS starting from configuration $\mathcal{I}$ ever reach configuration in which it stays for ever?"). Our main result is a sharp dichotomy between classes of SDSs whose behavior is "easy" to predict and those whose behavior is "hard" to predict. Specifically, we show the following.

1. The t-REACHABILITY, REACHABILITY and the FIXED POINT REACHABILITY problems for SDSs are **PSPACE**-complete, even when restricted to graphs of bounded bandwidth (and hence of bounded pathwidth and treewidth) and when the function associated with each node is *symmetric*. The result holds even for regular graphs of constant degree where all the nodes compute the same symmetric Boolean function.
2. In contrast, the t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems are solvable in polynomial time for SDSs when the Boolean function associated with each node is *symmetric and monotone*.

Two important consequences of our results are the following: (i) The close correspondence between SDSs and cellular automata (CA), in conjunction with with our lower bounds for SDSs, yields stronger lower bounds on the complexity of reachability problems for CA than known previously. (ii) REACHABILITY problems for hierarchically-specified linearly inter-connected copies of a single finite automaton are **EXPSPACE**-hard.

The results can be combined with our related results to show hardness of a number of equivalence relations for such automata. The results can also be used to demonstrate that determining the sensitivity to initial conditions of such automata (as proposed in [Mo90, BPT91]) is computationally intractable.

Classification: Computational Complexity, Dynamical Systems, Complexity Classes, Cellular Automata, PSPACE.

[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 addresses of authors: {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.

1 Introduction and Motivation

We study the computational complexity of combinatorial problems associated with a new class of finite discrete dynamical systems, called *Sequential Dynamical Systems* (henceforth referred to as SDS), proposed in [BR99, BMR99, BMR00]. A formal definition of such a system is given in Section 2. Sequential dynamical systems 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 [Wo86, Gu89]. Dynamical systems are closely related finite networks of communicating automata, finite and infinite transition systems and sequential digital circuits [Ra92, HT94, SH+96, AY98, AKY99, RH93, SM73, Hu73, HRS76, HR78].

In simple terms, a sequential dynamical system (SDS) $\mathcal{S}$ is a triple $(G, \mathcal{F}, \pi)$. $G(V, E)$ is an undirected graph (called the **underlying graph** of the SDS) with n nodes, with each node having a state with finite number of state values. $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is a set of **local transition functions**, where f_i is a function associated with node $v_i \in V$. The inputs to f_i are the values of the state of node v_i and those of v_i's neighbors in G. The range of f_i is the set of allowed state values. π is a permutation of (i.e., a total order on) the nodes in V. A single SDS transition is obtained by updating the states of nodes $v \in V$ by evaluating the function associated with each of the nodes, in the order specified by π. A **configuration** of SDS $\mathcal{S}$ is an n-tuple $(b_1, b_2, \ldots, b_n)$, where b_i is the value of the state of node v_i. Thus, a transition of an SDS can be envisioned as a change from one configuration to another. The **phase space** of $\mathcal{S}$ is a directed graph where each node represents a configuration and each directed edge $(\mathcal{C}, \mathcal{C}')$ indicates that the system moves from configuration $\mathcal{C}$ to configuration $\mathcal{C}'$ in one transition. For an SDS whose underlying graph has n nodes, the phase space has k^n nodes, where k is the number of allowed state values. A **fixed point** of an SDS $\mathcal{S}$ is a configuration $\mathcal{C}$ such that the only possible transition from the configuration $\mathcal{C}$ is to $\mathcal{C}$ itself.

In this abstract, 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 f_i is Boolean and symmetric. Our *hardness results* hold even for such simple-SDSs and thus imply analogous hardness results for more general models.

Here, we study the computational complexity of determining various phase space 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 $\mathcal{C}$ for ever?") that are commonly studied by the dynamical systems community. Specifically, we investigate whether such properties can be decided efficiently using computational resources that are polynomial in the size of the SDS representation, rather than in the size of the phase space (which is exponentially larger). We also study the computational complexity of such questions when the underlying systems are specified succinctly. The research is guided by the following overall goals: (i) identification of efficient inter-simulations that yield uniform lower bounds for various models, (ii) efficient reductions that can be naturally extended to succinctly presented and, in the limit, to infinite instances, and (iii) obtain meta-results that allow us to infer the easiness/hardness of analysis problems as a function of underlying parameters (functions, graph topology, domain size, etc).

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. The local interaction rules for entities and a dependency graph structure are by now accepted as standard aspects of discrete dynamical systems for modeling large-scale systems. The ordering aspect is somewhat new in a formal setting but has recently received attention by other researchers [HG99, Ga97, Rk94]. It is implicit in all discrete event simulations. Following [BPT91], we say that a system is predictable if basic phase space properties such as REACHABILITY and FIXED POINT EXISTENCE can be determined in time which 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 we 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.

Another motivation for studying SDSs is derived from the papers of Buss, Papadimitriou and Tsitsiklis [BPT91], Moore [Mo90, Mo91], Sutner [Su95] and Wolfram [Wo86]. Specifically, 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+00a, BH+00b] 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 Formulations

We begin with a formal definition of sequential dynamical systems. As stated in the introduction, 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, BMR00, MR99, Re00]. We also recall basic definitions of phase space parameters studied in this paper.

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 $1, 2, \ldots, n$.

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.

Note again 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 $\mathcal{C} : V \to \mathbb{F}_2$. 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 $\xi(\mathcal{S}, 0) = \mathcal{I}$. We also use $\xi(\mathcal{S}, t)(W)$ to denote the states of the nodes in $W \subseteq V$ and $\xi(\mathcal{S}, t)(v)$ to denote the state of a particular node $v \in V$ at time t.

2.1 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. When Boolean local transition functions are given as tables, $|\mathcal{S}| = O(m + |T|n)$, were $|T|$ 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.

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. The analysis problems considered in this paper 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 decide 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 decide whether $\mathcal{S}$ starting in configuration $\mathcal{I}$ ever reaches the configuration $\mathcal{B}$. (Note that, for $t \geq 2^n$, t-REACHABILITY is equivalent to REACHABILITY.) Given an SDS $\mathcal{S}$ and a configuration $\mathcal{I}$, the FIXED POINT REACHABILITY problem is to decide whether $\mathcal{S}$ starting in state $\mathcal{I}$ reaches a fixed point.

2.2 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 node 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. A further generalization is to consider a partial order on the nodes instead of a total order. We refer to such an SDS as a **DagDS** since any partial order can be represented as a directed acyclic graph (dag). Clearly, DagDSs generalize both SDSs and SySDSs. The definition of a SyDS can be extended to obtain an FR-SyDS and a Gen-SyDS in a manner similar to that of SDS. FR-DagDS and Gen-DagDS can also be defined in an analogous fashion. It can be seen that SDSs are the most restricted models and Gen-DagDS are the least restricted models. Whenever possible, we prove our **PSPACE**-completeness results for the most restricted SDS model, thereby obtaining stronger lower bound results.

Note that 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+00a, BH+00b].

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 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[4] for some $k \geq 1$, these problems can be solved in polynomial time.

[4]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.

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

1. how a given FR-SyDS with local transition functions that are not necessarily symmetric can be efficiently simulated by a SyDS, and

2. 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.

(1) Recalling results in [RH93] showing the **EXSPACE**-hardness of local STATE-REACHABILITY problems , for hierarchically-specified linearly inter-connected copies of a single finite automaton, the constructions here also imply that various local STATE-REACHABILITY problems are also **EXSPACE-hard**, for hierarchically-specified constantly bandwidth-bounded networks of simple SDSs. Using ideas from [SH+96, HR+01], this last result implies: i)determining any simulation equivalence relation or pre-order in the Linear-time/Branching-time hierarchies of [vG90, vG93] is **EXSPACE**-hard, for such hierarchically-specified networks of simple SDSs; and ii)in the sense of [BPT91], such hierarchically-specified networks of simple SDSs exhibit **EXSPACE**-hard sensitivity to initial conditions. Additionally following [BPT91], we emphasize concepts, techniques, etc., that can be used to characterize the *computationally-tractable* or *computationally-intractable sensitivity to initial values* of these models. Note that our models use "minimal" amount of concurrency to obtain the results.

(2) All our reductions are carried out 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.

(3) The results in [Su95, Gr87] prove the **PSPACE**-completeness of REACHABILITY and FIXED POINT REACHABILITY problems and the **NP**-completeness of the PREDECESSOR EXISTENCE problem ("Given a cellular automaton A and a configuration $\mathcal{B}$, is there a configuration from which A can reach $\mathcal{B}$ in one transition?") for CA. These authors did not consider 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 their hardness results to much simpler instances and also provide the first step in proving results that delineate polynomial time solvable and computationally intractable instances.

(4) 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. In their model, there are n identical automata, a global control rule, an initial state vector $\mathcal{I}$ and a positive integer T. The global control rule is given as a first order sentence and is independent of the identities of the automata. The automata do not interact with each other. At each stage, the automata independently evaluate their next state depending on the current state and the input received from the global controller. Following this, the global control rule reads the state of the automata and evaluates the control rule. If the rule evaluates to true then all automata receive 1 as their input, otherwise they receive a 0. The goal is to predict the state of the system after T time units. Note that their identity-independence assumption is similar to our symmetric function assumption, except that they consider first order formulas. Our results show that, in contrast to the polynomial time solvability of the reachability problem for globally controlled systems of independent automata, 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. (The interaction graph is obtained from a simple path by replacing individual nodes in the path by groups of nodes that interact only with nodes in neighboring groups.)

4 Related Work

As mentioned earlier, CA have been studied widely in the literature, owing to their simplicity on one hand and their ability to produce complex behavior on the other. Computational aspects of CA have been studied by a number of researchers (see [Mo91, Mo90, CPY89, Wo86, Gu89, Gr87, Su95] and the references therein). However, most of the work addresses computability issues for infinite CA. Other than the paper by Buss, Tsitsiklis and Papadimitriou [BPT91] discussed in the previous section, the papers that are most relevant to our work are the following: (i) The papers by Barrett, Mortveit and Reidys [BMR99, BMR00, MR99, Re00, Re00a] and Laubenbacher and Pareigis [LP00] investigate mathematical properties of sequential dynamical systems, (ii) The papers of Sutner [Su89, Su90, Su95] characterize the complexity of reachability and predecessor existence problems for finite CA and (iii) The papers of Moore [Mo90, Mo91] make an important connection between unpredictability of dynamical systems and undecidability of some of their properties. Moore formally shows that undecidability is a much stronger form of unpredictability.

Alur et. al. [AKY99, AY98] consider the complexity of several problems for hierarchically specified communicating finite state machines. SDSs can be viewed as very simple kinds of concurrent state machine: moreover the hardness proof obtained here can be extended to obtain **EXPSACE**-hardness, when we have exponentially many simple automata (vertices in our case) joined in form of 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 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. The proof of this result uses a reduction from Quantified Boolean Formulas (QBF) and exploits the quadratic nature of the allowed rules. Other references on discrete dynamical systems include [AM94, AMP95, BC96, Br95, CY88, CPY89, Du94, KCG94, Pi94].

5 Hardness results

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

Theorem 5.1 *(***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 node functions is at most 3, and (iii) The domain of each function is a small constant (depending only on the size of the LBA encoding). (Theorem stated below but proof omitted).
Step 2: Next, we show how to transform these problems for FR-SyDS into the corresponding problems for SDS (where the node functions may be different). See Section 5.2.
Step 3: Finally, we further extend the hardness result so that all the node functions are identical (same function and same degree) (proved in the appendix).

Theorem 5.2 *(***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) There are no more than 3 distinct functions f_i present at the nodes of the graph, and (iii) The domain of each function is a small constant (that depends only on the LBA encoding),*

5.1 Representation of symmetric Boolean functions

A symmetric Boolean function can be represented and computed in time proportional to the degree of a node (more accurately in time proportional to the fan-in). Consider a Boolean function f_l at a node v_l with degree $n-1$. Recall that the function takes n inputs, including the value at the node itself. Function f_l can be represented by the subset of $\{0,1\ldots,n\}$ denoting when the function takes value 1 as a function of the number of input variables that are 1 (with remaining input variables being 0). For instance, let $n=5$. A possible function f_l is $\{1,3,5\}$ which is the *exclusive-or* (EX-OR) of the five input variables. Another way to represent a symmetric Boolean function is to give an $(n+1)$-dimensional 0-1 vector such that the $i-th$ entry denotes the function value when i inputs are set to 1. Thus, another representation of EX-OR above is $\langle 0,1,0,1,0,1\rangle$.

5.2 SyDS with symmetric Boolean functions: Step 2

Definition 5.1 *Given $k \geq 1$, a distance-k coloring of a graph $G(V,E)$ is an assignment of colors $h: V \to N$, to the vertices of G such that $\forall u,v \in V$ where the distance between u and v is at most k, we have that $h(u) \neq h(v)$.*

Proposition 5.1 *A graph $G(V,E)$ with maximum degree Δ can be distance-2 colored using at most Δ^2+1 colors, and such a coloring can be obtained in polynomial time. Thus for a graph whose vertex 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 5.3 *For a given m and Δ, consider the class of Gen-SyDSs where the size of the state domain of each node is at most m and the degree of each node is at most Δ. There is a polynomial time reduction from a generalized SyDS $S=(G,\mathcal{F})$ in this class and configurations $\mathcal{I}$ and $\mathcal{B}$ for S to a usual (having symmetric Boolean functions) SyDS $S_1=(G_1,\mathcal{F}_1)$ and configurations $\mathcal{I}_1$ and $\mathcal{B}_1$ for S_1 such that*

1. *S starting in configuration $\mathcal{I}$ reaches $\mathcal{B}$ iff S_1 starting in configuration $\mathcal{I}_1$ reaches $\mathcal{B}_1$. Moreover, for each t, S reaches $\mathcal{B}$ in t steps iff S_1 reaches $\mathcal{B}_1$ in t steps.*
2. *S starting in configuration $\mathcal{I}$ reaches a fixed point iff S_1 starting in $\mathcal{I}_1$ reaches a fixed point.*

Proof sketch: Given S, the reduction first constructs a distance-2 coloring h of G, using at most Δ^2+1 colors, where the colors are consecutive integers, beginning with zero. Next, given graph $G(V,E)$ and coloring h, we construct graph $G_1(V_1,E_1)$. For each node $x_k \in V$ there are $(m-1)m^{h(x_k)}$ nodes in V_1. We refer to these nodes as x^k_{ij}, $1 \leq i < m$ and $1 \leq j \leq m^{h(x_k)}$. Informally, corresponding to a node x_k of S, V_1 contains $m-1$ sets of nodes (called *clumps*), each of size $m^{h(x_k)}$. For a given node $x_k \in V$, clump $\mathcal{X}^k_j$ refers to the nodes $x^k_{j,r}$ $1 \leq r \leq m^{h(x_k)}$. Additionally, we will use $\mathcal{X}^k = \mathcal{X}^k_1 \cup \mathcal{X}^k_2 \ldots \mathcal{X}^k_{m-1}$ to denote the set of all nodes in V_1 corresponding to x_k. E_1 consists of the following two kinds of edges:

1. For each node $x_k \in V$, there is an edge between each pair of distinct nodes in $\mathcal{X}^k$. Thus, the nodes in $\mathcal{X}^k$ form a complete graph.

2. For each $(x_k, x_r) \in E$, there is an edge between each node in $\mathcal{X}^k$ and each node in $\mathcal{X}^r$. Thus, each edge (x_k, x_r) is replaced by a complete bipartite graph between the sets of nodes used to replace the nodes x_k and x_r.

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

$$(\mathcal{A}(x^k_{ij}) = 1 \text{ and } i \geq p) \quad \Rightarrow \quad \mathcal{A}(x^k_{pq}) = 1.$$

In other words, a configuration $\mathcal{A}$ of $\mathcal{S}_1$ is *proper* if the value at any node in $\mathcal{X}^k_j$ equal to 1 implies that *all* nodes in $\mathcal{X}^k_1, \mathcal{X}^k_2, \ldots, \mathcal{X}^k_j$ are also 1. Note that if $\mathcal{A}$ is a configuration of $\mathcal{S}$, then $\mathcal{A}$ maps each element of V into a value from $\{0, \ldots, (m-1)\}$. The simulation of $\mathcal{S}$ by $\mathcal{S}_1$ is based on the following bijection (that will be true by construction) g between the configurations of $\mathcal{S}$ and the proper configurations of $\mathcal{S}_1$ $g : m^V \to 2^{V_1}$. For configuation $\mathcal{A}$ of $\mathcal{S}$, the corresponding configuration $g(\mathcal{A})$ of $\mathcal{S}_1$ is specified as for all $1 \leq i < m,\ 1 \leq j \leq m^{h(x_k)},\ g(\mathcal{A})(x^k_{ij}) = 1$ iff $\mathcal{A}(x_k) \geq i$.

Intuitively, we maintain the invariant that the value c at node x_k corresponds to having the nodes in clumps $\mathcal{X}^k_1 \cup \mathcal{X}^k_2, \ldots, \mathcal{X}^k_c$ equal to 1 and the nodes in clumps $\mathcal{X}^k_{c+1} \cup \mathcal{X}^k_{c+3}, \ldots, \mathcal{X}^k_{m-1}$ equal to 0. The initial configuration $\mathcal{I}_1$ of $\mathcal{S}_1$ is set to be $g(\mathcal{I})$.

The functions in the set $\mathcal{F}_1$ are defined as follows. Suppose that node $x_k \in V$ has neighbors $y_1, \ldots y_d$ in G and f_{x_k} is the function at node x_k. Consider a node x^k_{ij} in $\mathcal{X}^k$. Consider a proper configuration $\mathcal{A}_1$ of $\mathcal{S}_1$, corresponding to the configuration $\mathcal{A}$ of $\mathcal{S}$. Suppose that in configuration $\mathcal{A}_1$, exactly w of the input parameters to $f_{x^k_{ij}}$ are equal to 1. Since configuration $\mathcal{A}_1$ is proper and h is a distance-2 coloring of G, there exist unique integers $c_0, \ldots, c_d$, each in the range $0 \ldots m-1$, such that

$$w = c_0 m^{h(x_k)} + c_1 m^{h(y_1)} + c_2 m^{h(y_2)} + \ldots + c_d m^{h(x_d)}.$$

Since $\mathcal{A}_1 = g(\mathcal{A})$, it follows that $c_0, \ldots, c_d$ are the values of x_k and its neighbors in configuration $\mathcal{A}$ of $\mathcal{S}$. The function $f_{x^k_{ij}}$ is defined as follows:

$$f_{x^k_{ij}}(w) = 1 \qquad \text{iff} \qquad f_{x_k}(c_0, c_1, \ldots, c_d) \geq i.$$

Intuitively, the reduction is done in such a way that different vectors $C = \langle c_0, \ldots, c_d \rangle$ and $Q = \langle q_0, \ldots, q_d \rangle$ corresponding to the values at x_k and its neighbors produce different counts for how many inputs of symmetric function $f_{x^k_{ij}}$ are equal to 1, and so can be appropriately differentiated. For a proper configuration of $\mathcal{S}_1$, consider the count w of how many input variables to function $f_{x^k_{ij}}$ equal 1. Integer w can be uniquely decomposed into a sum of powers of m. Because node x_k and each of its neighbors in G is assigned a distinct color by h, the coefficient of each power of m encodes the value of one of the inputs to function f_{x_k}, thereby enabling $f_{x^k_{ij}}$ to play its role in simulating f_{x_k}. The next lemma summarizes the needed properties of the construction. The proof of the lemma is omitted due to lack of space.

Lemma 5.1 *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) $\forall t \geq 0,\ x_k \in V,\ 1 \leq i < m,\ 1 \leq j \leq m^{h(x_k)},\ \xi_1(\mathcal{S}_1, t))(x^k_{ij}) = 1$ iff $\xi(\mathcal{S}, t)(x_k) \geq i$.*

Next note that the graph we obtained in proof of Theorem 5.2 was of constant degree and thus using Proposition 5.1 can be distance-2 colored using a constant number of colors. Additionally note that in the proof of Theorem 5.2 the domain size was a small fixed constant. Putting these facts together we get that the graph produced in the above construction is of bounded degree. The proof of the theorem is now a direct consequence of the above results. ■

6 Polynomial Time Solvable Cases

In this section we consider polynomial time solvable cases of the analysis problems. Let $|\mathcal{S}|$ denote the size of an SDS $\mathcal{S}$. In the phase space $\mathcal{P}_{\mathcal{S}}$ of $\mathcal{S}$, a **transient** is a simple directed path such that no edge of the path appears in any cycle in $\mathcal{P}_{\mathcal{S}}$. Our polynomial time algorithms for answering the reachability questions are based on a simple sufficient condition: If an SDS $\mathcal{S}$ is such that (i) the number of nodes in every limit cycle in $\mathcal{P}_{\mathcal{S}}$ is bounded by a polynomial in $|\mathcal{S}|$ and (ii) the number of nodes in every transient in $\mathcal{P}_{\mathcal{S}}$ is bounded by a polynomial in $|\mathcal{S}|$, then t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems can be solved in time polynomial in $|\mathcal{S}|$. While developing polynomial time algorithms using these sufficient conditions, we also obtain useful results concerning structural aspects of these SDSs.

6.1 k-Simple-Threshold Functions

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 subsection.

Definition 6.1 *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 6.1 *The problems* t-REACHABILITY, REACHABILITY *and* FIXED POINT REACHABILITY *for any k-simple-threshold-SDS, $1 \leq k \leq n$, can be solved by executing at most $\frac{3m}{2}$ steps of the given SDS, where m is the number of edges in the underlying graph.*

Proof: The proof of the theorem is based on a *potential function* argument. 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 $T_1(v)$. Note that 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:

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

The following is an easy consequence of the definitions of $T_0(v)$, $T_1(v)$ and the fact that $k_v \geq 1$ is that 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 $P(G) = \sum_{v \in V} P(v) + \sum_{e \in E} P(e)$. The initial potential $P(G)$ (regardless of the initial configuration) can be upper bounded using above observations as follows:

$$P(G) = \sum_{v \in V} P(v) + \sum_{e \in E} P(e) \leq \sum_{v \in V} (\delta_v + 1) + \sum_{e \in E} 1 \leq 2m + n + m \leq 3m + n.$$

Further, since for each node $v \in V$, $P(v) \geq 1$ the potential of the SDS at any time is at least n. 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. Note that if the system has not reached a fixed point, then at least one node undergoes such a state change. This state change may modify the potential of v and the potentials of the edges incident on v. Let D_v denote the set of edges incident on v whose potential changed from 1 to 0 as a result of the state change at v. Similarly, let A_v denote the set of edges incident on v such that their potential changed from 0 to 1. Finally, let $\Delta_v = T_a(v) - T_b(v)$ denote the decrease in the potential of v. (Note that the value of Δ_v may be negative.) We claim that

$$|D_v| \geq T_b(v) \qquad \text{and} \qquad |A_v| \leq T_a(v) - 2.$$

To see the the first inequality, consider each node x such that $\{v, x\}$ is an edge in G and $s_x = b$ at the time when v is updated. The number of such nodes is at least $T_b(v)$ since the value of v changed from a to b. For each such edge $\{v, x\}$, the potential decreases from 1 to 0 since after the update to v, $s_v = s_x$. Thus, $|D_v| \geq T_b(v)$. The second inequality follows since at most $T(v) - 1 - T_b(v)$ of v's neighbors could have been assigned value a prior to updating v. By the observations above, we get that $T(v) - 1 - T_b(v) = T_a(v) - 2$. We claim that the state change at node v decreases the potential of the system. To see this, note that the total decrease in potential due to the state change at node v is given by

$$\Delta_v + D_v - A_v \geq T_a(v) - T_b(v) + T_b(v) - T_a(v) + 2 = 2.$$

Thus, 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. During the transition from one configuration to a different configuration, at least one node changes its state value. Each such transition causes a decrease of at least 2 in the potential of the system. 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. This immediately implies the polynomial solvability of t-REACHABILITY, REACHABILITY and FIXED POINT REACHABILITY problems for such SDSs. ■

Remarks:

(1) Theorem 6.1 points out an interesting contrast between CA and SDSs. It is easy to construct instances of k-simple-threshold-CA with limit cycles. An example of such a system is a simple k-regular bipartite graph $G(V \cup U, E)$. Let $|U| = |V| = n$. Each node in V is adjacent to k neighbors in U and vice versa. Initially, we assign nodes in V the value 1 and nodes in U value 0. It is now easy to see that the system oscillates between the two configurations $(0^n 1^n)$ and $(1^n 0^n)$; thus, it does not reach a fixed point. In contrast, by Theorem 6.1, k-simple-threshold-SDSs have fixed points but not limit cycles of length ≥ 2.

(2) It can be shown that starting at any initial configuration, a k-simple-threshold-SDS will reach a fixed point regardless of the order in which the states of the nodes are updated. As a matter of fact, a fixed point will be reached even if the order of updates is changed in every iteration. Moreover, the order of updates need not even be given by a permutation. As long as each node updates its state at least once in a polynomially long sequence of state updates, a k-simple-threshold-SDS will reach a fixed point within a polynomial number of steps. We refer the reader to [IMR00] for more details on this topic.

(3) Theorem 6.1 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 6.1 implies the polynomial time solvability of reachability problems for SDSs with symmetric monotone local transition functions. Theorem 6.1 also shows that for k-simple-threshold-SDSs, the length of any transient is at most $\frac{3m}{2}$.

Acknowledgements: Work done has been funded in part of the LDRD-DR project *Foundations of Simulation Science* and LDRD-ER project *Extremal Optimization.* We sincerely thank Paul Wollan and Predrag Tosic for a very careful and in-depth reading of the manuscript and suggesting a number of changes that substantially improved the readability of the paper. We also thank Gabriel Istrate, Stephen Kopp, Henning Mortveit, Allon Percus, Christian Reidys, Paul Wollan and Predrag Tosic 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),* Springer Verlag, 1999.

[AY98] R. Alur and M. Yannakakis. Model checking of hierarchical state machines. *6th ACM Symposium on the Foundations of Software Engineering,* pp. 175-188, 1998.

[Al00] R. Alur. Exploiting Hierarchical Structure for Efficient Formal Verification. *CONCUR* 2000, pp. 66-68.

[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),* pp. 459-467, Montreal, Canada, May 1994.

[AMP95] E. Asarin, O. Maler and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science (TCS),* 138(1), pp. 35-65, February 1995.

[AM94] E. Asarin and O. Maler. On some relations between dynamical systems and transition systems. *Proc. 21st International Colloquium, on Automata, Languages and Programming (ICALP),* 820, LNCS, Springer-Verlag, pp. 59-72, Jerusalem, Israel, July 1994.

[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.

[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.

[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.

[BH+00a] C. Barrett, H. Hunt III, M. Marathe, S. Ravi, D. Rosenkrantz. Elements of a theory of computer simulation V: computational complexity and universality. to be submitted, January 2001.

[BH+00b] C. Barrett, H. Hunt III, M. Marathe, S. Ravi, D. Rosenkrantz. Computational aspects of sequential dynamical systems II: design problems. in preparation, 2000.

[BC96] O. Bournez and M. Cosnard. On the computational power of dynamical systems and hybrid systems. *Theoretical Computer Science,* 168(2), pp. 417-459, 20 November 1996.

[Br95] M. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoretical Computer Science,* 138(1), pp. 67-100, February 1995.

[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 appeared in *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS),* Oct 1990.

[BS90] R. B. Bopanna and M. Sipser. The complexity of finite functions, Chapter 14 in *Handbook of Theoretical Computer Science,* Vol. A, Edited by J. van Leeuwen, MIT Press/Elsevier, Cambridge-Amsterdam, 1990.

[CPY89] K. Cullik, J. Pachl and S. Yu. On the limit sets of cellular automata. *SIAM J. Computing,* 18(4), pp. 831-842, 1989.

[CY88] K. Cullik and S. Yu. Undecidability of CA classification schemes. *Complex Systems,* 2(2), pp. 177-190, 1988.

[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.

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

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

[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.

[GSW94] R. Gunther, B. Schapiro and P. Wagner. Complex Systems, Complexity Measures, grammars and model-Inferring. *Chaos Solitons and Fractals* 4(5), pp. 635-651, 1994

[Ha] J.P. Hayes. Digital Simulation with Multiple Logic values. *IEEE Transactions on Computer Aided Design,* CAD-5, pp. 274-283, 1986.

[HLW92] F. Höfting, T. Lengauer and E. Wanke. Processing of hierarchically defined graphs and graph families. *Data Structures and Efficient Algorithms* (Final Report on the DFG Special Joint Initiative), Springer-Verlag, LNCS 594, pp. 44-69, 1992.

[Ho84] C. Hoare. *Communicating Sequential Processes.* Prentice Hall International, 1984.

[Ho91] G. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall, 1991.

[Hu73] H.B. Hunt III. On the Time and Tape Complexity of Languages. Ph.D. Thesis, Cornell University, Ithaca, NY, 1973. Some of the results of this thesis appear in On the Time and Tape Complexity of Languages I. were presented at the *Fifth Annual ACM Symposium on Theory of Computing (STOC)* 1973, pp. 10-19.

[HR78] H.B. Hunt III and D.J. Rosenkrantz. Computational Parallels Between the Regular and Context-Free Languages. *SIAM Journal on Computing (SICOMP)* 7(1), pp. 99-114, 1978.

[HRS76] H.B. Hunt III, D.J. Rosenkrantz, and T.G. Szymanski. On the Equivalence, Containment, and Covering Problems for the Regular and Context-Free Languages. *Journal of Computer and System Sciences (JCSS)* 12(2), pp. 222-268, 1976.

[Hu82] H.B. Hunt III. On the Complexity of Flowchart and Loop Program Schemes and Programming Languages. *Journal of the ACM (J ACM)* 29(1), pp. 228-249, 1982.

[HR+01] H. Hunt III, D. Rosenkrantz, C. Barrett, M. Marathe and S. Ravi, Complexity of Analysis and Verification Problems for Communicating Automata and Discrete Dynamical Systems submitted 2001.

[HT94] D.T. Huynh and L. Tian. On deciding some equivalences for concurrent processes, *Theoretical Informatics and Applications* 28(1), pp. 51-71, 1994.

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

[IMR00] G. Istrate, M. Marathe and S. Ravi, Adversarial models in evolutionary game dynamics. to appear in *Proc. of ACM Symposium on Discrete Algorithms* January 2001.

[KMP95] Y. Kesten, Z. Manna and A. Pnueli. Verifying Clocked Transition Systems. *Hybrid Systems* 1995, pp. 13-40. Complete version in *Acta Informatica* 36(11), pp. 837-912, 2000.

[KCG94] P. Koiran, M. Cosnard and M. Garzon. Computability with low-dimensional dynamical systems. *Theoretical Computer Science,* 132(1-2), pp. 113-128, 26 September 1994.

[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.

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

[Mi99] R. Milner. *Communicating and Mobile systems: the π-calculus.* Cambridge University Press, 1999.

[MH+98] M. V. Marathe, H. B. Hunt III, D. J. Rosenkrantz and R. E. Stearns. Theory of periodically specified problems: Complexity and Approximability. *Proc. 13th IEEE Conference on Computational Complexity*, Buffalo, NY, June, 1998.

[Mo91] C. Moore. Generalized shifts: unpredictability and undecidability in Dynamical Systems. *Nonlinearity*, 4, pp. 199-230, 1991.

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

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

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

[Pa94] C. Papadimitriou. *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.

[Pi94] G. Pighizzini. Asynchronous automata versus asynchronous cellular automata. *Theoretical Computer Science*, 132(1-2), pp. 179-207, 26 September 1994.

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

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

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

[RSW92] Y. Rabinovich, A. Sinclair and A. Wigderson. Quadratic dynamical systems. *Proc. 33rd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 304-313, Pittsburgh, October 1992.

[Re00] C. Reidys. On acyclic orientations and SDS. Advances in Applied Mathematics, to appear in 2000.

[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.

[SH+96] S.K. Shukla, H.B. Hunt III, D.J. Rosenkrantz and R.E. Stearns. On the Complexity of Relational Problems for Finite State Processes. *International Colloquium on Automata Programming and Languages (ICALP)*, pp. 466-477, 1996.

[Sm71] A. Smith. Simple computation-universal cellular spaces. *J. ACM*, 18(3), pp. 339-353, 1971.

[SM73] L.J. Stockmeyer and A.R. Meyer. Word problems requiring exponential time. *Proceedings 5th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 1-9, 1973.

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

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

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

[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.

[vG90] R.J. 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.J. van Glabbeek. The linear time-branching time spectrum II (the semantics of sequential systems with silent moves). *LNCS* 715, 1993.

[VW86] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. *Proc. 1st IEEE Symposium on Logic in Computer Science*, pp.332-344, 1986.

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

7 Appendix

7.1 Simulating SyDS by SDS

Theorem 7.1 *The* FIXED POINT EXISTENCE *problem for SyDS with symmetric Boolean functions (not necessarily identical) is polynomial time reducible to the* FIXED POINT *problem for SDS with symmetric Boolean functions.*

Proof: Given an SyDS $\mathcal{S}$ $(G(V,E),\mathcal{F},\mathcal{I})$, with $\mathcal{F}$ being the set of symmetric Boolean functions, we create an instance $\mathcal{S}_1$ $(G_1(V_1,E_1),\mathcal{F}_1,\pi,\mathcal{I}_1)$ of SDS as follows:
For each $x \in V$ we create a set of 9 nodes in V_1. Denote them by x_i, $1 \leq i \leq 9$. $\mathcal{S}_1$ will simulate $\mathcal{S}$ as follows. Letting $\xi(\mathcal{S},t)(x)$ to denote the state (value) of a vertex x in SDS $\mathcal{S}$ after t time steps, we will maintain the following invariant:

$$\xi(\mathcal{S}_1,t)(x_1) \equiv \xi(\mathcal{S}_1,t)(x_2) \equiv \xi(\mathcal{S}_1,t)(x_3) \equiv \xi(\mathcal{S}_1,t)(x_4) \equiv \xi(\mathcal{S},t)(x)$$

and

$$\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_6) \equiv \overline{\xi(\mathcal{S}_1,t)(x_7)} \equiv \overline{\xi(\mathcal{S}_1,t)(x_8)} \equiv \xi(\mathcal{S}_1,t)(x_9) \equiv \xi(\mathcal{S},t-1)(x)$$

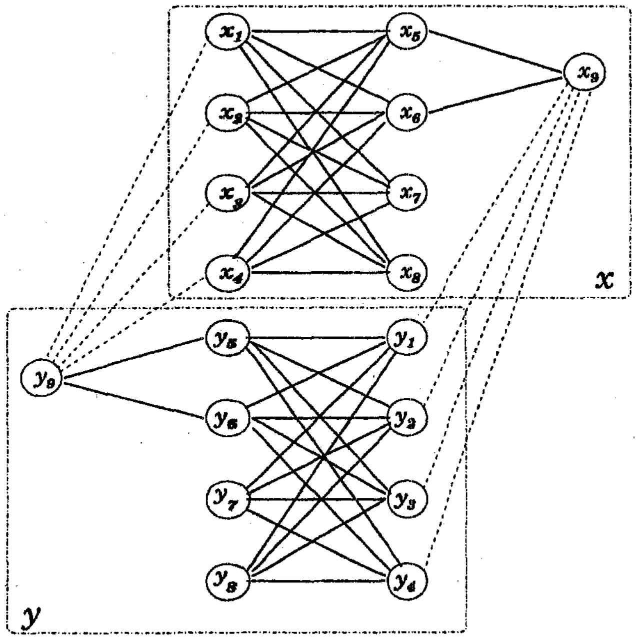

Figure 1: Figure explaining the construction of the Gadget. The dotted boxes show the replacements for vertices x and y. The dotted lines going between vertices across the dotted boxes tells how an edge (x,y) is replaced by a set of edges.

Informally speaking, we maintain the following semantics:

1. For $1 \leq i \leq 4$, $\xi(\mathcal{S}_1,t)(x_i)$, will hold the value (state) of the vertex x at the current time t and

2. For $5 \le i \le 9$, $\xi(S_1, t)(x_i)$, will hold the value of the state (or its complement) at x at time $t-1$.

The ideà behind the simulation is that at each time step S_1 will first compute x_i, $5 \le i \le 8$, by using values from x_i, $1 \le i \le 4$; thereby storing the value of x at time $t-1$. Then S' will compute the values of x^i based on the values of x^7 and x^8. Finally, S' will compute the values of x_i, $1 \le i \le 4$ using the newly computed value of y_i for all y such that y is a neighbor of x. We now describe the components of S_1.

1. *Graph G_1:* As shown in the figure the graph on x_i, $1 \le i \le 8$ is a complete bipartite graph with one side of the bipartition being x_i, $1 \le i \le 4$ and the other side of the bipartition is x_i, $5 \le i \le 8$. Finally the vertex x_9 is connected to x_5 and x_6. In addition, for each edge $(x, y) \in E$, S' contains the following eight edges
$$(x_9, y_1), (x_9, y_2), (x_9, y_3), (x_9, y_4)$$
$$(y_9, x_1), (y_9, x_2), (y_9, x_3), (y_9, x_4)$$

2. *Permutation π_1:* The permutation π has three components $\pi_1^1 \cdot \pi_2^1 \cdot \pi_3^1$, where each π_i is given as follows:
$$\pi_1^1 = (x_5, x_6, x_7, x_8), \qquad \forall x \in V$$
$$\pi_2^1 = x_9 \qquad \forall x \in V$$
$$\pi_3^1 = (x_1, x_2, x_3, x_4), \qquad \forall x \in V$$

3. *Initial state $\mathcal{I}_1$:* The initial state $\mathcal{I}_1$ is given as follows:
$$\mathcal{I}_1(x_1) = \mathcal{I}_1(x_2) = \mathcal{I}_1(x_3) = \mathcal{I}_1(x_4) = I(x), \quad \forall x \in V$$
$$\mathcal{I}_1(x_5) = \mathcal{I}_1(x_6) = \mathbf{0}, \quad \forall x \in V$$
$$\mathcal{I}_1(x_7) = \mathcal{I}_1(x_8) = \mathbf{1}, \quad \forall x \in V$$
$$\mathcal{I}_1(x_9) = \mathcal{I}(x), \forall x \in V$$

4. *Function Set $\mathcal{F}_1$:* The function set consists of four different functions: f_1 at nodes x_5, x_6, f_2 at nodes x_7, x_8, f_3 at x_9 and finally f_4 at $x_1, \ldots x_4$. Below we describe each of the functions in detail:

   (a) *Function f_1 at x_5 and x_6:* The nodes x_5 and x_6 have five neighbors and hence f_1 has 6 arguments. f_1 is 1 iff *at least* 4 out of its 6 arguments is 1 and is 0 otherwise. Formally, given a set of variables X, let $N(X)$ denote the number of variables set to 1. Then
$$f_1(X) = 1 \qquad \text{iff} \qquad N(X) \ge 4$$

   (b) *Function f_2 at x_7 and x_8:* f_2 is the complement of f_1; it is 1 iff less than 4 of its input values are 1. Formally,
$$f_2(X) = 1 \qquad \text{iff} \qquad N(X) < 4$$

   (c) *Function f_3 at x_9:* Node x_9 is connected to x_5 and x_6 among the copies of x and for each neighbor of x, x_9 is also connected to a group of $4\delta_x$ nodes (where δ_x is the degree of node x) as outlined earlier. f_3 is 1 iff the number of its arguments equaling 1 is congruent to **2** mod 4 or 3 mod 4. Formally,
$$f_3(X) = 1 \qquad \text{iff} \qquad N(X) \equiv 2 \bmod 4 \qquad \text{or} \qquad N(X) \equiv 3 \bmod 4$$

(d) *Function f_4 at x_1, x_2, x_3 and x_4:* Let k be the arity of f_4. Specifically, $\forall y_i \in V$ such that $(x, y_i) \in E$, we have y_i^9 connected to x_j, $1 \leq j \leq 4$. Then for $k \geq 2$, f_4 is equal to f_x[5] when $(k-2)$ of its parameters are 1. For $k < 2$, f_4 is equal to 0. Formally

$$\begin{aligned} f_4(X) &= f_x \quad \text{if} \quad k \geq 2 & (1) \\ &= 0 \quad \text{otherwise} & (2) \end{aligned}$$

We now prove the correctness of our reduction. For this we need a number of basic properties of the functions. Define $\xi(\mathcal{S}_1, -1)(x^1) = 0$.

Claim 7.1

$$\forall t \geq 0, \qquad \xi(\mathcal{S}_1,t)(x_1) \equiv \xi(\mathcal{S}_1,t)(x_2) \equiv \xi(\mathcal{S}_1,t)(x_3) \equiv \xi(\mathcal{S}_1,t)(x_4) \equiv \xi(\mathcal{S},t)(x)$$

$$\forall t \geq 0, \qquad \xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_6) \equiv \xi(\mathcal{S}_1,t-1)(x_1)$$

$$\forall t \geq 0, \qquad \xi(\mathcal{S}_1,t)(x_7) \equiv \xi(\mathcal{S}_1,t)(x_8) \equiv \overline{\xi(\mathcal{S}_1,t)(x_5)}$$

$$\forall t \geq 0, \qquad \xi(\mathcal{S}_1,t)(x_9) \equiv \xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S},t-1)(x)$$

In other words, after one round of execution, the state of x_1 in $\mathcal{S}_1$ is the same as the state of x in $\mathcal{S}$ and x_9 remembers the state of x in the previous time unit.

Proof: The proof is by induction on t. The basis for $t = 1$ can be verified using the definitions of $\mathcal{I}_1$ and $\xi(\mathcal{S}_1, -1)(x^1)$. Assume that the claim holds for all times $t' < t$ and consider time t.

(1): It is clear that for $1 \leq i \leq 4$, $\xi(\mathcal{S}_1,t)(x_i)$ are the same. Note that at time $t-1$, exactly two of x_i, $5 \leq i \leq 8$ are 1 and the other two are 0. Thus if in the operation of $\mathcal{S}$, at time t, exactly j of the inputs to x are equal to 1, then in the operation of $\mathcal{S}_1$, $j+2$ of its inputs are 1. This along with the definition of f_4 proves the required claim.

(2): First note that $\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_6)$. By induction hypothesis, $1 \leq i \leq 4$, $\xi(\mathcal{S}_1,t)(x_i)$ all have the same value. In particular they are all simultaneously 1 or 0. Thus by definition of $f_1()$, it is clear that $\xi(\mathcal{S}_1,t)(x_5)$ and $\xi(\mathcal{S}_1,t)(x_6)$ solely depend on the value of x_1, x_2, x_3 and x_4 and are independent of the value at x_9 and their own value at the previous time.

(3): The proof of the fact that $\forall t > 0$, $\xi(\mathcal{S}_1,t)(x_7) \equiv \xi(\mathcal{S}_1,t)(x_8)$ follows along the lines of proof for part (2). To show that they in turn are equivalent $\overline{\xi(\mathcal{S}_1,t)(x_5)}$ follows by the definition of $f_2()$.

(4): First note that the neighbors x_5 and x_6 of x_9 have the same values at time t and they come before x_9 in our permutation. Each set of 4 values from x_9's neighbors is identical and thus all these values sum up to 0 mod 4. Also note that $\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_6)$. We thus have two cases to consider.

Case 1: $\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_6) \equiv 1$ and $\xi(\mathcal{S}_1,t-1)(x_9) \equiv 1$. Then $N(X) \equiv 3 \bmod 4$ and hence $\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_9) \equiv 1$.

Case 2: $\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_6) \equiv 1$ and $\xi(\mathcal{S}_1,t-1)(x_9) \equiv 0$. Then $N(X) \equiv 2 \bmod 4$ and hence $\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_9) \equiv 0$.

Case 3: $\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_6) \equiv 0$ and $\xi(\mathcal{S}_1,t-1)(x_9) \equiv 0$. Then $N(X) \equiv 0 \bmod 4$ $\xi(\mathcal{S}_1,t)(x_5) \equiv \xi(\mathcal{S}_1,t)(x_9) \equiv 0$.

[5]Recall that f_x denotes the function at node x in $\mathcal{S}$.

Case 4: $\xi(S_1,t)(x_5) \equiv \xi(S_1,t)(x_6) \equiv 0$ and $\xi(S_1,t-1)(x_9) \equiv 1$. Then $N(X) \equiv 1 \bmod 4$ $\xi(S_1,t)(x_5) \equiv \xi(S_1,t)(x_9) \equiv 0$. ■

By the above claim, it is clear that there is a bijection between $\xi(S_1,t)$ and $\xi(S,t)$; thus the reachability problem and the fixed point problem for S are polynomial time reducible to the reachability and the fixed point problem respectively for S_1. This completes the proof of our theorem. ■

Using earlier discussion, we can thus prove the following

Theorem 7.2 *The* REACHABILITY *and* FIXED POINT *problems for SDS with symmetric (non-identical) Boolean functions are* **PSPACE***-hard, even when restricted to SDSs such that:*

- *Each node is the SDS graph is of constant degree, and*
- *the pathwidth and hence the treewidth of the graph is bounded by a constant that depends on m and h, and*
- *The number of distinct functions used is a constant (that depends only on the number of distinct functions used in the proof of Theorem 5.2).*

Proof: The proof follows by noting that the construction outlined in proof of Theorem 7.1 replaces each node of the SyDS S by a set of 9 nodes. Moreover, there can be no edges between two nodes in different sets if the original nodes they replaced did not have an edge. ■

7.2 Extension to identical functions

Theorem 7.3 *Given an SDS (or an SyDS) S $(G(V,E),\mathcal{F},\mathcal{I})$, with $\mathcal{F} = \{f^1, f^2, \ldots f^q\}$ being the set of distinct symmetric Boolean functions, we can create an instance S_2 $(G_2(V_2,E_2),\mathcal{F}_2,\pi,\mathcal{I}_2)$ of SDS (SyDS) in polynomial time such that*

1. *$\mathcal{F}_2 = \{f\}$; i.e. $\mathcal{F}_2$ has the same function at each node.*
2. *If the original graph has constant degree then G_2 also has constant degree.*

Proof: We do the proof in two steps: In the first step, we construct an instance S_1 $(G_1(V_1,E_1),\mathcal{F}_1,\pi)$ in which we have the same functions but of varying degrees at each node. In step 2, we show how to extend this idea to construct a new SDS S_2 such that each node has identical function.
Step 1: Consider a SDS or SyDS S. It can be modified so that all the Boolean functions are the same, as follows. (This first reduction gives the nodes varying degree, but they all have the "same" symmetric Boolean function.)

Let Δ be the maximum degree of the nodes in the graph G of S. Let q be the number of distinct Boolean functions occurring in S. Thus, for each node i, function f_i is function f^j for some j $(1 \le j \le q)$. The constructed system S_1 will involve a *single* Boolean function f; although they will have varying number of inputs. Since f is symmetric, we can describe f by specifying what its value is, as a function of how many of its input parameters are 1. Thus, we use the shorthand of specifying f using an integer as its parameter (this integer represents how many of its Boolean inputs are 1). The specification of f is:

$$f(0) = 0$$
$$f(1) = 1$$
$$f(2) = 1$$
$$f(k) = 0;\ \ 2 < k < \Delta + 2$$

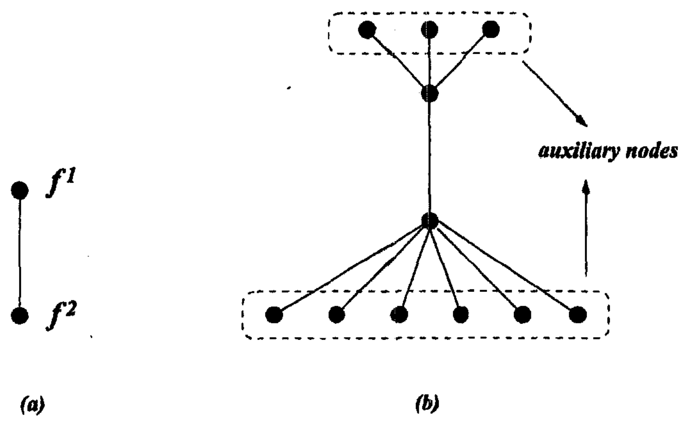

Figure 2: Figure explaining the construction of the single function in Theorem 7.3.

$$f(j(\Delta+2)+k) = f^j(k); \qquad 1 \le j \le q, \qquad 0 \le k \le \Delta+1$$

Note: If function f^j has fewer than k input arguments, then $f(j(\Delta+2)+k) = 0$.

$G_1(V_1, E_1)$ is given as follows: $V_1 = W \cup U$, where W is one to one correspondence with the nodes in V and $E_1 = E_W \cup E_{WU}$. The edges in E_W go between the nodes in W, the graph (W, E_W) is isomorphic to (V, E). The nodes in U are called auxiliary nodes. There are no edges between nodes in U. Each node in W is connected to a specified number of distinct auxiliary nodes in U. The auxiliary nodes have degree 1. The state of each auxiliary node is initialized to 1, and so (since $f(1) = f(2) = 1$) will be in state 1 throughout the operation of $\mathcal{S}_1$. Suppose that in $\mathcal{S}$, the function for node i is f^j. (Recall that $1 \le j \le q$.) Then, in G_1 node $i \in W$ is connected to $j(\Delta+2)$ auxiliary nodes. This ensures that in each step of the operation of $\mathcal{S}_1$, function f effectively selects the value that function f^j would select for the number of original input values that equal 1. The reduction as outlined works for SyDS since there is no permutation. To complete the reduction for SyDS, we need to specify the ordering of the vertices in G_1. $\pi_1 = \pi_1^1 \cdot \pi_2^1$, where π_1^1 is applied to vertices in W and is identical to π. π_2^1 is a permutation of U and does not really affect our construction. We let π_2^1 be the permutation in which the vertices in U appear in ascending order of their indices in G_1. Intuitively speaking, the function f represents a master function that can act like any function f^j. The auxiliary inputs drive the function into distinct range of 1-inputs (with different number of 1's).

Claim 7.2 *There is a bijection between $\xi(\mathcal{S}_1, t)$ and $\xi(\mathcal{S}, t)$. Specifically, at each time t, $\xi(\mathcal{S}_1, t)(W) \equiv \xi(\mathcal{S}, t)(V)$ and $\xi(\mathcal{S}_1, t)(U) = 1$.*

The claim completes the proof of step 1.

Step 2: A similar but a slightly more complicated construction can give all the nodes the same degree (as well as the same function). This more complicated construction is summarized as follows. Let $D = \Delta + q(\Delta+2)$. Recall that q denotes the number of distinct functions. Each node in $\mathcal{S}_2$ will have degree D. Function f is the same as described above, except that now $f(1) = 0$, and $f(3) = 1$. (For all other counts, the value of f is the same as given above.) There are now two classes of auxiliary nodes: auxiliary 1-nodes, that are initialized to 1, and auxiliary 0-nodes, that are initialized to 0. The auxiliary nodes described above all become auxiliary 1-nodes. In addition, each original node is padded with connections to enough auxiliary 0-nodes so that its degree is D.

The auxiliary nodes need to have their degree made D. Each auxiliary 1-node is connected to one other auxiliary 1-node, and possibly one original node. Its remaining D neighbors are auxiliary 0-nodes. This ensures, that at any time during the operation of $\mathcal{S}_2$, for a 1-node, the number of inputs to f that equal 1 is either 2 or 3, and so the state remains at 1 throughout the operation of $\mathcal{S}_2$.

The auxiliary 0-nodes are grouped into cliques of cardinality $D - 1$. (Additional auxiliary 0-nodes can be added to round out the final clique.) Each auxiliary 0-node is connected to at most one original node or auxiliary 1-node, and its remaining neighbors are other auxiliary 0-nodes. This ensures that its state remains 0 throughout the operation of S_2. ■

7.3 Extension to the weighted k-simple-Threshold functions

We extend the result to k-Threshold-SDSs to a weighted version of the problem. Informally, we have weights on edge of the graph. Now we treat 0's and 1's symmetrically as before but the threshold is formed by the weighted sum of inputs whose other end point is 0 (or dually 1). Formally, each edge $e = (u, v)$ has a numerical weight (could be negative) denoted by $w(e)$. For each node v, let

$$T(v) = \sum_{w \in N(v)} w((u, w))$$

i.e. it is the sum of the weights of the edges incident on v. For any assignment to the node variables, let

$$W_0(v) = \sum_{(v,w),\ \mathrm{v}[w]=0} w(u, w) \qquad \text{and} \qquad W_1(v) = \sum_{(v,w),\ \mathrm{v}[w]=1} w(u, w)$$

Note that since there are no self-loops $W_0(v)$ and $W_1(v)$ are independent of the value of node v.

Each node v has two values $T_0(v)$ and $T_1(v)$ such that $T_0(v) + T_1(v) > T(v)$. Let $g(v) = T0(v) + T1(v) - T(v)$ be called the "gap" for v. The function at node v is defined as follows:

$$\begin{aligned} \mathrm{v}[v] &= 0 \text{ if } W_0(v) \geq T_0(v) \\ &= 1 \text{ if } W_1(v) \geq T_1(v) \\ &= \text{unchanged} \end{aligned}$$

In other words, if the weighted number of 0-neighbors is greater than the 0-threshold $T_0(v)$ then node value becomes 0. Similarly, if the weighted number of 1-neighbors is greater than the 1-threshold $T_1(v)$ then node value becomes 1. Else the node does not change the value that it had before the update. We denote an SDS with weighted functions at nodes as above by Wt-k-Threshold-SDS.

Theorem 7.4 *The* t-REACHABILITY *problem and the* FIXED POINT REACHABILITY *problem for Wt-k-Threshold-SDSs,* $1 \leq k \leq n$, *are computable in polynomial time if if the edge weights are all polynomial in the size of the problem.*

Wt-k-Threshold-SDSs do not have limit cycles of length greater than 1 (i.e. they only have transients and fixed points).

Proof: The proof follows the earlier proof for unweighted case and is thus brief. For any assignment, we associate a cost to each edge and node as follows:

$$\begin{aligned} P(v) &= T_1(v) \quad \text{if } \mathrm{v}[v] = 1 \\ &= T_0(v) \quad \text{if } \mathrm{v}[v] = 0 \end{aligned}$$

The edge potential $P(e)$ of an edge $e = (u, v)$ is define as follows:

$$\begin{aligned} P(e) &= w \text{ if } e = (u, v),\ w(e) = w \text{ and } \mathrm{v}[u] \neq \mathrm{v}[v] \\ &= 0 \text{ otherwise} \end{aligned}$$

As in the proof of unweighted case we show that whenever the value at a node v changes, the total potential drops by at least the "gap" $g(v)$.

Consider a particular node v whose value changes from 0 to 1.

1. Edges incident on v where the other end point is 1 have their costs reduced to 0. For these edges we get a total reduction in potential is $W_1(v) \geq T_1(v)$.

2. Edges incident on v where the other end point is 0 have their costs increased to the edge weight for a total increase of $W_0(v)$ which is equal to $T(v) - W_1(v) \leq T(v) - T_1(v) = T_0(v) - g(v)$.

3. The node cost is increased by $T_1(v)$ and decreased to $T_0(v)$.

Combining the gains and losses gives a net decrease of

$$T_1(v) + T_0(v) - (T_0(v) - g(v)) \geq g(v).$$

By symmetry, when a node v changes from 1 to 0, we also get a reduction in the potential of at least $g(v)$. ∎

If we extend the model to allow edges to have unequal effects on its endpoints (or equivalently use a directed graph) then convergence is lost because bits can circulate from node to node. An open question is settle the complexity of the reachability problem under this model.