Discrete Mathematics and Theoretical Computer Science **AB(DMCS)**, 2003, 69–80

Predecessor and Permutation Existence Problems for Sequential Dynamical Systems

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

[1] *Basic and Applied Simulation Science (CCS-5) and National Infrastructure Simulation and Analysis Center (NISAC), Los Alamos National Laboratory, P.O. Box 1663, MS M997, Los Alamos, NM 87545. Email: {barrett, marathe}@lanl.gov. The work is supported by the U.S. Department of Energy under contract W-7405-ENG-36.*

[2] *Department of Computer Science, University at Albany - SUNY, Albany, NY 12222. Email: {hunt, ravi, djr, res}@cs.albany.edu. Supported by a grant from Los Alamos National Laboratory and by NSF Grants CCR-97-34936 and CCR-01-05536.*

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

A class of finite discrete dynamical systems, called **Sequential Dynamical Systems** (SDSs), was introduced in [BR99] as a formal model for analyzing simulation systems. Here, we address the complexity of two basic problems and their generalizations for SDSs.

Given an SDS $\mathcal{S}$ and a configuration $\mathcal{C}$, the PREDECESSOR EXISTENCE (or PRE) problem is to determine whether there is a configuration $\mathcal{C}'$ such that $\mathcal{S}$ has a transition from $\mathcal{C}'$ to $\mathcal{C}$. Our results provide separations between efficiently solvable and computationally intractable instances of the PRE problem. For example, we show that the PRE problem can be solved efficiently for SDSs with Boolean state values when the node functions are symmetric and the underlying graph is of bounded treewidth. In contrast, we show that allowing just one non-symmetric node function renders the problem **NP**-complete even when the underlying graph is a star (which has a treewidth of 1). Our results extend some of the earlier results by Sutner [Su95] and Green [Gr87] on the complexity of the PREDECESSOR EXISTENCE problem for 1-dimensional cellular automata.

Given two configurations $\mathcal{C}$ and $\mathcal{C}'$ of a partial SDS $\mathcal{S}$, the PERMUTATION EXISTENCE (or PME) problem is to determine whether there is a permutation of nodes such that $\mathcal{S}$ has a transition from $\mathcal{C}'$ to $\mathcal{C}$ in one step. We show that the PME problem is **NP**-complete even when the function associated with each node is a simple-threshold function. We also show that the problem can be solved efficiently for SDSs whose underlying graphs are of bounded degree and bounded treewidth. We consider a generalized version (GEN-PME) of the PME problem and show that the problem is **NP**-complete for SDSs where each node function is NOR and the underlying graph has a maximum node degree of 3. When each node computes the OR function or when each node computes the AND function, we show that the GEN-PME problem is solvable in polynomial time.

Keywords: Discrete Dynamical Systems, Cellular Automata, Predecessor Existence, Permutation Existence, Computational Complexity

1 Introduction and Motivation

We study the computational complexity of some basic problems that arise in the context of a new class of discrete finite dynamical systems, called **Sequential Dynamical Systems** (henceforth referred to as SDSs), proposed in [BR99]. A formal definition of such a system is given in Section 2. SDSs are closely related to classical Cellular Automata (CA), a widely studied class of dynamical systems for modeling complex systems. They are also closely related to an extension of CA called **graph automata** [NR98]. Decidability issues for dynamical systems in general and CA in particular have been widely studied in the literature [Wo86, Gu89]. In contrast, computational complexity questions arising in the study of CA and related dynamical systems have received comparatively less attention.

In simple terms, an SDS $\mathcal{S} = (G, \mathcal{F}, \pi)$ consists of three components. $G(V,E)$ is an undirected graph with n nodes with each node having a state. $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, with f_i denoting a function associated with node v_i. π is a permutation of (or a total order on) the nodes in V. A **configuration** of an SDS is an n-vector $(b_1, b_2, \ldots, b_n)$, where b_i is the value of the state of node v_i $(1 \leq i \leq n)$. A single SDS transition from one configuration to another is obtained by updating the state of each node using the corresponding function. These updates are carried out in the order specified by π.

The research reported here is a part of a program[†] to provide a formal basis for the design and analysis of *large-scale computer simulations*, especially for socio-technical systems [BE+01]. Examples of such systems include various national infrastructures including transportation, power and communication. Such systems are prototypical examples of discrete coupled dynamical systems. An important aspect of such systems is that unlike many physical systems, these systems do not have an *equation of state*. For additional information regarding how SDSs can be used to model simulations, we refer the reader to [BE+01].

In [BH+01b, BE+01], we studied the complexity of some configuration reachability problems for SDSs. Here, we focus on the complexity of two basic problems for SDSs, namely PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE. We now discuss these problems informally and defer the formal definitions to Section 2.4. Given an SDS $\mathcal{S} = (G, \mathcal{F}, \pi)$ and a configuration $\mathcal{C}$, the PREDECESSOR EXISTENCE problem (abbreviated as PRE) is to determine whether the configuration $\mathcal{C}$ has a predecessor; that is, whether there is a configuration $\mathcal{C}'$ such that $\mathcal{S}$ has a transition from $\mathcal{C}'$ to $\mathcal{C}$ in one step. Given a partially specified SDS $\mathcal{S}$ consisting of the underlying graph $G(V,E)$, the set $\mathcal{F}$ of functions associated with the nodes and two configurations $\mathcal{C}$ and $\mathcal{C}'$, the PERMUTATION EXISTENCE problem (abbreviated as PME) is to determine whether there is a permutation π of the nodes such that under permutation π, there is a transition from $\mathcal{C}'$ to $\mathcal{C}$ in one step. The PRE problem is a classical problem studied by the dynamical systems community in the context of CA [Su95, Gr87]. The PME problem is important in the context of SDSs since two different node permutations may give rise to totally different behaviors of the underlying dynamical system. An investigation of these problems is helpful in obtaining a better understanding of the dynamical systems modeled by SDSs. The predecessor existence question is directly related to certain liveness properties of some network protocols [BE+01, Pe97]. Our conclusion is that these questions are, in general, computationally intractable. However, we identify a number of special classes of SDSs for which the questions can be answered efficiently. Several of our results are also applicable to cellular automata (CA) and graph automata (GA).

Both PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE problems occur naturally while reverse engineering sequential dynamical systems from time series data. Examples of reverse engineering

[†] See `http://tsasa.lanl.gov` for additional details on this program.

problems where sequential dynamical system framework has been applied recently include the work of Laubenbacher et.al. [LSS03].

The remainder of the paper is organized as follows. In Section 2 we provide the necessary definitions. Section 3 summarizes our results and discusses related results from the literature. Section 4 presents a proof to illustrate the ideas behind some of our efficient algorithms.

2 Definitions and Problem Formulations

2.1 Sequential Dynamical Systems

A **Sequential Dynamical System** (SDS) $\mathcal{S}$ over a given domain $\mathbb{D}$ of state values is a triple $(G, \mathcal{F}, \pi)$, whose components are as follows:

1. $G(V,E)$ is a finite 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. For each node i of G, $\mathcal{F}$ specifies a **local transition function**, denoted by f_i. This function maps $\mathbb{D}^{\delta_i+1}$ into $\mathbb{D}$, where δ_i is the degree of node i. Letting $N(i)$ denote the set consisting of node i itself and its neighbors, each parameter of f_i corresponds to a member of $N(i)$. Throughout the paper, we assume that the local transition function associated with each node can be evaluated in polynomial time.

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.

A **configuration** $\mathcal{C}$ of $\mathcal{S}$ can be interchangeably regarded as an n-vector $(c_1, c_2, \ldots, c_n)$, where each $c_i \in \mathbb{D}$, $1 \le i \le n$, or as a function $\mathcal{C} : V \to \mathbb{D}$. From the first perspective, c_i is the state value of node i in configuration $\mathcal{C}$, and from the second perspective, $\mathcal{C}(i)$ is the state value of node i in configuration $\mathcal{C}$.

Computationally, each step of an SDS (i.e., the transition from one configuration to another), involves n substeps, where the nodes are processed in the *sequential* order specified by permutation π. The "processing" of a node consists of computing the value of the node's local transition function and changing its state to the computed value. The following pseudocode shows the computations involved in one transition.

---

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 $\pi(i)$.) Let x denote the value computed.
(ii) Node $\pi(i)$ sets its state $s_{\pi(i)}$ to x.
end-for

We let $F_{\mathcal{S}}$ denote the **global transition function** associated with $\mathcal{S}$. This function can be viewed either as a function that maps $\mathbb{D}^n$ into $\mathbb{D}^n$ or as a function that maps $\mathbb{D}^V$ into $\mathbb{D}^V$. $F_{\mathcal{S}}$ represents the transitions between configurations, and can therefore be considered as defining the dynamic behavior of SDS $\mathcal{S}$. Let I denote the designated configuration of $\mathcal{S}$ at time 0. Starting with I, the configuration of $\mathcal{S}$ after

t steps (for $t \geq 0$) is denoted by $\xi(\mathcal{S}, I, t)$. Note that $\xi(\mathcal{S}, I, 0) = I$ and $\xi(\mathcal{S}, I, t+1) = F_{\mathcal{S}}(\xi(\mathcal{S}, I, t))$. Consequently, for all $t \geq 0$, $\xi(\mathcal{S}, I, t) = F_{\mathcal{S}}{}^{t}(I)$.

Recall that a configuration $\mathcal{C}$ can be viewed as a function that maps V into $\mathbb{D}$. As a slight extension of this view, we use $\mathcal{C}(W)$ to denote the states of the nodes in $W \subseteq V$. $\mathcal{C}(W)$ is called a **subconfiguration** of $\mathcal{C}$.

The **phase space** $\mathcal{P}_{\mathcal{S}}$ of an SDS $\mathcal{S}$ is a directed graph defined as follows: There is a node in $\mathcal{P}_{\mathcal{S}}$ for each configuration of $\mathcal{S}$. There is a directed edge from a node representing configuration $\mathcal{C}$ to that representing configuration $\mathcal{C}'$ if $F_{\mathcal{S}}(\mathcal{C}) = \mathcal{C}'$. In such a case, we also say that configuration $\mathcal{C}$ is a **predecessor** of configuration $\mathcal{C}'$. Since SDSs are deterministic, each node in its phase space has an outdegree of 1. In general, the phase space $\mathcal{P}_{\mathcal{S}}$ may have an infinite number of nodes. When the domain $\mathbb{D}$ of state values is finite, the number of nodes in the phase space is $|\mathbb{D}|^n$.

2.2 Variations of the Basic SDS Model

The above definition of an SDS imposes no restrictions on either the domain $\mathbb{D}$ of state values or the local transition functions, except that the ranges of the local transition functions must be subsets of $\mathbb{D}$. SDSs that model simulation systems can be obtained by appropriately restricting $\mathbb{D}$ and/or the local transition functions. We use the notation "(x,y)-SDS" to denote an SDS where 'x' specifies the restriction on the domain and 'y' specifies the restriction on the local transition functions. Some restrictions studied in this paper are discussed below.

References [BR99, BMR00, MR99, Re00] studied SDSs with Boolean domains and symmetric Boolean local transition functions. (The definition of symmetric Boolean functions is given in Section 2.3.) We denote such an SDS by (BOOL, SYM)-SDS. Symmetric functions provide 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]. As pointed out in Section 3, symmetry plays an important role in separating efficiently solvable and computationally intractable versions of the PRE problem for SDSs whose underlying graphs have bounded treewidth.

A restricted class of (BOOL, SYM)-SDSs is obtained by requiring the local transition functions to be *monotone* as well. It can be seen that a Boolean function is symmetric and monotone if and only if it is a k-simple threshold function (defined in Section 2.3) for some integer $k \geq 0$. Therefore, over the Boolean domain, the class of SDSs where each node function is symmetric and monotone coincides with the class of SDSs where each node function is a k-simple-threshold function for some integer $k \geq 0$. We denote this restricted class by (BOOL, ST)-SDS. A class of symmetric but not monotone Boolean functions is that of *k-tally* functions (defined in Section 2.3) for any integer $k \geq 0$. The class of SDSs in which each node computes the k-tally function for some nonnegative integer k is denoted by (BOOL, TALLY)-SDS. SDSs in which the local transition functions are not necessarily symmetric are also considered in the companion papers [BH+01a, BH+01b].

Some of our results are for SDSs over the Boolean domain where the node functions are from an explicitly specified set of Boolean functions. For example, when each node function is AND, the corresponding class of SDSs is denoted by (BOOL, AND)-SDS. If the set of node functions has cardinality two or more, then standard set notation is used as the restriction on the local transition functions in the notation for such SDSs. For example, when each node function is from the set {XOR, XNOR}, the corresponding class of SDSs is denoted by (BOOL, {XOR, XNOR})-SDS.

We also consider SDSs over an algebraic field where the node functions are linear combinations of the inputs. The class of such SDSs is denoted by (FIELD, LINEAR)-SDS. Each local transition function

$f_i,\ 1 \le i \le n,$ has the following form:

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

Here, α_i and a_{ij} ($1 \le i \le n$ and $1 \le j \le r$) are (scalar) constants, s_j is the state value of node v_j, and '$+$' (addition) and '$*$' (scalar multiplication) are the operators of the field.

It is also of interest to consider dynamical system models obtained by modifying some components of an SDS. One such model is a **Synchronous Dynamical System** (SyDS), which is 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 restrictions on domain and local transition functions for SDSs are applicable to SyDSs as well.

2.3 Other relevant Definitions

Several special classes of Boolean functions are considered in this paper. We provide formal definitions of these classes below.

A **symmetric** Boolean function is one whose 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. The k-**simple-threshold** (Boolean) function has value 1 if at least k of the inputs have value 1; otherwise, the value of the function is zero. The k-**tally** (Boolean) function has value 1 if exactly k of the inputs have value 1; otherwise, the value of the function is zero. It should be noted that both k-simple-threshold and k-tally functions are symmetric. The k-simple-threshold functions are also monotone.

In this paper, **NP**-completeness results are established using reductions from variants of the Satisfiability (SAT) problem. An instance of SAT is specified by a collection $X = \{x_1, x_2, \ldots, x_n\}$ of n Boolean variables and a collection $C = \{c_1, c_2, \ldots, c_m\}$ of m clauses, where each clause is a set of literals. The question is whether there is an assignment of Boolean values to the variables so that each clause is satisfied (i.e., contains at least one true literal).

Comparing SyDS and SDS. The main difference between graph automata (and SyDS) and SDSs is that in the former, node states are updated in parallel while in latter, they are updated in a specified sequential order. From a modeling standpoint, having a partial order on the update seems more appropriate since it captures causality between agents. Examples where such an ordering seems appropriate are considered in [BE+01, HG99]. Reference [BE+01] discusses an example in which the movement of cars on a single-lane road is modeled using an SDS. It is shown that the order in which the state information for cars is updated yields completely different dynamics. For instance, when the state information is updated from front to back, the system acts like a perfect predictor and thus never yields clusters of vehicles. On the other hand, updating from back to front yields more realistic traffic dynamics. Recently, other authors [HG99, Ga97, Rk94] have also considered sequential updates. In particular, Huberman and Glance [HG99] discuss experimentally how certain simulations of n-person games exhibit very different (but probably more realistic) dynamics when the cells are updated sequentially as opposed to when they are updated in parallel.

Computationally, SDS and SyDS also exhibit interesting differences as well as similarities. Our polynomial time results for class of bounded treewidth graphs apply to both SyDS and SDS. Similar conclusions hold for many other polynomial time solvable cases as well. On the other hand, it is known that the phase space of SyDS and SDS when all local functions are simple threshold functions show important differences. Specifically, the phase space of SDS has only fixed points (for every permutation) while SyDS can have both fixed points and limit cycles of length 2. As another example, Mortveit and Reidys [MR99] completely characterize the question of when an SDS is invertible. However, this is a well known open question for SyDSs. See [BE+01, BH+01b] for additional discussion on this topic.

2.4 Problems Considered

In this paper, we study two basic problems and their extensions that arise in the context of SDSs. Some of these problems have been studied in the context of CA. We provide formal definitions of the two basic problems below.

Given an SDS $\mathcal{S} = (G(V,E), \mathcal{F}, \pi)$ and a configuration $\mathcal{C}$, the PREDECESSOR EXISTENCE problem (abbreviated as PRE) is to determine whether there is a configuration $\mathcal{C}'$ such that $F_{\mathcal{S}}(\mathcal{C}') = \mathcal{C}$. (Note that $\mathcal{C}$ has a predecessor if and only if $\mathcal{C}$ is not a garden of Eden configuration.) Given a partially specified SDS $\mathcal{S}$ consisting of graph $G(V,E)$, the set $\mathcal{F}$ of local transition functions associated with the nodes of G, an initial configuration $\mathcal{C}'$ and a final configuration $\mathcal{C}$, the PERMUTATION EXISTENCE problem (abbreviated as PME) is to determine whether there is a permutation π for $\mathcal{S}$ such that $F_{\mathcal{S}}(\mathcal{C}') = \mathcal{C}$.

As stated, the PRE problem asks for an immediate predecessor; that is, whether there is a configuration $\mathcal{C}'$ from which a given configuration $\mathcal{C}$ can be reached in one transition. It is possible to generalize the problem to the case where we are given an integer $t \geq 1$, and the goal is to determine whether there is a configuration $\mathcal{C}'$ from which $\mathcal{C}$ can be reached in exactly t transitions. We call this the t-PRE problem.

Given an SDS $\mathcal{S} = (G(V,E), \mathcal{F}, \pi)$, let $W = \{v_{i_1}, \ldots, v_{i_k}\}$ be a subset of nodes in V. Recall that the states of nodes in W is the subconfiguration $(b_{i_1}, b_{i_2}, \ldots, b_{i_k})$. Using this notation, we can state the SDS analogs of some problems considered ‡ by Green [Gr87] in the context of infinite CA. These problems can be viewed as extensions of the t-PRE problem.

Given an SDS $\mathcal{S} = (G(V,E), \mathcal{F}, \pi)$, a subset of nodes W, a vector $B = (b_{i_1}, b_{i_2}, \ldots, b_{i_k})$ that specifies state values for the nodes in W and an integer $t \geq 1$, the t-SUBCONFIGURATION PREDECESSOR EXISTENCE problem (abbreviated as t-SUB-PRE) is to determine whether there are configurations $\mathcal{C}'$ and $\mathcal{C}$ such that ${F_{\mathcal{S}}}^t(\mathcal{C}') = \mathcal{C}$ and B is a subconfiguration of $\mathcal{C}$. The t-SUBCONFIGURATION RECURRENCE problem (abbreviated as t-SUB-RECUR) is to determine whether there are configurations $\mathcal{C}'$ and $\mathcal{C}$ such that ${F_{\mathcal{S}}}^t(\mathcal{C}') = \mathcal{C}$ and B is a subconfiguration of both $\mathcal{C}'$ and $\mathcal{C}$; in other words, the question is whether the subconfiguration represented by B will occur again after exactly t time steps.

Given an SDS $\mathcal{S} = (G(V,E), \mathcal{F}, \pi)$, an integer $t \geq 1$ and a temporal sequence $\langle b_v(1), b_v(2), \ldots, b_v(t)\rangle$ of t state values of a given node v, the t-TEMPORAL SEQUENCE PREDECESSOR EXISTENCE problem (abbreviated as t-TEMP-SEQ-PRE) is to determine whether there is a configuration $\mathcal{C}$ such that ${F_{\mathcal{S}}}^j(\mathcal{C})(v) = b_v(j),\ 1 \leq j \leq t$.

Another way of specifying a subconfiguration of an SDS is to give suitable state values for some nodes and "don't-care" values for other nodes. The corresponding subconfiguration is obtained by retaining only the non-don't-care values. Using this idea, it is possible to formulate a generalized version of the PME problem as follows: Given a partially specified SDS $\mathcal{S}$ consisting of graph $G(V,E)$, the set $\mathcal{F}$ of local transition functions associated with the nodes of G, an initial configuration $\mathcal{C}'$ and a final configuration $\mathcal{C}$

‡ The actual definitions in [Gr87] are slightly different. Since the main goal of [Gr87] was to prove **NP**-completeness, the problems were formulated with time t equal to the number of nodes in the subconfiguration.

possibly containing don't-care values, the GENERALIZED PERMUTATION EXISTENCE problem (abbreviated as GEN-PME) is to determine whether there is a permutation π for $\mathcal{S}$ such that $F_{\mathcal{S}}(\mathcal{C}')$ and $\mathcal{C}$ agree in all the components of $\mathcal{C}$ that have non-don't-care values. (In other words, $\mathcal{C}$ succinctly specifies a set of configurations, and $F_{\mathcal{S}}(\mathcal{C}')$ may be any one of these configurations.)

3 Summary of Results and Related Work

As stated earlier, we study the complexity of PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE problems for sequential dynamical systems. The paper makes two basic contributions. First, we classify the complexity of these problems providing in many cases tight separation between *easy* and *hard* instances. These results also show a trade-off between the underlying connectivity structure and the class of local functions used in determining the complexity of the problem. For example, PRE problem is easy for (BOOL, AND)-SDSs on general graphs and for (BOOL, SYM)-SDSs when underlying graphs have bounded treewidth. In contrast, PRE problem is **NP**-complete when each node computes the same k-simple-threshold function for any $k \geq 2$. Notice that as the graph structure becomes more restrictive, more general functions can be accommodate to obtain polynomial time results. In classifying the complexity, we initiate the study of discrete dynamical systems on graphs of bounded treewidth. As discussed later, this strictly includes 1-dimensional CAs with bounded radii. Second, in obtaining our easiness results, we show a strong connection between PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE problems and the classical generalized satisfiability problems that have been extensively studied in Computer Science (See e.g. [CKS01a]). This allows us to use the known polynomial time algorithms for such satisfiability problems to obtain analogous easiness results for PREDECESSOR EXISTENCE and PERMUTATION EXISTENCE problems. While we will not discuss this further here, most of our easiness results for treewidth and degree bounded graphs hold not only for Boolean domains but arbitrary finite domains. Moreover, using the general framework presented by Stearns and Hunt [SH96], we can show that our results extend to variants of these two basic problems, e.g. count the number of predecessors, count the number of predecessors with specified value for some of the nodes, etc. Answering such questions is useful in the context of reverse engineering problems such as those discussed in [LSS03].

A natural question to ask is whether a different local transition function is needed at each node. From a modeling perspective, this is certainly beneficial. For instance, in our simulation of deregulated electrical markets [BE+01], we needed to represent various types of players: monopolistic, oligopolistic, etc. A similar need arises in the context of simulating wireless communication systems, wherein different local transition functions correspond to the use of different protocols at each node. Computationally, most of our hardness results hold even when all local transition functions are identical. In contrast, many of our polynomial time results hold when the local transition functions are not necessarily identical (although the results assume that each local transition function is chosen from a given class of functions).

3.1 PREDECESSOR EXISTENCE *Problem*

Sutner [Su95] and Green [Gr87] considered the PRE problem and its generalizations in the context of CA. Their work motivated our study the PRE problem for SDSs.

We show that the PRE problem is **NP**-complete for each of the following restricted classes of SDSs: (i) (BOOL, ST)-SDSs where each node computes the same k-simple-threshold function for any $k \geq 2$ (ii) (BOOL, TALLY)-SDSs in which each node computes the same k-tally function for any $k \geq 1$, (iii) (BOOL, $\{$AND, OR$\}$)-SDSs and (iv) (BOOL, SYM)-SDSs whose underlying graphs are planar.

We present polynomial time algorithms for the PRE problem for (BOOL, AND)-SDSs and (BOOL, OR)-SDSs with no restrictions on the underlying graph. We also present polynomial time algorithms for the PRE problem for (BOOL, SYM)-SDSs whose underlying graphs have bounded treewidth. In contrast, we show that even if one nonsymmetric local transition function is permitted, the PRE problem is **NP**-complete for SDSs whose domain is Boolean and whose underlying graphs are star graphs. In addition, we present polynomial time algorithms for the PRE problem for the following cases: (a) (FIELD, LINEAR)-SDSs, with no restriction on the underlying graph and (b) SDSs whose underlying graphs have bounded degree and bounded treewidth with no restriction on the local transition functions (other than that the function evaluations can be done efficiently).

These algorithms can be extended to solve the t-PRE, t-SUB-PRE, t-SUB-RECUR and t-TEMP-SEQ-PRE problems in polynomial time when when t is a constant. Many of these polynomial time algorithms are obtained by reducing the corresponding problem to a generalized satisfiability problem [CKS01a] that can be solved in polynomial time.

Many of the polynomial time results can be directly extended to SyDSs. Our results extend the earlier work of Sutner [Su95] and Green [Gr87] on the complexity of the PREDECESSOR EXISTENCE problem for CA. For instance, our polynomial time solvability results for the class of SDSs whose underlying graphs are both degree and treewidth bounded, extend Sutner's result since CA can be seen to have bounded treewidth (in fact, they are of bounded bandwidth). Finally, our polynomial time results can be extended to solve a number of other variants when instances are restricted to degree and treewidth bounded graphs. Examples other than those mentioned above include the problem of finding a predecessor with maximum (or minimum) number of state values being 1 (or 0), etc.

Our lower bounds for PREDECESSOR EXISTENCE problem restricted to (BOOL, ST)-SDS directly imply analogous lower bounds for Hopfield networks with sequential update and to questions related to communicating finite state machines [BZ83, Mi99, Pe97] To our knowledge, such results have not been reported earlier.

3.2 *The* PERMUTATION EXISTENCE *Problem*

The PME problem is unique to SDSs since state updates in CA are carried out in a synchronous fashion. As mentioned in Section 1, the motivation for the PME problem comes from the fact that the behavior of an SDS under two different permutations may be very different.

We show that the PME problem is **NP**-complete for (BOOL, ST)-SDSs as well as for SDSs whose underlying graphs are stars. However, we show that the problem can be solved efficiently for the following cases: (a) SDSs whose underlying graphs are of bounded degree and bounded treewidth with no restriction on the local transition functions, (b) (BOOL, NOR)-SDSs and (c) (BOOL, NAND)-SDSs. We also show that the GEN-PME problem is **NP**-complete for each of the following restricted classes of SDSs: (i) (BOOL, NOR)-SDSs ((BOOL, NAND)-SDSs) where the maximum node degree in the underlying graph is 3 and (ii) (BOOL, SYM)-SDSs whose underlying graphs are planar. We present polynomial time algorithms for the GEN-PME problem for (BOOL, OR)-SDSs and (BOOL, AND)-SDSs. These results show an interesting contrast between the complexities of GEN-PME and PME problems for (BOOL, NOR)-SDSs and (BOOL, NAND)-SDSs.

3.3 *Previous Work*

Computational aspects of CA have been studied by a number of researchers; see for example [Mo90, Wo86, Gu89, Gr87, Su95]. Much of this work addresses decidability of properties for infinite CA. Barrett, Mortveit and Reidys [BMR00, MR99, Re00] and Laubenbacher and Pareigis [LP00] investigate the mathematical properties of SDSs. The PRE problem was shown to be **NP**-complete for finite CA by Sutner [Su95] and Green [Gr87]. Sutner also showed that PRE problem for finite 1-dimensional CA with a fixed neighborhood radius can be solved in polynomial time. As mentioned earlier, Green [Gr87] studied generalized versions of the PRE problem for infinite CA. The problems were so formulated that his results are also applicable to finite 1-dimensional CA. References [Su95, Gr87] do not consider other restrictions on CA that lead to polynomial algorithms for the PRE problem. Our approach allows us to identify a number of restricted classes of SDSs for which the PRE problem can be solved efficiently.

4 A Selected Proof Sketch

Due to lack of space, only one proof is included in this version. The purpose of this proof is to illustrate the ideas used in the design of our efficient algorithm for the PRE problem for (BOOL, SYM)-SDSs, when the underlying graph of the SDS is of bounded treewidth. A complete version of the paper can be obtained from the authors.

Theorem 4.1 *Let $\mathcal{S}$ be a* (BOOL, SYM)*-SDS whose underlying graph has bounded treewidth. The* PRE *problem for $\mathcal{S}$ can be solved in polynomial time.*

Proof: Let k be the treewidth of the underlying graph $G(V,E)$. It is well known that a tree decomposition $(\{X_i \mid i \in I\}, T = (I,F))$ of G can be constructed in time that is a polynomial in the size of G. Moreover, this can be done so that T is a binary tree; that is, each node of T has at most two children [Bo88].

For a given node i of the tree decomposition, we refer to the SDS nodes in X_i as **explicit nodes** of i. If a given explicit node of i is also an explicit node of the parent of i, we refer to this node as an **inherited node** of i; and if it does not occur in the parent of i, we refer to it as an **originating node** of i. We refer to the set of all explicit nodes occurring in the subtree of T rooted at i that are not explicit nodes of i as **hidden nodes** of i. (Thus, the hidden nodes of i are the union of the originating and hidden nodes of the children of i.)

Given the underlying graph $G(V,E)$ of $\mathcal{S}$, the permutation π of $\mathcal{S}$and two disjoint subsets Y and Z of V, we define the set $N(Y,Z)$ as follows.

> $N(Y,Z)$ is the set of nodes $w \in G$ such that for each node $y \in Y$, $\{w,y\} \in E$, y precedes w in π, and there is no node $z \in Z$ such that $\{w,z\} \in E$ and z precedes w in π.

Intuitively, $N(Y,Z)$ is the set of nodes w of G, not in Y, such that in a transition of $\mathcal{S}$, the old value of w is an input parameter to the computation of the new value of every node in Y, but is not an input parameter to the computation of the new value of any of the nodes in Z.

Let $\mathcal{C}$ be the configuration specified in the given instance of the PRE problem for $\mathcal{S}$. Consider a given node i of the tree decomposition. Suppose α is a given assignment of state values to the explicit nodes of i and β is a given assignment of state values to the hidden nodes of i. We say that the combined assignment $\alpha \cup \beta$ is **viable** for i if for every hidden node w of i, the evaluation of the local transition function f_w gives the value $\mathcal{C}(w)$, using the value $\beta(w)$ for w, the value $\alpha \cup \beta(u)$ for every neighbor u of w that follows w in π, and the value $\mathcal{C}(u)$ for every neighbor u of w that precedes w in π. (Note that the definition of a tree decomposition ensures that every neighbor of a hidden node w is either an explicit node or a hidden node of i.)

We say that the combined assignment $\alpha \cup \beta$ is **strongly viable** for i if the above condition holds for every node w that is either a hidden node or an originating node of i. (The definition of a tree decomposition ensures that every neighbor of an originating node of i is either an explicit node or a hidden node of i.)

For a given node i of the tree decomposition, and a given assignment β to the states of the hidden nodes of i, define a function $h_\beta : 2^{X_i} \longrightarrow \mathbb{N}$ from the nonempty subsets of X_i to the set $\mathbb{N}$ of natural numbers as follows. For a subset Y of X_i,

$h_\beta(Y)$ is the number of hidden nodes w of i such that $w \in N(Y, X_i - Y)$ and $\beta(w) = 1$.

For a given node i of the tree decomposition, and a given assignment α to the states of the explicit nodes of i, define the set H_α to be the set of functions h from $2^{X_i} \longrightarrow \mathbb{N}$ such that there exists an assignment β to the states of the hidden nodes of i such that $\alpha \cup \beta$ is viable for i and h is h_β.

Since for any given β and Y, the maximum possible value of $h_\beta(Y)$ is the number of hidden nodes of i, and $|X_i| \leq k$ (where k is the treewidth), an upper bound on $|H_\alpha|$ is n^{2^k}.

Note: The function h_β could have been defined as a function whose domain is the nonempty subsets of X_i. However, including the empty set $\emptyset$ in the domain of h_β makes the final decision at the root node of the tree decomposition easier to describe.

We solve the PRE problem for $\mathcal{S}$ by using bottom-up dynamic programming on the decomposition tree. For each node i of T, we compute a table with an entry for each assignment α to the states of the explicit nodes of i. The value of the entry for each such assignment α is the set H_α. We refer to this table as J_i. Since the treewidth k is a constant, the size of the table for each node of the decomposition tree is a polynomial in n, the number of nodes of the underlying graph $G(V,E)$.

For a leaf node i of the decomposition tree, every entry H_α in the table consists of the single function h that maps every subset X_i into zero. Thus, the table for each leaf node of the decomposition tree can be computed in polynomial time.

For a nonleaf node i of the tree decomposition, the table for i can be computed in polynomial time from the tables of its children. To facilitate this computation, we utilize the following concepts and notation.

For a given node i of the tree decomposition, let $\widehat{X_i}$ denote the set of inherited nodes of i. (Thus, $\widehat{X_i} \subseteq X_i$.)

For a given node i of the tree decomposition, and a given assignment $\widehat{\beta}$ to the states of the originating nodes and the hidden nodes of i, define a function $\widehat{h}_{\widehat{\beta}} : 2^{\widehat{X_i}} \longrightarrow \mathbb{N}$ from the nonempty subsets of $\widehat{X_i}$ to the set $\mathbb{N}$ of natural numbers, as follows. For a subset Y of $\widehat{X_i}$,

$\widehat{h}_{\widehat{\beta}}(Y)$ is the number of nodes w such that w is an originating or a hidden node of i, $w \in N(Y, X_i - Y)$ and $\widehat{\beta}(w) = 1$.

For a given node i of the tree decomposition, and a given assignment ψ to the states of the inherited nodes of i, define $\widehat{H}_\psi$ to be the set of functions h from $2^{\widehat{X_i}} \longrightarrow \mathbb{N}$ such that there exists an assignment $\widehat{\beta}$ to the states of the originating and hidden nodes of i such that $\psi \cup \widehat{\beta}$ is strongly viable for i and h is $\widehat{h}_{\widehat{\beta}}$.

For each node i of the tree decomposition, we define $\widehat{J_i}$ as a table with an entry for each assignment ψ to the states of the inherited nodes of i. The value of the entry for each such assignment is the set $\widehat{H}_\psi$.

For each node i of the tree decomposition, given table J_i, the table $\widehat{J_i}$ can be constructed in polynomial time. For each assignment α to X_i and function h in H_α, consider the assignment α to be the union of an assignment ψ to the states of the inherited nodes of i, and an assignment γ to the states of the originating nodes of i. In polynomial time, we can determine whether the combination of ψ, γ and h corresponds to a combined assignment that is strongly viable for i. If so, we combine γ and h to obtain a function $\widehat{h}$ that we union into the $\widehat{J_i}$ entry for ψ. By considering each such pair (α, h) in J_i, we can construct $\widehat{J_i}$ in polynomial time.

Consider a nonleaf node i of the tree decomposition. Suppose that i has only one child, say the tree decomposition node i'. Given the table $\widehat{J_{i'}}$ for i', the table J_i can be constructed in polynomial time, as follows. Consider an assignment α to the states of the explicit nodes of i. Let ψ denote the projection of the assignment α on to $\widehat{X_i}$, the inherited nodes of i'. Then the entry for α in the table J_i for i is the set $\widehat{H}_\psi$ in the entry for ψ in the table for $\widehat{J_{i'}}$ for i', with the modification that every function $h : 2^{\widehat{X}_{i'}} \longrightarrow \mathbb{N}$ in the set $\widehat{H}_\psi$ is extended to be a function $g : 2^{\widehat{X}_i} \longrightarrow \mathbb{N}$ by setting $g(Y) = 0$ for every subset Y of X_i that contains at least one member of $X_i - \hat{X}_{i'}$.

Now suppose that nonleaf node i of the tree decomposition has two children, say nodes i' and i" of the tree decomposition. The tables $\widehat{J_{i'}}$ and $\widehat{J_{i''}}$ for tree nodes i' and i" are combined to produce table J_i for i as follows. Consider an assignment α to the states of the explicit nodes of i. Let ψ' and ψ" denote the projection of assignment α onto the inherited nodes of i' and i" respectively. Let every function h' in $\widehat{H}_{\psi'}$ for node i' be extended to a function $g' : 2^{\widehat{X}_i} \longrightarrow \mathbb{N}$, and every function h'' in $\widehat{H}_{\psi''}$ for node i" be extended to a function $g" : 2^{\widehat{X}_i} \longrightarrow \mathbb{N}$; both extensions are done as described above. Define the function $g' + g" : 2^{\widehat{X}_i} \longrightarrow \mathbb{N}$ as follows: $(g' + g")(Y) = g'(Y) + g"(Y)$. Thus, the entry for α in table J_i for i is the set of all functions $(g' + g")$ that can be constructed in the above manner from some h' in $\widehat{H}_{\psi'}$ from $\widehat{J_{i'}}$ and h" in $\widehat{H}_{\psi''}$ from $\widehat{J_{i''}}$.

Let r be the root node of the tree decomposition. The root node has no inherited nodes, so $\widehat{X}_r = \emptyset$. Consequently, table $\widehat{J_r}$ for the root node r consists of the single entry: $(\emptyset, \widehat{H}_\emptyset)$. Set $\widehat{H}_\emptyset$ is nonempty if and only if there exists an assignment to the states of all the nodes of SDS $\mathcal{S}$ that is strongly viable for r, that is, if and only if configuration $\mathcal{C}$ has a predecessor. ∎

Acknowledgments: We thank the referees for their suggestions. This research was also funded by the LDRD-DR project *Foundations of Simulation Science* and the LDRD-ER project *Advanced Methods in Discrete Simulations* at the Los Alamos National Laboratory.

References

[BE+01] C. Barrett, S. Eubank, M. Marathe, H. Mortveit and C. Reidys, invited paper to appear in *Proc. 1st International Conference on Grand Challenges in Simulations* held as a part of *Western Simulation Conference*, San Antonio Texas, 2002.

[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. *Proc. International Conf. 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. Analysis Problems for Sequential Dynamical Systems and Communicating State Machines. *Proc. International Symp. Mathematical Foundations of Computer Science* (MFCS'01), Czech Republic, August 2001, pp. 159–172.

[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.

[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.

[BZ83] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2), Apr. 1983, pp. 323–342,

[Bo88] Bodlaender, H. L. (1988), NC Algorithms for Graphs with Bounded Treewidth, *in* "Proceedings, Workshop on Graph Theoretic Concepts in Computer Science," pp. 1–10.

[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.

[CKS01a] N. Creignou, S. Khanna, and M. Sudan Complexity classifications of Boolean constraint satisfaction problems *SIAM Monographs on Discrete Mathematics and Applications* 7, 2001.

[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.

[Gr87] F. Green. NP-Complete Problems in Cellular Automata. *Complex Systems*, 1(3), 1987, pp. 453–474.

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

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

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

[LSS03] R. Laubenbacher, Shah, and Stigler, Simulation of Polynomial Systems. *Simulation and the Health Sciences*, Anderson, J.G. and Katzper M (eds), Soc. for Modeling and Simulation Intl., San Diego, CA, 2003.

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

[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.

[Pe97] W Peng. Deadlock Detection in Communicating Finite State Machines by Even Reachability Analysis. *Mobile Networks* (MONET), 2(3), 1997, pp. 251-257.

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

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

[SH96] R. E. Stearns and H. B. Hunt III. An Algebraic Model for Combinatorial Problems. *SIAM Journal on Computing*, Vol. 25, No. 2, April 1996, pp. 448–476.

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

[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.

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