Synchronous Boolean Finite Dynamical Systems on Directed Graphs over XOR Functions

Mitsunori Ogihara
Department of Computer Science, University of Miami, FL, USA
m.ogihara@miami.edu

Kei Uchizawa
Graduate School of Science and Engineering, Yamagata University, Japan
uchizawa@yz.yamagata-u.ac.jp

Abstract

In this paper, we investigate the complexity of a number of computational problems defined on a synchronous boolean finite dynamical system, where update functions are chosen from a template set of exclusive-or and its negation. We first show that the reachability and path-intersection problems are solvable in logarithmic space-uniform AC^1 if the objects execute permutations, while the reachability problem is known to be in P and the path-intersection problem to be in UP in general. We also explore the case where the reachability or intersection are tested on a subset of objects, and show that this hardens complexity of the problems: both problems become NP-complete, and even Π_2^p-complete if we further require universality of the intersection. We next consider the exact cycle length problem, that is, determining whether there exists an initial configuration that yields a cycle in the configuration space having exactly a given length, and show that this problem is NP-complete. Lastly, we consider the t-predecessor and t-Garden of Eden problem, and prove that these are solvable in polynomial time even if the value of t is also given in binary as part of instance, and the two problems are in logarithmic space-uniform NC^2 if the value of t is given in unary as part of instance.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Computational complexity, dynamical systems, Garden of Eden, predecessor, reachability

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.76

Funding This work was supported by JSPS KAKENHI Grant Numbers JP19K11817.

Acknowledgements We thank the reviewers for their constructive criticisms and many valuable suggestions.

1 Introduction

A discrete dynamical system is a network consisting of objects with states and state update functions assigned to the objects. The state update function of an object receives the states from the objects of the network, including itself, and determines the next state of the object. In a discrete dynamical system, the updates occur in discrete time steps. After receiving initial state values for the objects (called an initial state configuration), the system commences its computation by applying the state update functions to the nodes according to their update schedule, thereby generates an indefinitely long series of state configurations.

A directed graph naturally represents the direct dependencies among objects for state updates, where the edge from an object, u, to another, v, represents that the update function of v is dependent on the state of u. In the case where the dependencies are mutual, all edges are bidirectional, and so an undirected graph offers a simpler representation of the dependencies. These graphs may contain self-loops, since the state update functions may depend on their own values. A variety of discrete dynamical systems exists depending on

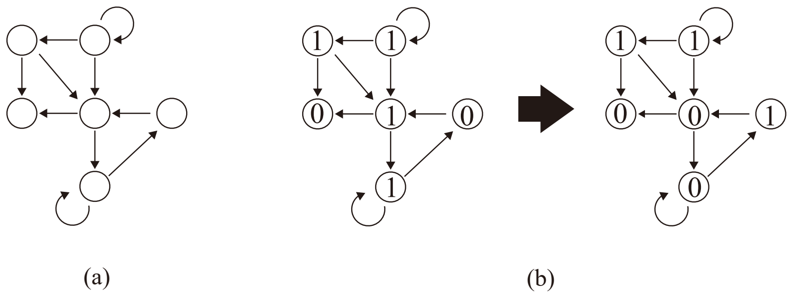

Figure 1 (a) A synchronous BFDS with six objects, where inputs of an update function of an object are the ones whose outgoing edges connected to the object; and (b) Suppose that every object has the exclusive-or XOR as its update function. Then, a single update for the BFDS with the given configuration depicted in the left panel yields the one in the right panel.

whether the update schedules are synchronous, whether or not the underlying graph is undirected, the state set (e.g., binary, finite, and infinite), and the types of update functions (e.g., boolean, symmetric, monotone). A synchronous boolean finite dynamical system (synchronous BFDS for short) [6] is one in which each object has just one boolean value as its state, the dependency graph is directed, and the updates are synchronous (see Fig. 1). Because the number of objects is finite, there are finitely many possible configurations in a BFDS (2^n for an n-object system). With synchronicity, a BFDS, in a finite number of steps (within 2^n steps in the case of an n-object system), either converges to a fixed point or enters a nontrivial simple cycle (see Fig 2).

An article by Barrett *et al.* [6] is the first to consider the synchronous BFDS model (they also introduced some other models, which are not the subjects of this paper). A wide stream of research has followed the paper studying the model from the computational complexity perspective. There, the questions have been to pinpoint the computational complexity of predicting the behavior of a model with or without a specific initial configuration. For example, Kosub [10] shows that there is a dichotomy between P and NP-complete regarding the question of whether the system possess a fixed point for any initial configuration. Kosub and Human extend this to a dichotomy between the function classes FP and #P-complete functions if the concern is for counting the number of initial configurations with a fixed point [11]. Rosenkrantz *et al.* provide a general framework for analyzing BFDS through subgraph embedding and show that various problems about BFDS are NP-complete, but if the representation graph is undirected with bounded tree-width and the update functions are r-symmetric, the computational complexity drops to P [13].

The stream of research also finds that some problems about a synchronous BFDS offer a rich theory of computational complexity by considering the types of functions permissible for the templates. Specifically, researchers have studied the conjunction AND, disjunction OR, exclusive-or XOR, and the negation of exclusive-or NXOR, and their combinations as the possible types, and asked the following three problems:

- *The reachability problem.* The problem asks whether, given a synchronous BFDS and two specific configurations, whether the system generates the second of the two from the first.
- *The path-intersection problem.* A variant of the reachability problem, it asks whether the paths of configurations starting from two initial configurations have a common element.
- *The cycle length problem.* The problem asks if the cycle that the system enters with a specific initial configuration has length greater than or equal to a specific value.

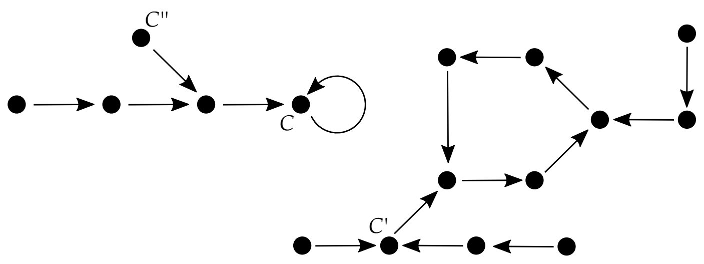

Figure 2 Example of a configuration space of a synchronous BFDS, where dots represent configurations, and arrows do transitions through updates. The configuration $\mathcal{C}$ is a fixed point. Two series of configurations from $\mathcal{C}'$ and $\mathcal{C}''$ do not have a common element, but ones for $\mathcal{C}$ and $\mathcal{C}''$ do. A cycle obtained by starting from an initial configuration $\mathcal{C}'$ has length five. The configuration $\mathcal{C}'$ has two predecessors, and one of them is a Garden of Eden.

Various complexity results exist about the problems. The reachability problem is PSPACE-complete in general, but it is so even if the underlying graph is an undirected bounded-degree graph and the update functions are symmetric [2], the underlying graph is undirected and update functions are threshold functions [1], or the underlying graph is directed and the template set is $\{\text{AND}, \text{OR}\}$ [12]. These PSPACE-complete results contrast well with that the problem is in P if the template set is one of $\{\text{AND}\}$, $\{\text{OR}\}$ and $\{\text{XOR}, \text{NXOR}\}$ [12]. As for the path-intersection and cycle length problems, they are PSPACE-complete too, if the underlying graph is directed and the template set is $\{\text{AND}, \text{OR}\}$. However, the path-intersection problem becomes solvable in UP and the cycle length problem becomes solvable in UP $\cap$ coUP [12] and in BQP [8], if the template set is one of $\{\text{AND}\}$, $\{\text{OR}\}$ and $\{\text{XOR}, \text{NXOR}\}$.

Testing reversibility has been a popular topic of computational complexity theoretic studies of synchronous BFDS [3, 4, 5, 13]. A predecessor of a given configuration is the one which the system produces from the given configuration. A *Garden of Eden* of BFDS is a configuration without predecessors. Typical problems in the reversibility of synchronous BFDS are the following two:

- *The t-predecessor problem, $t \geq 1$.* The problem asks if a configuration in a synchronous BFDS has a t-th predecessor, i.e., a configuration from which t successive applications of the updates produces the given configuration.
- *The t-Garden of Eden problem, $t \geq 0$.* The problem asks if a configuration in a synchronous BFDS has a t-th predecessor that is a Garden of Eden.

Article [4] shows that the 1-predecessor problem is in P if the underlying graph is an undirected graph with bounded tree-width and update functions are r-symmetric, but the problem is NP-complete if the underlying graph is an undirected star graph and the update functions are non-symmetric. With the use of incomplete (or degenerate) basis functions as the function types, the computational complexity of the 1-predecessor problem shows a wide variety: the problem is NP-complete if each update function is one of the 3-input OR and the 2-input AND, NL-complete if each update function is one of the 2-input OR and the 2-input AND, is in AC^0 if each update function is either OR only or AND only [9]. As for the case where $t = 2$, the t-Garden of Eden problem is Σ_2^p-complete if each function is either the 3-input OR or the 2-input AND, while the problem is NP-complete if the functions are either OR only or AND only, and is in AC^0 if the functions are either exclusively 2-input OR or exclusively 2-input AND [9].

In this paper, we investigate the computational complexity of the aforementioned problems where the update functions are all XOR, are all NXOR, or each function is chosen from $\{\text{XOR}, \text{NXOR}\}$. We can view a synchronous BFDS over XOR and NXOR as a linear transformation over $\mathbb{Z}_2$, and this connection makes the computational complexity problem interesting. We first consider the reachability and path intersection problems. We show that, while the reachability problem is in P and the path-intersection problem is in UP in general, both problems are solvable in logarithmic space-uniform AC^1 if the objects execute permutations (that is, every node in the underlying graph has in-degree 1 and out-degree 1). We then consider a variant of the problems where the update functions are still permutations but the reachability and path-intersection are on some subset of the objects. We show that, with this modification, the problems become NP-complete, and even Π_2^p-complete if we ask whether the reachability or the path-intersection holds for an arbitrary initialization of some subset of the objets. We next consider a variant of the cycle length problem, which we call the exact cycle length problem, where the question is whether or not there is an initial configuration that produces a simple cycle with a specific length, and we show that this variant is NP-complete. Lastly, we consider the t-predecessor and t-Garden of Eden problems. We prove that the problems are solvable in polynomial time even if the value of t is presented in binary, and the two problems are in logarithmic space-uniform NC^2 if the value either is constant or given in unary.

2 Preliminaries

In this section, we formally define terms and the problems on a synchronous BFDS.

2.1 Synchronous boolean finite dynamical systems

For an integer $n \geq 1$, a *synchronous boolean finite dynamical system* (synchronous BFDS, for short) $\mathcal{F}$ of n objects, $v_1, \ldots, v_n$, which respectively hold boolean variables, $x_1, \ldots, x_n$, and synchronously update their variables by evaluating boolean functions $f_1, \ldots, f_n$, respectively, where the inputs to the functions are the values of $x_1, \ldots, x_n$ immediately before the update. The values stored in the objects of a synchronous n-object BFDS constitutes a *state configuration* (or simply a *configuration*). A configuration can be represented as an n-dimensional boolean (or 0/1) vector. A synchronous BFDS thus can be naturally viewed as a function that maps the set of n-dimensional boolean vectors to itself. A synchronous BFDS commences its computation with a set of values stored in the objects. This initial set of values is called an *initial state configuration* (or an *initial configuration*).

Given a configuration $\mathcal{C}$ and a variable x (which is stored in some unique object of the system), $\mathcal{C}[x]$ is used to represent the value of x in the configuration x. The action of $\mathcal{F}$ on a state configuration $\mathcal{C}$ is given by $\mathcal{F}(\mathcal{C}) = (f_1(\mathcal{C}), f_2(\mathcal{C}), \ldots, f_n(\mathcal{C}))$. In other words, for all $i, 1 \leq i \leq n$, $\mathcal{F}(\mathcal{C})[x_i] = f_i(\mathcal{C})$. Given an initial state configuration $\mathcal{C}_{\mathrm{ini}}$, the synchronous BFDS generates a sequence of state configurations by iterative applications of $\mathcal{F}$: For all $t \geq 0$, the t-th element in the configuration sequence starting from $\mathcal{C}_{\mathrm{ini}}$ (we count from 0 with 0 representing the "initial") is given by $\mathcal{C} = \mathcal{F}^t(\mathcal{C}_{\mathrm{ini}})$.

The update functions of a BFDS can depend only on part of the configurations, and thus, can be expressed as a function of a smaller input dimension. In some literature, each update function, f_i, is expected to always depend on x_i, but in the present paper, this restriction is removed: f_i is allowed not to depend on x_i. The dependency among the objects of a BFDS can be represented as a directed graph whose nodes are the objects of the system, and each directed edge (u, v) represents that the function at v is dependent on the value at u.

Let $\mathcal{B}$ be a family of boolean functions. We say that a BFDS *has template set* $\mathcal{B}$ if the function of each object comes from the family $\mathcal{B}$. In this paper, we mainly consider the cases where $\mathcal{B}$ is either {XOR} or {NXOR}, but in some places, the additional bases of {AND} and {OR}.

It is known and not hard to see that computing $\mathcal{F}^t(\mathcal{C}_{\mathrm{ini}})$, given $\mathcal{F}$, $\mathcal{C}_{\mathrm{ini}}$, and t, is in P if $\mathcal{B}$ is one of $\{\mathrm{AND}\}$, $\{\mathrm{OR}\}$, $\{\mathrm{XOR}\}$, $\{\mathrm{NXOR}\}$, and $\{\mathrm{XOR}, \mathrm{NXOR}\}$ [12]:

▶ **Lemma 1.** *If $\mathcal{B}$ is one of $\{\mathrm{AND}\}$, $\{\mathrm{OR}\}$ and $\{\mathrm{XOR}, \mathrm{NXOR}\}$, we can compute $F^t(a)$ in time polynomial in $n + \log t$.*

2.2 The reachability and path-intersection problems of synchronous boolean finite dynamical systems

We are concerned with two general properties of synchronous BFDS: the reachability problem between two configurations in BFDS and the path-intersection problem generated from two distinct initial configurations.

▶ **Definition 2.** *Let $\mathcal{B}$ be a template set. The reachability problem for $\mathcal{B}$ asks, given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a configuration $\mathcal{C}_{\mathrm{ini}}$, and a configuration $\mathcal{C}_{\mathrm{fin}}$, whether there exists $t \geq 0$ such that $\mathcal{F}^t(\mathcal{C}_{\mathrm{ini}}) = \mathcal{C}_{\mathrm{fin}}$.*

▶ **Definition 3.** *Let $\mathcal{B}$ be a template set. The path-intersection problem for $\mathcal{B}$ asks, given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, and configurations $\mathcal{C}_1$ and $\mathcal{C}_2$, whether there exist $s \geq 0$ and $t \geq 0$ such that $\mathcal{F}^s(\mathcal{C}_1) = \mathcal{F}^t(\mathcal{C}_2)$.*

We consider a variant of the above problems, where the equality between two configurations is tested using a subset of objects. Let $\mathcal{F}$ be a synchronous BFDS, and R a subset of objects in $\mathcal{F}$. We say that $\mathcal{D}$ is a sub-configuration of $\mathcal{C}$ on R if $\mathcal{D}$ is obtained from $\mathcal{C}$ retaining only those objects in R, and we say that $\mathcal{C}$ is a completion of $\mathcal{D}$. We use the notation $\mathcal{C}|_R$ to mean that this is a vector constructed by selecting only those variables corresponding to R.

▶ **Definition 4.** *Let $\mathcal{B}$ be a template set. The projection reachability problem for $\mathcal{B}$ asks, given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a subset R of the objects of $\mathcal{F}$, a configuration $\mathcal{C}_{\mathrm{ini}}$, and a configuration $\mathcal{C}_{\mathrm{fin}}$, whether there exists $t \geq 0$ such that $\mathcal{F}^t(\mathcal{C}_{\mathrm{ini}})|_R = \mathcal{C}_{\mathrm{fin}}|_R$.*

▶ **Definition 5.** *Let $\mathcal{B}$ be a template set. The projection path-intersection problem for $\mathcal{B}$ asks, given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a subset R of the objects of $\mathcal{F}$, and configurations $\mathcal{C}_1$ and $\mathcal{C}_2$, whether there exist $s \geq 0$ and $t \geq 0$ such that $\mathcal{F}^s(\mathcal{C}_1)|_R = \mathcal{F}^t(\mathcal{C}_2)|_R$.*

In addition to the above, we consider the universal projection reachability problem and the universal projection path-intersection problem by considering the question of whether the reachability or the path-intersection property holds for all possible initial value assignments to a specified set of objects.

▶ **Definition 6.** *The universal projection reachability problem is the problem of deciding, given a BFDS $\mathcal{F}$, an initial sub-configuration $\mathcal{C}$ of $\mathcal{F}$, a final configuration $\mathcal{D}$, and a projection P, whether for all completions $\mathcal{C}'$ of $\mathcal{C}$, there exists some t such that $\mathcal{F}^t(\mathcal{C}')|_P = \mathcal{D}|_P$.*

▶ **Definition 7.** *The universal projection path-intersection problem is the problem of deciding, given a BFDS $\mathcal{F}$, two initial sub-configurations $\mathcal{C}_1$ and $\mathcal{C}_2$ of $\mathcal{F}$, projection P, whether for all completions $\mathcal{C}_1'$ and $\mathcal{C}_2'$ of $\mathcal{C}_1$ and $\mathcal{C}_2$, there exist s and t such that $\mathcal{F}^s(\mathcal{C}_1')|_P = \mathcal{F}^t(\mathcal{C}_2')|_P$.*

2.3 The exact cycle length problem

Given a synchronous BFDS $\mathcal{F}$ and a configuration $\mathcal{C}$ of $\mathcal{F}$, we say that there exists a cycle of length l starting from $\mathcal{C}$ if $\mathcal{F}^k(\mathcal{C}) \neq \mathcal{C}$ for each $1 \leq k < l$ and $\mathcal{F}^l(\mathcal{C}) = \mathcal{C}$. We denote by $L_{\mathcal{F}}(\mathcal{C})$ the length l of the cycle. We consider a variant of the cycle length problem as follows;

▶ **Definition 8.** *Let $\mathcal{B}$ be a template set. The exact cycle length problem for $\mathcal{B}$ asks, given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a sub-configuration $\mathcal{D}_{\text{ini}}$, and an integer l, whether there exists a completion $\mathcal{E}_{\text{ini}}$ of $\mathcal{D}_{\text{ini}}$ such that $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}}) = l$.*

2.4 The predecessor problems

Given a synchronous BFDS $\mathcal{F}$ and a final configuration $\mathcal{C}_{\text{fin}}$ of $\mathcal{F}$, we say that a configuration $\mathcal{C}$ is a *t-th predecessor of* $\mathcal{C}_{\text{fin}}$, $t \geq 1$, if $\mathcal{F}^t(\mathcal{C}) = \mathcal{C}_{\text{fin}}$. Note that if $\mathcal{F}(\mathcal{C}_{\text{fin}}) = \mathcal{C}_{\text{fin}}$, we have $\mathcal{F}^t(\mathcal{C}_{\text{fin}}) = \mathcal{C}_{\text{fin}}$ for any $t \geq 1$. We will omit the word *first* in the case where $t = 1$. By convention, we define the 0-th predecessor of $\mathcal{C}_{\text{fin}}$ to be $\mathcal{C}_{\text{fin}}$ itself. We say that a configuration $\mathcal{C}$ is a Garden of Eden if $\mathcal{C}$ has no predecessor. We consider the predecessor and Garden-of-Eden problem defined in the paper [9] as follows:

▶ **Definition 9.** *Let $\mathcal{B}$ be a template set. The t-PRED Problem for $\mathcal{B}$ asks, given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a configuration $\mathcal{C}_{\text{fin}}$, and an integer t, whether $\mathcal{C}_{\text{fin}}$ has a t-th predecessor.*

▶ **Definition 10.** *Let $\mathcal{B}$ be a template set. The t-GOE Problem for $\mathcal{B}$ asks, given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a configuration $\mathcal{C}_{\text{fin}}$, and an integer t, whether $\mathcal{C}_{\text{fin}}$ has a t-th Garden of Eden, i.e., a t-th predecessor that is a Garden of Eden.*

In the paper [9], the computational complexity of t-PRED and t-GOE are extensively studied over various degenerative (i.e., insufficient to express all boolean functions) bases, where t is a fixed constant or appears in unary as part of input. Note that in the above definitions, t is given in binary and so the problems are more flexible than the ones defined in [9].

3 Reachability and Path-intersection Problems

In this section, we study the complexity of reachability and path-intersection problems of synchronous BFDS. We start with a special case of synchronous BFDS in which the objects execute permutations. In such systems, the nodes in the directed graphs representing the system updates have in-degree 1 and out-degree 1.

3.1 Reachability and path-intersection in permutation systems

▶ **Theorem 11.** *The reachability problem of permutational BFDS can be solved in polynomial time.*

Proof. Let $\mathcal{F}$ be a permutational BFDS with n objects. Let $\mathcal{C}_{\text{ini}}$ be an initial configuration, and $\mathcal{C}_{\text{fin}}$ a final configuration. The reachability problem asks whether there exists some integer $t \geq 0$ such that $\mathcal{F}^t(\mathcal{C}_{\text{ini}}) = \mathcal{C}_{\text{fin}}$. Let $M_{\mathcal{F}}$ be the matrix that represents this permutation. The permutation that $\mathcal{F}$ executes is decomposed as the disjoint union of independent cycles. Let $r, 1 \leq r \leq n$, be the number of the disjoint cycles of $\mathcal{F}$. Let $O_1, O_2, \ldots, O_r$ be the r cycles. For each $k, 1 \leq k \leq r$, let o_k be $|O_k|$, the length of O_k. Obviously, for all $k, 1 \leq k \leq r$, $1 \leq o_k \leq n$. Let G be the least common multiple of $o_1, \ldots, o_r$. There exists some t such that $\mathcal{F}^t(\mathcal{C}_{\text{ini}}) = \mathcal{C}_{\text{fin}}$ if and only if

(*) there exists some $t, 0 \leq t \leq G - 1$, such that for each $k, 1 \leq k \leq r$, $\mathcal{F}^{t \bmod o_k}(\mathcal{C}_{\text{ini}})|_{O_k} = \mathcal{C}_{\text{fin}}|_{O_k}$.

Below, we will show a method for testing (*).

For each k, $1 \leq k \leq r$, let $Q_k = \{t \bmod o_k \mid \mathcal{F}^t(\mathcal{C}_{\text{ini}})|_{O_k} = \mathcal{C}_{\text{fin}}|_{O_k}\}$. The set Q_k is a subset of $[o_k - 1]$ for all $k, 1 \leq k \leq r$. The question (*) is equivalent to whether there exists some value for $t, 0 \le t \le G-1$, such that for all $k, 1 \le k \le r$, $t \bmod o_k$ is a member of Q_k. In other words, whether there is a solution to the system of modular equations:

$$(\forall k, 1 \le k \le r)(\exists a \in Q_k)\ [\ t \equiv a \pmod{o_k}\ ] \tag{1}$$

The reason that we use the existential quantifier in Eq. (1) is that the cardinality of Q_k is not necessarily 0 or 1. Obviously, if Q_k has cardinality 0, then Eq. (1) has no solution, and so $\mathcal{C}_{\mathrm{fin}}$ is unreachable from $\mathcal{C}_{\mathrm{ini}}$. If Q_k has cardinality greater than 1, we can reduce Q_k to a smaller set so there is only one element in it as follows: Suppose Q_k has more than one element. Let a and b, $a < b$, be two members of Q_k. Because both $\mathcal{F}^a(\mathcal{C}_{\mathrm{ini}})$ and $\mathcal{F}^b(\mathcal{C}_{\mathrm{ini}})$ are identical to $\mathcal{C}_{\mathrm{fin}}$ on O_k, $\mathcal{F}^{b-a}(\mathcal{C}_{\mathrm{fin}})$ is equal to $\mathcal{C}_{\mathrm{fin}}$ on O_k. Thus, for all $a, b, c \in Q_k$, $a + |b - c| \bmod o_k \in Q_k$. This means that there is some divisor of o_k, d, and some $a, 0 \le a \le d-1$, such that Q_k is the set of all integers $i, 0 \le i \le o_k - 1$, such that $i \equiv a \pmod{d}$. For each $k, 1 \le k \le r$, such that Q_k has more than one element, let d_k be the value of d and a_k thus defined; for each $k, 1 \le k \le r$, such that Q_k has only one element, let $d_k = o_k$ and a_k be the unique element in Q_k.

Now Eq. (1) is equivalent to whether there is a solution to the system of modular equations:

$$(\forall k, 1 \le k \le r)[\ t \equiv a_k \pmod{d_k}\ ] \tag{2}$$

Whether this system has a solution or not can be checked by testing whether for each pair (k, k'), the equation for k is consistent with the equation for k'. The consistency between k and k' holds if and only if for the greatest common divisor s of d_k and $d_{k'}$, whether $a_k \equiv a_{k'} \pmod{s}$.

Based upon the observation, we develop the following algorithm for testing the reachability.

1. Obtain the cycles, $O_1, \ldots, O_r$, of the permutation represented by $\mathcal{F}$.
2. Compute $Q_1, \ldots, Q_r$.
3. Check if all of $Q_1, \ldots, Q_r$ are nonempty. If the check fails, reject the input immediately.
4. For each k such that Q_k has only one element, compute d_k and a_k for the remaining values for k as the unique element in Q_k and o_k.
5. For each k such that Q_k has more than one element, compute a_k as the smallest number in Q_k and d_k as the difference between the first and the second smallest numbers in Q_k.
6. For each pair (k, k'), $1 \le k < k' \le r$, compute $s = \gcd(d_k, d_{k'})$, and test if $a_k \equiv a_{k'} \pmod{s}$. If the test passes for all (k, k'), accept; otherwise, reject.

It is easy to see that all these steps can be carried out in time polynomial in n. This proves the theorem. ◀

It is possible to carry out the algorithm stated in the proof in logarithmic space and with logarithmic-space uniform AC^1. To simplify the computation, instead of counting the number of distinct orbits, r, we will let each index $j, 1 \le j \le n$, represent the orbit starting from j. In this manner, an orbit having multiple elements will be represented as many times as there are elements in the orbit. However, for two elements appearing in the same orbit, the sets Q are same, and so the overall solvability of the system of congruences is unchanged.

So, assume that with the possible redundancy, the number of the orbits, r, is equal to n, and for all $k, 1 \le k \le n$, O_k is the orbit starting from k. Let $\mathcal{Q}$ denote the permutation represented by $\mathcal{F}$. For an arbitrary index $j, 1 \le j \le n$, consider an index sequence $j, \mathcal{Q}^1(j), \mathcal{Q}^2(j), \ldots$. Starting from j, the elements of this sequence can be generated one after another. In the case of logarithmic-space computation, the generation can be carried by scanning the matrix $\mathcal{F}$. The smallest index m at which $\mathcal{Q}^m(j) = j$ is the length of the orbit, o_j, and the values that appear as the elements are the members of the orbit. Thus, for each j, the number of elements in the orbit containing j, for each j and for each h, whether h is a member of the orbit containing j, and for each j, for each h, and for each s, whether h is the s-th element of the orbit starting from j can be answered in logarithmic space. Since the iterative permutation and the elements of the orbit can be generated iteratively, in logarithmic space it is possible to answer whether $a \in Q_k$ for each a and for each k. Thus, it is possible to compute, in logarithmic space, a_k and d_k. For each pair, (k, k'), of indexes, we then compute a_k and d_k as well as $a_{k'}$ and $d_{k'}$ in binary, and then compute the greatest common divisor, s, of d_k and $d_{k'}$, and check whether $a_k \equiv a_{k'} \pmod{s}$. Since the binary numbers have $O(\log n)$ bits, the test can be carried out in logarithmic space. Hence, the reachability problem is solvable in logarithmic space.

To show that the circuit complexity of the reachability problem is logarithmic space-uniform AC^1, first note that the matrix multiplication of the permutation matrix can be computed in AC^0, and so their powers up to the n-th power can be computed in AC^1 via repeated squaring. Once these permutation matrices have been computed, Step 1 of the algorithm (obtaining the orbits) can be executed by multiplying these matrices by 0/1-vectors with only one 1 appearing in them. The circuit may use an n-bit sequence to represent the membership for each orbit (therefore, n^2 bits). Since these multiplications can be carried out in parallel, this step requires AC^0. For Step 2, by multiplying the matrices by $\mathcal{C}_{\mathrm{ini}}$ and comparing the results with $\mathcal{C}_{\mathrm{fin}}$ on the orbits, Q_k can be obtained in parallel. Again, the representation can be an n-bit sequence for each orbit (a total of n^2 bits) and the computation requires AC^0. After this, if for some k, Q_k is empty, then the circuit produces the output of 0; otherwise, a_k and d_k can be computed by first checking whether Q_k has only one element and then if more than one element exists, as the first element and the difference between the second and the first. Again, this can be carried out in AC^0. Compatibility testing is the final step of the algorithm. We imagine that the circuit is equipped with a module for all possible compatibility tests. There are only $O(n^4)$ possible combinations of two residue-modulus pairs. Given two integers d and d' represented in binary, their greatest common divisor can be computed in $O(\log n)$ space using Euclid's algorithm, and so precomputing the tests for n requires $O(\log n)$ space. It is not hard to see that the other components of the circuit can be computed in $O(\log n)$ space as well. Hence, the problem is in logarithmic space-uniform AC^1.

We have thus proven:

▶ **Theorem 12.** *The reachability problem of permutational BFDS belongs to logarithmic space and the logarithmic space-uniform AC^1.*

Since permutations are reversible, the question of whether the paths starting from two initial configurations intersect on a permutational BFDS is equivalent to the question of whether the path starting from one of the two initial configurations reaches the other initial configuration. Thus, we have the following corollary:

▶ **Corollary 13.** *The path-intersection problem of permutational BFDS belongs to* P*, the logarithmic space, and the logarithmic space-uniform AC^1.*

3.2 The reachability and path-intersection problems with projection and universality

In this section, we consider the variants of the reachability and path-intersection problems, and show their hardness. The proofs are omitted.

▶ **Theorem 14.** *The projection reachability problem of synchronous BFDS with* XOR *as template is NP-complete.*

We can show that the above proposition holds with OR in place of XOR. Then, by exchanging the role between 0 and 1 in the proof for OR, we obtain that the proof holds for AND.

Also, returning to the XOR proof, consider adding two new objects, a and b, whose values are initially 1, whose update functions are NXOR of a and b, and changing all the other update functions of the form $\mathrm{XOR}(u_1, \ldots, u_k)$ to $\mathrm{NXOR}(u_1, \ldots, u_k, a)$, we obtain that the result holds for NXOR.

▶ **Corollary 15.** *The projection reachability problem of BFDS is NP-complete with any of* OR*,* AND*, and* NXOR *as the basis.*

As before, the above results hold for path-intersection as well.

▶ **Corollary 16.** *The projection path-intersection problem of synchronous BFDS is NP-complete with one of* XOR*,* OR*,* AND*, and* NXOR *as the basis.*

We can show that the universal projection reachability problem is Π_2^p-complete.

▶ **Theorem 17.** *The universal projection reachability problem is Π_2^p-complete with the basis of* XOR*.*

As before, the reachability problem can be viewed as a special case of path intersection problem.

▶ **Corollary 18.** *For synchronous BFDS with template chosen from* $\{\mathrm{XOR}\}$, $\{\mathrm{NXOR}\}$, $\{\mathrm{OR}\}$, *and* $\{\mathrm{AND}\}$*, both the universal projection reachability problem and the universal projection path-intersection problem are Π_2^p-complete.*

4 Cycle Length Problems

In this section, we show that, while the cycle length problem is known to be in $\mathrm{UP} \cap \mathrm{coUP}$ and in BQP, the exact cycle length problem is NP-complete.

▶ **Theorem 19.** *The exact cycle length problem is* NP*-complete if* $\mathcal{B} = \{\mathrm{XOR}, \mathrm{NXOR}\}$*.*

Proof. We reduce 3-occurrence 3SAT to the problem. Let ϕ be a given 3CNF formula, where the number of appearances of each variable is limited to three. Suppose ϕ consists of n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots C_m$, each of which has at most three literals (either $l_{j,1}$ and $l_{j,2}$ or $l_{j,1}, l_{j,2}$ and $l_{j,3}$). We say that a literal $l_{j,k}$ is the first and second positive (resp., negative) literal of x_i.

We first construct the desired BFDS $\mathcal{F}$, $\mathcal{D}_{\mathrm{ini}}$ and integer l such that $\mathcal{F}$ has a l-cycle starting from $\mathcal{E}_{\mathrm{ini}}$ if and only if ϕ is satisfiable.

[Construction of $\mathcal{F}$, P, $\mathcal{D}_{\mathrm{ini}}$ and l]

For each $j \in [m]$, we denote by p_j the jth odd prime. By the Prime Number Theorem (see, e.g., [7]), $p_n = O(n \log n) = o(n^2)$, and so $|p_n| = O(\log n)$. Thus, using the trial division, $p_1, \ldots, p_n$ can be found in polynomial time. For $j \in [m]$ and $k \in [3]$, we construct a BFDS $M_{j,k}$ as follows: $M_{j,k}$ has p_j objects $x_{j,k,0}, x_{j,k,1} \ldots, x_{j,k,p_j-1}$ computing XORs, and compose a cycle:

$$\mathcal{F}[x_{j,k,h}] = x_{j,k,h-1 \bmod p_j}. \tag{3}$$

We construct $\mathcal{F}$ by combining these $M_{j,k}$ as follows. For each $i \in [n]$, if there are either two positive literals or two negative literals of x_i, denoted by $l_{j,k}$ and $l_{j',k'}$, we add an object y_i computing XOR such that

$$\mathcal{F}[y_i] = y_i \oplus \bigoplus_{h=0}^{p_j-1} x_{j,k,h} \oplus \bigoplus_{h=0}^{p_{j'}-1} x_{j',k',h}. \tag{4}$$

In addition, for a pair of the first positive literal $l_{j,k}$ of x_i and the first negative literal $l_{j',k'}$ of x_i, we add an object z_i computing NXOR such that

$$F[z_i] = \overline{\left(z_i \oplus \bigoplus_{h=0}^{p_j-1} x_{j,k,h} \oplus \bigoplus_{h=0}^{p_{j'}-1} x_{j',k',h} \right)}. \tag{5}$$

This completes the construction of $\mathcal{F}$. We define P as a set of all the objects in $\mathcal{F}$ other than $x_{j,k,0}$ for every $1 \le j \le m$ and $1 \le k \le 3$. We set all the values in P to zeros. Thus, for any extension $\mathcal{E}_{\text{ini}}$ of $\mathcal{D}_{\text{ini}}$, it holds that for any $j \in [m]$, $k \in [3]$ and $t \ge 1$,

$$\sum_{h=0}^{p_j-1} \mathcal{F}^t(\mathcal{E}_{\text{ini}})[x_{j,k,h}] = \mathcal{E}_{\text{ini}}[x_{j,k,0}] \in \{0,1\}. \tag{6}$$

We finally define $l = \prod_{j=1}^{m} p_j$.

Since $M_{j,k}$'s are disjoint cycles and do not interact with each other, we have the following claim.

▷ Claim 20. Let $S(\mathcal{E}_{\text{ini}}) = \{j \mid \exists k, \mathcal{E}_{\text{ini}}[x_{j,k,0}] = 1\}$. If there exists i such that either $\mathcal{F}(\mathcal{E}_{\text{ini}})[y_i] = 1$ or $\mathcal{F}(\mathcal{E}_{\text{ini}})[z_i] = 1$ holds, then $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}})$ is even; and, otherwise, $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}}) = \prod_{j \in S(\mathcal{E}_{\text{ini}})} p_j$.

Proof. Suppose there exists $i \in [n]$ such that $\mathcal{F}(\mathcal{E}_{\text{ini}})[y_i] = 1$ holds. Then Eq. (4) implies that $\mathcal{E}_{\text{ini}}[x_{j,k,0}] \neq \mathcal{E}_{\text{ini}}[x_{j',k',0}]$, where j, j', k, k' are the ones specified in Eq. (4). Thus, Eq. (6) implies that $\mathcal{F}^t(\mathcal{E}_{\text{ini}})[y_i] = 1$ for odd t while $\mathcal{F}^t(\mathcal{E}_{\text{ini}})[y_i] = 0$ for even t. Therefore, $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}})$ is even. We can similarly verify the claim for the case where $\mathcal{F}(\mathcal{E}_{\text{ini}})[z_i] = 1$.

Suppose $\mathcal{F}(\mathcal{E}_{\text{ini}})[y_i] = \mathcal{F}(\mathcal{E}_{\text{ini}})[z_i] = 0$ for every $i \in [n]$. Then Eqs. (4)-(6) imply that $\mathcal{F}^t(\mathcal{E}_{\text{ini}})[y_i] = \mathcal{F}^t(\mathcal{E}_{\text{ini}})[z_i] = 0$ for any $i \in [n]$ and any positive integer t. Moreover, for every $j \in [m] \backslash S(\mathcal{E}_{\text{ini}})$, we have $\mathcal{F}^t(\mathcal{E}_{\text{ini}})[x_{j,k,0}] = \mathcal{F}(\mathcal{E}_{\text{ini}})[x_{j,k,1}] = \cdots = \mathcal{F}(\mathcal{E}_{\text{ini}})[x_{j,k,p_j-1}] = 0$ for every $k \in [3]$. Thus, $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}})$ is determined by $M_{j,k}$s for $j \in S(\mathcal{E}_{\text{ini}})$. Since $M_{j,k}$ are disjoint cycles of different primes, we have $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}}) = \prod_{j \in S(\mathcal{E}_{\text{ini}})} p_j$. ◀

[⇐] We here prove that if ϕ is satisfiable, then there exists an extension $\mathcal{E}_{\text{ini}}$ of $\mathcal{D}_{\text{ini}}$ such that $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}}) = \prod_{j \in [m]} p_j$.

Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \{0,1\}^n$ be a satisfying assignment for ϕ. We define $\mathcal{E}_{\text{ini}}$ as follows: For every $j \in [m]$, and $k \in [3]$, $\mathcal{E}_{\text{ini}}[x_{j,k,0}] = 1$ if either "$l_{j,k}$ is a positive literal of x_i and $\alpha_i = 1$" or "$l_{j,k}$ is a negative literal of x_i and $\alpha_i = 0$," and otherwise, $\mathcal{E}_{\text{ini}}[x_{j,k,0}] = 0$. Eqs. (4)-(6) imply that $\mathcal{F}(\mathcal{E}_{\text{ini}})[y_i] = \mathcal{F}(\mathcal{E}_{\text{ini}})[z_i] = 0$ for every $i \in [n]$, and hence $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}}) = \prod_{j \in [m]} p_j$, as desired.

[⇒] We prove that if there exists an extension $\mathcal{E}_{\text{ini}}$ of $\mathcal{D}_{\text{ini}}$ such that $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}}) = l$ then ϕ is satisfiable.

Since $L_{\mathcal{F}}(\mathcal{E}_{\text{ini}}) = l$, the claim implies that $S(\mathcal{E}_{\text{ini}}) = \{p_1, p_2, \ldots, p_m\}$, and hence, for every j, $1 \leq j \leq m$, there exists k_j, $1 \leq k_j \leq 3$, such that $\mathcal{E}_{\text{ini}}[x_{j,k_j,0}] = 1$. We thus define $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \{0,1\}^n$ as follows: For each i, $1 \leq i \leq n$,

$$\alpha_i = \begin{cases} 1 & \text{if } l_{j,k} \text{ is a positive literal and } \mathcal{E}_{\text{ini}}[x_{j,k_j,0}] = 1; \\ 0 & \text{if } l_{j,k} \text{ is a negative literal and } \mathcal{E}_{\text{ini}}[x_{j,k_j,0}] = 1. \end{cases}$$

Since l is not even, the claim implies that $\mathcal{F}(\mathcal{E}_{\text{ini}})[y_i] = \mathcal{F}(\mathcal{E}_{\text{ini}})[z_i] = 0$ for every $i \in [n]$, and hence we can set α_i without any confliction. Since $\alpha_i = 1$ if l_{j,k_j} is a positive literal, and $\alpha_i = 0$ if l_{j,k_j} is a negative literal, α clearly satisfies the literal l_{j,k_j} for every $j \in [m]$. ◀

5 Predecessor Problems

In this section, we show that the t-predecessor problem and the t-Garden of Eden problem can be answered in polynomial time.

▶ **Theorem 21.** *The t-predecessor problem and the t-Garden of Eden problem are solvable in polynomial time if the basis $\mathcal{B}$ is $\{\text{XOR}, \text{NXOR}\}$, where the value of t can be given in binary as part of the instance.*

Proof. We first describe our algorithms for the two problems where the basis contains only XOR. Suppose we are given a synchronous BFDS $\mathcal{F}$ of n objects $x_1, x_2, \ldots x_n$, a configuration $\mathcal{C}_{\text{fin}}$, and an integer t.

We can think of $\mathcal{F}$ as an $n \times n$ 0/1 incidence matrix $M_{\mathcal{F}}$ whose i-th row corresponds to the update function f_i in such a manner that the j-th entry of the row is 1 if and only if f_i takes x_j as one of its inputs. By viewing a configuration as an 0/1 column vector, the application of the system for one step can be viewed as multiplying $M_{\mathcal{F}}$ by the vector from right, with the arithmetic carried out in $\mathbb{Z}_2$. The t-predecessor problem can be restated as the question of whether the linear system of equations:

$$M_{\mathcal{F}}^t \mathcal{D} = \mathcal{C}_{\text{fin}} \tag{7}$$

has a solution, where $\mathcal{D}$ is a column vector of length n. This needs a bit of explanation. Given a solution $\mathcal{D}$ to the system of linear equations, we have that $[\mathcal{D}, M_{\mathcal{F}}(\mathcal{D}), M_{\mathcal{F}}^2(\mathcal{D}), \ldots, M_{\mathcal{F}}^t(\mathcal{D})]$ is a sequence of configurations that takes $\mathcal{D}$ to $\mathcal{C}_{\text{fin}}$ in t steps, and so $\mathcal{D}$ is a t-th predecessor of $\mathcal{C}_{\text{fin}}$. Conversely, given a t-th predecessor, $\mathcal{E}$, of $\mathcal{C}_{\text{fin}}$, there is a sequence $[\mathcal{C}_{\text{fin}}, \mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_{t-1}, \mathcal{E}]$ that takes $\mathcal{C}_{\text{fin}}$ back to $\mathcal{E}$, where applying $M_{\mathcal{F}}$ to an element excluding $\mathcal{C}_{\text{fin}}$ in the sequence produces the element immediately to the left in the sequence. This means that $\mathcal{E}$, after applying $M_{\mathcal{F}}$ t times consecutively, becomes $\mathcal{C}_{\text{fin}}$, and so $\mathcal{E}$ is a solution to the system.

To answer the question of whether the system has a solution, we first compute $M_{\mathcal{F}}^t$ by the iterative squaring method. The calculation can be done in polynomial time (in the length of the input) even if t is given in binary.

We then check the solvability of the system by converting the system so that the matrix is in a row echelon form; if a row is not all 0, at the position of its leading 0, all the other rows have 0. This conversion can be carried out using the Gaussian Elimination method. By concurrently applying the operation on the vector, we obtain the equivalent system:

$$H\mathcal{D} = \mathcal{C}' \tag{8}$$

where H is in a row echelon form. Checking whether this system has a solution is simple: the system has a solution if and only if the column vector $\mathcal{C}'$ has a 0 for each all-0 row of the matrix H. Since Gaussian Elimination can be carried out in polynomial time, this implies that the t-predecessor problem is solvable in polynomial time.

The t-Garden of Eden problem asks whether there is a solution of Eq. (7), $\mathcal{S}$, such that $M_{\mathcal{F}}x = \mathcal{S}$ does not have a solution. Since Eq. (8) is an equivalent system to Eq. (7), we can instead ask whether there is a solution, $\mathcal{S}$, of the equivalent system for which $M_{\mathcal{F}}x = \mathcal{S}$ has no solution. Clearly, t-th Garden of Eden exists only if t-th predecessor exists. Thus, the instance in which Eq. (7) has no solution can be outright asserted not have a t-th Garden of Eden. If the equation has a solution, each solution can be described as a vector whose elements are either constants or linear functions as follows:

Group 1 If a row r of H has a leading 1 at position i and the remainder of the row is all 0, $x_i = b_i$, where b_i is the i-th element of the vector $\mathcal{C}'$.

Group 2 If a row r of H has a leading 1 at position i and the remainder of the row is not all 0, $x_i = x_{j_1} + \cdots + x_{j_k} + b_i$, where $j_1, \ldots, j_k$ are the positions of 1's on row r other than i and b_i is the i-th element of the vector $\mathcal{C}$.

Group 3 If x_i does not receive an assignment through either of the above, i.e., no rows of the echelon form has leading 0 at position i, then x_i remains as an independent variable; i.e., $x_i = x_i$.

Let $\mathcal{S}$ be the vector thus determined. The vector $\mathcal{S}$ can be expressed as the product of an $n \times (n+1)$ 0/1 matrix, Q and an $n+1$ dimensional column vector $\mathcal{X} = (x_1, \ldots, x_n, 1)^T$. More specifically, in Q, the first n columns represent the appearance of $x_1, \ldots, x_n$ in the solution in the n rows and the last column represents the value of the constants b_i's. In other words, if $x_i = x_{j_1} + \cdots + x_{j_k} + b_i$ is the solution for x_i, then row i of the $n \times (n+1)$ matrix Q has 1 at the positions $j_1, \ldots, j_k$ and at the last position if b_i is 1. We apply Gaussian Elimination on $M_{\mathcal{F}}$ for converting it to a row echelon form and apply the operations concurrently to the matrix Q, to obtain a new equation:

$$M'\mathcal{D} = Q'\mathcal{X} \tag{9}$$

As before, a solution to Eq. (8) can be configured so that the solution has no predecessor if and only if there is a row index i such that the i-th row in M' is all 0 while the i-th row in Q' contains a 1. This condition can be tested once the matrix Q has been obtained and the echelon form has been computed. Since Gaussian Elimination can be carried out in polynomial time, we have that the t-Garden of Eden problem can be solved in polynomial time.

For the case where $\mathcal{B} = \{\text{XOR}, \text{NXOR}\}$, we consider $\mathcal{F}$ as $n \times (n+1)$ matrix $M'_{\mathcal{F}}$ consisting of $M_{\mathcal{F}}$ together with a column vector of length n whose i-th element is one if f_i is NXOR; and zero, otherwise. Let $\mathcal{C}'_{\text{fin}}$ be a column vector consisting of $\mathcal{C}_{\text{fin}}$ followed by a single element one. We then apply the above procedure based on the linear system $(M'_{\mathcal{F}})^t\mathcal{D} = \mathcal{C}_{\text{fin}}$. This proves the theorem. ◀

The problem is solvable even in logarithmic space-uniform NC^2.

▶ **Theorem 22.** *With the basis* $\{\text{XOR}, \text{NXOR}\}$*, the t-predecessor problem and the t-Garden of Eden problem are solvable in logarithmic space-uniform NC^2 if t is a constant, t is given in unary, or $M^t_{\mathcal{F}}$ is given as part of instance.*

References

1 C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Reachability problems for sequential dynamical systems with threshold functions. *Theoretical Computer Science*, 295(1–3):41–64, 2003.

2 C. L. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Complexity of reachability problems for finite discrete dynamical systems. *Journal of Computer and System Sciences*, 72(8):1317–1345, 2006.

3 C. L. Barrett, H. B. Hunt III, M. V. Marathe, D. J. Rosenkrantz S. S. Ravi, R. E. Stearns, and M. Thakur. Predecessor existence problems for finite discrete dynamical systems. *Theoretical Computer Science*, 386(1–2):3–37, 2007.

4 C. L. Barrett, H. B.Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Predecessor and permutation existence problems for sequential dynamical systems. In *Proceedings of Discrete Mathematics and Theoretical Computer Science*, pages 69–80, 2003.

5 C. L. Barrett, H. B.Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and P. T. Tosic. Gardens of eden and fixed points in sequential dynamical systems. In *Proceedings of Discrete Mathematics and Theoretical Computer Science*, pages 95–110, 2001.

6 C. L. Barrett, H. S. Mortveit, and C. M. Reidys. Elements of a theory of simulation II: Sequential dynamical systems. *Applied Mathematics and Computation*, 107(2-3):121–136, 2000.

7 G. H. Hardy and E. M. Wrigth. *An Introduction to the Theory of Numbers. 6th edition. In R. Heath-brown et al. (Eds.).* Oxford University Press, 2008.

8 A. Kawachi. Personal communication, 2016.

9 A. Kawachi, M. Ogihara, and K. Uchizawa. Generalized predecessor existence problems for boolean finite dynamical systems on directed graphs. *Theoretical Computer Science*, 762:25–40, 2019.

10 S. Kosub. Dichotomy results for fixed-point existence problems for boolean dynamical systems. *Mathematics in Computer Science*, 1(3):487–505, 2008.

11 S. Kosub and C. M. Homan. Dichotomy results for fixed point counting in boolean dynamical systems. In *Proceedings of the Tenth Italian Conference on Theoretical Computer Science*, pages 163–174, 2007.

12 M. Ogihara and K. Uchizawa. Computational complexity studies of synchronous boolean finite dynamical systems on directed graphs. *Information and Computation*, 256:226–236, 2017.

13 D. J. Rosenkrantz, M. V. Marathe, H. B. Hunt III, S. S. Ravi, and R. E. Stearns. Analysis problems for graphical dynamical systems: A unified approach through graph predicates. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1501–1509, 2015.