Generalized Predecessor Existence Problems for Boolean Finite Dynamical Systems

Akinori Kawachi[1], **Mitsunori Ogihara**[2], **and Kei Uchizawa**[3]

1 **Graduate School of Engineering, Osaka University, Osaka, Japan**
2 **Department of Computer Science, University of Miami, Coral Gables, FL, USA**
3 **Graduate School of Science and Engineering, Yamagata University, Yamagata, Japan**

Abstract

A Boolean Finite Synchronous Dynamical System (BFDS, for short) consists of a finite number of objects that each maintains a boolean state, where after individually receiving state assignments, the objects update their state with respect to object-specific time-independent boolean functions synchronously in discrete time steps. The present paper studies the computational complexity of determining, given a boolean finite synchronous dynamical system, a configuration, which is a boolean vector representing the states of the objects, and a positive integer t, whether there exists another configuration from which the given configuration can be reached in t steps. It was previously shown that this problem, which we call the t-Predecessor Problem, is NP-complete even for $t = 1$ if the update function of an object is either the conjunction of arbitrary fan-in or the disjunction of arbitrary fan-in.

This paper studies the computational complexity of the t-Predecessor Problem for a variety of sets of permissible update functions as well as for polynomially bounded t. It also studies the t-Garden-Of-Eden Problem, a variant of the t-Predecessor Problem that asks whether a configuration has a t-predecessor, which itself has no predecessor. The paper obtains complexity theoretical characterizations of all but one of these problems.

1998 ACM Subject Classification F.1.1 Models of Computation

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

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.8

1 Introduction

A *dynamical system* is a time-dependent network of objects that models evolution, where each object holds a state value that is an element of a state set. The configuration of the system is the collective state of the objects and is the vector that assembles the states of all the objects in a certain order. Given an initial configuration, the system evolves over time through state update, where the state of an object is updated by a function that takes as input state values of some nodes, possibly its own state value. Variants of dynamical systems can be defined by considering the state set (binary, discrete, countably infinite, and uncountable), the types of permissible update functions, whether the number of states is fixed, and the order in which the updates are performed (either in a fixed order or all at the same time).

The simplest dynamical systems are those with the boolean state set, a fixed finite number of objects, and synchronous updates, where the state update function does not depend on the time distance from the start. These systems are called *boolean finite dynamical systems*

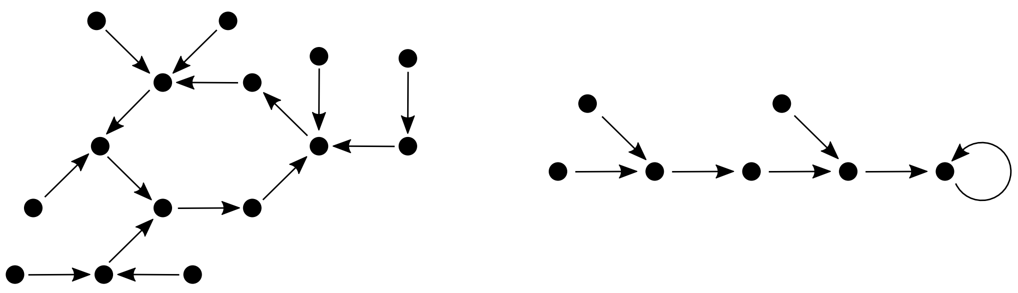

Figure 1 Configuration space with a loop, a fixed point, and flows into them. Dots represent configurations, and arrows do transitions. A node of in-degree 0 represents a Garden of Eden.

(BFDS) [1]. Quite often, the BFDS model is further simplified by assuming that the update functions are chosen from a collection of templates, such as the exclusive-or, the negation, the conjunction, and the disjunction.

Given a BFDS $\mathcal{F}$ of n objects, the number of possible configurations of the system is 2^n. Due to the assumptions that the updates are synchronized and that the update function does not change over time, the imposed finiteness of the system configuration space leads to important facts: for each initial configuration $\mathbf{a}$, the system starting from $\mathbf{a}$ either converges to a fixed point or enters some loop having length ≥ 2 and that the convergence or the entrance to a loop takes place within 2^n steps from departure (see Figure 1). These facts mean that the dichonomical fate of an initial configuration in a BFDS can be tested in the linear space. In fact, the fate-determination problem (or the convergence problem) as well as its variant, the reachability problem (whether a configuration be reached from another configuration with respect to a given BFDS), is PSPACE-complete if a complete boolean basis is available for building update functions and the complexity is lower for both problems otherwise [3]. The BDFS offers a rich theory not only in terms of fixed points, reachability, and cycles, but also in terms of the reversal, that is, the action of going back in time starting from a given configuration. Since the system changes two distinct configurations the same configuration with a single update, so a given configuration may have multiple predecessors. Also, there may exist configurations without predecessors. We call such a configuration *Garden of Eden (GOE)* (see Figure 1).

If the update functions are each polynomial time computable, which is indeed the case where the functions are chosen from a predetermined set of templates, testing whether a given configuration of a BFDS has a predecessor can be answered in NP, and thus, whether the configuration is a GOE can be answered in coNP. In fact, it is NP-complete to decide whether a configuration is *not* a GOE [2]. This paper makes a deeper investigation into this problem by asking how much simplification can be given the template set for update functions to retain this completeness. We show: if the templates are either only conjunction or only disjunction the GOE problem is in AC^0; if the templates are the two-fan-in conjunction and the two-fan-in disjunction, then the GOE problem is NL-complete; if the templates are either the combination of the two-fan-in conjunction and the three-fan-in disjunction or the combination of the two-fan-in disjunction and the three-fan-in conjunction, then the GOE problem is coNP-complete.

We generalize the GOE Problem further in two ways. First, for an integer $t \geq 1$, we ask whether we can go back from a given configuration successively t times, by cleverly choosing at each time one of the possible predecessors, if any at all. We call this problem the t-Predecessor Existence Problem (the t-PRED Problem, for short). Second, for an integer

$t \geq 0$, we ask whether we can go back from a given configuration successively t times and arrive at a GOE, by cleverly choosing at each time one of the possible predecessors, if any at all. We call this problem the t-Garden Of Eden Existence Problem (the t-GOE Problem, for short). The GOE Problem we mentioned earlier is indeed the 0-GOE Problem in this extension. It is easy to see that the 1-PRED Problem is exactly complementary to the 0-GOE Problem; but, for $t \geq 2$, the t-PRED is not necessarily complementary to the $(t-1)$-GOE Problem.

In this paper we ask the complexity of these two extensions with the functions restricted to be disjunction and conjunction. Except for the 1-GOE Problem with 2-fan-in disjunction and 2-fan-in conjunction, we obtain complete characterization of the constant-bounded as well as the polynomial-bounded t-PRED Problem and t-GOE Problem for all the templates consisting of conjunction and disjunction (see Table 1 in Section 2).

We note here that the papers [5, 6] show that the problem of computing fixed points in a BFDS exhibits a curious dichotomy between P and #P-complete and that [8] studies the problem of calculating the length of a cycle that an initial configuration is eventually taken to, and shows that the problem is polynomial-time solvable for some template sets, computable in UP for some, and PSPACE-complete for some. Along with these prior papers, our paper shows BFDS offers a rich theory of computational complexity.

This paper is organized as follows: In the next section we go over the definitions and prove some useful lemmas and propositions. We then show the results on the predecessor existence problems in Section 3 and the results on the Garden of Eden problems in Section 4. We will conclude the paper in Section 5.

2 Preliminaries

For an integer $n \geq 1$, a *synchronous boolean finite dynamical system* (synchronous BFDS, for short) $\mathcal{F}$ of n objects consists of n variables $x_1, \ldots, x_n$ and n boolean functions $(f_1, f_2, \ldots, f_n)$ such that for each i, $1 \leq i \leq n$, f_i is a boolean function that takes input from $x_1, \ldots, x_n$. A *state configuration* $\mathbf{a}$ (or simply a *configuration*) of $\mathcal{F}$ is an n-dimensional boolean vector and for each variable x, $\mathbf{a}[x] \in \{0, 1\}$ represents the component of $\mathbf{a}$ corresponding to x. The action of $\mathcal{F}$ on an state configuration $\mathbf{x}$ is defined by: $\mathcal{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x}))$. In other words, the elements of $\mathcal{F}(\mathbf{x})$ are obtained by applying the n boolean functions $f_1, \ldots, f_n$ concurrently on the variables $x_1, \ldots, x_n$.

In the remainder of the paper, boolean dynamical systems are defined without giving an explicit ordering among the objects. We will use the notation $\mathcal{F}[x]$ to mean the function of the dynamical system $\mathcal{F}$ for the object x.

Given an initial state configuration $\mathbf{x}^0$, the synchronous BFDS generates a sequence of state configurations by iterative applications of $\mathcal{F}$: For all $t \geq 0$, $\mathbf{x}^{t+1} = \mathcal{F}(\mathbf{x}^t)$, where $\mathbf{x}^t = (x_1^t, x_2^t, \ldots, x_n^t)$. In other words, for all $t \geq 0$, $\mathbf{x}^t = \mathcal{F}^t(\mathbf{x}^0)$.

Although we use the notation to mean that all the n variables are fed to each f_i, in reality some f_i may depend on a proper subset of the variables. Let g be a boolean function possibly with arity less than n. We say that f_i has template g to mean that f_i is equivalent to g with input variables properly chosen from $x_1, \ldots, x_n$. Given a collection of boolean functions, we say that $\mathcal{F}$ *has template set* $\mathcal{B}$ if each function in $\mathcal{F}$ has template in $\mathcal{B}$. We are interested in the following functions as template:

- id: this is the identity function with only one input that outputs the value of its input without changing it;
- AND_k, $k \geq 1$: this is the conjunction of arity k;
- OR_k, $k \geq 1$: this is the disjunction of arity k.

Note that for all $k \geq 2$ and $m < k$ AND_k (OR_k, respectively) can be used as a template for AND_m (OR_m, respectively) by repeating some of the inputs. Note also that both AND_1 and OR_1 are identical to id.

We are interested in the following template sets:

- $\mathcal{B}_{\mathrm{id}} = \{\mathrm{id}\}$,
- $\mathcal{B}_{2\mathrm{OR}} = \{\mathrm{OR}_2\}$ and $\mathcal{B}_{2\mathrm{AND}} = \{\mathrm{AND}_2\}$,
- $\mathcal{B}_{\mathrm{OR}} = \{\mathrm{OR}_k \mid k \geq 1\}$ and $\mathcal{B}_{\mathrm{AND}} = \{\mathrm{AND}_k \mid k \geq 1\}$,
- $\mathcal{B}_{2\mathrm{OR},2\mathrm{AND}} = \{\mathrm{OR}_2, \mathrm{AND}_2\}$,
- $\mathcal{B}_{3\mathrm{OR},2\mathrm{AND}} = \{\mathrm{OR}_3, \mathrm{AND}_2\}$ and $\mathcal{B}_{2\mathrm{OR},3\mathrm{AND}} = \{\mathrm{OR}_2, \mathrm{AND}_3\}$.

Given a synchronous BFDS $\mathcal{F}$ and a configuration $\mathbf{a}$ of $\mathcal{F}$, we say that another configuration $\mathbf{b}$ is the *t-th predecessor of* $\mathbf{a}$, $t \geq 1$, if $\mathcal{F}^t(\mathbf{b}) = \mathbf{a}$. We will omit the word *first* in the case where $t = 1$. By convention, we define the 0-th predecessor of $\mathbf{a}$ to be $\mathbf{a}$ itself. We say that $\mathbf{a}$ is a Garden of Eden if $\mathbf{a}$ has no predecessor. We then consider the following problems.

- Let t, $t \geq 1$, be a fixed constant. Given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$ and a configuration $\mathbf{a}$, the t-PRED Problem for $\mathcal{B}$ asks whether $\mathbf{a}$ has a t-th predecessor.
- Let t, $t \geq 0$, be a fixed constant. Given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a configuration $\mathbf{a}$, the t-GOE Problem for $\mathcal{B}$ asks whether $\mathbf{a}$ has a t-th Garden of Eden, i.e., a t-th predecessor that is a Garden of Eden.

We also consider the polynomial version of the two problems.

- Given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a configuration $\mathbf{a}$, and p presented in unary as 1^p, the Poly-PRED Problem for $\mathcal{B}$ asks whether $\mathbf{a}$ has a p-th predecessor.
- Given a synchronous BFDS $\mathcal{F}$ with template set $\mathcal{B}$, a configuration $\mathbf{a}$, and p presented in unary as 1^p, the Poly-GOE Problem for $\mathcal{B}$ asks whether $\mathbf{a}$ has a p-th Garden of Eden.

Note that as long as the predecessor existence and Garden of Eden problems go, by exchanging simultaneously between AND and OR and between true and false, any complexity result with respect to $\mathcal{B}_{\mathrm{AND}}$ holds with respect to $\mathcal{B}_{\mathrm{OR}}$. The same relation holds between $\mathcal{B}_{2\mathrm{AND}}$ and $\mathcal{B}_{2\mathrm{OR}}$ and between $\mathcal{B}_{2\mathrm{OR},3\mathrm{AND}}$ and $\mathcal{B}_{3\mathrm{OR},2\mathrm{AND}}$.

The 1-PRED and 0-GOE are generally called the Predecessor Existence Problem and the Garden-of-Eden Problem, respectively, and have been well studied. We note here that the synchronous BFDS often assumes that for each object its update function takes its state as part of the input; that is, x_i is one of the inputs to f_i. In this paper we remove that restriction, since if for all i it holds that f_i is either disjunction or conjunction whose inputs include x_i, the system $\mathcal{F}$ is monotone and converges within n steps, which gives little room for exploration.

For GOE Problem, in [2] it is shown that for the sequential dynamical systems (that is, the systems in which updates are performed one variable at a time with respect to a predetermined order), t-PRED is NP-complete even if the template set consists of AND's and OR's of any arity. The proof of this result does not directly imply the same result for the synchronous dynamical system.

In the case of sequential and synchronous dynamical systems, we have the following:

▶ **Proposition 1** ([2]). *The Poly-PRED Problem is solvable in polynomial time if $\mathcal{B}$ is one of the following: (i) ANDs of any fan-in and their negation, (ii) ORs of any fan-in and their negation, and (iii) XORs of any fan-in and their negation.*

For GOE Problem, the following is known:

▶ **Proposition 2** ([2]). *0-GOE Problem is* coNP*-complete in general, but is solvable in polynomial time if 1-PRED Problem is solvable in polynomial time.*

Table 1 The results from this paper. The left panel is for the PRED Problems and the right panel is for the GOE problems. For “?” it is only known that the problem is NL-hard and in NP. The “-C” stands for “-complete”.

PRED	t		
	1	≥ 2	poly
$\mathcal{B}_{\text{id}}$	AC^0	AC^0	DL-C
$\mathcal{B}_{\text{2OR}}$, $\mathcal{B}_{\text{2AND}}$	AC^0	AC^0	NL-C
$\mathcal{B}_{\text{OR}}$, $\mathcal{B}_{\text{AND}}$	AC^0	AC^0	NL-C
$\mathcal{B}_{\text{2OR,2AND}}$	NL-C	NP-C	NP-C
$\mathcal{B}_{\text{2OR,3AND}}$	NP-C	NP-C	NP-C
$\mathcal{B}_{\text{3OR,2AND}}$	NP-C	NP-C	NP-C

GOE	t			
	0	1	≥ 2	poly
$\mathcal{B}_{\text{id}}$	AC^0	AC^0	AC^0	DL-C
$\mathcal{B}_{\text{2OR}}$, $\mathcal{B}_{\text{2AND}}$	AC^0	AC^0	AC^0	NL-C
$\mathcal{B}_{\text{OR}}$, $\mathcal{B}_{\text{AND}}$	AC^0	NP-C	NP-C	NP-C
$\mathcal{B}_{\text{2OR,2AND}}$	NL-C	?	NP-C	NP-C
$\mathcal{B}_{\text{2OR,3AND}}$	coNP-C	Σ_2^p-C	Σ_2^p-C	Σ_2^p-C
$\mathcal{B}_{\text{3OR,2AND}}$	coNP-C	Σ_2^p-C	Σ_2^p-C	Σ_2^p-C

Table 1 summarizes the results shown in this paper.

We prove the following lemma, which will be useful in proving results on $\mathcal{B}_{\text{OR}}$ and $\mathcal{B}_{\text{2OR}}$.

▶ **Definition 3.** Let $\mathcal{F}$ be an n-variable synchronous BFDS, let $\mathbf{a}$ be a configuration of $\mathcal{F}$, and let $t \geq 1$ be an integer. Define a directed graph $G[\mathcal{F}, \mathbf{a}, t] = (V, E)$ with V partitioned into two groups K and L as follows:

- $V = V_0 \cup \ldots \cup V_t$ and for each i, $0 \leq i \leq t$, $V_i = \{v_{i,1}, \cdots, v_{i,n}\}$;
- $E = \{(v_{i,p}, v_{i+1,q}) \mid 0 \leq i \leq t-1$ and the function for x_p takes input from x_q $\}$.
- $L = L_0 \cup \ldots \cup L_t$, where $L_0 = \{v_{0,j} \mid$ the value of x_j in $\mathbf{a}$ is false $\}$ and for each i, $1 \leq i \leq t$, L_i is the set of all nodes in V_i that are reachable from L_0.
- $K = K_0 \cup \ldots \cup K_t$ and for each $i, 0 \leq i \leq t$, $K_i = V_i - L_i$; specifically, $K_0 = \{v_{0,j} \mid$ the value of x_j in $\mathbf{a}$ is true $\}$.

▶ **Lemma 4.** *Let $\mathcal{F}$ be an n-variable synchronous BFDS whose template set is the OR function. Let $t \geq 1$ and $\mathbf{a}$ a configuration of $\mathcal{F}$. Let $G = (V, E) = G[\mathcal{F}, \mathbf{a}, t]$ be as defined in the above and (K, L) be the partition of V. Then*

1. $\mathbf{a}$ *has a t-th predecessor if and only if for each $u \in K_0$ there is a path to a node in K_t that does not visit any node in L.*
2. $\mathbf{a}$ *has a t-th Garden-of-Eden predecessor if and only if there exist some $M = \{v_{t,j_1}, \ldots, v_{t,j_m}\} \subseteq V_t$ and $v_{t,j_0} \in K_t \backslash M (= V_t - (L_t \cup M))$ such that:*
   1. *for each $u \in K_0$ there is a path to a node in $K_t \setminus M$ that does not visit any node in $L \cup M$,*
   2. *each input of the function for x_{j_0} is an input of the function for one of $x_{j_1}, \cdots, x_{j_m}$, and*
   3. *the cardinality of M is no greater than the arity of the function for x_{j_0}.*

Proof. (The Predecessor Case) Suppose the condition in the lemma is satisfied. For each $u \in K_0$ choose one L-free path to some node in K_t. For each $i, 0 \leq i \leq t$, define $B_i = \{v_{i,j} \mid v_{i,j}$ appears on one of the chosen paths $\}$. Then $B_i \subseteq K_i$ for all i. For each $i, 0 \leq i \leq t$, define configuration $\mathbf{b}_i$ by setting the value of a variable x_j true if $v_{i,j} \in B_i$ and false otherwise. Then $\mathbf{b}_0 = \mathbf{a}$ and for all $i, 0 \leq i \leq t-1$, $\mathcal{F}(\mathbf{b}_{i+1}) = \mathbf{b}_i$. Thus, $\mathbf{b}_t$ is a t-th predecessor. On the other hand, suppose $\mathbf{a}$ has a t-th predecessor. Let $\mathbf{b}_0 = \mathbf{a}$. Select configurations $\mathbf{b}_1, \ldots, \mathbf{b}_t$ so that for each $i, 0 \leq i \leq t-1$, $\mathbf{b}_{i+1}$ is a predecessor of $\mathbf{b}_i$. For each $i, 0 \leq i \leq t$, let $S_i = \{v_{i,j} \mid \mathbf{b}_i$ assigns true to $x_{i,j}$ $\}$. Then $S_0 = K_0$. Because of the predecessor relations, for each $i, 1 \leq i \leq t$, $S_i \subseteq K_i$ and each node $u \in S_i$ has at least one incoming edge from S_{i-1}. Thus, the property holds for $\mathcal{F}$, $\mathbf{a}$, and t.

(The Garden Of Eden Case) Suppose that the conditions stated in the lemma hold. As before for each $u \in K_0$ choose a path that is free on $M \cup L$ but ensure that one of the paths arrive at some node in $K_t \setminus M$. Then, following the argument as before, the configurations induced by the path nodes form a series of t configurations arriving at $\mathbf{a}$. For $\mathbf{b}_t$, the value of x_{j_0} is true and the values of $x_{j_1}, \ldots, x_{j_m}$ are false. Each input of x_{j_0} is also an input of one of $x_{j_1}, \ldots, x_{j_m}$. Any predecessor of $\mathbf{b}_t$ must assign true to one of the inputs of x_{j_0} and must assign false to all of the inputs of $x_{j_1}, \ldots, x_{j_m}$, but that is not possible. Thus, there is no predecessor of $\mathbf{b}_t$.

On the other hand, suppose that there is a t-th predecessor of $\mathbf{a}$ that is a Garden of Eden. Select such one and define $\mathbf{b}_0, \ldots, \mathbf{b}_t$ and $S_0, \ldots, S_t$ as before. We have, as we have observed previously, for all $i, 0 \leq i \leq t$, $S_i \subseteq K_i$, and each node in $S_0 \cup \cdots \cup S_t$ is along an L-free path from K_0 (which is equal to S_0) to S_t. Let R be the variables that supply input to the variables corresponding to the nodes in $V_t - S_t$. In any predecessor of $\mathbf{b}_t$, the value of each variable in R must be false while each variable in $X - R$ can be set to true if needed. Then, that $\mathbf{b}_t$ is a Garden of Eden implies that there is some variable $u \in S_t$ all of whose input belong to R. Select one such u and for each input h of u, select a variable R that takes input from h. Construct M by placing the chosen variables from R. Then u and M satisfy the property in question. ◀

We have straightforward upper bounds on the predecessor and Garden-Of-Eden problems.

▶ **Proposition 5** ([2]). *Suppose the template set $\mathcal{B}$ consists only of polynomial-time computable boolean functions. Then the Poly-PRED Problem is in* NP.

▶ **Proposition 6.** *Suppose the template set $\mathcal{B}$ consists only of polynomial-time computable boolean functions. Then the Poly-GOE Problem is in Σ_2^p.*

The following proposition is useful to reduce the predecessor problems to Garden-Of-Eden problems.

▶ **Proposition 7.** *Given a synchronous BFDS $\mathcal{F}$, its configuration $\mathbf{a}$, and $t \geq 1$, we can add $t+2$ variables to $\mathcal{F}$ and $\mathbf{a}$ to create a new BFDS $\mathcal{F}'$ and $\mathbf{a}'$ so that:*

- *if $\mathbf{a}$ has a t-th predecessor in $\mathcal{F}$, then $\mathbf{a}'$ has a t-th predecessor in $\mathcal{F}'$ and none of its t-th predecessors have a predecessor; and*
- *if $\mathbf{a}$ does not have a t-th predecessor in $\mathcal{F}$, then $\mathbf{a}'$ does not have a t-th predecessor in $\mathcal{F}'$.*

Proof. Let $\mathcal{F}$, $\mathbf{a}$, and t be given. Introduce $t+2$ variables $e_0, \ldots, e_{t+1}$. We define the function of e_0 to be $\mathrm{id}(e_0)$ and the function for $e_i, 1 \leq i \leq t+1$, to be $\mathrm{id}(e_{i-1})$. This is $\mathcal{F}'$. We then add to $\mathbf{a}$ the values of e_i as all false except e_{t+1}. Then, for each i, $1 \leq i \leq t$, the i-th predecessor of the additional part has false for $e_0, \ldots, e_{t-i}$ and true for e_{t-i+1}. Specifically we have that the t-th predecessor on this part has false for e_0 and true for e_1. Since e_0 and e_1 take input from e_0 and is the identify function, clearly, such a t-th predecessor cannot have a predecessor. Since the new variables and functions are disjoint with those in the original, the new part does not affect the invertibility of the original part. This proves the proposition. ◀

The last general result we present in this section shows that if a predecessor problem with a certain template set (respectively, a GOE problem with a certain template set) for some t is hard a problem H, then the problem is hard for $t+1$. We omit the proof of this lemma.

▶ **Lemma 8.** *Let $\mathcal{B}$ be a template set containing the two-fan-in OR. Suppose there is a many-one reduction g from a problem H to the t-predecessor problem with $t \geq 1$ (respectively,* *the t-GOE problem with $t \geq 0$). Then there is a many-one reduction g' from H to the $(t+1)$-predecessor problem (respectively, the $(t+1)$-GOE problem). Furthermore, if g is logspace computable (polynomial-time computable) using $\mathcal{B}$ as the oracle, then so is g'.*

3 The Complexity of Predecessor Problems

We first consider the case of $\mathcal{B}_{\mathrm{2OR,3AND}}$ and $\mathcal{B}_{\mathrm{3OR,2AND}}$, and prove that the problems are NP-complete.

▶ **Theorem 9.** *For $\mathcal{B}_{\mathrm{2OR,3AND}}$ and $\mathcal{B}_{\mathrm{3OR,2AND}}$, the t-PRED Problem is* NP*-complete for all constants $t \geq 1$.*

Proof. The inclusion in NP follows from Proposition 5.

We consider $\mathcal{B}_{\mathrm{3OR,2AND}}$ for NP-hardness and provide a polynomial-time many-one reduction from 3SAT to the 1-PRED Problem. Let φ be a 3CNF formula with n variables and m clauses. We introduce variables w, $c_1, \ldots, c_m$, $y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1}$, and $z_{1,0}, z_{1,1}, \ldots, z_{n,0}, z_{n,1}$. We associate $y_{i,1}$ with the positive literal of the i-th variable of y and $y_{i,0}$ with the negative literal of the i-th variable. We define the functions for these variables as follows:

- w, $y_{i,0}$, $y_{i,1}$: the function is $\mathrm{id}(w)$.
- $z_{i,0}$: $\mathrm{OR}(y_{i,0}, y_{i,1})$.
- $z_{i,1}$: $\mathrm{AND}(y_{i,0}, y_{i,1})$.
- c_j: Let the three literals of C_j be $y_{p,b}$, $y_{q,c}$, and $y_{r,d}$. Then the function is $\mathrm{OR}(y_{p,b}, y_{q,c}, y_{r,d})$.

We set the values of the variables in $\mathbf{a}$ to true for w, $y_{i,0}$, $y_{i,1}$, and $z_{i,0}$ and false for $z_{i,1}$.

Suppose $\mathbf{a}$ has a predecessor $\mathbf{b}$. Then for all i, since $z_{i,0}$ is true and $z_{i,1}$ is false in $\mathbf{a}$, in $\mathbf{b}$ exactly one of $y_{i,0}$ and $y_{i,1}$ is true and the values of y's in $\mathbf{b}$ can be viewed as a truth-assignment to the n variables of φ. Then, for all j, c_j is true in $\mathbf{a}$ and the inputs to the function for c_j correspond to the literals of C_j, it must be the case that the truth-assignment as represented by the y's in $\mathbf{b}$ form a satisfying assignment of φ. This means that φ is satisfiable.

On the other hand, suppose that φ is satisfiable. We take one satisfying assignment α of φ and build $\mathbf{b}$ by setting the values to y's according to α, true to w, and an arbitrary value to z's. Then $\mathcal{F}(\mathbf{b}) = \mathbf{a}$ and so $\mathbf{a}$ has a predecessor.

Clearly, this reduction can be computed in polynomial time, and so the theorem holds for $t = 1$. By combining this with Lemma 8, we obtain the proof for $t \geq 2$. ◀

We then consider the case of $\mathcal{B}_{\mathrm{2OR,2AND}}$. In this case, the problem is tractable only when $t = 1$.

▶ **Theorem 10.** *For $\mathcal{B}_{\mathrm{2OR,2AND}}$, the 1-PRED Problem is* NL*-complete and for all constants $t \geq 2$ the problem is* NP*-complete.*

Proof. It is easy to see that the above proof can be carried out for 2SAT with a logspace computable many-one reduction and so the 1-PRED Problem with $\mathcal{B}_{\mathrm{2OR,2AND}}$ is NL-hard. To show that the problem is in NL, note that in the case of $\mathcal{B}_{\mathrm{2OR,2AND}}$, given a configuration $\mathbf{a}$, the value assignments to its predecessor can be written as a 2CNF formula with possible single-literal clauses; e.g., $\mathrm{OR}(x, y) = \text{true}$ can be expressed as $x \vee y$.

For the 2-PRED Problem, the main idea is to break the computation of three-literal ORs into the OR to two two-variable ORs. We introduce alternating variables w_1 and w_2 whose functions are $\mathrm{id}(w_2)$ and $\mathrm{id}(w_1)$, respectively, and set their values in $\mathbf{a}$ to be true and false, respectively. Then for any predecessor of $\mathbf{a}$, their values should be false and true, respectively, and for any second predecessor of $\mathbf{a}$, their values should be true and false, respectively. We use variables $y_{i,0}, y_{i,1}, z_{i,0}, z_{i,1}$ as in the case of $\mathcal{B}_{3\mathrm{OR},2\mathrm{AND}}$ and add $u_{i,0}, u_{i,1}$. For each j, we introduce three variables c_j, d_j, and e_j and define their functions to be $\mathrm{OR}(d_j, e_j)$, $\mathrm{OR}(y_{p,c}, y_{q,d})$, and $\mathrm{OR}(y_{p,c}, y_{r,e})$ where $y_{p,c}, y_{q,d}$, and $y_{r,e}$ are the three literals of C_j. We define the functions for $y_{i,b}$ to be $\mathrm{OR}(y_{i,b}, w_1)$ for each $b \in \{0,1\}$, the functions for $u_{i,b}$ to be $\mathrm{id}(z_{i,b})$, the functions for $z_{i,0}$ to be $\mathrm{OR}(y_{i,0}, y_{i,1})$, and the functions for $z_{i,1}$ to be $\mathrm{AND}(y_{i,0}, y_{i,1})$.

In $\mathbf{a}$ we set the value of w_2 and all $u_{i,1}$ to false and set everything else to true. Assume $\mathbf{a}$ has a predecessor $\mathbf{a}'$ and $\mathbf{a}'$ has a predecessor $\mathbf{a}''$. The values of $y_{i,0}$ and $y_{i,1}$ in $\mathbf{a}''$ are OR-ed and AND-ed and stored in $z_{i,0}$ and $z_{i,1}$, respectively, in $\mathbf{a}'$ and then preserved in $u_{i,0}$ and $u_{i,1}$, respectively, in $\mathbf{a}$. So, it must be the case that exactly one of $y_{i,0}$ and $y_{i,1}$ is true in $\mathbf{a}''$ and thus we can view these as truth-assignments to the variables of φ. For each clause C_j, the first and the second literals of C_j as appearing in $\mathbf{a}''$ are OR-ed and stored in $\mathbf{a}'$ as d_j as well as the first and the second literals as e_j. These two are then joined by an OR in $\mathbf{a}$ as c_j. Thus, for $\mathbf{a}''$ to exist the y's in $\mathbf{a}''$ must represent a satisfying assignment of φ. Because y's are OR-ed with w_1 and $\mathbf{a}''$ should have true for w_1, y's in $\mathbf{a}'$ are all true. This means that z's, d's, and e's become all true in $\mathbf{a}$. Thus, $\mathbf{a}$ has a second predecessor if and only if φ is satisfiable.

The hardness for the case $t \geq 3$ follows from Lemma 8. ◀

If $\mathcal{B}$ contains only conjunction or only disjunction, the problems are significantly easy.

▶ **Theorem 11.** *For $\mathcal{B}_{\mathrm{OR}}$ and $\mathcal{B}_{\mathrm{AND}}$, the t-PRED Problem is in AC^0 for all constants $t \geq 1$.*

Proof. Suppose that $\mathcal{B}$ is $\mathcal{B}_{\mathrm{OR}}$, and we are given $\mathcal{F}$ and $\mathbf{a}$. Let $G = (V, E) = G[\mathcal{F}, \mathbf{a}, t]$, and (K, L) be the partition of V as given in Definition 3. By Lemma 4, we have only to, for each $u \in K_0$, enumerate all the paths from u to V_t and then check if the reachable nodes in K_t can be reachable from L_0. Since the number of all the paths in G is $O(n^t)$, this can be carried out using an AC^0 circuit. Thus, we have done. ◀

We now consider Poly-PRED Problems where t is part of the input. By combining Proposition 5 and Theorem 9, we obtain the following corollary for the case where $\mathcal{B}$ contains both conjunction and disjunction.

▶ **Corollary 12.** *Suppose $\mathcal{B}$ is one of $\mathcal{B}_{2\mathrm{OR},2\mathrm{AND}}$, $\mathcal{B}_{2\mathrm{OR},3\mathrm{AND}}$, and $\mathcal{B}_{3\mathrm{OR},2\mathrm{AND}}$. Then the Poly-PRED Problem with template set $\mathcal{B}$ is* NP*-complete.*

In the case of $\mathcal{B}_{\mathrm{id}}$, the problem is L-complete, and in the case of $\mathcal{B}_{\mathrm{OR}}$ and $\mathcal{B}_{\mathrm{AND}}$, the problem is NL-complete. The proofs are omitted due to the page limitation.

▶ **Theorem 13.** *The Poly-PRED Problem with template set $\mathcal{B}_{\mathrm{id}}$ is* L*-complete under logspace-uniform AC^0-reductions.*

▶ **Theorem 14.** *Suppose $\mathcal{B}$ is either $\mathcal{B}_{\mathrm{OR}}$ or $\mathcal{B}_{\mathrm{AND}}$. Then the Poly-PRED Problem with template set $\mathcal{B}$ is* NL*-complete under logspace-uniform AC^0-reductions.*

4 The Complexity of Garden-of-Eden Problems

In this section we prove our results on the Garden of Eden problems. In the previous section we prove our hardness results by producing a many-one reduction from a language L to a synchronous BFDS and $\mathbf{a}$ so that there is a t-th predecessor if the input is a member of L and so that there is a $(t-1)$-st predecessor but no t-th predecessor if the input is not a member of L. One may think that this construction can be used to produce a many-one reduction from the complement of L to the $(t-1)$-st GOE problem with respect to the same BFDS. Unfortunately, this is not the case because according to the construction there may be multiple first predecessors of $\mathbf{a}$ and so even in the case where the input is a member of L, not every $(t-1)$-st predecessor of $\mathbf{a}$ has a predecessor. Thus, to prove the hardness of a Garden of Eden problem for a language L the construction must be such that if the input is in L, there is a $(t-1)$-st predecessor that is a Garden of Eden and such that if the input is not in L, *every* $(t-1)$-st predecessor has a predecessor.

As mentioned earlier, unlike other $t \geq 1$, 0-GOE Problem is complementary to 1-PRED Problem. Noting that NL is closed under complementation [4, 7], we have the following result from Theorems 9 and 10.

▶ **Theorem 15.** *The* 0*-GOE Problem is in* AC^0 *if the template set is one of* $\mathcal{B}_{\mathrm{id}}$, $\mathcal{B}_{2\mathrm{OR}}$, $\mathcal{B}_{2\mathrm{AND}}$, $\mathcal{B}_{\mathrm{OR}}$, *and* $\mathcal{B}_{\mathrm{AND}}$, NL*-complete if the template set is* $\mathcal{B}_{2\mathrm{OR},2\mathrm{AND}}$, *and* coNP*-complete if the template set is either* $\mathcal{B}_{2\mathrm{OR},3\mathrm{AND}}$ *or* $\mathcal{B}_{3\mathrm{OR},2\mathrm{AND}}$.

Consider then the case where $t \geq 1$. We first observe an upper bound on the problems for the case of $\mathcal{B}_{\mathrm{OR}}$ and $\mathcal{B}_{\mathrm{AND}}$.

▶ **Lemma 16.** *The Poly-GOE Problem with template* $\mathcal{B}_{\mathrm{OR}}$ *or* $\mathcal{B}_{\mathrm{AND}}$ *is in* NP.

Proof. Let $\mathcal{F}$ be a synchronous BFDS with template $\mathcal{B}_{\mathrm{OR}}$ with n variables $x_1, \ldots, x_n$ and let $\mathbf{a}$ be a configuration of $\mathcal{F}$. Suppose we are testing whether $\mathbf{a}$ is a t-th predecessor and is a GOE in the system $\mathcal{F}$ and t is given as 1^t as part of input. Consider a layered digraph $G = (V, E)$ whose variable set V has $t+2$ layers, $X_0, \ldots, X_t, X_{t+1}$, where for each j, $0 \leq j \leq t+1$, X_j consists of n variables $x_{1,j}, \ldots, x_{n,j}$. We draw a directed edge $(x_{i,j}, x_{i',j'})$ if $0 \leq j \leq t$, $j' = j+1$, and $x_{i'}$ is an input to the OR function for x_i.

Let S be the set of all $x_{i,0}$ such that the value of x_i is true in $\mathbf{a}$ and Let S' be the set of all $x_{i,0}$ such that the value of x_i is false in $\mathbf{a}$. Let R be the set of all nodes reachable from S'. Let $T' = R \cap X_t$ and $T = X_t - T'$.

We claim that $\mathbf{a}$ has a t-th predecessor that is a GOE if and only if there exists a set $D \subseteq T$ such that (i) from each node in S there is a path to a node in D that does not visit R and (ii) there is a node u in D such that every node v in X_{t+1} is reached from u is reached from some node in $X_t - D$.

Suppose there is such a D. Define a configuration $\mathbf{b}$ by setting the value of u to true if and only if it belongs to D. Since D is reachable from S without going through the nodes in R and $D \cap R = \emptyset$, we have $\mathcal{F}^t(\mathbf{b}) = \mathbf{a}$ and so $\mathbf{a}$ is a t-th predecessor of $\mathbf{a}$. Each predecessor $\mathbf{b}$ of must have a value true for at least one of the variables v that is reachable from u, but since $\mathbf{b}$ has value false for all the nodes in $X_t - D$, all of them must have value false. These two requirements are conflicting with each other and cannot be satisfied at the same time. Thus, $\mathbf{b}$ is a GOE.

On the other hand, suppose there is no such a D. Suppose there is no D satisfying (i), it means that $\mathbf{a}$ does not have a t-th predecessor, and so, $\mathbf{a}$ does not have a t-th predecessor that is a GOE. Suppose there is a D satisfying (i) but each such D fails to satisfy (ii). For each such D we define $\mathbf{b}$ to be configuration that assigns true to all variables in D and false to the rest. Also, for each such D we can select for each $u \in D$ select one node v in $X_{t+1} - R$ that is reachable from u. By setting the value to true for all v's thus selected to false for the remainder, we obtain a configuration $\mathbf{c}$. Then $\mathbf{b}$ is a t-th predecessor of $\mathbf{a}$ and $\mathbf{c}$ is a predecessor of $\mathbf{b}$ and so every t-th predecessor of $\mathbf{a}$ has a predecessor.

To test the existence of a D in question, we calculate S, S', R, T', T and then try all possible combinations of $D \subseteq T$ and $u \in D$, and so the test can be carried out in NP.

This proves the lemma. ◀

Unlike predecessor problems, Garden-of-Eden problem is NP-complete even if $\mathcal{B}$ is either $\mathcal{B}_{\mathrm{OR}}$ or $\mathcal{B}_{\mathrm{AND}}$ and t is part of the input.

▶ **Theorem 17.** *The* 1-*GOE Problem with template* $\mathcal{B}_{\mathrm{OR}}$ *or* $\mathcal{B}_{\mathrm{AND}}$ *is* NP-*complete.*

Proof. Because we have Lemma 16, we have only to show that the 1-GOE Problem is NP-hard. We will reduce 3SAT to the problem. Let φ be a 3CNF formula of n variables and m clauses. Our synchronous BFDS uses variables, among other, $x_{i,0}, x_{i,1}, y_{i,0}, y_{i,1}, u_i, z_i, 1 \leq i \leq n$, $c_j, 1 \leq j \leq m$. The idea is to use $y_{i,0}$ and $y_{i,1}$ to encode purported value assignments to the negative and the positive literals of x_i. These assignments are not necessarily opposite to each other. We use the variable u_i to generate $y_{i,0}$ as the OR of $x_{i,0}$ and u_i and generate $y_{i,1}$ as the OR of $x_{i,1}$ and u_i. We generate z_i as the OR of $y_{i,0}$ and $y_{i,1}$. We use the value assignments to the literal $y_{i,b}$'s to evaluate the variables c_j's, which are corresponding to the clauses. If φ is satisfiable, it is possible to choose the values for $y_{i,b}$'s so that for each i, $y_{i,0}$ and $y_{i,1}$ are opposite to each other and so u_i is false. If φ is not satisfiable, to make the value of c_j true for all i, we need to assign true to both $y_{i,0}$ and $y_{i,1}$ and so for such an i, u_i can be set to true. So, assuming that z_i's and c_j's are all true, φ is satisfiable if and only if we can choose the values for $y_{i,b}$'s so that u_i's is all false.

We formalize this idea using additional variables p_0, p_1, p_2, u_0, and v_0. The functions for p_0, p_1, and p_2 are respectively $\mathrm{id}(p_2)$, $\mathrm{id}(p_0)$, and $\mathrm{id}(p_1)$; that is, they rotate the values among them, from p_0 to p_1, p_1 to p_2, and then p_2 to p_0. We set the values for them in $\mathbf{a}$ to be false, false, and true, and so, in each predecessor of $\mathbf{a}$, if any, their values should be false, true, false, and in each second predecessor of $\mathbf{a}$, if any, their values should be true, false, false. The function for u_0 is the OR of $u_1, \ldots, u_n$ and the function for v_0 is the OR of u_0 and p_0.

The functions for the other variables are as follows:

- u_i: the OR of u_i and p_0;
- z_i: the OR of $y_{i,0}, y_{i,1}$, and p_0;
- $y_{i,0}$: the OR of $x_{i,0}$ and u_i;
- $y_{i,1}$: the OR of $x_{i,1}$ and u_i;
- $x_{i,0}$: the OR of $x_{i,0}$ and p_0;
- $x_{i,1}$: the OR of $x_{i,1}$ and p_0;
- c_j: the OR of $y_{k,b}$, $y_{l,c}$, $y_{m,d}$, and p_0 where $y_{k,b}$, $y_{l,c}$, $y_{m,d}$ are the variables corresponding to the three literals of the jth clause in φ.

Figure 2 shows how these variables interact.

The configuration $\mathbf{a}$ is all true except for p_0 and p_1. Let us speak of a hypothetical predecessor $\mathbf{b}$ of $\mathbf{a}$ and a hypothetical predecessor $\mathbf{c}$ of $\mathbf{b}$. As mentioned earlier, the value of p_0 is true in $\mathbf{c}$ and false in $\mathbf{b}$. Because of this, all the variables that take p_0 as part of input must be true in $\mathbf{b}$; they are p_1, v_0, all u_i's, all z_i's, all c_j's, and all $x_{i,b}$'s. Since these are all true, in $\mathcal{F}(\mathbf{b})$ the value should be true for u_0, all u_i's, all $y_{i,b}$'s, and $x_{i,b}$'s, which is consistent with their value assignments in $\mathbf{a}$. Also, since v_0 has value true in $\mathbf{a}$, in $\mathbf{b}$ the value of u_0 should be true. The only values that are not determined in $\mathbf{b}$ are those of $y_{i,b}$'s. The constraints are that for each i, either $y_{i,0}$ or $y_{i,1}$ must be true and that these assignments satisfy all the clauses.

Suppose φ is satisfiable. We pick one satisfying assignment of φ and select the values of $y_{i,b}$ accordingly. Then $y_{i,0}$ and $y_{i,1}$ are opposite to each other for all i, and so in $\mathbf{c}$ u_i's must be all false. However, since u_0 has value true in $\mathbf{b}$ and it is the OR of all u_i's, to make it happen the value of at least one u_i must be true in $\mathbf{c}$. These two requirements are conflicting and so $\mathbf{c}$ cannot exist and thus $\mathbf{b}$ is a GOE.

On the other hand, suppose φ is not satisfiable. We determine the values of $y_{i,b}$'s in $\mathbf{b}$ so that they turn c_j's all true in $\mathbf{a}$ and for each i, at least one of $y_{i,0}$ and $y_{i,1}$ is true. Because φ

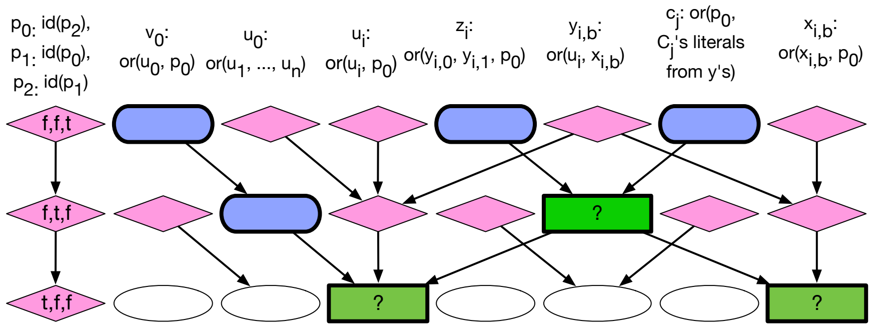

Figure 2 The construction for the 1-GOE Problem for $\mathcal{B}_{\mathrm{OR}}$. The top layer is **a**, the middle layer is a predecessor of **a**, and the bottom layer is a second predecessor of **a**. The arrows show where the inputs come from but inputs from p_i's are not shown. The variables in each diamond shaped are fixed because of the use of p_0. Unless specified, they are all true. The rectangles with rounded sides show blocks of variables whose values are expected to be true. They take input from the rectangles marked with the question mark. They are variables corresponding to the value assignments to the literals and the u_i's.

is not satisfiable, there must be at least one i such that both $y_{i,0}$ and $y_{i,1}$ are true. Select s to be such an i. In **c** set u_s to true and other u_i's to false and set the value of $x_{i,b}$ in **c** equal to the value of $y_{i,b}$ in **b**. Also, set all the remaining variables except the p_i's false. Then this **c** is indeed in a predecessor of **b**. This means that each predecessor **b** of **a** has a predecessor and thus there is no predecessor that is a GOE.

This proves the theorem. ◀

From the above and Lemmas 8 and 16, we have the following corollary.

▶ **Corollary 18.** *Let $\mathcal{B}$ be either $\mathcal{B}_{\mathrm{OR}}$ or $\mathcal{B}_{\mathrm{AND}}$. The Poly-GOE Problem with template $\mathcal{B}$ and t-GOE Problem for $t \geq 1$ with template $\mathcal{B}$ are* NP*-complete.*

If the arities of conjunction and disjunction are bounded by two, the problems are tractable.

▶ **Theorem 19.** *Let $\mathcal{B}$ be one of $\mathcal{B}_{\mathrm{id}}$, $\mathcal{B}_{2\mathrm{OR}}$, and $\mathcal{B}_{2\mathrm{AND}}$. For all constants $t \geq 0$, the t-GOE Problem with template $\mathcal{B}$ is in* AC^0.

Proof. We have only to test the conditions as stated in Lemma 4 part 2. Specifically, we need to compute $G = (V, E)$, the partition (K, L) of V, and then try all possible combinations for u and M to see whether the three properties hold. The conditions can be tested in AC^0 because M has cardinality at most 2. ◀

However, we show that the problem for $\mathcal{B}_{\mathrm{id}}$ and that for $\mathcal{B}_{2\mathrm{OR}}$ or $\mathcal{B}_{2\mathrm{AND}}$ belong to different complexity classes if t is part of the input.

▶ **Theorem 20.** *The Poly-GOE Problem with template $\mathcal{B}_{\mathrm{id}}$ is* L*-complete.*

Proof. Theorem 13 shows that the Poly-PRED Problem with template $\mathcal{B}_{\mathrm{id}}$ is L-complete. By Proposition 7, this implies that Theorem 13 shows that the Poly-GOE Problem with template $\mathcal{B}_{\mathrm{id}}$ is L-hard.

To show that the problem is in L, we use Lemma 4. Since the available function template is id, the set M in the proposition has cardinality 1. As we have seen in Theorem 13, the reachability part of the test can be carried out in L. As for the remaining properties, we have only to try all possible combinations of u and M. ◀

▶ **Theorem 21.** *The Poly-GOE Problem with template $\mathcal{B}_{2\mathrm{OR}}$ or $\mathcal{B}_{2\mathrm{AND}}$ is* NL*-complete.*

Proof. The NL-hardness follows from Theorem 14 and Proposition 7. To show that the problem is in NL, we use Lemma 4. Since the arity of the functions is at most 2, there are only polynomially many possibilities for the choice of z and M and so the test can be done in NL. ◀

In the case where $\mathcal{B}$ contains both conjunction and disjunction, the complexity of problems depends on whether $\mathcal{B}$ contains a function of arity more than two or not for any $t \geq 2$.

▶ **Theorem 22.** *The t-GOE Problem with template $\mathcal{B}_{2\mathrm{OR},2\mathrm{AND}}$ for $t \geq 2$ and the Poly-GOE Problem with template $\mathcal{B}_{2\mathrm{OR},2\mathrm{AND}}$ are* NP*-complete.*

Proof. The hardness follows from Theorem 10, Proposition 7, and Lemma 8. The membership in NP comes from the fact that in the case where the functions have arity 2, whether a configuration has a predecessor can be expressed as a 2CNF formula. We have only to guess a series of t configurations, verify the series flows into $\mathbf{a}$, and the t-th predecessor in the series does not have a predecessor. ◀

▶ **Theorem 23.** *Let $\mathcal{B}$ be either $\mathcal{B}_{2\mathrm{OR},3\mathrm{AND}}$ or $\mathcal{B}_{3\mathrm{OR},2\mathrm{AND}}$. The t-GOE Problem with template $\mathcal{B}$ for $t \geq 1$ and the Poly-GOE Problem with template $\mathcal{B}$ are Σ_2^p-complete.*

We omit the proof of Thorem 23.

5 Conclusions

In this paper we studied the complexity of the Predecessor Existence Problem and the Garden of Eden Problem for various orders of predecessor and for various template sets. Other than those stating containment in AC^0 and the 1-Garden of Eden Problem for $\mathcal{B}_{2\mathrm{OR},2\mathrm{AND}}$, the problems are shown to be complete for standard complexity classes and the classes that appear are diverse: $\mathrm{L}, \mathrm{NL}, \mathrm{NP}, \mathrm{coNP}$, and Σ_2^p. An obvious next step for the paper is to pinpoint the complexity for the remaining case. Also, it will be interesting to look at other template sets.

References

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

2 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*, 95–110, 2001.

3 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*, 340(3):496–513, 2005.

4 N. Immerman. Nondeterministic space is closed under complementation. SIAM Journal on Computing, 17:935–938, 1988.

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

6 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.

7 R. Szelepcsényi. The method of forcing for nondeterministic automata. Bulletin of the EATCS, 33:96–100, 1987.

8 M. Ogihara and K. Uchizawa. Computational complexity studies of synchronous boolean finite dynamical systems. In *Proceedings of the 12th Conference on Theory and Applications of Models of Computation*, pages, 87–98, 2015.