State Complexity of GF(2)-Concatenation and GF(2)-Inverse on Unary Languages

Alexander Okhotin, Elizaveta Sazhneva

State complexity of GF(2)-concatenation and GF(2)-inverse on unary languages*

Alexander Okhotin[0000−0002−1615−2725] and Elizaveta Sazhneva

St. Petersburg State University, 7/9 Universitetskaya nab.,
Saint Petersburg 199034, Russia
alexander.okhotin@spbu.ru, sazhneva.eliza@yandex.ru

Abstract. The paper investigates the state complexity of two operations on regular languages, known as GF(2)-concatenation and GF(2)-inverse (Bakinova et al., "Formal languages over GF(2)", LATA 2018), in the case of a one-symbol alphabet. The GF(2)-concatenation is a variant of the classical concatenation obtained by replacing Boolean logic in its definition with the GF(2) field; it is proved that GF(2)-concatenation of two unary languages recognized by an m-state and an n-state DFA is recognized by a DFA with $2mn$ states, and this number of states is necessary in the worst case, as long as m and n are relatively prime. This operation is known to have an inverse, and the state complexity of the GF(2)-inverse operation over a unary alphabet is proved to be exactly $2^{n-1}+1$.

1 Introduction

Union and concatenation of formal languages are defined in terms of conjunction and disjunction: a string is in $K \cup L$ if it is in K *or* in L, and a string w is in $K \cdot L$, if, for some partition $w = uv$, $u \in K$ *and* $v \in L$—a disjunction of $|w|+1$ conjunctions. New variants of these two operations, obtained by replacing disjunctions with *exclusive OR*, have recently been proposed by Bakinova et al. [1]. Union ($K \cup L$) is thus replaced with symmetric difference ($K \triangle L$), whereas for concatenation ($K \cdot L$), once the disjunction is replaced with exclusive OR, the condition of the *existence of a partition* turns into the condition that *the number of partitions must be odd.*

$$K \cdot L = \{\, w \mid \#\text{ of partitions } w = uv, \text{ with } u \in K \text{ and } v \in L, \text{ is non-zero} \,\}$$
$$K \odot L = \{\, w \mid \#\text{ of partitions } w = uv, \text{ with } u \in K \text{ and } v \in L, \text{ is odd} \,\}$$

The latter operation is called *GF(2)-concatenation*, because it is actually a weighted concatenation with weights in the GF(2) field. For example, $\{\varepsilon, a\} \cdot \{\varepsilon, a\} = \{\varepsilon, a, aa\}$, but $\{\varepsilon, a\} \odot \{\varepsilon, a\} = \{\varepsilon, aa\}$, because two partitions of a cancel each other. Notably, GF(2)-concatenation is *invertible*: for every language $L \subseteq \Sigma^*$ with $\varepsilon \in L$, there exists a unique language $L^{-1} \subseteq \Sigma^*$ that

* Supported by Russian Science Foundation, project 18-11-00100.

	Union	Concatenation		Star	
Unambiguous	$(\uplus)$	$mn-1$ [6]	(UNAMB$\cdot$)	$m2^{n-1}-2^{n-2}$ [4]	(UNAMB$*$)	$\frac{3}{8}2^n+1$ [6]
Classical	$(\cup)$	mn [9]	$(\cdot)$	$m2^n-2^{n-1}$ [9]	$(*)$	$\frac{3}{4}2^n$ [9]
GF(2)	$(\triangle)$	mn [2]	$(\odot)$	$m\cdot 2^n$ [1]	$(^{-1})$	2^n+1 [1]

Table 1. State complexity of unambiguous, classical and GF(2)-variants of union, concatenation and star.

satisfies $L \odot L^{-1} = L^{-1} \odot L = \{\varepsilon\}$. For instance, $\{\varepsilon, a\}^{-1} = a^*$, because $\{\varepsilon, a\}^{-1} \odot a^* = \{\varepsilon\}$: indeed, every non-empty string in the latter GF(2)-concatenation has two partitions. Symmetric difference is the *GF(2)-union*.

Using GF(2)-operations instead of the classical operations gives rise to a new variant of formal language theory. For instance, *GF(2)-grammars*, defined by Bakinova et al. [1] and subsequently studied by Makarov and Okhotin [8], are incomparable in power to classical grammars with union and concatenation, but still have a parsing algorithm working in time $O(n^\omega)$, with $\omega < 3$, and can be parsed by circuits of depth $O((\log n)^2)$.

The family of regular languages is closed under both the GF(2)-concatenation and the GF(2)-inversion operations. For a pair of languages recognized by an m-state and an n-state DFA, their GF(2)-concatenation is recognized by a DFA with $m \cdot 2^n$ states; this number of states is necessary in the worst case, witnessed by automata over a 2-symbol alphabet [1]. Similarly, the GF(2)-inverse of a language recognized by an n-state DFA is recognized by a DFA with $2^n + 1$ states, and this bound is tight for alphabets containing at least 3 symbols [1].

To compare with the classical case, classical concatenation has state complexity $m2^n - 2^{n-1}$, and classical Kleene star, or the quasi-inverse, has state complexity $\frac{3}{4}2^n$ [9]. Another point of comparison is with *unambiguous concatenation* and *unambiguous star*, defined by restricting the arguments, so that each string has a unique representation; classical operations and GF(2)-operations are two incomparable generalizations of the unambiguous operations. Unambiguous concatenation has state complexity $m2^{n-1}-2^{n-2}$ [4], whereas the state complexity of the unambiguous star is $\frac{3}{8}2^n+1$ [6]. The state complexity of unambiguous, classical and GF(2)-variants of the three main operations on formal languages is compared in Table 1. All results refer to the case of DFA.

The goal of this paper is to investigate the state complexity of the GF(2)-operations in the case of a unary alphabet [1]. In general, unary state complexity is substantially different from the case of multiple-symbol alphabets. The trade-offs between different types of automata over a unary alphabet were studied by Chrobak [3], Mereghetti and Pighizzini [10], Geffert et al. [5], Kunc and Okhotin [7], Okhotin [11], and others. The state complexity of operations on unary DFA was first investigated by Yu et al. [13], who proved that concatenation is representable with mn states, and this bound is tight for relatively prime m and n; the state complexity of star on unary languages is $(n-1)^2 + 1$.

How do the GF(2)-operations stand in comparison? For the GF(2)-concatenation of unary languages recognized by DFA with m and n states, the results generally resemble the classical case: it shall be established in Section 2 that $2mn$ states are sufficient, and for relatively prime m, n, this number of states is necessary. On the other hand, the case of GF(2)-inverse of a unary language is substantially different from the classical case; the state complexity turns out to be $2^{n-1}+1$, which is established in Sections 3–6 by determining a connection between the states of a DFA recognizing a GF(2)-inverse and the coefficients of a certain associated sequence of polynomials over GF(2).

2 GF(2)-concatenation

As usual, a DFA is defined as a quintuple $(\Sigma, Q, q_0, \delta, F)$, where Σ is the input alphabet and Q is a finite set of states, with initial state $q_0 \in Q$, transition function $\delta\colon Q \times \Sigma \to Q$ and accepting states $F \subseteq Q$. This paper considers DFA over a unary alphabet $\Sigma = \{a\}$, where the transition function defines a sequence of states $q_0, q_1, \ldots$, with $q_{i+1} = \delta(q_i, a)$. Let j be the least number with $q_j = q_i$ for some $i < j$. This is the point where the automaton starts to behave periodically; the states $q_0, \ldots, q_{i-1}$ are called the *tail*, and the periodic part $q_i, \ldots, q_{j-1}$ is called the *cycle*. If the tail is empty, the automaton is called *cyclic*.

The known construction for GF(2)-concatenation of two given DFA, $\mathcal{A} = (\Sigma, P, p_0, \eta, E)$ and $\mathcal{B} = (\Sigma, Q, q_0, \delta, F)$, works as follows [1]. The language $L(\mathcal{A}) \odot L(\mathcal{B})$ is recognized by a DFA $\mathcal{C}$ with the set of states $P \times 2^Q$. In a state (p, S), with $p \in P$ and $S \subseteq Q$, the automaton simulates the computation of $\mathcal{A}$ in the first component p, while S is the set of all states reached *an odd number of times* in the ongoing simulated computations of $\mathcal{B}$. The initial state of $\mathcal{C}$ is $(p_0, \{q_0\})$ if $\varepsilon \in L(\mathcal{A})$ and $(p_0, \varnothing)$ otherwise. Its transition function, $\pi\colon (P \times 2^Q) \times \Sigma \to P \times 2^Q$, is defined on a pair (p, S) as follows. Each currently simulated computation of $\mathcal{B}$ is continued, represented by the following set S'.

$$S' = \{\, q' \mid \text{the number of states } q \in S, \text{ with } q' = \delta(q, a), \text{is odd} \,\}$$

If the simulated automaton $\mathcal{A}$ passes through an accepting state, then the transition $\pi((p, S), a)$ also adds a new computation to the set S'.

$$\pi((p, S), a) = \begin{cases} (\eta(p, a), S'), & \text{if } \eta(p, a) \notin E \\ (\eta(p, a), S' \triangle \{q_0\}), & \text{if } \eta(p, a) \in E \end{cases}$$

a state (p, S) is marked as accepting, if S contains an odd number of accepting states of $\mathcal{B}$.

$$F' = \left\{\, (p, S) \;\middle|\; |S \cap F| \text{ is odd} \,\right\}$$

This completes the known construction, which is valid for every alphabet.

In the case of a unary alphabet, the state complexity of GF(2)-concatenation on two unary DFA is first investigated in the special case of both automata being cyclic, each with a unique accepting state. Under these restrictions, the state complexity depends only on the number of states in the given automata.

Lemma 1. *Let $\mathcal{A}$ and $\mathcal{B}$ be two minimal cyclic DFA over a unary alphabet, one with the cycle of length m and the other with the cycle of length n, and each with a single accepting state. Then, the GF(2)-concatenation $L(\mathcal{A}) \odot L(\mathcal{B})$ is recognized by a cyclic DFA with period of length $\frac{2mn}{\gcd(m,n)}$, and this is the minimal DFA for this language.*

Proof. Let $\mathcal{A} = (\{a\}, P, 0, \eta, E)$, with $P = \{0, \ldots, m-1\}$, $\eta(i, a) = i + 1 \bmod m$ for all $i \in P$, and $E = \{e\}$. Similarly, let $\mathcal{B} = (\{a\}, Q, 0, \delta, F)$, with $Q = \{0, \ldots, n-1\}$, $\delta(i, a) = i + 1 \bmod n$ for all $i \in Q$, and $F = \{f\}$.

Let $C = (\Sigma, Q', q_0, \pi, F')$ be the DFA recognizing the GF(2)-concatenation $L(\mathcal{A}) \odot L(\mathcal{B})$, with the set of states $Q' = \{\, (p, S) \mid p \in P,\ S \subseteq Q \,\}$, and with accepting states $F' = \{\, (p, S) \mid p \in P,\ f \in S \,\}$.

Claim. For all $p \in P$ and $k \in \{1, \ldots, \frac{n}{\gcd(m,n)} - 1\}$, there are exactly two reachable subsets $S_1, S_2 \subseteq Q$ with $|S_1| = |S_2| = k$ and $\pi((p, S_1), a) \neq \pi((p, S_2), a)$. For $k = 0$ or $k = \frac{n}{\gcd(m,n)}$, there is a unique reachable subset (p, S) with $(p, S) \in Q'$ and $|S| = k$. Accordingly, Q' contains $\frac{2mn}{\gcd(m,n)}$ reachable states.

Assume that $e \neq 0$. After reading a^e, the automaton reaches the state $(e, \{0\})$. From this point on, consider the states reached by the automaton after reading repetitive blocks a^m. The state in the first component is rejecting until the last symbol in the block, and hence the states in both components are cyclically shifted by m, until the accepting state e reappears in the first component. At the last step, a new state 0 is added to the second component, while all pre-existent states in the second component have been shifted by $m \pmod n$. Altogether, the following states are visited.

$$\begin{aligned}
\pi((0, \varnothing), a^e) &= (e, \{0\}) \\
\pi((e, \{0\}), a^m) &= (e, \{0, k_1\}), && \text{where } k_1 \equiv m \pmod n \\
\pi((e, \{0, k_1\}), a^m) &= (e, \{0, k_1, k_2\}) && \text{where } k_2 \equiv 2m \pmod n, \text{ etc.}
\end{aligned}$$

The subset continues to grow until some j-th step, with $\delta(k_j, a^m) = 0$. This means that $(j+1)m \equiv 0 \pmod n$. Since j is the least such number, it must be $j = \frac{n}{\gcd(m,n)} - 1$. Therefore, $\pi((e, \varnothing), a^{jm}) = (e, \{0, k_1, k_2, \ldots k_j\})$.

From this point on, consider further computations upon reading repetitive blocks a^m. The states in the second component keep cyclically shifting, and the states 0 added in the end of each block cancel out these states in the same order as they were added.

$$\begin{aligned}
\pi((e, \{0, k_1, k_2, \ldots k_j\}), a^m) &= (e, \{k_1, k_2, \ldots k_j\}) \\
\pi((e, \{k_1, k_2, \ldots k_j\}), a^m) &= (e, \{k_2, k_3, \ldots k_j\}), && \text{etc.}
\end{aligned}$$

In the end, $\pi((e, \{0, k_1, k_2, \ldots k_j\}), a^{jm}) = (e, \varnothing)$. Finally, after reading a^{m-e}, the automaton returns to its initial state.

$$\pi((e, \varnothing), a^{m-e}) = (0, \varnothing)$$

Overall, the automaton for the GF(2)-concatenation is a cycle on $\frac{2mn}{\gcd(m,n)}$ states.

Claim. The period $\frac{2mn}{\gcd(m,n)}$ is minimal.

The idea of the argument is that $\mathcal{C}$ has a block of $m+n$ consecutive rejecting states, and that this block occurs only once in the automaton. □

Theorem 1. *Let $\mathcal{A}$ and $\mathcal{B}$ be any two DFA over a unary alphabet, with m and with n states, respectively. Then, the GF(2)-concatenation $L(\mathcal{A}) \odot L(\mathcal{B})$ is recognized by a DFA with $2mn$ states.*

For relatively prime m and n, this number of states is necessary in the worst case.

Proof (a sketch). Let $\mathcal{A}$ have the set of states $P = \{0, 1, \ldots, m-1\}$, with accepting states $E \subseteq P$. For each state $i \in P$, define a DFA $\mathcal{A}_i$ by setting i in $\mathcal{A}$ as the only accepting state. Then, $L(\mathcal{A}) = \bigcup_{i \in E} L(\mathcal{A}_i)$, and the union is disjoint. Let the language be periodic starting from k, with period $m-k$.

Similarly, let the set of states of $\mathcal{B}$ be $Q = \{0, 1, \ldots, n-1\}$, with accepting states $F \subseteq Q$. Let $\mathcal{B}_j$ be $\mathcal{B}$ with j as the only accepting state, so that $L(\mathcal{B}) = \bigcup_{i \in F} L(\mathcal{B}_i)$. Let the periodic part begin at ℓ, with period $n-\ell$.

Then, the desired GF(2)-concatenation can be represented as follows.

$$L(\mathcal{A}) \odot L(\mathcal{B}) = \Big(\bigcup_{i \in E} L(\mathcal{A}_i) \Big) \odot \Big(\bigcup_{j \in F} L(\mathcal{B}_j) \Big) = \Big(\bigtriangleup_{i \in E} L(\mathcal{A}_i) \Big) \odot \Big(\bigtriangleup_{j \in F} L(\mathcal{B}_j) \Big) = \\ = \bigtriangleup_{\substack{i \in E \\ j \in F}} L(\mathcal{A}_i) \odot L(\mathcal{B}_j)$$

Each of these $|E| \cdot |F|$ languages is periodic beginning from $k+\ell-1$, with period $2(m-k)(n-\ell)$; the proof is omitted due to space constraints. These languages are then joined into a single automaton with at most $2mn$ states.

For the lower bound, Lemma 1 with relatively prime m, n provides the desired witness languages. □

3 Automaton for GF(2)-inverse

With respect to GF(2)-concatenation, every language L containing the empty string is *invertible*, in the sense that there is a language L^{-1} satisfying $L \odot L^{-1} = L^{-1} \odot L = \{\varepsilon\}$. The *GF(2)-inverse* operation, $f(L) = L^{-1}$, preserves regularity, and its state complexity is $2^n + 1$ [1].

Theorem A (Bakinova et al. [1, Thm. 2]) *For every language L over an alphabet Σ, with $\varepsilon \in L$, a string $w \in \Sigma^*$ is in L^{-1} if and only if it has an odd number of representations of the form $w = w_1 w_2 \ldots w_k$, with $k \geqslant 0$ and $w_1, \ldots, w_k \in L \setminus \{\varepsilon\}$.*

As proved by Bakinova et al. [1], for every n-state DFA $\mathcal{A} = (\{a\}, Q, q_0, \delta, F)$, with $\varepsilon \in L(\mathcal{A})$, the language $L(\mathcal{A})^{-1}$ is recognized by a DFA $\mathcal{C} = (\{a\}, 2^Q \cup \{q_0'\}, q_0', \delta', F')$ defined as follows. The states of $\mathcal{C}$ are all subsets of Q and a new initial state q_0', Its transition function is $\delta' \colon (2^Q \cup \{q_0'\}) \times \{a\} \to 2^Q \cup \{q_0'\}$. The transition in the state q_0' produces a singleton state corresponding to a single computation of $\mathcal{A}$.

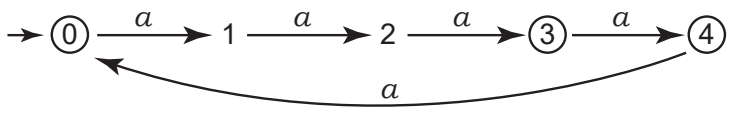
$$\delta'(q_0', a) = \{\delta(q_0, a)\}$$

In a state $S \subseteq Q$, first let $S' = \{\, q \mid \#\text{ of states } p \in S \text{ with } \delta(p, a) = q \text{ is odd} \,\}$. Then the transition is defined as follows.

$$\delta'(S, a) = \begin{cases} S', & \text{if } |S \cap F| \text{ is even} \\ S' \mathbin{\triangle} \delta(q_0, a), & \text{if } |S \cap F| \text{ is odd} \end{cases}$$

The set of accepting states is $F' = \left\{\, S \;\middle|\; |S \cap F| \text{ is odd} \,\right\} \cup \{q_0'\}$.

Example 1. Consider the following 5-state unary DFA.



The DFA for its inverse, constructed by the above method, is shown in Figure 1. It has a cycle of length 15 and a tail of length 2 (along with 16 unreachable states).

In the general case, the DFA for the GF(2)-inverse $L(\mathcal{A})^{-1}$ of an n-state DFA $\mathcal{A}$ has $2^n + 1$ states, and it is known that this number is necessary in the worst case, for alphabets with at least three symbols [1]. It turns out that in the unary case it is always sufficient to use $2^{n-1} + 1$ states.

This upper bound is easy to establish for non-cyclic automata.

Lemma 2. *Let $\mathcal{A} = (\{a\}, Q, 0, \delta, F)$ be an n-state non-cyclic DFA with $0 \in F$. Then, the DFA for the GF(2)-inverse $L(\mathcal{A})^{-1}$ constructed as above, has at most $2^{n-1} + 1$ reachable states.*

Proof. Indeed, no subset containing the state 0 is ever reached, since this state is not reachable by any transitions. □

For cyclic automata $\mathcal{A}$, a deeper analysis of the automaton for its inverse is needed, since the set of unreachable states is harder to specify. The first result to be established is the following dependence between the membership of individual states in the subsets.

Lemma 3. *Let $\mathcal{A} = (\{a\}, Q, 0, \delta, F)$ be a cyclic DFA with $Q = \{0, \ldots, n-1\}$, $\delta(i, a) = i + 1 \bmod n$ for all i, and $0 \in F$. Let $\mathcal{C} = (\{a\}, 2^Q \cup \{q_0'\}, q_0', \delta', F')$*

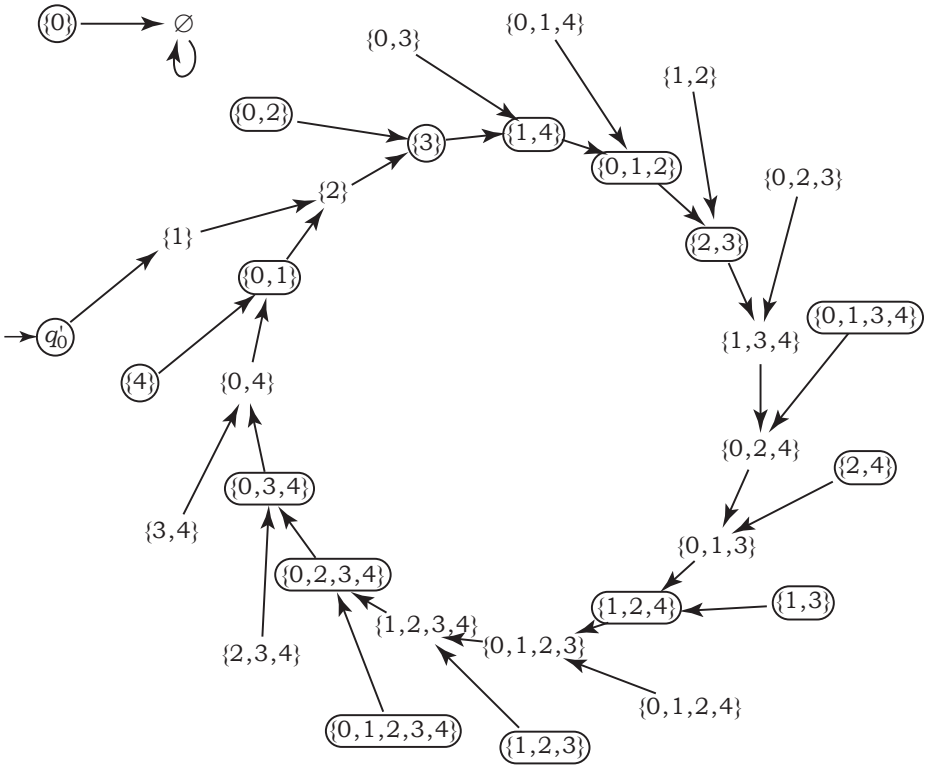


Fig. 1. DFA for the GF(2)-inverse of the language in Example 1.

be the DFA recognizing the GF(2)-inverse of $L(\mathcal{A})$, defined as above. For each $i \geqslant 1$, let $S_i \subseteq Q$ be the state of $\mathcal{C}$ reached upon reading the string a^i.

Denote the membership of the j-th state in S_i by a Boolean value $S_i^j \in \{0,1\}$, with $S_i^j = 1$ if $j \in S_i$, and $S_i^j = 0$ otherwise. Then, the membership of state 1 in each set S_i is determined by the set S_{i-1} by the following formula.

$$S_i^1 = \sum_{f \in F \setminus \{0\}} S_{i-1}^f \qquad (\textit{for } i \geqslant 2)$$

Furthermore, the membership of 0 in S_i depends on its membership in the previous $n-1$ states as follows.

$$S_i^0 = \sum_{f \in F \setminus \{0\}} S_{i-f}^0 \qquad (\textit{for } i \geqslant n+1)$$

Proof (a sketch). The formula for S_i^1 is directly inferred from the definition of automata. The second formula is inferred from this one using the following two observations: first, 0 is in S_i if and only if 1 is in $S_{i-(n-1)}$; second, a state f is in S_{i-n} if and only if 0 is in S_{i-f}. □

4 Polynomials for GF(2)-inverse

The subsets reached by the automaton for the GF(2)-inverse of a cyclic language L have a useful characterization in terms of certain polynomials over GF(2). First, based on the automaton for L, a sequence of polynomials shall be constructed, and then the leading coefficients of these polynomials shall correspond to the membership of state 0 in the subsets of the automaton for L^{-1}.

A few definitions are due. Let $f(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ be a polynomial of degree $n-1$ over GF(2), with $a_0, \ldots, a_{n-1} \in \{0,1\}$ and with $a_{n-1} = 1$.

For every $i \geqslant 0$, let $p_i(x)$ be the polynomial obtained by taking x^i modulo $f(x)$; this is a polynomial of degree at most $n-2$. The polynomials $p_i(x)$ form a sequence, in which the first term is $p_0(x) = 1$, and every succeeding term is obtained from the previous term as follows: if a polynomial $p_i(x)$ does not contain the monomial x^{n-2}, then the next term is $p_{i+1}(x) = x \cdot p_i(x)$; and if there is a monomial x^{n-2} in $p_i(x)$, then the next term is $p_{i+1}(x) = x \cdot p_i(x) + f(x)$; in other words, x^{n-1} is replaced with $a_{n-2}x^{n-2} \ldots + a_1 x + a_0$.

The first n terms in the sequence $\{x^i \bmod f(x)\}_{i=0}$ are 1, x, x^2, …, x^{n-2}, $x^{n-1} + f(x)$. The form of the subsequent terms non-trivially depends on f.

A polynomial $f(x)$ of degree $n-1$ is called *primitive*, if this sequence contains all $2^{n-1} - 1$ non-zero polynomials. For each $n \geqslant 2$, primitive polynomials are known to exist.

For all $i \geqslant 0$ and $j \in \{0, \ldots, n-2\}$, let $b_{i,j} \in \{0,1\}$ be the coefficient at the monomial x^j in $p_i(x)$. The coefficient at the term x^{n-2} in $p_i(x)$ depends on the coefficients at the same term in the preceding $n-1$ polynomials in the sequence.

Lemma 4. *For every $i \geqslant n-1$, the coefficient $b_{i,n-2}$ depends on the earlier coefficients as follows, with all arithmetic in GF(2).*

$$b_{i,n-2} = \sum_{k=0}^{n-2} b_{i-(n-1-k),\, n-2} a_k$$

Let $\mathcal{A} = (\{a\}, Q, 0, \delta, F)$ be a cyclic unary DFA with the set of states $Q = \{0, \ldots, n-1\}$ and with $0 \in F$. The corresponding polynomial over GF(2) is defined as $f(x) = \sum_{j \in F} x^{n-1-j}$. In the sequence of polynomials x^i modulo $f(x)$, let $b_{i,n-2}$ be the coefficient at x^{n-2} in the i-th polynomial. Then, by Lemma 4, each coefficient is expressed through the preceding $n-1$ coefficients as follows.

$$b_{i,n-2} = \sum_{k=0}^{n-2} \left(b_{i-(n-1-k),n-2} \cdot (n-1-k \stackrel{?}{\in} F) \right) = \sum_{j \in F \setminus \{0\}} b_{i-j,n-2}$$

This is the same recurrent formula as in Lemma 3.

Example 2 (continued from Example 1). For the 5-state cyclic automaton with accepting states $F = \{0,3,4\}$, the corresponding polynomial is $f(x) = x^4 + x + 1$. Then, the sequence x^i modulo $f(x)$ begins with the following polynomials. $p_0(x) = 1$, $p_1(x) = x$, $p_2(x) = x^2$, $p_3(x) = x^3$, $p_4(x) = x+1$, $p_5(x) = x^2 + x$, etc.

The following table puts the subsets reachable by the automaton alongside the polynomials in this sequence. The acceptance status of each subset is provided for reference.

i	S_{i+2}	$p_i(x)$	$a^{i+2} \stackrel{?}{\in} L^{-1}$
0	$\{2\}$	1	$-$
1	$\{3\}$	x	$+$
2	$\{1,4\}$	x^2	$+$
3	$\{\mathbf{0},1,2\}$	$\boldsymbol{x^3}$	$+$
4	$\{2,3\}$	$x+1$	$+$
5	$\{1,3,4\}$	x^2+x	$-$
6	$\{\mathbf{0},2,4\}$	$\boldsymbol{x^3}+x^2$	$-$
7	$\{\mathbf{0},1,3\}$	$\boldsymbol{x^3}+x+1$	$-$
8	$\{1,2,4\}$	x^2+1	$+$
9	$\{\mathbf{0},1,2,3\}$	$\boldsymbol{x^3}+x$	$-$
10	$\{1,2,3,4\}$	x^2+x+1	$-$
11	$\{\mathbf{0},2,3,4\}$	$\boldsymbol{x^3}+x^2+x$	$+$
12	$\{\mathbf{0},3,4\}$	$\boldsymbol{x^3}+x^2+x+1$	$+$
13	$\{\mathbf{0},4\}$	$\boldsymbol{x^3}+x^2+1$	$-$
14	$\{\mathbf{0},1\}$	$\boldsymbol{x^3}+1$	$+$

For every i, the state 0 is in the subset S_{i+2} if and only if the polynomial contains the term x^3. This is not a coincidence, and this correspondence shall now be established in the general case.

5 Upper bound for the GF(2)-inverse

The following two binary sequences turn out to be identical. First, there is the sequence $\{S_i^0\}$ representing the membership of the state 0 in the subsets reached by the automaton for the GF(2)-inverse. The other sequence is the sequence $\{b_{i,n-2}\}$ of coefficients at x^{n-2}.

Lemma 5. *Let $\mathcal{A} = (\{a\}, Q, 0, \delta, F)$ be a cyclic DFA with $Q = \{0, \ldots, n-1\}$ and $0 \in F$. For each $i \geqslant 1$, let $S_i \subseteq Q$ be the state of the automaton for the inverse given in Section 3, reached upon reading the string a^i. For each $i \geqslant 1$, let $p_i(x)$ be x^i taken modulo $f(x) = \sum_{j \in F} x^{n-1-j}$. Then, for every $i \geqslant 0$, the state 0 is in S_{i+2} if and only if the monomial x^{n-2} is in $p_i(x)$.*

The proof is by induction on i: the base cases are $i \in \{0, 1, \ldots, n-2\}$, on which the sequences coincide. The induction step follows by Lemmata 3 and 4, since both sequences are defined by the same formulae on the same data.

Lemma 6. *Assume that the sequence $\{S_i^0\}_{i=0}^{\infty}$ has period p beginning at ℓ, in the sense that $S_i^0 = S_{i+p}^0$ for all $i \geqslant \ell$. Then, the sequence of states $\{S_i\}_{i=0}^{\infty}$ has period p beginning at ℓ.*

Lemma 7. *Let $\mathcal{A} = (\{a\}, Q, 0, \delta, F)$ be an n-state cyclic DFA with $0 \in F$. Then, the DFA for the GF(2)-inverse $L(\mathcal{A})^{-1}$, constructed as in Section 3, has at most $2^{n-1}+1$ reachable states.*

Proof (a sketch). The sequence of polynomials x^i modulo $f(x) = \sum_{j \in F} x^{n-1-j}$ contains at most $2^{n-1}-1$ distinct polynomials. By Lemma 5, this sequence coincides with the sequence of state 0, and then, by Lemma 6, the sequence of subsets has the same total length of the tail and the period. This gives the desired upper bound on the number of states. □

6 Lower bound for the GF(2)-inverse

The lower bound shall be established using cyclic witness languages. The following property of GF(2)-inverses of cyclic unary languages comes useful.

Lemma 8. *Let $L \subseteq a^*$, with $\varepsilon \in L$, be a unary language recognized by an n-state cyclic DFA. Then, its GF(2)-inverse L^{-1} contains a string a^ℓ, with $\ell \geqslant n+1$, if and only if the number of representations $a^\ell = a^{\ell-j}a^j$, with $a^{\ell-j} \in L^{-1}$, $a^j \in L \setminus \{\varepsilon\}$ and $j < n$, is odd.*

The proof is by establishing the equivalence with the condition in Theorem A.

For any language $L \subseteq a^*$, with $\varepsilon \in L$, let $\alpha_i = 1$ if $a^i \in L^{-1}$, and $\alpha_i = 0$ otherwise. Then the condition in Lemma 8 can be written down as the following formula.

Lemma 9. *Let $L \subseteq a^*$ be a language, with $\varepsilon \in L$, recognized by an n-state cyclic DFA. Then, $\alpha_i = \sum_{j \in F \setminus \{0\}} \alpha_{i-j}$ for $i \geqslant n+1$.*

As in Section 4, let $f(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ be a primitive polynomial over GF(2), with $a_{n-1} = a_0 = 1$ (primitive polynomials of any degree over GF(2) are known to exist). For every $i \geqslant 0$, let $p_i(x) = b_{i,n-2}x^{n-2} + \ldots + b_{i,1}x + b_{i,0}$ be x^i modulo $f(x)$. Since f is primitive, by definition, all polynomials $p_0(x), \ldots, p_{2^{n-1}-2}$ are pairwise distinct, and then $p_{2^{n-1}-1} = p_0(x) = 1$.

It turns out that the sequence of coefficients at x^{n-2} has the same period as the sequence of full polynomials.

Lemma 10. *The minimal period of the sequence $\{b_{i,n-2}\}_{i=0}^{\infty}$ is $2^{n-1}-1$.*

Lemma 11. *The sequence $\{b_{i,n-2}\}_{i=0}^{\infty}$ contains all binary substrings of length $n-1$, except for $(0, \ldots, 0)$.*

A cyclic automaton $\mathcal{A}_f$ corresponding to this primitive polynomial $f(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ is defined as $\mathcal{A}_f = (\{a\}, Q, 0, \delta, F)$, with $Q = \{0, \ldots, n-1\}$, $\delta(i, a) = i + 1 \bmod n$ for all i, and $F = \{\, n-1-i \mid a_i = 1 \,\}$ (by the same principle as in Lemma 5).

Let $L = L(\mathcal{A}_f)$, and consider the sequence $\{\alpha_i\}_{i=0}^{\infty}$ defined as above. The goal is to prove that $(\alpha_2, \ldots, \alpha_n) \neq (0, \ldots, 0)$.

	Sum	Concatenation		Star	
Unambiguous	$(\uplus)$ $\leqslant \frac{1}{2}mn$ [6]	(UNAMB$\cdot$)	$m+n-1$ [6]	(UNAMB$*$)	$n-2$ [6]
Classical	$(\cup)$ $\leqslant mn$ [12]	$(\cdot)$	$\leqslant mn$ [13]	$(*)$	$(n-1)^2+1$ [13]
GF(2)	$(\triangle)$ mn	$(\odot)$	$\leqslant 2mn$	$(^{-1})$	$2^{n-1}+1$

Table 2. State complexity of unambiguous, classical and GF(2)-variants of sum, concatenation and star: the case of a unary alphabet.

Lemma 12. *Let L be a unary language with $\varepsilon \in L$ and $L \neq \{\varepsilon\}, a^*$, which is recognized by a DFA with n states. Then, the inverse L^{-1} contains a string of length between 2 and n.*

Since the sequence $\{\alpha_i\}_{i=2}^{\infty}$ begins with something other than $n-1$ zeroes, by Lemma 10, the binary substring $(\alpha_2, \ldots, \alpha_n)$ occurs somewhere in the sequence $\{b_{i,n-2}\}_{i=0}^{\infty}$. By Lemma 9, the rest of the terms of the sequence $\{\alpha_i\}$ are defined by the same formula as the sequence $\{b_{i,n-2}\}$, which makes the binary strings $\alpha_2, \ldots, \alpha_{2^{n-1}}$ and $b_{0,n-2}, \ldots, b_{2^{n-1}-2}$ identical up to a cyclic shift. In particular, the period of the sequence $\{\alpha_i\}_{i=2}^{\infty}$ is $2^{n-1}-1$.

It remains to determine the length of the tail. Since the construction in Section 3 produces $2^{n-1}+1$ states, the length of the tail is at most 2. It turns out that it cannot be shortened, because the strings a and $a^{2^{n-1}}-1$ have different membership status.

Lemma 13. *$\alpha_1 \neq \alpha_{2^{n-1}+1}$, and therefore the length of the tail is 2.*

The following theorem has thus been established.

Theorem 2. *For every $n \geqslant 2$, there exists a language L, with $\varepsilon \in L$, recognized by n-state unary cyclic DFA, for which the minimal DFA recognizing its GF(2)-inverse L^{-1} has $2^{n-1}+1$ states.*

7 Future work

The results of this paper are summarized and compared to related results in Table 2.

A problem proposed for future research is to determine the number of states in NFA needed to represent these operations. There are two different modes of nondeterminism involved: existential nondeterminism in NFA and parity nondeterminism in both GF(2)-operations. Intuitively, one kind of nondeterminism cannot help implementing another kind, and the following straightforward construction might actually turn out to be the best possible: first, determinize the arguments, with a blow-up of the order $e^{(1+o(1))\sqrt{n \ln n}}$ [3]; and then, apply the constructions for deterministic automata presented in this paper. Could this construction be substantially improved upon?

References

1. E. Bakinova, A. Basharin, I. Batmanov, K. Lyubort, A. Okhotin, E. Sazhneva, "Formal languages over GF(2)", *Language and Automata Theory and Applications* (LATA 2018, Bar-Ilan near Tel Aviv, Israel, 9–11 April 2018), LNCS 10792, 68–79.
2. J. A. Brzozowski, "Quotient complexity of regular languages", *Journal of Automata, Languages and Combinatorics*, 15:1/2 (2010), 71–89.
3. M. Chrobak, "Finite automata and unary languages", *Theoretical Computer Science*, 47 (1986), 149–158. Errata: 302 (2003), 497–498.
4. M. Daley, M. Domaratzki, K. Salomaa, "Orthogonal concatenation: Language equations and state complexity", *Journal of Universal Computer Science*, 16:5 (2010), 653-675.
5. V. Geffert, C. Mereghetti, G. Pighizzini, "Converting two-way nondeterministic unary automata into simpler automata", *Theoretical Computer Science*, 295:1–3 (2003), 189–203.
6. G. Jirásková, A. Okhotin, "State complexity of unambiguous operations on deterministic finite automata", *Descriptional Complexity of Formal Systems* (DCFS 2018, Halifax, Canada, 25–27 July 2018), LNCS 10952, 188–199.
7. M. Kunc, A. Okhotin, "Describing periodicity in two-way deterministic finite automata using transformation semigroups", *Developments in Language Theory* (DLT 2011, Milan, Italy, 19–22 July 2011), LNCS 6795, 324–336.
8. V. Makarov, A. Okhotin, "On the expressive power of GF(2)-grammars", *SOFSEM 2019: Theory and Practice of Computer Science* (Nový Smokovec, Slovakia, 27–30 January 2019), LNCS 11376, 310–323.
9. A. N. Maslov, "Estimates of the number of states of finite automata", *Soviet Mathematics Doklady*, 11 (1970), 1373–1375.
10. C. Mereghetti, G. Pighizzini, "Optimal simulations between unary automata", *SIAM Journal on Computing*, 30:6 (2001), 1976–1992.
11. A. Okhotin, "Unambiguous finite automata over a unary alphabet", *Information and Computation*, 212 (2012), 15–36.
12. G. Pighizzini, J. Shallit, "Unary language operations, state complexity and Jacobsthal's function", *International Journal of Foundations of Computer Science*, 13:1 (2002), 145–159.
13. S. Yu, Q. Zhuang, K. Salomaa, "The state complexity of some basic operations on regular languages", *Theoretical Computer Science*, 125 (1994), 315–328.