# Ambiguity and Structural Ambiguity of Symmetric Difference NFAs

Brink van der Merwe, Lynette van Zijl, and Jaco Geldenhuys * **

Department of Computer Science
Stellenbosch University, Private Bag X1, 7602 Matieland, SOUTH AFRICA
abvdm@cs.sun.ac.za,lvzijl@sun.ac.za,jaco@cs.sun.ac.za

**Abstract.** Okhotin showed an exponential trade-off in the conversion from nondeterministic unary automata to unambiguous nondeterministic unary automata. We show that the trade-off in the case of unary symmetric difference automata to finitely (structurally) ambiguous unary symmetric difference automata is linear with constant 1 in the number of states. In particular, for every $n$-state unary nondeterministic symmetric difference automaton, there is an equivalent finitely (structurally) ambiguous $n$-state unary symmetric difference nondeterministic automaton. We also consider the complexity of deciding unambiguity for $k$-deterministic finite automata and investigate the interplay between ambiguity and structural ambiguity.

**Keywords:** nondeterminism, ambiguity

## 1 Introduction

Symmetric difference nondeterministic finite automata ($\oplus$-NFAs) are well-suited to the investigation of periodic or cyclic behaviour in regular languages. The succinctness of $\oplus$-NFAs has been investigated in some detail [18], but little work has been done on the language-theoretic properties of $\oplus$-NFAs. In this work, we therefore consider the issue of the *ambiguity* of unary $\oplus$-NFAs.

The ambiguity of a nondeterministic finite automaton (NFA) $M$ refers to the maximum number of different accepting paths of $M$ for all the words in the language accepted by $M$. For example, if $M$ has only one accepting path for any word, then $M$ is unambiguous. Or, $M$ may have no more than $c$ accepting paths for any word (with $c$ a constant), in which case $M$ is finitely ambiguous. Similarly, $M$ may be polynomially or exponentially ambiguous, if the number of accepting paths is at most polynomial or exponential in the number of letters in an accepted word. The ambiguity of NFAs has been extensively investigated (see for example [5, 6, 12]), and recently Okhotin [11] considered the difference between unary NFAs and NFAs with larger alphabets as far as ambiguity is concerned.

* This research was supported by NRF grant #69872.
** This paper is an extended and revised version of [8].

The structural ambiguity [7] of an NFA is determined by not taking the final states of an NFA into account. To be more precise, in order to determine the structural ambiguity of a given NFA $M$ with state set $Q$ and final state set $F \subseteq Q$, we consider the number of acceptance paths for words when replacing $F$ with any subset of the form $\{q\}$, where $q \in Q$. From the definitions it follows that a finitely structurally ambiguous NFA is in fact finitely ambiguous.

In previous work [20], we investigated the ambiguity of $\oplus$-NFAs (as opposed to the ambiguity of traditional NFAs). We showed the existence of families of $\oplus$-NFAs for each ambiguity class, and also considered the descriptional complexity of ambiguous $\oplus$-NFAs. In particular, we showed that for each ambiguity class, there exists an $n$-state binary $\oplus$-NFA for which the minimal equivalent DFA has $O(2^n)$ states. Here we show that for every $n$-state unary $\oplus$-NFA, there is in fact an equivalent finitely structurally ambiguous (and not only finitely ambiguous) $n$-state unary $\oplus$-NFA. This paper extends (some) results from [8] to both the non-unary case and also from results on ambiguity to structural ambiguity.

The remainder of this article is organised as follows: Sect. 2 gives background and definitions, and specifically establishes the algebraic background required in the rest of the paper. In Sect. 3 we prove the state trade-off between unary $\oplus$-NFAs and unary finitely structurally ambiguous $\oplus$-NFAs. The next section notes some related results, such as the ambiguity of $k$-deterministic finite automata. We conclude in Sect. 5.

## 2 Background

$\oplus$-NFAs were defined in [18], and Vuillemin and Gama give an overview of the mathematical basis for $\oplus$-NFAs [17]. We therefore only briefly summarize the necessary definitions and background. We assume that the reader has a basic knowledge of automata theory and formal languages, as for example in [14], and a background in linear algebra, as for example in [13]. Note that symmetric difference is used in the usual set theoretic sense: for any two sets $A$ and $B$, the symmetric difference of $A$ and $B$ is defined as $A \oplus B = (A \cup B) \setminus (A \cap B)$. Also note that for $n$ sets $A_1, \ldots, A_n$, the expression $A_1 \oplus \ldots \oplus A_n$ is equal to the set of elements appearing in an odd number of the sets $A_1, \ldots, A_n$.

### 2.1 Definition of $\oplus$-NFAs

**Definition 1.** *A $\oplus$-NFA $M$ is a 5-tuple $M = (Q, \Sigma, \delta, Q_0, F)$, where $Q$ is the finite non-empty set of states, $\Sigma$ is the finite non-empty input alphabet, $Q_0 \subseteq Q$ is the set of start states, $F \subseteq Q$ is the set of final states and $\delta$ is the transition function such that $\delta : Q \times \Sigma \to 2^Q$.* □

The transition function $\delta$ can be extended to $\delta : 2^Q \times \Sigma \to 2^Q$ by defining

$$\delta(A, a) = \bigoplus_{q \in A} \delta(q, a)$$

for any $a \in \Sigma$ and $A \in 2^Q$. The transition function of the $\oplus$-NFA can be extended to $\delta^* : 2^Q \times \Sigma^* \to 2^Q$ by defining $\delta^*(A, \epsilon) = A$ and $\delta^*(A, aw) = \delta^*(\delta(A, a), w)$ for any $a \in \Sigma$, $w \in \Sigma^*$ and $A \in 2^Q$.

Note that, if the size of the alphabet is one (that is, $|\Sigma| = 1$), then the $\oplus$-NFA is called a *unary* $\oplus$-NFA.

**Definition 2.** *Let $M$ be a $\oplus$-NFA $M = (Q, \Sigma, \delta, Q_0, F)$, and let $w$ be a word in $\Sigma^*$. Then $M$ accepts $w$ if and only if $|F \cap \delta(Q_0, w)| \bmod 2 \neq 0$.* □

For any word $w = w_0 w_1 \dots w_k \in \Sigma^*$ read by a $\oplus$-NFA $M$, there is at least one associated sequence of states $s_0, s_1, \dots, s_{k+1}$ such that $\delta(s_i, w_i) = s_{i+1}$ (unless $\delta(s_i, w_i) = \emptyset$ for some $i$). Such a sequence of states is a *path* for the word $w$. All possible paths on the word $w$ can be combined into an *execution tree* of $M$. A path in the execution tree is an *accepting path* if it ends in a final state. It is important to note that in the execution tree of a $\oplus$-NFA, if there is an even number of occurrences of a state $s_i$ on a given level $j$, then those states cancel out under the symmetric difference operation, and those paths terminate. If an odd number of occurrences of a state $s_i$ occurs on a level $j$, then none of the $s_i$ cancel out and all their paths remain in the execution tree.

In other words, a $\oplus$-NFA accepts a word $w$ by parity — if there is an odd number of accepting paths for $w$ in the execution tree, then $w$ is accepted; else it is rejected. This parity acceptance is motivated by the algebraic foundations of $\oplus$-NFAs, where unary $\oplus$-NFAs correspond to pseudo-noise sequences [4].

*Example 1.* Let $M = (\{q_1, q_2, q_3\}, \{a\}, \delta, \{q_1\}, \{q_3\})$ be a $\oplus$-NFA where $\delta$ is given by

| $\delta$ | $a$ |
|---|---|
| $q_1$ | $\{q_2\}$ |
| $q_2$ | $\{q_3\}$ |
| $q_3$ | $\{q_1, q_3\}$. |

Figure 1 shows a graphical representation of $M$; note that there is no visual difference to a traditional NFA. To find the DFA $M'$ equivalent to $M$, we apply the subset construction using the symmetric difference operation instead of union. The transition function $\delta'$ of $M'$ is

| $\delta'$ | $a$ |
|---|---|
| $[q_1]$ | $[q_2]$ |
| $[q_2]$ | $[q_3]$ |
| $*\,[q_3]$ | $[q_1, q_3]$ |
| $*\,[q_1, q_3]$ | $[q_1, q_2, q_3]$ |
| $*\,[q_1, q_2, q_3]$ | $[q_1, q_2]$ |
| $[q_1, q_2]$ | $[q_2, q_3]$ |
| $*\,[q_2, q_3]$ | $[q_1]$. |

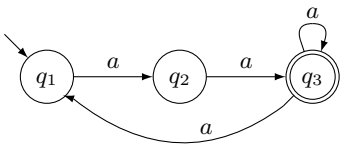


**Fig. 1.** The $\oplus$-NFA for Example 1.

□

Note that each accepting state is marked by a '$*$'.

It is easy to see that any unary $\oplus$-NFA is an autonomous linear machine (see [16, 18] for a formal exposition). As such, one can encode the transition table of a unary $\oplus$-NFA $M$ as a binary matrix $\mathbf{A}$:

$$a_{ji} = \begin{cases} 1 \text{ if } q_j \in \delta(q_i, a) \\ 0 \text{ otherwise,} \end{cases}$$

and successive matrix multiplications in the Galois field $GF(2)$ reflect the subset construction on $M$.

$\mathbf{A}$ is called the *characteristic matrix* of $M$, and $c(x) = \det(\mathbf{A} - x\mathbf{I})$ is known as its characteristic polynomial.

Similarly, we can encode any set of states $X \subseteq Q$ as an $n$-entry row vector $\boldsymbol{v}$ by defining

$$v_i = \begin{cases} 1 \text{ if } q_i \in X \\ 0 \text{ otherwise.} \end{cases}$$

Note that we place an arbitrary but fixed order on the elements of $Q$. We refer to $\boldsymbol{v}$ as the *vector encoding* of $X$, and to $X$ as the *set encoding* of $\boldsymbol{v}$.

If $\boldsymbol{y}$ encodes the initial states of a $\oplus$-NFA $M$, and $\mathbf{A}$ is its characteristic matrix, then $\mathbf{A}\boldsymbol{y}^{\mathrm{T}}$ encodes the states reachable from the initial state after reading one letter, $\mathbf{A}^2\boldsymbol{y}^{\mathrm{T}}$ encodes the states reachable after two letters, and in general $\mathbf{A}^k\boldsymbol{y}^{\mathrm{T}}$ encodes the states reachable after $k$ letters. If $\boldsymbol{z}$ encodes the final states of $M$, then standard linear algebra shows the following:

$$M \text{ accepts } a^k \text{ if and only if } \boldsymbol{z}\mathbf{A}^k\boldsymbol{y}^{\mathrm{T}} = 1.$$

In the general case where we consider also non-unary symmetric difference automata, one can associate a matrix $\mathbf{T}_a$ to each symbol $a \in \Sigma$. Then a word $w = w_1 \dots w_n$, with $w_i \in \Sigma$, is accepted if and only if $\boldsymbol{z}\mathbf{T}_{w_1} \dots \mathbf{T}_{w_n}\boldsymbol{y}^{\mathrm{T}} = 1$.

*Example 2.* Consider the $\oplus$-NFA in Example 1. Its characteristic matrix is

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

and its characteristic polynomial is $c(x) = x^3 + x^2 + 1$. Interested readers may note that $c(x)$ is a primitive polynomial in $GF(2)$. If we encode the start state as a column vector $\boldsymbol{y}^T$, with only the first component of $\boldsymbol{y}$ equal to one, and compute $\mathbf{A}^k\boldsymbol{y}^T$, we end up with the $k$-th entry in the on-the-fly subset construction on $M$. For example, with the start state $q_1$ encoded as $\boldsymbol{y} = [\,1\ 0\ 0\,]$, we see that $\mathbf{A}^4$ is given by

$$\mathbf{A}^4 = \begin{bmatrix} 1\,1\,0 \\ 1\,1\,1 \\ 1\,0\,1 \end{bmatrix},$$

and hence $\mathbf{A}^4\boldsymbol{y}^T$ is given by $[\,1\ 1\ 1\,]$. This corresponds to the state $[\,q_1, q_2, q_3\,]$, which is reached after four applications of the subset construction on $M$. Similarly, $\mathbf{A}^6\boldsymbol{y}^T$ is given by $[\,0\ 1\ 1\,]$, which corresponds to $[\,q_2, q_3\,]$. □

### 2.2 Analysis of ⊕-NFA Behaviour

In [18] we formally showed that the state behaviour of a unary ⊕-NFA is the same as that of a linear feedback shift register (LFSR). The similarity is intuitively straightforward, as an LFSR is a linear machine over $GF(2)$, and we can encode a unary ⊕-NFA as a linear machine over $GF(2)$ as shown above. This correspondence means that we can exploit the wealth of literature on LFSRs to analyse the behaviour of unary ⊕-NFAs, and in particular their cyclic behaviour (see, for example, [3] or [16]). For ⊕-NFAs in general (whether unary or non-unary), standard techniques in linear algebra is often of use. This includes making a change of basis in order to obtain a more convenient form of transition matrices, initial vectors or final vectors. This is precisely the technique that we use to obtain the next theorem and also Theorem 3 in section 3.

**Theorem 1.** *Let $M = (Q, \Sigma, \delta, Q_0, F)$ be a ⊕-NFA accepting $L$. Then for any non-empty subset of states $X \subseteq Q$ there exists ⊕-NFAs $M'$ and $M''$, both accepting $L$, such that:*

1. *$M' = (Q, \Sigma, \delta', X, F')$, and*
2. *$M'' = (Q, \Sigma, \delta'', Q_0'', X)$.*

*Proof.* Both claims are based on the same principle; we only show the first. Let $\boldsymbol{s}$, $\boldsymbol{f}$, and $\boldsymbol{x}$ be the vector encodings of the sets $Q_0$, $F$, and $X$, respectively, and let $\mathbf{T_a}$ be the matrix corresponding to the transition table of $a \in \Sigma$. Choose a non-singular matrix $\mathbf{P}$ such that $\boldsymbol{x}^T = \mathbf{P}\boldsymbol{s}^{\mathrm{T}}$. (Note that such a matrix $\mathbf{P}$ must exist. We can obtain $\mathbf{P}$ as the product $\mathbf{P_1}\mathbf{P_2}^{-1}$, where $\mathbf{P_1}$ and $\mathbf{P_2}$ are non-singular matrices obtained as follows: Denote by $\boldsymbol{e_1}$ the vector with a 1 in the first component and 0's in all the other components. Let $\mathbf{P_1}$ and $\mathbf{P_2}$ be non-singular matrices such that $\mathbf{P_1}\boldsymbol{e_1}^{\mathrm{T}} = \boldsymbol{x}^{\mathrm{T}}$ and $\mathbf{P_2}\boldsymbol{e_1}^{\mathrm{T}} = \boldsymbol{s}^{\mathrm{T}}$. We construct for example $\mathbf{P_1}$ (and similarly $\mathbf{P_2}$) by setting the first column of $\mathbf{P_1}$ to be equal to $\boldsymbol{x}^{\mathrm{T}}$ and then select the remaining columns in any way such that the column vectors of $\mathbf{P_1}$ are linearly independent.) Now, if $L$ is non-empty, then $Q_0$ and $F$ are non-empty

sets, and thus $\boldsymbol{s}$ and $\boldsymbol{f}$ are non-zero. Also if $L$ is empty, then we may choose either $Q_0$ or $F$ to be non-empty. Let $\boldsymbol{y} = \boldsymbol{f}\mathbf{P}^{-1}$, and let $\mathbf{S_a} = \mathbf{P}\mathbf{T_a}\mathbf{P}^{-1}$ be the matrix corresponding to the transition table of $a \in \Sigma$. Let $F'$ be the set encoding of $\boldsymbol{y}$ and define $\delta'$ by

$q_i \in \delta'(q_j, a)$ if and only if the entry in row $i$ and column $j$ in $\mathbf{S_a}$ equals 1.

Then

$$\begin{aligned}
&M' \text{ accepts } w_1 w_2 \dots w_k, \text{ for } w_i \in \Sigma \\
\Leftrightarrow\ & \boldsymbol{y}\mathbf{S_{w_1}}\mathbf{S_{w_2}}\dots\mathbf{S_{w_k}}\boldsymbol{x}^{\mathrm{T}} = 1 \\
\Leftrightarrow\ & (\boldsymbol{f}\mathbf{P}^{-1})(\mathbf{P}\mathbf{T_{w_1}}\mathbf{P}^{-1})(\mathbf{P}\mathbf{T_{w_2}}\mathbf{P}^{-1})\dots(\mathbf{P}\mathbf{T_{w_k}}\mathbf{P}^{-1})(\mathbf{P}\boldsymbol{s}^{\mathrm{T}}) = 1 \\
\Leftrightarrow\ & \boldsymbol{f}\mathbf{T_{w_1}}\mathbf{T_{w_2}}\dots\mathbf{T_{w_k}}\boldsymbol{s}^{\mathrm{T}} = 1 \\
\Leftrightarrow\ & M \text{ accepts } w_1 w_2 \dots w_k.
\end{aligned}$$

□

In essence, the technique in Theorem 1 makes use of a change in basis. This makes it possible to transform any $\oplus$-NFA into an equivalent automaton while controlling either the choice of start states, final states, or one of the transition matrices of a letter in $\Sigma$.

*Example 3.* Consider again the $\oplus$-NFA $M$ in Example 1. To transform it to an automaton where all of the states are start states, we solve $\boldsymbol{x} = \mathbf{P}\boldsymbol{s}^{\mathrm{T}}$. We know from the definition of $M$ that $\boldsymbol{s} = [1\ 0\ 0]$ and we want $\boldsymbol{x} = [1\ 1\ 1]$. We can take for example

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

and therefore

$$\mathbf{B} = \mathbf{P}\mathbf{A}\mathbf{P}^{-1} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

and the vector encoding for the final states is $\boldsymbol{y} = \boldsymbol{f}\mathbf{P}^{-1} = [0\ 1\ 1]$. The resulting automaton is shown in Figure 2. One can also verify the fact that the language is not changed by the change in basis, by calculating the DFA corresponding to the new $\oplus$-NFA:

| $\delta'$ | $a$ |
|---|---|
| $[\,q_1, q_2, q_3\,]$ | $[\,q_2, q_3\,]$ |
| $[\,q_2, q_3\,]$ | $[\,q_3\,]$ |
| $*\,[\,q_3\,]$ | $[\,q_1, q_2\,]$ |
| $*\,[\,q_1, q_2\,]$ | $[\,q_1, q_3\,]$ |
| $*\,[\,q_1, q_3\,]$ | $[\,q_1\,]$ |
| $[\,q_1\,]$ | $[\,q_2\,]$ |
| $*\,[\,q_2\,]$ | $[\,q_1, q_2, q_3\,]$. |

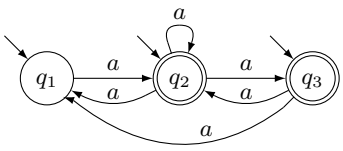


**Fig. 2.** The $\oplus$-NFA from Example 3.

It is easy to check that this DFA is isomorphic to the DFA in Example 1. □

### 2.3 Ambiguity

We briefly state the formal definitions for ambiguity. Although we give the definitions only for $\oplus$-NFAs, exactly the same definitions are applicable to traditional NFAs.

**Definition 3. Unambiguous***: A $\oplus$-NFA is unambiguous if every word in the language is accepted with at most one accepting path.*

**Definition 4. Finitely ambiguous***: A $\oplus$-NFA is finitely ambiguous if every word in the language is accepted with at most $k$ accepting paths, where $k$ is a positive integer.*

**Definition 5. Polynomially ambiguous***: A $\oplus$-NFA is polynomially ambiguous if every word in the language is accepted with at most $k$ accepting paths, where $k$ is bound polynomially in the length of the input word.*

**Definition 6. Exponentially ambiguous***: A $\oplus$-NFA is exponentially ambiguous if every word in the language is accepted with at most $k$ accepting paths, where $k$ is bounded exponentially in the length of the input word.*

Similar to the definitions given in [6] for NFAs, we also define structural ambiguity for $\oplus$-NFAs.

**Definition 7. Structural Ambiguity***: A $\oplus$-NFA $M = (Q, \Sigma, \delta, Q_0, F)$ is structurally unambiguous, finitely structurally ambiguous, polynomially structurally ambiguous or exponentially structurally ambiguous if for all $q \in Q$ the $\oplus$-NFA $M_q = (Q, \Sigma, \delta, Q_0, \{q\})$ is unambiguous, finitely ambiguous, polynomially ambiguous or exponentially ambiguous, respectively.*

Note that a finitely structurally ambiguous, polynomially structurally ambiguous or exponentially structurally ambiguous NFA or $\oplus$-NFA is finitely, polynomially or exponentially ambiguous, respectively.

Ambiguity for a given NFA or $\oplus$-NFA can be visually demonstrated by drawing the execution tree of the corresponding automaton. Given an NFA or $\oplus$-NFA $M$, we assume that the root is on level 0 of the execution tree, and is a start state of $M$. Note that, if $M$ has multiple start states, then one considers the forest of execution trees, where the root of each tree is one of the start states of $M$.

## 3 Ambiguity of Unary $\oplus$-NFAs

It is known that the conversion of a traditional $n$-state NFA to a DFA has an upper bound of $O(2^n)$ states, and this bound is tight [9, 10]. This does not hold in the case of unary NFAs, where the bound is $g(n) + n^2$ states, where $g(n)$ is Landau's function [1]. In the case of unary $n$-state $\oplus$-NFAs, we recall a tight upper bound of $2^n - 1$ for the $\oplus$-NFA to DFA conversion [19].

Okhotin [11] showed that the $g(n) + n^2$ bound also holds for the unary NFA to unary unambiguous NFA trade-off. Surprisingly, the unary $\oplus$-NFA to unary finitely ambiguous $\oplus$-NFA trade-off is simply linear with constant 1 – any unary $n$-state $\oplus$-NFA has an equivalent $n$-state finitely ambiguous $\oplus$-NFA, as we show in detail below.

For any state in a $\oplus$-NFA and $a \in \Sigma$, its *$a$-indegree* denotes the number of transitions on $a$ entering the state in its graphical representation.

**Theorem 2.** *Any $\oplus$-NFA such that each state has $a$-indegree at most two for any $a \in \Sigma$, is finitely structurally ambiguous.*

*Proof.* Let $N$ be any $\oplus$-NFA such that each state has $a$-indegree at most two and let $w = w_1 w_2 \dots w_k$, with $w_i \in \Sigma$. Next we consider the execution forest of $w$. We have an execution tree associated with each initial state in $N$, but it is enough to show that there are finitely many execution paths for a given word in each of the execution trees. Thus we may in fact assume that we have only one initial state in $N$ and hence a single execution tree. First we show by induction that each state appears at most once on a given level in the execution tree. This is certainly true at the root of the tree where we have only the initial state. Assume the result for level $i-1$ in the execution tree and consider level $i$ ($i \leq k$), and let $q$ be any state in $N$. Since the $w_i$-indegree of $q$ is at most two, precisely one of the following three statements hold:

- there is no transition on $w_i$ from a state at level $i-1$ to state $q$, and therefore state $q$ is not present at level $i$;
- there is a transition on $w_i$ from a single state at level $i-1$ to state $q$;
- there are transitions on $w_i$ from precisely two states at level $i-1$ to state $q$, and thus by definition of an $\oplus$-NFA, state $q$ is not present at level $i$.

We thus conclude that each state appears at most once on a given level in the execution tree of $w$. □

*Remark 1.* Let $M$ be a $\oplus$-NFA with a single initial state, such that each state has $a$-indegree at most two for any $a \in \Sigma$. Then the maximum number of accepting paths for any word $w$ can be determined, by using the fact from the previous proof that each state can appear at most once at each level of the execution tree of a word, as follows. First determinize $M$ in order to obtain $N$, and consider each state in $N$ as a subset of states of $M$ in the usual way. Now the maximum number of accepting paths for any word, which is equal to the maximum number of accepting states at any level of the execution tree of any

word, is simply equal to the maximum odd number of accepting states of $M$ appearing in the representation of a state of $N$.

It should also be observed that the converse to the above theorem does not hold. For example, suppose there is a state $q$ in a $\oplus$-NFA $M$ that has indegree three and all other states have indegree two or smaller. Then, if it is not possible to simultaneously be in the three parent states of $q$ while reading a word, $M$ will still be finitely structurally ambiguous.

**Theorem 3.** *Let $M$ be any unary $\oplus$-NFA. Then there exists a finitely structurally ambiguous unary $\oplus$-NFA $N$, accepting the same language as $M$, and also with the same number of states as $M$.*

*Proof.* Let $\mathbf{A}$ be the characteristic matrix of $M$.

First recall from linear algebra that the companion matrix of the monic polynomial $p(x) = c_0 + c_1x + \ldots + c_{n-1}x^{n-1} + x^n$ over $GF(2)$ is the square matrix

$$\mathbf{C}(p) = \begin{bmatrix} 0 & 0 \ldots 0 & c_0 \\ 1 & 0 \ldots 0 & c_1 \\ 0 & 1 \ldots 0 & c_2 \\ \vdots & \vdots \quad \vdots & \vdots \\ 0 & 0 \ldots 1 & c_{n-1} \end{bmatrix}.$$

A classic result from linear algebra (see [13], Theorem 7.14), states that $\mathbf{A}$ has a rational canonical form with entries from $GF(2)$. More precisely, there exists an invertible matrix $\mathbf{Q}$ such that $\mathbf{B} = \mathbf{Q}^{-1}\mathbf{A}\mathbf{Q} = \text{diag}[\mathbf{L}(f_1), \mathbf{L}(f_2), \ldots, \mathbf{L}(f_s)]$, where $\mathbf{L}(f_i)$ is the companion matrix for a monic polynomial $f_i$. We denote by $\text{diag}[\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_s}]$, with the $\mathbf{A}_i$'s square matrices, the matrix with the $\mathbf{A}_i$'s on the diagonal and all other entries equal to 0. In our context, the important property of the matrix $\text{diag}[\mathbf{L}(f_1), \mathbf{L}(f_2), \ldots, \mathbf{L}(f_s)]$ is that the indegree of each state of a $\oplus$-NFA $N$, with characteristic matrix $\text{diag}[\mathbf{L}(f_1), \mathbf{L}(f_2), \ldots, \mathbf{L}(f_s)]$, must be at most two. One can see this by noting that we have at most two non-zero entries in each row of $\text{diag}[\mathbf{L}(f_1), \mathbf{L}(f_2), \ldots, \mathbf{L}(f_s)]$. We thus apply a change of basis (by using $Q$) to $\mathbf{A}$ to obtain the characteristic matrix $\mathbf{B}$ of a $\oplus$-NFA $N$, and also to the initial and accepting vectors of $M$ to obtain the initial and accepting vectors for $N$. Since a change of basis preserves the language accepted by a $\oplus$-NFA, $M$ and $N$ accept the same language. Note that $M$ and $N$ will also have the same number of states, and from Theorem 2 we have that $N$ is finitely structurally ambiguous.

□

## 4 Further Ambiguity Results

First we consider the relationship between the ambiguity of $\oplus$-NFAs accepting a language $L$ and those accepting the complement of $L$, denoted by $\overline{L}$. In particular, we investigate the relationship, in terms of ambiguity, between the $\oplus$-NFAs

$M$ and $M^c$, where $M^c$ is obtained from $M$ by adding a single new state $q^c$ to the states of $M$, with $q^c$ both a start and an accepting state, but not adding any additional transitions except for $\delta(q^c, a) = q^c$ for any $a \in \Sigma$. Observe that $M^c$ accepts the complement of the language accepted by $M$. It is interesting to note that in the case of traditional unary NFAs [11], the state complexity of complementation for unambigous unary finite automata lies between the bounds $\frac{1}{42}n\sqrt{n}$ (for $n \geq 867$) and $e^{\Theta}(\sqrt[3]{n \ln^2 n})$.

**Theorem 4.** *Assume the language $L$ is accepted by an $n$-state $\oplus$-NFA $M$. Then the $(n+1)$ state $\oplus$-NFA $M^c$, accepting $\overline{L}$, is finitely structurally ambiguous if and only if $M$ is finitely structurally ambiguous.*

*Proof.* The proof follows from the following observation. For $w \in \Sigma^*$, denote by $\mathcal{F}_{M,w,n}$ and $\mathcal{F}_{M^c,w,n}$ the set of states at level $n$ of the execution forest of $w$ when using $M$ and $M^c$, respectively. Then $\mathcal{F}_{M^c,w,n} = \mathcal{F}_{M,w,n} \cup \{q^c\}$. Thus the number of accepting states in $\mathcal{F}_{M,w,n}$ and $\mathcal{F}_{M^c,w,n}$ have opposite parity. Therefore $M^c$ accepts the complement of the language accepted by $M$, and the cardinality of $\mathcal{F}_{M^c,w,n}$ is finite or polynomial in $n$ if and only if this is also the case for $\mathcal{F}_{M,w,n}$. □

**Corollary 1.** *Assume the unary language $L$ is accepted by an $n$-state ambiguous $\oplus$-NFA. Then there is an $n+1$-state finitely structurally ambiguous (and thus also finitely ambiguous) $\oplus$-NFA accepting $\overline{L}$.*

*Proof.* Directly from Theorem 3, Theorem 4 and the fact that a finitely structurally ambiguous $\oplus$-NFA is a finitely ambiguous $\oplus$-NFA. □

We next consider the ambiguity of $k$-deterministic $\oplus$-FAs. A $k$-deterministic FA ($k$-DFA) and similarly a $k$-$\oplus$-DFA is a deterministic finite automaton, except that it has multiple initial states. Hence the only nondeterminism in a $k$-DFA or $k$-$\oplus$-DFA occurs in its multiple initial states.

**Definition 8.** *A $k$-DFA (or $k$-$\oplus$-DFA) $M$ is a 5-tuple $M = (Q, \Sigma, \delta, Q_0, F)$, where $Q$ is the finite non-empty set of states, $\Sigma$ is the finite non-empty input alphabet, $Q_0 \subseteq Q$ is the set of start states, $F \subseteq Q$ is the set of final states and $\delta$ is the transition function such that $\delta : Q \times \Sigma \to Q$.*

Note that, as before, the difference between a traditional $k$-DFA and a $k$-$\oplus$-DFA lies in the application of the subset construction to get the equivalent DFA (without multiple initial states).

It is to be expected that a unary $k$-DFA should be either unambiguous or finitely ambiguous, and this is indeed the case both for the traditional $k$-DFA and the $k$-$\oplus$-DFA:

**Theorem 5.** *Any $k$-DFA or $k$-$\oplus$-DFA is finitely ambiguous (finitely structurally ambiguous), with the constant for the finite ambiguity no more than $|Q_0|$.*

*Proof.* The multiple initial states lead to a forest of disconnected execution trees, such that each tree is a single deterministic branch. In the case of $k$-DFAs the ambiguity is determined by the number of possible final states on each level, which is bounded below by zero and above by $|Q_0|$. Note that the number of trees stay constant. Hence, the $k$-DFA is finitely ambiguous.

In the case of $k$-$\oplus$-DFAs, the number of deterministic trees in the forest of disconnected execution trees cannot be more than $|Q_0|$, but may become less if an even number of identical states occur on the same level. The result holds by the same argument as above. □

It is interesting to note, however, that there exists a family $\{M_n\}_{n\geq 2}$ of $k$-DFAs that are finitely ambiguous when considered as $k$-DFAs, but are unambiguous when considered a $k$-$\oplus$-DFAs.

**Theorem 6.** *There exists a family $\{M_n\}_{n\geq 2}$ of unary $k$-DFAs that are finitely ambiguous when considered as $k$-DFAs, but are unambiguous when considered as $k$-$\oplus$-DFAs.*

*Proof.* Let $M_n = (Q, \{a\}, \delta, Q_0, F)$ be defined by $Q = \{q_1, \ldots, q_{n-1}\}$, $Q_0 = Q$, $F = \{q_{n-1}\}$ and $\delta(q_i, a) = q_{i+1}$ for $1 \leq i < n-1$, and $\delta(q_{n-1}, a) = q_{n-2}$.

Consider first the $k$-DFA case. The forest of execution trees contains $n = |Q_0|$ disconnected trees with no branches. Hence, on any level, there are exactly $n$ states. The transition function ensures that each tree has the form $q_i \to q_{i+1} \to \ldots q_{n-2} \to q_{n-1} \to q_{n-2} \to q_{n-1} \ldots$. In other words, each tree consists of a single branch with consecutive states, until the last two states alternate indefinitely. Now, since all the elements in $Q_0$ are distinct, the trees all reach the state $q_{n-1}$ within $n-1$ steps. Hence, from step $n$ onwards, there are at most $|Q_0|$ final states at any level, and hence the $k$-DFA is finitely ambiguous.

In the case of the $k$-$\oplus$-DFA, we note that on level one of the execution tree, there are $n$ distinct states $\{q_1, q_2, \ldots, q_{n-1}\}$, and hence one final state. On level two, states $q_{n-3}$ and $q_{n-1}$ both result in state $q_{n-2}$, and symmetric difference causes the trees with state $q_{n-2}$ to terminate. This process continues with the other branches, until only one tree remains which alternates between states $q_{n-1}$ and $q_{n-2}$, or which ends in an empty set of states. Hence, the $k$-$\oplus$-DFA is unambiguous. □

We note that Stearns and Hunt [15] showed that there exists a polynomial time algorithm to determine whether any given NFA is unambiguous. We show that similarly it can be determined in polynomial time if a $k$-$\oplus$-DFA is unambiguous.

**Theorem 7.** *There is a $k$-th degree polynomial time algorithm to determine whether a $k$-$\oplus$-DFA is unambiguous (or structurally unambiguous).*

*Proof.* Let $M$ be a $k$-$\oplus$-DFA with states $Q$ and accepting states $F \subseteq Q$. We construct a new $\oplus$-NFA $N$ from M by taking $k$ disjoint copies of $M$, with each of the copies the same as $M$, with the exception that we give each copy precisely one of the initial states of $M$. By considering execution forests, it is easy to see

that $M$ and $N$ accept the same language. Also note that each state will appear at most once on a level of the execution forest of a given word, when using $N$. Now determinize $N$ to obtain $N^D$. Note that since $M$ is a $k$-$\oplus$-DFA, we may consider each state of $N^D$ as a subset of $Q^k$ (the set of $k$-dimensional vectors over $Q$), if $M$ is total. If $M$ is not total, we add the new state $q_d$ to $Q$ to obtain $\bar{Q}$, and we consider the states of $N^D$ to be a subset of $\bar{Q}^k$. Although $M$ and $N^D$ accept the same language, it is not immediately clear how to determine if $M$ is unambiguous by using $N^D$. In order to achieve this, we construct $N^D$ in a modified way. While constructing $N^D$ on the fly from $N$, if it is the case that an even number of the $k$ components of a constructed state is labeled with the same state from $Q$, we relabel these components with $q_d$. Once a state of $N^D$, considered as an element from $\bar{Q}^k$, has at a specific component the label $q_d$, all states in $N^D$ reachable from this state will also have $q_d$ at this component. The important aspect to notice about $N^D$, is that while reading a word $w$, the components of the state names of $N^D$, with $q_d$ deleted, specifies precisely the states present (with multiplicity) at the levels of the execution forest of $w$, when using $M$. Also note that $N^D$ has at most $|\bar{Q}|^k$ states, where $|\bar{Q}|$ denotes the cardinality of $\bar{Q}$. From the definition of when a $\oplus$–NFA is unambiguous, it follows that $M$ is unambiguous if and only if each state of the (modified) DFA $N^D$ that contains an odd number of components labeled by elements from $F$, contains in fact only one component labeled by an element from $F$. Similarly, $M$ is structurally unambiguous if and only if each state in the (modified) DFA $N^D$ does not have components labeled by the same element from $Q$. □

## 5 Conclusion and Future Work

We showed that, for any $n$-state unary $\oplus$-NFA, there is an equivalent $n$-state unary $\oplus$-NFA which is finitely ambiguous. This implies that, for unary regular languages, there are no languages which are strictly polynomially or strictly exponentially ambiguous with $\oplus$-NFAs. This is most likely not the case for non-unary languages. It thus remains to investigate ambiguity issues in more detail for non-unary languages.

## References

1. Chrobak, M.: Finite automata and unary languages. Theoretical Computer Science 47(3), 149–158 (1986), erratum appeared as [2]
2. Chrobak, M.: Errata on "Finite automata and unary languages". Theoretical Computer Science 302(1–3), 497–498 (Jun 2003)
3. Dornhoff, L., Hohn, F.: Applied Modern Algebra. Macmillan Publishing Company (1978)
4. Goresky, M., Klapper, A.: Pseudonoise sequences based on algebraic feedback shift registers. IEEE Transactions on Information Theory 52(4), 1649–1662 (Apr 2006)
5. Leung, H.: Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata. SIAM Journal of Computing 27(4), 1073–1082 (Aug 1998)

6. Leung, H.: Descriptional complexity of NFA of different ambiguity. International Journal of Foundations of Computer Science 16(5), 975–984 (Oct 2005)
7. Leung, H.: Structurally unambiguous finite automata. In: Ibarra, O., Yen, H.C. (eds.) Proceedings of the 11th International Conference on the Implementation and Applicatino of Automata. pp. 198–207. LNCS #4094, Springer-Verlag (2006)
8. van der Merwe, B., van Zijl, L., Geldenhuys, J.: Ambiguity of unary symmetric difference nfas. In: Cerone, A. (ed.) Proceedings of the International Colloquium on Theoretical Aspects of Computing. pp. 256–266. LNCS #6916, Springer-Verlag (Sep 2011)
9. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: Proceedings of the 12 Annual IEEE Symposium on Switching and Automata Theory. pp. 188–191 (Oct 1971)
10. Moore, F.R.: On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic and two-way finite automata by deterministic automata. IEEE Transactions on Computing C–20(10), 1211–1219 (Oct 1971)
11. Okhotin, A.: Unambiguous finite automata over a unary alphabet. Information and Computation 212, 15–36 (March 2012)
12. Ravikumar, B., Ibarra, O.H.: Relating the type of ambiguity of finite automata to the succinctness of their representation. SIAM Journal of Computing 18(6), 1263–1282 (Dec 1989)
13. Roman, S.: Advanced Linear Algebra. Springer-Verlag (1992)
14. Sipser, M.: Introduction to the Theory of Computation. International Thomson Publishing (1996)
15. Stearns, R., Hunt, H.: On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. SIAM Journal of Computing 14, 598–611 (1979)
16. Stone, H.: Discrete Mathematical Structures. Science Research Associates (1973)
17. Vuillemin, J., Gama, N.: Efficient equivalence and minimization for nondeterministic XOR automata. Tech. rep., INRIA (Dec 2009)
18. van Zijl, L.: Generalized Nondeterminism and the Succinct Representation of Regular Languages. Ph.D. thesis, Stellenbosch University (Nov 1997)
19. van Zijl, L.: Magic numbers for symmetric difference NFAs. International Journal of Foundations of Computer Science 16(5), 1027–1038 (Oct 2005)
20. van Zijl, L., Geldenhuys, J.: Descriptional complexity of ambiguity in symmetric difference NFAs. Journal of Universal Computer Science (2011), accepted for publication