Descriptional Complexity of Non-Unary Self-Verifying Symmetric Difference Automata

Laurette Marais

Meraka Institute, CSIR and

Dept. of Computer Science, Stellenbosch University

Lynette van Zijl

Dept. of Computer Science, Stellenbosch University

Received (15 January 2018)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

Unary self-verifying symmetric difference automata have a known tight bound of $2^{n-1}-1$ for their state complexity. We now consider the non-unary case and show that, for every $n \geq 2$, there is a regular language $\mathcal{L}_n$ accepted by a non-unary self-verifying symmetric difference nondeterministic automaton with n states, such that its equivalent minimal deterministic finite automaton has 2^{n-1} states. Furthermore, given any SV-XNFA with n states, it is possible, up to isomorphism, to find at most another $|GL(n, \mathbb{Z}_2)| - 1$ equivalent SV-XNFA. Finally, we show that for a certain set of non-unary SV-XNFA, 2^{n-1} is a tight bound on the state complexity.

Keywords: descriptional complexity; symmetric difference automata; self-verifying automata.

1. Introduction

Symmetric difference nondeterministic finite automata (XNFA) are interesting from a state complexity point of view. Determinising XNFA is done via the subset construction as for NFA, but instead of taking the union of sets, the symmetric difference is taken. This means that $2^n - 1$ is an upper bound on the state complexity of XNFA. This has been shown to be a tight bound for unary alphabets [10].

Self-verifying automata (SV-NFA) were described in [1,4,5] as having two kinds of final states: accept states and reject states. Non-final states are called neutral states. It is required that for any word, at least one such a final state is reached, and that only one kind of final state is reached on any path, so that any word is either explicitly accepted or explicitly rejected by the automaton. It was shown in [5] that $e^{\Theta\sqrt{n \ \ln n}}$ is an upper bound for the unary case, but not a tight bound, while in the non-unary case, $g(n)$, where $g(n)$ grows like $3^{\frac{n}{3}}$, is a tight upper bound.

In [7], we extended the notion of self-verification (SV) to XNFA to obtain SV-XNFA. We showed that $2^n - 1$ is not a tight upper bound for SV-XNFA in the case of a unary alphabet. A lower bound of $2^{n-1} - 1$ was established for the unary case, and we showed this to be a tight bound in [6].

In this paper, we now consider the state complexity of SV-XNFA with *non-unary* alphabets. We give an upper bound of $2^n - 1$ and a lower bound of 2^{n-1}.

Furthermore, any XNFA can be transformed into an equivalent XNFA by performing a change of basis operation [9]. We show that this holds also for SV-XNFA, and that for any given SV-XNFA, up to isomorphism, at most another $|GL(n, \mathbb{Z}_2)| - 1$ equivalent SV-XNFA can be found.

The rest of this paper is organised as follows: in Section 2, preliminary definitions are given, while Section 3 gives the algebraic proofs necessary for the analysis of non-unary self-verifying XNFA. The proofs for the bounds on non-unary self-verifying XNFA are then considered in Section 4, followed by the conclusion.

2. Preliminaries

An NFA N is a five-tuple $N = (Q, \Sigma, \delta, Q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to 2^Q$ is a transition function (where 2^Q indicates the power set of Q), $Q_0 \subseteq Q$ is a set of initial states, and $F \subseteq Q$ is the set of final, or acceptance, states.

The transition function can be extended to $\delta : 2^Q \times \Sigma \to 2^Q$ in the following way:

$$\delta(A, \sigma) = \bigcup_{q \in A} \delta(q, \sigma) \ .$$

Note that δ can be extended to strings in the Kleene closure Σ^* of the alphabet. For $w = \sigma_0\sigma_1 \ldots \sigma_k$ and $a \in \Sigma$,

$$\delta(q, \varepsilon) = q, \text{ and } \delta(q, wa) = \delta(\delta(w), a) \ .$$

An NFA N is said to accept a string $w \in \Sigma^*$ if $\delta(Q_0, w) \cap F \neq \emptyset$, and the set of all strings (also called words) accepted by N is the language $\mathcal{L}(N)$ accepted by N. Any NFA has an equivalent DFA which accepts the same language. The DFA $N_D = (Q_D, \Sigma, \delta_D, Q_0, F_D)$ that is equivalent to a given NFA is found by performing the subset construction [3], so that Q_D consists of sets of states from Q. In essence, the subset construction keeps track of all the states that the NFA may be in at the same time, and forms the states of the equivalent DFA by a grouping of the states of the NFA. In short,

$$\delta_D(A, \sigma) = \bigcup_{q \in A} \delta(q, \sigma)$$

for any $A \subseteq Q$ and $\sigma \in \Sigma$. Any A is a final state in the DFA if $A \cap F \neq \emptyset$.

2.1. *Symmetric difference automata (XNFA)*

A symmetric difference NFA (XNFA) is defined similarly to an NFA, except that their behaviour is defined by the symmetric difference set operation. For any two sets A and B, the symmetric difference is given by

$$A \oplus B = (A \cup B) \setminus (A \cap B) \ .$$

More specifically, an XNFA $N_{\oplus}$ is a five-tuple $N_{\oplus} = (Q, \Sigma, \delta, Q_0, F)$, with each element defined as for NFA with the exception of δ, which is extended to $\delta : 2^Q \times \Sigma \to 2^Q$ in the following way:

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

Just as for NFA, δ can be extended to strings in the Kleene closure Σ^* of the alphabet.

An XNFA $N_{\oplus}$ is said to accept a string $w \in \Sigma^*$ if $|\delta(Q_0, w) \cap F|$ is odd (also known as parity acceptance), as an analogy to the symmetric difference set operation [12]. In other words, an odd number of paths labeled w must lead to final states. The set of all words accepted by $N_{\oplus}$ is the language $\mathcal{L}(N_{\oplus})$.

For a given XNFA, to determine the equivalent deterministic finite automaton, which we denote with XDFA for clarity, the subset construction is applied as

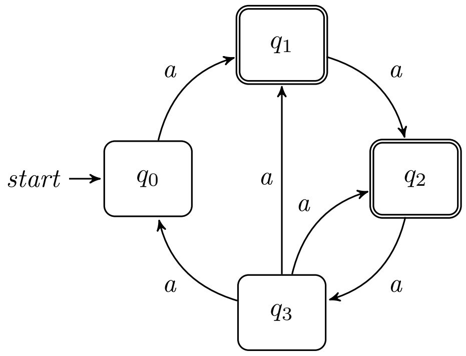
$$\delta_D(A, \sigma) = \bigoplus_{q \in A} \delta(q, \sigma)$$

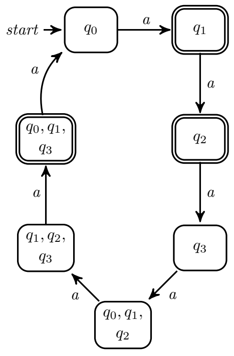
for any $A \subseteq Q$ and $\sigma \in \Sigma$.

The XDFA is denoted with $N_{D,\oplus} = (Q_D, \Sigma, \delta_D, Q_0, F_D)$. An XDFA final state contains an odd number of final XNFA states. For any given XNFA, an equivalent deterministic finite automaton can be found for any given XNFA. Since the deterministic finite automata accept the regular languages, it follows that the XNFA also accept the class of regular languages [12]. For clarity, we indicate XDFA states with square brackets and use curly brackets when referring specifically to sets of states in the context of XNFA. However, we may still treat XDFA states as sets, by, for example, using the standard notation for indicating the membership of elements, i.e. $q_0 \in [q_0, q_1, q_2]$.

We illustrate the concepts above with a short example.

Example 1. *Consider the unary XNFA N and its equivalent XDFA below.*





When N is converted to the equivalent XDFA, one sees for example that $\delta'(\{q_1, q_2, q_3\}, a) = \{q_2\} \oplus \{q_3\} \oplus \{q_0, q_1, q_2\} = \{q_0, q_1, q_3\}$. With the final states of N as $\{q_1, q_2\}$, it follows that in the XDFA, state $[q_1, q_2, q_3]$ is not a final state, since it contains an even number of final states from N. On the other hand, state $[q_0, q_1, q_3]$ is final in the XDFA, since it contains only one final state of N.

Given parity acceptance, XNFA have been shown to be equivalent to weighted automata over the finite field of two elements, or GF(2) [9,12]. For an XNFA $N = (Q, \Sigma, \delta, Q_0, F)$, the transitions for each alphabet symbol σ can be represented as a matrix over GF(2). Each row represents a mapping from a state $q \in Q$ to a set of states $P \in 2^Q$. The set P is written as a vector with a one in position i if $q_i \in P$, and a zero in position i if $q_i \notin P$. Hence, the transition table is represented as a matrix M_σ of zeroes and ones (see Example 6). This is known as the characteristic or transition matrix for σ of the XNFA. In the rest of this paper, we consider only XNFA with non-singular matrices, whose cycle structures do not include transient heads, i.e. states that are only reached once before a cycle is reached. The cycle structure of n-state XNFA with singular matrices are not interesting for our purposes, as it involves only short cycles with at most n states [8].

Initial and final states are similarly represented by vectors, and appropriate vector and matrix multiplications over GF(2) represent the behaviour of the XNFA[a]. For instance, in the unary case we would have a single matrix M_a that describes the transitions on a for some XNFA with n states. We encode the initial states Q_0 as vector of length n over GF(2), namely $v(Q_0) = [q_{0_0}\ q_{0_1}\ \cdots\ q_{0_{n-1}}]$, where $q_{0_i} = 1$ if $q_i \in Q_0$ and 0 otherwise. Similarly, we encode the final states as a length n vector $v(F) = [q_{F_0}\ q_{F_1}\ \cdots\ q_{F_{n-1}}]$. Then $v(Q_0)M_a$ is a vector that encodes the states reached after reading the symbol a exactly once, and $v(Q_0)M_a^k$ encodes the states reached after the symbol a was read k times. The weight of a word a_k of length k is given by

$$\Delta(a_k) = v(Q_0)M_a^k v(F)^T \ .$$

[a]In GF(2), $1 + 1 = 0$.

In Example 1 above, the initial state vector is $v(Q_0) = [1\ 0\ 0\ 0]$, the final state vector is $v(F) = [0\ 1\ 1\ 0]$, and the matrix M is given by

$$M = \begin{bmatrix} 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,1\,1\,0 \end{bmatrix} \quad .$$

We can say that M_a represents the word a, and $M_{a^k} = M_a^k$ represents the word a^k. In the binary case, there are two matrices, namely, M_a for transitions on a and M_b for transitions on b. Reading an a corresponds to multiplying by M_a, while reading a b corresponds to multiplying by M_b. Let M_w be the result of the appropriate multiplications of M_a and M_b representing some $w \in \{a, b\}^*$, then the weight of w is given by $\Delta(w) = v(Q_0)M_w v(F)^T$.

We now show that, in the unary case, a so-called change of basis is possible, where for some $n \times n$ transition matrix M_a of an XNFA and any non-singular $n \times n$ matrix A, $M'_a = A^{-1}M_aA$ is the transition matrix of an equivalent XNFA with $v(Q'_0) = v(Q_0)A$ and $v(F')^T = A^{-1}v(F)^T$. For any word a_k of length k, we have the following:

$$\begin{aligned}\Delta'(a_k) &= v(Q'_0)M'^k_a v(F')^T \\ &= v(Q_0)A(A^{-1}M_aA)^k A^{-1}v(F)^T \\ &= v(Q_0)M_a^k v(F)^T \\ &= \Delta(a_k) \quad .\end{aligned}$$

This also applies to the binary case. For some XNFA N, let $M_w = \prod_{i=1}^k M_{\sigma_i}$ represent a word $w = \sigma_1\sigma_2 \cdots \sigma_k$, where $M_{\sigma_i} = M_a$ if $\sigma_i = a$, and similarly for b. Now, let N' be an XNFA whose transition matrices are $M'_a = A^{-1}M_aA$ and $M'_b = A^{-1}M_bA$ for some non-singular A. Then w is represented by

$$\begin{aligned}M'_w &= \prod_{i=1}^k M'_{\sigma_i} \\ &= M'_{\sigma_1}M'_{\sigma_2} \cdots M'_{\sigma_k} \\ &= (A^{-1}M_{\sigma_1}A)(A^{-1}M_{\sigma_2}A) \cdots (A^{-1}M_{\sigma_k}A) \\ &= A^{-1}M_{\sigma_1}M_{\sigma_2} \cdots M_{\sigma_k}A \\ &= A^{-1}M_wA \ .\end{aligned}$$

Therefore, the weight of any word w on N' is

$$\begin{aligned}\Delta'(w) &= v(Q'_0)M'_w v(F')^T \\ &= v(Q_0)A(A^{-1}M_wA)A^{-1}v(F)^T \\ &= v(Q_0)M_w v(F)^T \\ &= \Delta(w) \ .\end{aligned}$$

Note that the above discussion does not rely on the fact that there are only two alphabet symbols, and so applies in general to the r-ary case as well.

The so-called characteristic polynomial $c(X)$ of an XNFA is defined as the determinant $c(X) = \det(XI - M)$ of the characteristic matrix, where I is the identitiy matrix. It is often convenient to use the polynomial rather than the matrix, and the polynomial fully captures the cycle structure of the XNFA. In Example 1 above, the characteristic polynomial is $c(X) = X^4 + X^2 + X + 1$. The polynomial is called reducible if it can be factorised; otherwise, it is irreducible. A primitive polynomial in GF(2) is the minimum polynomial of a primitive element (a primitive element generates the multiplicative group of the field) [8]. It is known that an n-state unary XNFA with a primitive characteristic polynomial has an equivalent XDFA with a single cycle of length $2^n - 1$ [9].

2.2. *Self-verifying automata (SV-NFA)*

Self-verifying NFA (SV-NFA) [1, 4, 5] are automata with two kinds of final states, namely accept states and reject states, as well as neutral non-final states. It is required that for any word, one or more of the paths for that word reach a single kind of final state. That is, either accept states or reject states are reached, but not both. Consequently, self-verifying automata reject words explicitly if they reach a reject state, in contrast to NFA, where rejection is the result of a failure to reach an accept state.

Definition 2. *A self-verifying nondeterministic finite automaton (SV-NFA) is a 6-tuple $N = (Q, \Sigma, \delta, Q_0, F^a, F^r)$, where Q, Σ, δ and Q_0 are defined as for standard NFA. The sets $F^a \subseteq Q$ and $F^r \subseteq Q$ are the sets of accept and reject states, respectively, and $F^a \cap F^r = \emptyset$. The remaining states, that is, the states belonging to $Q \setminus (F^a \cup F^r)$, are called neutral states. For each input string w in Σ^*, it is required that there exists at least one path ending in either an accept or a reject state; that is, $\delta(q_0, w) \cap (F^a \cup F^r) \neq \emptyset$ for any $q_0 \in Q_0$, and there are no strings w such that both $\delta(q_0, w) \cap F^a$ and $\delta(q_1, w) \cap F^r$ are nonempty, for any $q_0, q_1 \in Q_0$.*

Since any SV-NFA either accepts or rejects any string $w \in \Sigma^*$ explicitly, its equivalent DFA must do so too. The path for each w in a DFA is unique, so each state in the DFA is an accept or reject state. Hence, for any DFA state d, there is some SV-NFA state $q_i \in d$ such that $q_i \in F^a$ (and hence $d \in F^a_D$) or $q_i \in F^r$ (and hence $d \in F^r_D$). Since each state in the DFA is a subset of states of the SV-NFA, accept and reject states cannot occur together in a DFA state. That is, if d is a DFA state, then for any $p, q \in d$, if $p \in F^a$ then $q \notin F^r$ and vice versa.

2.3. *Self-verifying symmetric difference automata (SV-XNFA)*

In [7], self-verifying symmetric difference automata (SV-XNFA) were defined as a combination of the notions of symmetric difference automata and self-verifying automata, but only the unary case was examined. We now restate the definition of SV-XNFA in order to present results on larger alphabets in Section 4. Note, however, that the definition is slightly amended: in [7], the implicit assumption was made that no SV-XNFA state could be both an accept state and a reject state. This assumption is explored in detail for the unary case in [6], but for our current purposes it suffices to say that such a requirement removes the equivalence between XNFA and weighted automata over GF(2), which is essential for certain operations on XNFA, such as minimisation [9]. This implies that parity acceptance applies to SV-XNFA, where the condition for self-verification (SV-condition) is that for any word, an odd number of paths end in either accept states or reject states, but not both. In terms of the equivalent XDFA, this is equivalent to requiring that any XDFA state contain either an odd number of accept states or an odd number of reject states, but not both. If an XNFA state is both an accept state and a reject state, it contributes to both counts.

Definition 3. *A self-verifying symmetric difference finite automaton (SV-XNFA) is a 6-tuple $N = (Q, \Sigma, \delta, Q_0, F^a, F^r)$, where Q, Σ, δ and Q_0 are defined as for XNFA, and F^a and F^r are defined as for SV-NFA, except that $F^a \cap F^r$ need not be empty. That is, each state in the SV-XDFA equivalent to N must contain an odd number of states from either F^a or F^r, but not both, and some SV-XNFA states may belong to both F^a and F^r.*

We refer to the equivalent DFA of some SV-XNFA as its equivalent SV-XDFA to indicate that every state must accept or reject and that parity acceptance holds given the subset construction. Any SV-XDFA is equivalent to an XDFA, so SV-XNFA accept the class of regular languages. The SV-condition for XNFA implies that if a state in the SV-XDFA of an SV-XNFA N contains an odd number of states from F^a, it may also contain an even number of states from F^r, and hence belong to F^a_D, and vice versa. An SV-XDFA state may contain any number of neutral states from N.

The choice of F^a and F^r for a given SV-XNFA N is called an *SV-assignment* of N. An SV-assignment where either F^a or F^r is empty, is called a *trivial SV-assignment.* Otherwise, if both F^a and F^r are nonempty, the SV-assignment is non-trivial. We now restate a core result concerning unary SV-XNFA from [6].

Theorem 4. *[6] Any matrix M has an SV-assignment, if and only if its characteristic polynomial has $X + 1$ as a factor.*

In order to ultimately prove state complexity bounds on SV-XNFA, it is necessary to investigate the cycles that form in their equivalent XDFA. To that effect, the next section describes the cyclic behaviour of XDFA.

$$M = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & \cdots & 1 \\ c_0 & c_1 & \cdots & c_{n-2} & c_{n-1} \end{bmatrix}$$

Fig. 1. Normal form matrix for $c(X) = X^n + c_{n-1}X^{n-1} + ... + c_1X + c_0$

3. XNFA as linear machines over GF(2)

In [11] it is shown that the cycle structure of unary XNFA are equivalent to that of linear feedback shift registers (LFSRs), with their rich literature in circuit design, cryptology, and other applications [8]. Specifically, a matrix M with characteristic polynomial $c(X)$ is associated with a certain cycle structure of sets of XNFA states (that is, XDFA states), and the choice of Q_0 determines which cycle represents the behaviour of a specific unary XNFA. The cycle structure is induced by $c(X)$, so any matrix that has $c(X)$ as its characteristic polynomial has the same cycle structure, although the states ocurring in the cycles differ according to each specific matrix. This allows us to use the theory underlying LFSRs in the analysis of XNFA (for example, [2]).

For the r-ary case, the transition matrix for each symbol is associated with its own cycle structure, and the choice of Q_0 determines which cycle is realised in the r-ary XNFA for each symbol. There are $2^n - 1$ possible choices for Q_0 (we exclude the empty set). Evidently, the cycles associated with each symbol might overlap, and so the structure of the r-ary XNFA would not be cyclic itself, although the transitions for each symbol would exhibit cyclic behaviour. Specifically, for an r-ary XNFA N and some symbol $\sigma \in \Sigma$, we refer to the cycle structure of N on σ to indicate the cycle structure resulting from considering only transitions on σ. Our main results will be derived from examining the cycle structure induced by each symbol of the alphabet of the automaton, as well as the ways in which the cycles overlap.

For any $c(X) = X^n + c_{n-1}X^{n-1} + \cdots + c_1X + c_0$ there is a normal form matrix M of the form given in Fig. 1, such that $c(X) = \det(XI - M)$, where I is the identity matrix. We say that M is in canonical form.

In the next lemma, it will be convenient to represent XDFA states $d_s \subseteq Q$ as $s = \langle s_{n-1}, s_{n-2}, ..., s_1, s_0\rangle$, where $s_i = 1$ if $q_i \in d_s$ and 0 otherwise. The lemma is adapted from [8] on the basis of the equivalence between unary XNFA and LFSRs.

Lemma 5. *Let M_σ be a transition matrix representing transitions on σ for some XNFA N, with characteristic polynomial $c_\sigma(X)$, and let M_σ be in canonical form. Let f be a bijection of the states of the equivalent XDFA N_D onto polynomials of degree $n-1$, such that f maps the state $s = \langle s_{n-1}, s_{n-2}, ..., s_1, s_0\rangle$ into the*

$$M_a = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Fig. 2. Example 6, transition matrix for a

$$M_b = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Fig. 3. Example 6, transition matrix for b

Table 1. Transitions on δ correspond to multiplication by X

$\delta_D(s,\sigma)$	$Xf(s)$ mod $c_\sigma(X)$	
$\delta_D(\{q_0\},a)=\{q_1\}$	$X(1)$	$=X$
$\delta_D(\{q_3\},a)=\{q_0,q_1,q_2\}$	$X(X^3)$	$=X^4$ mod $c_a(X)$
		$=X^2+X+1$
$\delta_D(\{q_0,q_2,q_3\},a)=\{q_0,q_2,q_3\}$	$X(X^3+X^2+1)$	$=X^4+X^3+X$ mod $c_a(X)$
		$=X^3+X^2+1$
$\delta_D(\{q_1\},b)=\{q_2\}$	$X(X)$	$=X^2$
$\delta_D(\{q_0,q_1,q_3\},b)=\{q_0,q_2,q_3\}$	$X(X^3+X+1)$	$=X^4+X^2+X$ mod $c_b(X)$
		$=X^3+X^2+1$
$\delta_D(\{q_1,q_2,q_3\},b)=\{q_0,q_1,q_2\}$	$X(X^3+X^2+X)$	$=X^4+X^3+X^2$ mod $c_b(X)$
		$=X^2+X+1$

polynomial $f(s)=s_{n-1}X^{n-1}+s_{n-2}X^{n-2}+\cdots+s_1X+s_0$. *Then* f *maps the state* $s\cdot M_\sigma$ *into the polynomial* $Xf(s)$ *mod* $c_\sigma(X)$.

Lemma 5 provides a mapping between polynomials over $GF(2)$ and the states of XDFA. The XDFA state arrived at after a transition from state s on σ corresponds to the polynomial which results from multipying $f(s)$ by X in the polynomial algebra of $GF(2)[X]$ modulo $c_\sigma(X)$.

Example 6. *Let* N *be a binary XNFA (shown in Figure 4), where* M_a *is the normal form matrix of* $c_a(X)=X^4+X^2+X+1$ *and* M_b *is the normal form matrix of* $c_b(X)=X^4+X^3+X+1$. *The matrices* M_a *and* M_b *are given in Fig. 2 and 3. The resulting XDFA is shown in Figure 5, while some examples comparing state transitions and polynomial multiplication are shown in Table 1. Note that, for now, the focus is on the cyclic behaviour of the equivalent XDFA, and so we do not refer to any final states.*

In addition to a normal form matrix, each polynomial is also associated with a so-called *companion matrix*, as stated in the next theorem.

Theorem 7. *[8] Every matrix* M *over* $GF(2)$ *with characteristic polynomial* $c(X)$

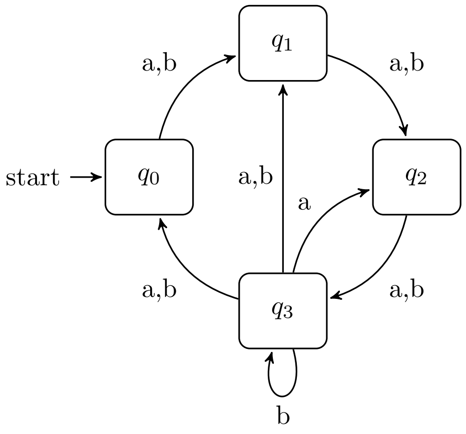


Fig. 4. Example 6, N

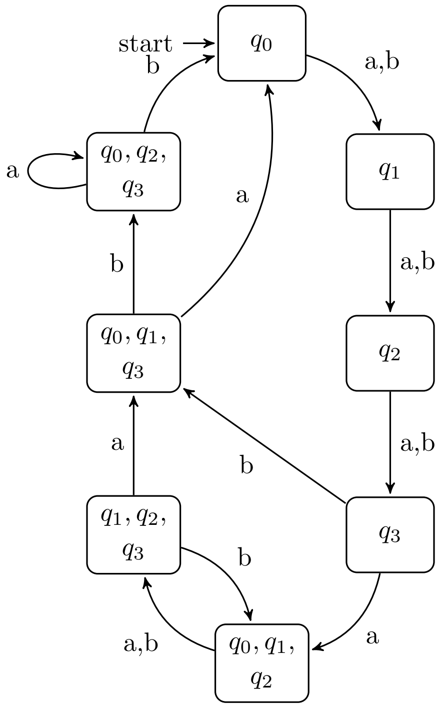


Fig. 5. Example 6, N_D

is similar[b] *to a matrix M' of the form shown in Figure 6, where each of the submatrices A_i is a normal form matrix of a polynomial that is irreducible over $GF(2)$ or a power of a polynomial that is irreducible over $GF(2)$, and the 0's are 0-submatrices of appropriate sizes. M' is said to be the companion matrix of $c(X)$.*

$$M' = \left[\begin{array}{c|c|c|c} A_1 & 0 & \dots & 0 \\ \hline 0 & A_2 & \dots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \dots & A_m \end{array}\right]$$

Fig. 6. Block diagonal companion matrix of normal form matrices

Each block in the companion matrix M of some $c(X)$ represents a smaller automaton of which the characteristic polynomial is irreducible or is a power of an

[b]Two $n \times n$ matrices A and A' are similar if there exists some non-singular $n \times n$ matrix B so that $A' = B^{-1}AB$.

irreducible polynomial. If $c(X)$ has different factors, so that M has more than one block on the diagonal, the automaton can be thought of as a composite machine, where M is divided into two or more blocks, and cycles from the different blocks combine to form new cycles, as stated in Lemma 8.

In the discussion that follows, *empty cycle* refers to the cycle that contains the "empty" XDFA state, or the empty subset of XNFA states, which corresponds to the zero polynomial. Since $0 \cdot X = 0$, according to Lemma 5, we would have $\delta(\emptyset, a) = \emptyset$. The following lemma is taken from [8].

Lemma 8. *[8] Let $c(X)$ be a reducible polynomial of degree n over $GF(2)$ that does not have X as a factor. We consider its companion matrix as consisting of two blocks, B_1 and B_2, which in turn may consist of blocks of normal form matrices associated with the factors of $c(X)$. For each cycle of length k_1 induced by block B_1 in the companion matrix and for each cycle of length k_2 induced by block B_2, $c(X)$ has $gcd(k_1, k_2)$ cycles of length $lcm(k_1, k_2)$.*

The proof of the lemma considers an XNFA N as a composite machine whose companion matrix contains the blocks B_1 and B_2 on the diagonal, where B_1 and B_2 represent two XNFA, say N_1 and N_2. In the simplest case, B_1 and B_2 represent normal form matrices associated with the factors of $c(X)$, but they may also in turn represent composite automata. Starting in some state belonging to N is equivalent to simultaneously starting in a state belonging to N_1 and a state belonging to N_2. The cycle belonging to N is complete, i.e. its start state reached again, when the two start states from N_1 and N_2 are reached again simultaneously.

Theorem 9. *Let N be a unary XNFA with its associated characteristic polynomial $c(X) = (X+1)\phi(X)$, where $X+1$ is not a factor of $\phi(X)$. Then the cycle structure associated with N consists of pairs of cycles of the same length, where one cycle in each pair does not have an SV-assignment.*

Proof. Let M_ϕ be the companion matrix of $\phi(X)$. Then the companion matrix M_c of $c(X)$ is given by

$$M_c = \left[\begin{array}{c|c} M_\phi & 0 \\ \hline 0 & 1 \end{array}\right] .$$

That is, M_c is a block diagonal matrix, with the companion matrix M_ϕ on the diagonal, as well as the companion matrix for $X + 1$, which is the 1×1 matrix containing a 1. The cycle structure associated with $X + 1$ consists of two cycles of length one, namely the empty cycle and the cycle containing the XDFA state that consists of only one XNFA state, namely $[q_{n-1}]$.

Now, by Lemma 8, the cycles of N are found by considering pairwise composite cycles associated with M_ϕ and the companion matrix of $X+1$. Any cycle C associated with M_ϕ composed with the empty cycle is simply C, while C composed with the cycle containing the state $[q_{n-1}]$ results in the following cycle: for every state s

in C, the composite cycle, which we may term $C_{q_{n-1}}$, contains the state $s \cup [q_{n-1}]$. The resulting composite machine N therefore contains pairs of cycles of the same length.

By Theorem 4, any cycle that is simply a copy of a cycle from the cycle structure of M_ϕ does not have an SV-assignment, since $X+1$ is not a factor of $\phi(X)$. Hence, in each pair of cycles C and $C_{q_{n-1}}$ belonging to N, the cycle C does not have an SV-assignment. $\square$

Example 10. *Let $c(X) = X^4 + X^3 + X^2 + 1$ and $\phi(X) = X^3 + X + 1$. Then $c(X) = (X+1)\phi(X)$, and since $\phi(X)$ is a primitive irreducible polynomial, it does not have $X+1$ as a factor. The companion matrices for $\phi(X)$ and $c(X)$ are given in Figures 7 and 8, respectively. Figure 9 gives the cycle structure of M_c. However, notice that the two cycles on the left are exactly those cycles that comprise the cycle structure of M_ϕ, while the two cycles on the right are copies of the cycles on the left, except that each state additionally includes q_3. By Theorem 4, the two cycles on the left have no SV-assignments, since $X+1$ is not a factor of $\phi(X)$. We see, however, that any choice of F^a and F^r where q_0, q_1 and q_2 each belong to either F^a or F^r, and where $q_3 \in F^a$ and $q_3 \in F^r$, is an SV-assignment for the two cycles on the right.*

$$M_\phi = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Fig. 7. Example 10, companion matrix of $\phi(X)$

$$M_c = \left[\begin{array}{ccc|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Fig. 8. Example 10, companion matrix for $c(X)$

We now use Theorem 9 to investigate bounds on the state complexity of non-unary SV-XNFA.

4. Bounds on the state complexity of non-unary SV-XNFA

For any n-state non-unary SV-XNFA, the equivalent minimal XDFA can have no more than $2^n - 1$ states, since this is the number of non-empty subsets for any set of n XNFA states. This represents the standard upper bound. In this section we establish a lower bound on state complexity, namely 2^{n-1}, and we identify a set of non-unary SV-XNFA for which this is also an upper bound.

Consider a unary XNFA with $c(X) = (X+1)\phi(X)$. In this case, we show that in its equivalent XDFA, any state with an odd number of XNFA states must necessarily go to another state with an odd number of XNFA states.

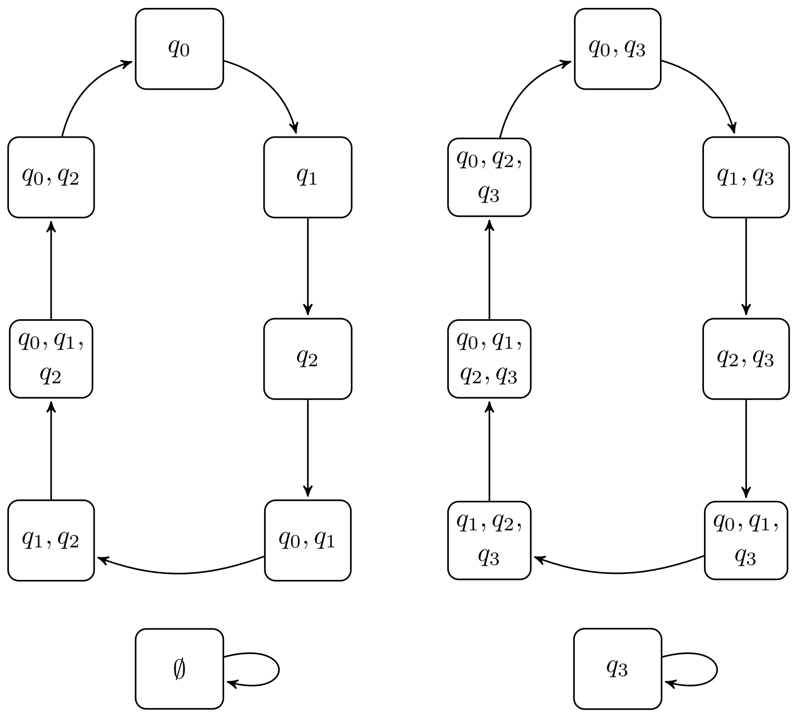


Fig. 9. Example 10, cycles

Lemma 11. *Let M be the normal form matrix of some $c(X) = (X + 1)\phi(X)$. Let $\delta : 2^Q \times \Sigma \to 2^Q$ be the transition function encoded by M. Since δ describes unary transitions, we let $\Sigma = \{a\}$. Then for any $d \in 2^Q$, $|\delta_D(d, a)|$ is odd if and only if $|d|$ is odd.*

Proof. Let $|Q| = n$. Then M is an $n \times n$ matrix that has the form given below:

$$M = \begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & & & 0 & 0 \\ 0 & 0 & 0 & \ddots & & 0 & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & & & 1 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ c_0 & c_1 & c_2 & \cdots & \cdots & c_{n-2} & c_{n-1} \end{bmatrix}.$$

We note that in $GF(2)$, $a - b = a + b$ and hence, if $X + 1$ is a factor of $c(X)$, then 1 is a root, i.e. $(1)^n + c_{n-1}(1)^{n-1} + ... + c_1(1) + c_0 = 0$. Consequently, an odd number among $c_0, c_1, \ldots, c_{n-2}, c_{n-1}$ are 1's. Therefore, we see that $|\delta(q_{n-1}, a)|$ is odd by inspecting the bottom row of M, while $|\delta(q_i, a)| = 1$ for $0 \le i \le n - 2$. We consider two cases regarding d.

Case 1: $q_{n-1} \notin d$. Let $d = [q_{i_1}, q_{i_2}, \ldots, q_{i_k}]$. By the argument above, if $q_{n-1} \notin d$, then k must be odd. Then $\delta_D(d, a) = [q_{i_1+1}, q_{i_2+1}, \ldots, q_{i_k+1}]$, and hence $|d| = |\delta_D(d, a)| = k$, which is odd.

Case 2: $q_{n-1} \in d$. Let $d = [q_{i_1}, q_{i_2}, \ldots, q_{i_k}, q_{n-1}]$. Again, since $q_{n-1} \in d$, it must hold that k is even. Let $D = [q_{i_1}, q_{i_2}, \ldots, q_{i_k}]$. Then $\delta_D(d, a) = \delta_D(D, a) \oplus \delta_D([q_{n-1}], a)$.

We have $\delta(D, a) = [q_{i_1+1}, q_{i_2+1}, \ldots, q_{i_k+1}]$, and so $|\delta_D(D, a)| = |D| = k$, which is even. Also, $|\delta_D([q_{n-1}], a)|$ is odd. Therefore, $|\delta_D(D, a)| + |\delta_D([q_{n-1}], a)|$ is odd. The symmetric difference of two sets is the set of elements that occur in exactly one of the two sets. If some element occurs in both, then it is not in the symmetric difference. We can therefore "cancel out" elements occuring in both $\delta_D(D, a)$ and $\delta_D([q_{n-1}], a)$ in pairs, until only those remain that occur in exactly one of the sets. But successively removing pairs of elements from an odd number of elements leaves an odd number of elements. So $\delta_D(D, a) \oplus \delta_D([q_{n-1}], a)$ must be odd, and hence $|\delta_D(d, a)|$ is odd.

Similarly, if $|d|$ is even, then $|\delta(d, a)|$ is even. □

Now extend Lemma 11 to an r-ary alphabet.

Theorem 12. *Let M_{σ_1}, M_{σ_2}, ..., M_{σ_r} be the normal form matrices of r polynomials $c_{\sigma_1}(X) = (X+1)\phi_{\sigma_1}(X)$, $c_{\sigma_2}(X) = (X+1)\phi_{\sigma_2}(X)$, ..., $c_{\sigma_r}(X) = (X+1)\phi_{\sigma_r}(X)$, respectively, and let M_{σ_1}, M_{σ_2}, ..., M_{σ_r} be the transition matrices of some r-ary XNFA N with $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_r\}$ and $Q_0 = \{q_0\}$. Then the number of states in the equivalent XDFA N_D does not exceed 2^{n-1}. Furthermore, any choice of F^a and F^r such that $F^a \cup F^r = Q$ and $F^a \cap F^r = \emptyset$ is an SV-assignment.*

Proof. By Lemma 11, since $Q_0 = \{q_0\}$ and $|\{q_0\}|$ is odd, and $c_{\sigma_1}(X)$, $c_{\sigma_2}(X)$,…,$c_{\sigma_r}(X)$ have $X + 1$ as a factor, only odd-sized states are reached on any transition. The number of XDFA states d such that $|d|$ is odd is $2^n/2 = 2^{n-1}$, and so N_D can have at most 2^{n-1} states. Since every XDFA state contains an odd number of XNFA states, any choice of F^a and F^r such that $F^a \cup F^r = Q$ and $F^a \cap F^r = \emptyset$ is an SV-assignment. □

The following lemma provides further information on the cycle structure induced by polynomials with $X + 1$ as a factor.

Lemma 13. *Let $c_\sigma(X) = (X + 1)\phi(X)$. Then, in the normal form matrix M_σ of $c_\sigma(X)$, which is the transition matrix on some symbol σ for an XNFA, the state mapped to $\phi(X)$ as described in Lemma 5, i.e. d_ϕ, is contained in a cycle of length one, when considering only transitions on σ.*

Proof. Consider the following:

$$\begin{aligned}(X+1)\phi(X) &= c_\sigma(X)\\ X\phi(X)+\phi(X) &= c_\sigma(X)\\ X\phi(X) &= \phi(X)+c_\sigma(X)\end{aligned}$$

Therefore, $X\phi(X) = \phi(X)$ in the representation of $GF(2^n)$ as polynomials over GF(2) modulo $c_\sigma(X)$. By Lemma 5, this corresponds to $\delta_D(d_\phi, \sigma) = d_\phi$. □

We now present a witness language for any n to show that 2^{n-1} is a lower bound on the state complexity of SV-XNFA with non-unary alphabets. First, we restate the following theorem from [7].

Theorem 14. *For any $n \geq 2$, there is an SV-XNFA N whose equivalent N_D has $2^{n-1} - 1$ states. For any $n \geq 2$, this an SV-XNFA has characteristic polynomial $c(X) = (X + 1)\phi(X)$, with $\phi(X)$ a primitive polynomial over GF(2).*

Lemma 15. *Let $\phi(X) = X^{n-1} + \phi_{n-2}X^{n-2} + \cdots + \phi_1 X + \phi_0$ be any primitive polynomial of degree $n-1$. Let N be a binary XNFA, and let the transition matrix on a be the normal form matrix of $c_a(X) = (X + 1)\phi(X)$ and the transition matrix on b be the normal form matrix of $c_b(X) = X^n + \phi(X)$. Then the equivalent XDFA of the XNFA with $Q_0 = \{q_0\}$ contains exactly 2^{n-1} odd-sized states.*

Proof. We write $c_a(X)$ and $c_b(X)$ in the following way:

$$\begin{aligned}c_a(X) &= X^n + c_{n-1}X^{n-1} + \cdots + c_1X + c_0\\ c_b(X) &= X^n + \phi_{n-1}X^{n-1} + \phi_{n-2}X^{n-2} + \cdots + \phi_1X + \phi_0 \quad .\end{aligned}$$

Since $\phi(X)$ is primitive, it has no roots in GF(2), including 1, so it must have an odd number of non-zero terms. Therefore, by Lemma 11, $|d_\phi|$ is odd. Furthermore, $c_b(X)$ has an even number of non-zero terms, and so has 1 as a root. Consequently, $c_b(X)$ has $X + 1$ as a factor.

The transition matrices M_a and M_b are given in Fig. 10 and Fig. 11. Note that they are both non-singular. Let $Q_0 = \{q_0\}$. Then by Theorem 14, the cycle structure on a is equivalent to an XDFA cycle with $2^{n-1} - 1$ states, all of which, by Lemma 11, have odd size. Also, by Lemma 13, d_ϕ is not contained in this cycle. This means that on a, every odd-sized state in the XDFA is reached except for d_ϕ. Now, from M_b it follows directly that $\delta_D(\{q_{n-1}\}, b) = d_\phi$. Furthermore, since $X+1$ is a factor of c_b, every transition from an odd-sized state on b is to an odd-sized state. Consequently, the binary XNFA N is equivalent to an XDFA that reaches all 2^{n-1} odd-sized states and none other. □

Theorem 16. *For any $n \geq 2$, there is a language $\mathcal{L}_n$ so that some n-state binary SV-XNFA accepts $\mathcal{L}_n$ and the minimal SV-XDFA that accepts $\mathcal{L}_n$ has 2^{n-1} states.*

$$M_a = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & \cdots & 1 \\ c_0 & c_1 & \cdots & c_{n-2} & c_{n-1} \end{bmatrix}$$

Fig. 10. Lemma 15, transition matrix for a

$$M_b = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & \cdots & 1 \\ \phi_0 & \phi_1 & \cdots & \phi_{n-2} & \phi_{n-1} \end{bmatrix}$$

Fig. 11. Lemma 15, transition matrix for b

Proof. Let $c_a(X) = (X+1)\phi(X)$ and $c_b = X^n + \phi(X)$, where $\phi(X)$ is a primitive polynomial and let $c_a(X)$ and $c_b(X)$ have degree n. We construct an SV-XNFA N with n states whose equivalent XDFA N_D has 2^{n-1} states as in Lemma 15, and let $F^a = \{q_0\}$ and $F^r = Q \setminus F^a$. Recall that for N, we have $\delta : Q \times \Sigma \to 2^Q$, and for N_D, we have $\delta_D : 2^Q \times \Sigma \to 2^Q$.

Let $\mathcal{L}_n^1 = a^{(2^{n-1}-1)i+j}$ for $i \geq 0$ and $j \in J$, where J is some set of integers, represent a subset of the language accepted by N that consists only of strings containing a. Now, from the transition matrix of N it follows that $0, n \in J$, while $1, 2, ..., n-1 \notin J$, since $q_0 \in \delta(q_0, a^n)$, but $q_0 \notin \delta(q_0, a^m)$ for $m < n$.

If there is an N'_D with fewer than $2^{n-1} - 1$ states that accepts $\mathcal{L}_n^1$, then there must be some $d_j \in Q_D$ such that $\{q_0\} \subset d_j$, $q_0 \in \delta_D(d_j, a^n)$ and there is no $m < n$ so that $q_0 \in \delta_D(d_j, a^m)$. That is, if N'_D exists, then on N_D, $\delta_D(\{q_0\}, a) = \delta_D(d_j, a)$, and $\delta_D(\{q_1\}, a) = \delta_D(d_{j+1}, a)$ etc.

Let d_k be any state in N_D such that $d_k \neq \{q_0\}$. Let $\max(d_k)$ be the largest subscript of any SV-XNFA state in d_k. Then $\max(d_k) > 0$. Let $m = n - \max(d_k)$, so $m < n$. Then, from the transition matrix of N, it follows that $q_0 \in \delta_D(d_k, a^m)$. That is, for any d_k there is an $m < n$ so that $q_0 \in \delta_D(d_k, a^m)$. Therefore, there is no N'_D with fewer than $2^{n-1} - 1$ states that accepts $\mathcal{L}_n^1$.

Now, let $\mathcal{L}_n^2 = b^n a^*$, which is also a subset of the language accepted by N. In order to accept this language, after reading b^n, a state must have been reached whereafter every transition on a must result in an accept state, i.e. an XDFA state containing q_0. But there is only one such state, and that is d_ϕ, since $\delta_D(d_\phi, a) = d_\phi$, which is excluded from the cycle needed to accept $\mathcal{L}_n^1$. Therefore, all 2^{n-1} odd-sized states are necessary to accept $\mathcal{L}_n^1 \cup \mathcal{L}_n^2$. Let $\mathcal{L}_n$ be the language accepted by N, then since $\mathcal{L}_n^1 \cup \mathcal{L}_n^2 \subset \mathcal{L}_n$, at least 2^{n-1} states are necessary to accept $\mathcal{L}_n$. □

We illustrate Theorem 16 for $n = 4$.

Example 17. *Let $\phi(X) = X^3 + X + 1$, which is a primitive polynomial. Now, let N be an XNFA with transition matrices M_a and M_b. The matrix M_a is the normal form matrix of $c_a(X) = (X+1)\phi(X) = X^4 + X^3 + X^2 + 1$ and M_b the normal form matrix of $c_b(X) = X^4 + \phi(X) = X^4 + X^3 + X + 1$. Let $Q_0 = \{q_0\}$ and let $F^a = \{q_0\}$ and $F^r = \{q_1, q_2, q_3\}$. The matrices M_a and M_b are shown in Figures 12 and 13, while N and its equivalent XDFA N_D are shown in Figures 14 and 15. We*

$$M_a = \begin{bmatrix} 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&0&1&1 \end{bmatrix}$$

Fig. 12. Example 17, transition matrix for a

$$M_b = \begin{bmatrix} 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&1&0&1 \end{bmatrix}$$

Fig. 13. Example 17, transition matrix for b

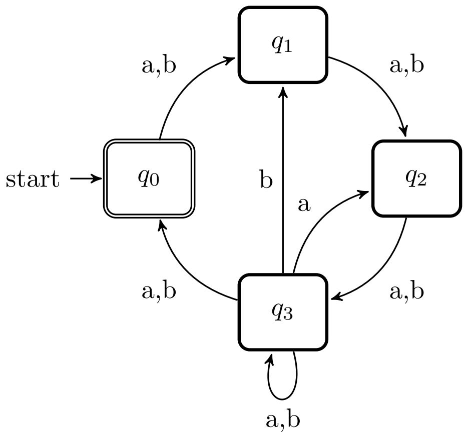


Fig. 14. Example 17, N

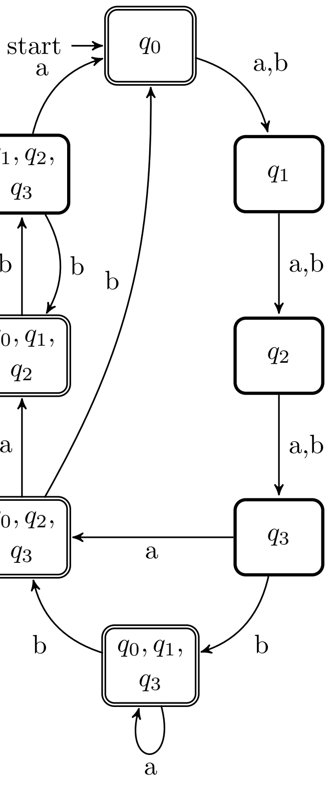


Fig. 15. Example 17, N_D

have $\mathcal{L}^1 = a^{7i+j}$ *for* $i \geq 0$ *and* $j \in \{0, 4, 5\}$ *and* $\mathcal{L}^2 = bbbba^*$.

The following is a simple corollary of Theorem 16.

Corollary 18. *For any* $m, n \geq 2$*, there is a language* $\mathcal{L}'_n$ *so that some* n*-state* m*-ary SV-XNFA accepts* $\mathcal{L}'_n$ *and the minimal SV-XDFA that accepts* $\mathcal{L}'_n$ *has* 2^{n-1} *states.*

We now show that any given SV-XNFA can be used to obtain another one via a so-called change of basis.

Theorem 19. *Given any SV-XNFA $N = (Q, \Sigma, \delta, Q_0, F^a, F^r)$ with n states and transition matrices M_{σ_1}, M_{σ_2}, ..., M_{σ_r}, and any non-singular $n \times n$ matrix A, we encode Q_0 as a vector $v(Q_0)$ of length n over GF(2) and F^a and F^r as vectors $v(F^a)$ and $v(F^r)$, respectively. Then there is an SV-XNFA $N' = (Q, \Sigma, \delta', Q'_0, F'^a, F'^r)$ where $M'_{\sigma_i} = A^{-1}M_{\sigma_i}A$ for $1 \le i \le r$, $v(Q'_0) = v(Q_0)A$, $v(F'^a)^T = A^{-1}v(F^a)^T$ and $v(F'^r)^T = A^{-1}v(F^r)^T$, and N' accepts the same language as N.*

Proof. In the discussion in Section 2.1 we showed that for XNFA, the change of basis described on an XNFA N that results in N', $\Delta'(w) = \Delta(w)$. We extend this to SV-XNFA by defining two new functions. Recall that M_w represents the sequence of matrix multiplications for some w of length k, and that $M'_w = A^{-1}M_wA$. Then, let

$$\begin{aligned} accept(w) &= v(Q_0)M_wv(F^a)^T \\ reject(w) &= v(Q_0)M_wv(F^r)^T \quad . \end{aligned}$$

The SV-condition is that $accept(w) \neq reject(w)$ for any $w \in \Sigma^*$. Similar to $\Delta(w)$, we have

$$\begin{aligned} accept'(w) &= v(Q'_0)M'_wv(F'^a)^T \\ &= v(Q_0)A(A^{-1}M_wA)A^{-1}v(F^a) \\ &= v(Q_0)M_wv(F^a) \\ &= accept(w) \end{aligned}$$

and

$$\begin{aligned} reject'(w) &= v(Q'_0)M'_wv(F'^r)^T \\ &= v(Q_0)A(A^{-1}M_wA)A^{-1}v(F^r) \\ &= v(Q_0)M_wv(F^r) \\ &= reject(w) \quad . \end{aligned}$$

Clearly, the SV-condition is met by $accept'$ and $reject'$, and so N' is an SV-XNFA that accepts the same language as N. □

The number of non-singular $n \times n$ matrices over GF(2) (including the identity matrix) is $|GL(n, \mathbb{Z}_2)| = \prod_{k=0}^{n-1}(2^n - 2^k)$, and so, up to isomorphism, for any SV-XNFA at most another $|GL(n, \mathbb{Z}_2)| - 1$ equivalent SV-XNFA can be found.

Theorem 20. *Let N be an r-ary XNFA with transitions matrices M_{σ_1}, M_{σ_2},...,M_{σ_r}, for which $c_{\sigma_1}(X) = (X+1)\phi_{\sigma_1}(X)$, and where $c_{\sigma_i}(X) = (X+1)^{k_i}\phi_{\sigma_i}(X)$ for $1 < i \le r$. Furthermore, for any $1 \le i \le r$, $X+1$ is not a factor of $\phi_{\sigma_i}(X)$. Then for any choice of Q_0 such that N is an SV-XNFA, the minimal equivalent XDFA has at most 2^{n-1} states.*

Proof. By Theorem 9, the cycle structure of $c_{\sigma_1}(X)$ consists of pairs of cycles of the same length, where one cycle in each pair does not have an SV-assignment.

Hence, at least $\frac{2^n}{2} = 2^{n-1}$ states belong to cycles on σ_1 that do not have SV-assignments. That is, 2^{n-1} belong to cycles where no choice of F^a and F^r satisfies the SV-condition, and hence, any XDFA that reaches such cycles cannot have an SV-assignment. Therefore, any r-ary SV-XNFA may only reach the remaining 2^{n-1} states. □

5. Conclusion

We have given an upper bound of $2^n - 1$ on the state complexity of SV-XNFA with alphabet size larger than one, and a lower bound of 2^{n-1}. Furthermore, we identified a set of non-unary SV-XNFA for which this is also an upper bound. We have also shown that, given any SV-XNFA with n states, it is possible, up to isomorphism, to find at most another $|GL(n, \mathbb{Z}_2)| - 1$ equivalent SV-XNFA via a change of basis.

References

[1] I. Assent and S. Seibert, An upper bound for transforming self-verifying automata into deterministic ones, *RAIRO-Theoretical Informatics and Applications-Informatique Théorique et Applications* **41**(3) (2007) 261–265.

[2] L.-L. Dornhoff and F.-E. Hohn, *Applied modern algebra.* (Macmillan Publishing Co., Inc., Collier Macmillan Publishers, 1978).

[3] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 1st edn. (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990).

[4] J. Hromkovič and G. Schnitger, *Proceedings of Automata, Languages and Programming: 26th International Colloquium, ICALP'99 Prague, Czech Republic, July 11–15, 1999* (Springer, Berlin, Heidelberg, 1999), ch. On the Power of Las Vegas II. Two-Way Finite Automata.

[5] G. Jirásková and G. Pighizzini, Optimal simulation of self-verifying automata by deterministic automata, *Information and Computation* **209**(3) (2011) 528 – 535, Special Issue: 3rd International Conference on Language and Automata Theory and Applications (LATA 2009).

[6] L. Marais and L. van Zijl, State complexity of Unary SV-XNFA with Different Acceptance Conditions, Submitted for publication.

[7] L. Marais and L. van Zijl, *Proceedings of Descriptional Complexity of Formal Systems: 18th IFIP WG 1.2 International Conference, DCFS 2016, Bucharest, Romania, July 5-8, 2016* (Springer International Publishing, 2016), ch. Unary Self-verifying Symmetric Difference Automata, pp. 180–191.

[8] H. S. Stone, *Discrete Mathematical Structures and their Applications* (Science Research Associates Chicago, 1973).

[9] B. van der Merwe, H. Tamm and L. van Zijl, *Proceedings of the 14th International Conference on the Descriptional Complexity of Formal Systems (DCFS 2012), Braga, Portugal, July 23-25, 2012* (Springer, Berlin, Heidelberg, 2012), ch. Minimal DFA for Symmetric Difference NFA, pp. 307–318.

[10] L. van Zijl, On binary $\oplus$-NFAs and succinct descriptions of regular languages, *Theoretical Computer Science* **328** (2004) 161 – 170.

[11] L. van Zijl, J.-P. Harper and F. Olivier, *Implementation and Application of Automata: 5th International Conference, CIAA 2000 London, Ontario, Canada, July*

24–25, 2000 Revised Papers (Springer, Berlin, Heidelberg, 2001), ch. The MERLin Environment Applied to $\star$-NFAs, pp. 318–326.

[12] J. Vuillemin and N. Gama, *Proceedings of the 14th International Conference on the Implementation and Application of Automata (CIAA 2009), Sydney, Australia, July 14-17, 2009* (Springer, Berlin, Heidelberg, 2009), ch. Compact Normal Form for Regular Languages as Xor Automata, pp. 24–33.