Journal of Automata, Languages and Combinatorics **u** (v) w, x–y

Large NFA Without Mergible States

CEZAR CÂMPEANU[1]
Department of Computer Science and Information Technology, University of Prince Edward Island
550 University Avenue, Charlottetown, PEI, Canada C1A 4P4
e-mail: ccampeanu@upei.ca

NICOLAE SÂNTEAN[2]
Department of Computer Science, University of Western Ontario, Middlesex College, London, Ontario, Canada N6A 5B7
e-mail: nic@csd.uwo.ca

and

SHENG YU[2]
Department of Computer Science, University of Western Ontario, Middlesex College, London, Ontario, Canada N6A 5B7
e-mail: syu@csd.uwo.ca

ABSTRACT

Merging states in finite automata is a main method of reducing the size of the representation of regular languages. The process has been extensively studied for deterministic finite automata (DFA), where the conditions for merging states can be efficiently computed. The matter is more complex in the case of nondeterministic finite automata (NFA), where merging states can be done in different ways, and the cost of detecting mergible states is high. In a recent paper the authors have studied one type of state mergibility and proven that one cannot have an arbitrarily large (in terms of number of states) NFA for a given language such that no states can be merged. In this paper, we study a different type of state mergibility for NFA, which is similar to the state mergibility in a DFA. We prove that there are situations where state merging is impossible for arbitrary large equivalent NFA.

Keywords: Nondeterministic Finite Automata, States, Minimization

1. Introduction

Succinct representations of regular languages have always been a popular topic in formal languages theory. Besides its theoretical importance, small representations of regular languages have also strong practical motivations, e.g., lexicons and spelling checkers. A natural process

[1]Supported by the Natural Science and Engineering Research Council of Canada grant DGP-I249600. Corresponding author
[2]Supported by the Natural Science and Engineering Research Council of Canada grant OGP0041630.

of reducing the size of a finite automaton is by merging states without changing the accepting power of the automaton. This can be efficiently done in a deterministic finite automaton (DFA). Not as easy is the case of nondeterministic finite automata (NFA), where one can have non-minimal machines which have no mergible states and where the notion of merging states is not a clear cut. Merging states that have various relations has been studied for the simplification of NFA in quite a number of papers, e.g., [4, 5, 2]. Despite the difficulties concerning merging states in NFA, in [1] we proved that one cannot "grow" an NFA indefinitely and still avoid mergible states. However, our work was partial, in the sense that it treated only one way of merging states: collapsing one state into another and consolidating all input and output transitions. By this type of merging, one should only make sure that no extra words are added to the language. It is important to notice that the relation among states given by this type of merging is symmetrical, i.e., if we can merge p into q, we can merge q into p as well. This property allow us to extend the notion to groups of mergible states.

Here we continue that endeavor by studying a different type of state merging. According to this second method, in an NFA a state p can be merged into a state q if by redirecting all input transitions of p to q and eliminating p together with all its output transitions, the accepted language does not change. Considering that the initial state has a "free-starting" input transition and all final states have a "free-ending" output transition, the case when such states are involved in a merging is self-explained. It is worth mentioning that according to this asymmetric definition, it is somehow difficult to define the notion of groups of mergible states.

In this paper we are trying to solve the same problem as in [1], however, the second method of merging is involved this time. We state the main problem of the paper in the following:

`Problem.` Does there exist a regular language L such that there is an infinite number of distinct NFA for L that have no mergible states? Provide a constructive proof if the answer is positive.

We have found that, surprisingly, the situation turns quite the opposite to what we found in [1]. In the next section, we present basic concepts used in this paper. In Section 3, we give a positive answer to the above question and construct an NFA, with no mergible states, accepting the simple language represented by $(1+1')(0+1+0'+1')^*$, while the size of the NFA is p^3+1 with p being an arbitrary large prime number.

2. Basic Notions and Notations

We assume the reader to be familiar with basic notions of formal language theory, particularly with finite automata concepts. Here we briefly recall some definitions and introduce basic notations used in this paper. For further details concerning notions and notations, the reader is referred to [3, 6].

For a set T, $\#T$ is the number of elements of T. The set of words over a finite alphabet Σ is denoted by Σ^*, and the empty word is λ. The length of a word $w \in \Sigma^*$ is denoted by $|w|$. A deterministic finite automaton (DFA) is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite and non-empty set of states and Σ is the finite and non-empty input alphabet, $q_0 \in Q$, $F \subseteq Q$, and $\delta : Q \times \Sigma \to Q$ is the transition function. We can extend δ from $Q \times \Sigma$ to $Q \times \Sigma^*$ by:

$$\overline{\delta}(s,\lambda) = s, \text{ and } \overline{\delta}(s,wa) = \delta(\overline{\delta}(s,w),a),$$

where $a \in \Sigma$ and $w \in \Sigma^*$. We usually denote $\overline{\delta}$ just by δ for simplicity. The language recognized by the automaton A is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. Two automata are equivalent if they recognize the same language.

A nondeterministic finite automaton (NFA) is a quintuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q, $q_0 \in Q$ and F are the same as for *DFA*, and $\delta : Q \times \Sigma \to 2^Q$ is a transition function. We can extend δ from $Q \times \Sigma$ to $2^Q \times \Sigma^*$ by:

$$\overline{\delta}(S, \lambda) = S, \text{ and } \overline{\delta}(S, wa) = \bigcup_{s \in \overline{\delta}(S,w)} \delta(s, a),$$

Yet again, we denote $\overline{\delta}$ by δ for simplicity. The language recognized by the automaton A is $L(A) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset\}$.

Definition 1 *In an NFA $A = (Q, \Sigma, \delta, q_0, F)$ we say that p is mergible to q ($p \preceq q$), for $p, q \in Q$, if the automaton $A' = (Q - \{p\}, \Sigma, \delta', q'_0, F - \{p\})$ is equivalent to A, where:*

1. *$q'_0 = q_0$ if $p \neq q_0$, and $q'_0 = q$ if $p = q_0$,*
2. $\delta'(s, a) = \begin{cases} \delta(s, a) & \text{if } p \notin \delta(s, a) \\ (\delta(s, a) - \{p\}) \cup \{q\} & \text{if } p \in \delta(s, a) \end{cases}$.

For two languages $L_1, L_2 \subseteq \Sigma^*$, we denote

$$L_1 L_2 = \{w \mid w = xy, x \in L_1, y \in L_2\}.$$

If L_1 or L_2 is a singleton language, we can write the word instead of the language, e.g., wL_2 instead of $\{w\}L_2$.

Two particular alphabets are used in the paper: $\Sigma = \{0, 1\}$ and $\Sigma' = \{0', 1'\}$. By h we denote the projection of $\Sigma \cup \Sigma'$ onto Σ, i.e., $h(0) = 0, h(1) = 1, h(0') = h(1') = \lambda$. Similarly, we define the projection h' of $\Sigma \cup \Sigma'$ onto Σ'. For a word $u \in \Sigma^*$ we denote by $(u)'$ the word obtained by applying the isomorphism $(0)' = 0', (1)' = 1'$ to w. Similarly, if $v \in \Sigma'^*$, then $(v)''$ is the word obtained by applying the inverse isomorphism $(0')'' = 0, (1')'' = 1$.

For a string $s \in 1\{0, 1\}^*$, $value(s)$ denotes the integer value of s considered as a number in base 2. For $n \in \mathbb{N}$, $n_{(2)}$ is the binary representation of the number n (without leading zeroes). We extend the definition to include the special case 0: $value(0) = 0$ and $0_{(2)} = 0$.

For all $s \in 1\{0, 1\}^*$ and $n \in \mathbb{N}$ we have the following relations:

1. $value(n_{(2)}) = n$,
2. $(value(s))_{(2)} = s$.

For $x, y \in \mathbb{N}$, $value(x_{(2)} y_{(2)}) = x 2^{|y_{(2)}|} + y$. Also, if $x \in 1\{0, 1\}^*$ and $y \in \{0, 1\}^*$, $value(xy) = value(x) 2^{|y|} + value(y)$, where $value(\lambda) = 0$ and $value(0z) = value(z)$ for any binary word z ($z \in \{0, 1\}$).

For $p > 2$, the set $\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}$ is the set of remainders modulo p. Recall that $(\mathbb{Z}_p, +, \cdot, 0, 1)$ is a ring, and that when p is prime $\mathbb{Z}_p$ becomes a field (any non-zero element has an inverse with respect to multiplication).

3. Avoiding Mergibility in Large NFA

Our purpose is to construct an infinite family of equivalent NFA (hence containing arbitrary large NFA) such that each automaton in the family has no mergible states, in the sense of Definition 1. For this purpose, we first prove the following number-theoretic lemma.

Lemma 1 *For any prime $p > 5$ the following property holds:*

for all $0 \le r, r', i, j < p$ *with* $r \neq r'$, *there exits* $x \in \mathbb{N}$ *such that*

1. $r \cdot 2^{|x_{(2)}|} + x \equiv i(\mod p)$ *and*
2. $r' \cdot 2^{|x_{(2)}|} + x \not\equiv j(\mod p)$.

Proof. Let p be a prime number with $p > 5$. Let i, j, r, r' be arbitrary numbers such that $0 \le i, j, r, r' < p$ and $r \neq r'$. We need to find $x \in \mathbb{N}$ satisfying conditions 1 and 2. Take $t = |p_{(2)}| + 1$. Because $p > 5$, one can easily check that $\#\{x \in \mathbb{N} \mid |x_{(2)}| = t\} = 2^t > 2p$. Therefore, for every $m \ge t$, one can find at least p consecutive numbers $(x_i)_{0 \le i < p}$ $(x_i + 1 = x_{i+1})$ of length m with $x_i \equiv i(\mod p)$. We take $x = x_{(p - r2^m + i)(\mod p)}$ (a parameterized choice, with parameter m), so $x \equiv p - r2^m + i(\mod p)$, i.e., $r \cdot 2^{|x_{(2)}|} + x \equiv i(\mod p)$.

Assume that $r' \cdot 2^{|x_{(2)}|} + x \equiv j(\mod p)$. Then $r'2^m + p - r2^m + i \equiv j(\mod p)$, equivalent to $2^m(r' - r) \equiv (j - i)(\mod p)$. Since p is prime, $\mathbb{Z}_p$ is a field, and because $r \neq r'$, $r' - r \neq 0$, it has an inverse in $\mathbb{Z}_p$. Hence, $2^m \equiv (j - i)(r' - r)^{-1}(\mod p)$.

Since $m \ge t$ is the length of the chosen x and $g.c.d.(2, p) = 1$, we have $\{2^m(\mod p) \mid m \ge t\} = \mathbb{Z}_p \setminus \{0\}$. This proves that the congruence $2^m \equiv (j - i)(r' - r)^{-1}(\mod p)$ can not be satisfied for all $m \ge t$ since the right hand side is a fixed value, whereas the left hand side traverses $\mathbb{Z}_p \setminus \{0\}$. In particular, if we choose an m such that the congruence does not hold, the corresponding x, of length m, satisfies conditions 1 and 2. □

Remark 1
1. One can observe that the above lemma holds for all values $0 \le i, j < p$, including the case $i = j$.
2. If $r = r'$, but $i \neq j$, the existence of an x satisfying the conditions 1 and 2 of Lemma 1 is obvious.
3. The value of m can be as small as $t + 1$, since 2^m has two different values for $m = t$ and $m = t + 1$. In this case we can choose an x with $|x_{(2)}| \le t + 1 < p - 1$.

Anticipating our construction, we aim at building a "three-dimensional" automaton, consisting of a cubic net of states and transitions. The position of its final states will be given by a structure supported by the following property:

Lemma 2 *For an arbitrary $p \ge 1$, we can construct p square matrices $(A_k)_{0 \le k < p}$ of size $p \times p$ with component values in $\{0, 1\}$ such that:*

1. *for arbitrary i, j, $0 \le i, j < p$, if $A_k[i, j] = 1$ for some k, $0 \le k < p$, then $A_l[i, j] = 0$ for all $l \neq k$, $A_k[q, j] = 0$ for all $q \neq i$, and $A_k[i, t] = 0$ for all $t \neq j$;*
2. *for every i, j, with $0 \le i, j < p$, there exists an index k, $0 \le k < p$, such that $A_k[i, j] = 1$.*

Consequently, the total number of matrix components having value 1 *is* p^2.

Proof. For $p = 1$, the lemma holds trivially ($A_0[0,0] = 1$). We now consider the cases when $p \geq 2$. Clearly, there are p permutations $t_0, t_1, \ldots, t_{p-1}$, of the numbers $0, 1, \ldots, p-1$ such that the numbers at each position of the p permutations also form a permutation of the p numbers $0, 1, \ldots, p-1$. That is, for each i, $0 \leq i < p$, $(t_{0,i}, t_{1,i}, \ldots, t_{p-1,i})$ is also a permutation. For example, let $p = 3$. We have 3 permutations of the three numbers $0, 1, 2$: $(0,1,2), (2,0,1), (1,2,0)$, where the numbers at each position of the three permutations also form a permutation, $(0,2,1)$ at position 1, $(1,0,2)$ at position 2, and $(2,1,0)$ at position 3. For an arbitrary p, such p permutations can be obtained simply by first choosing an arbitrary permutation, and then circularly shifting it for $p-1$ times to the right.

We associate each of such p permutations to one of the p square matrices as follows. Let the permutation be $(k_0, k_1, \ldots, k_{p-1})$ and the matrix $A[i, j]$, $i, j = 0, \ldots, p-1$. Then we assign $A[0, k_0] = 1$, $A[1, k_1] = 1$, ..., $A[p\text{-}1, k_{p-1}] = 1$, and all other entries 0.

We claim that the p square matrices, A_i, $0 \leq i < p$, satisfy the two conditions in the lemma. Since for each position i of the permutations, $(t_{0,i}, t_{1,i}, \ldots, t_{p-1,i})$ is also a permutation, if $A_k[i, j] = 1$, clearly $A_l[i, j] \neq 1$ for all $l \neq k$. It is also clear that $A_k[i,t] \neq 1$ and $A_k[q, j] \neq 1$ for all $t \neq j$ and $q \neq i$, because of the way we link the permutation to the matrix. Also, because the values at each position of the permutations is also a permutation of all the p numbers, for each row i and each column j, there is a k such that $A_k[i, j] = 1$. □

Let p be a prime number and take the matrices $(A_k)_{0 \leq k < p}$, as described in Lemma 2. To each matrix A_k we associate the following DFA: $D_k = (Q_k, \Sigma \cup \Sigma', \delta_k, s, F_k)$, given by

- $Q_k = \{< k, r, i > \mid r, i \in Z_p\} \cup \{s\}$, $F_k = \{< k, r, i > \mid A_k[r, i] = 1\}$,
- for all $0 \leq r, i < p$, δ_k is given by the following formulae:
  $\delta_k(< k, r, i >, a) = < k, value(r_{(2)}a)(\mod p), i >$, for all $a \in \{0, 1\}$,
  $\delta_k(< k, r, i >, a) = < k, r, value(i_{(2)}a)(\mod p) >$, for all $a \in \{0', 1'\}$,
- $\delta(s, 1) = < k, 1, 0 >$ and $\delta(s, 1') = < k, 0, 1 >$.

Since the only final states of D_k are those given by the characteristic array A_k, we have that

$$\begin{aligned} L(D_k) = \{x \in \{1,1'\}\{0,1,0',1'\}^* \mid\, & value(h(x)) \equiv r(\mod p)\ , \\ & value(h'(x)) \equiv i(\mod p), \text{ and } A_k[r, i] = 1\}. \end{aligned}$$

We recall that h is the projection onto Σ and h' is the projection onto Σ'. An example of automaton D_k for $p = 5$ and $A = I$ (the unit matrix) is depicted in Figure 1. Notice that these automata are minimal, for $k \neq l$ the states of D_k and D_l are disjoint, and D_k and D_l accept disjoint languages.

Our purpose is to construct an automaton based on D_k as subautomata (building blocks). Consider the automaton $N_p = \left(\bigcup_{k=0}^{p-1} Q_k, \Sigma \cup \Sigma', \delta, s, \bigcup_{k=0}^{p-1} F_k\right)$, where the transition relation δ is the union of all transition functions δ_k. Certainly, N_p is nondeterministic. It follows that

$$\delta(s, w) = \Big\{ < k, value(h(w))(\mod p), value(h'(w))(\mod p) > \mid 0 \leq k < p \Big\},$$

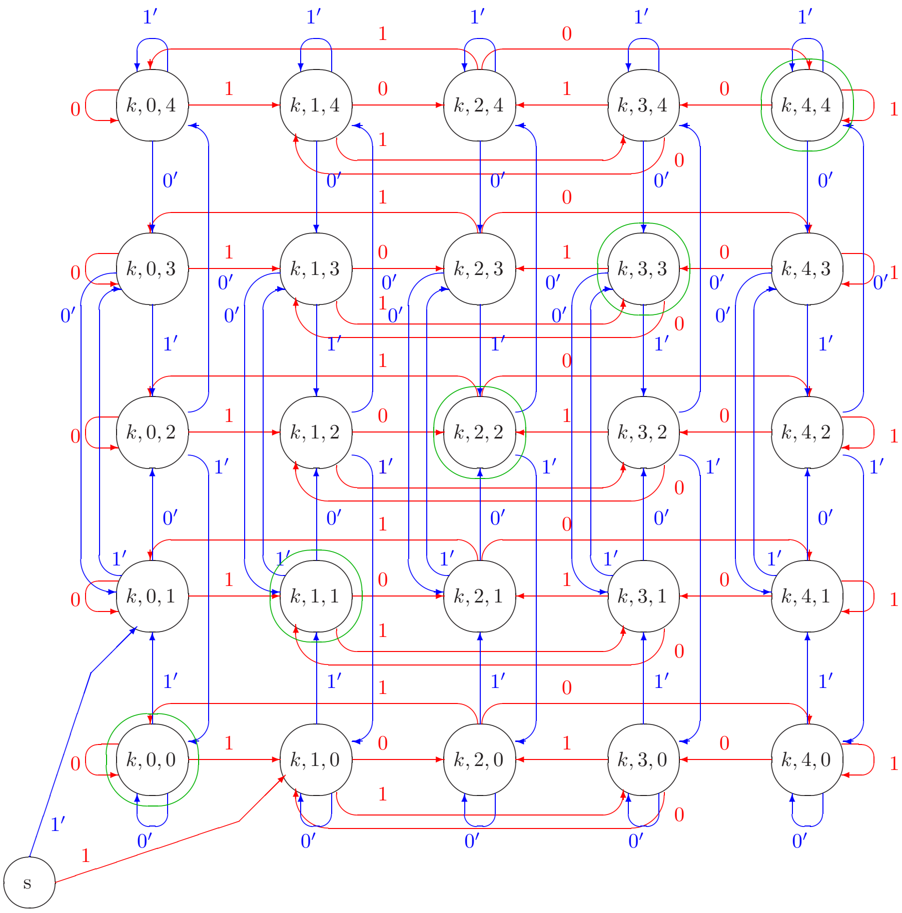

Figure 1: Example of automaton D_k for $p = 5$ and $A_k = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

for all $w \in \{1,1'\}\{0,1,0',1'\}^*$. One can easily check that

$$L(N_p) = \{1,1'\}\{0,1,0',1'\}^* = \bigcup_{k=0}^{p-1} L(D_k) \ .$$

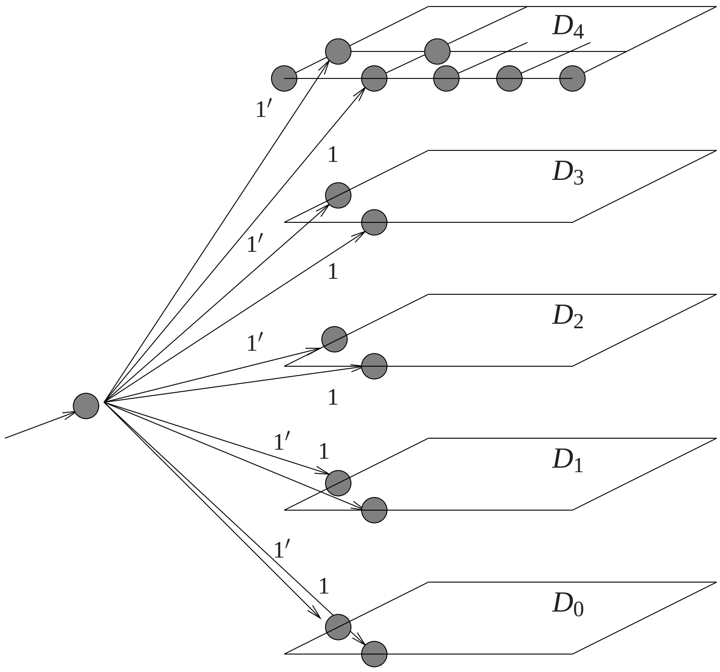

Figure 2: The NFA N_p, for $p = 5$

Theorem 3 *For any prime* $p > 5$*, the automaton* N_p *has no mergible states.*

Proof. Assume by contradiction that N_p contains two mergible states q and q', with $q \preceq q'$ (q is the state that is merged into q', i.e., it disappears after the merging). Denote by N'_p, the new automaton obtained from N_p by merging q into q', and let $L' = L(N'_p)$. We distinguish the following cases:

1. $q = s,\ q' =< k, r', j >,\ 0 \le k, r', j < p$;
2. $q' = s,\ q =< k, r, i >\ , 0 \le k, r, i < p$;
3. $q =< k, r, i >,\ q' =< k, r', j >,\ 0 \le k, r, i, j < p$ and $i \ne j$ or $r \ne r'$;
4. $q =< k, r, i >,\ q' =< l, r', j >,\ 0 \le r, r', i, j, k, l < p$ and $k \ne l$.

`Case` $q = s,\ q' =< k, r', j >,\ 0 \le k, r', j < p$.

In this case state s and its output transitions disappear, and state $< k,r',j >$ becomes the start state. Then, the states in Q_k are the only ones that are reachable and, even more, the new automaton will accept words starting with 0. Clearly, in this case N'_p and N_p are not equivalent.

`Case` $q' = s,\ \ q =< k,r,i >\ \ ,0 \le k,r,i < p.$

By the definition of $\{A_k\}_k$, there exists an index $u \in \{0,...,p-1\}$ such that $A_k[u,i] = 1$. Then, we first prove that there exists a word $0x \in \Sigma^*$ such that $\delta_k(< k,r,i >,0x) =< k,u,i >$. Indeed, if $t = |p_{(2)}|+1$, there exist at least p consecutive words of length t. Then x may be any word of length t such that $h(0x) \equiv u(\mod p)$.

Let $y = r_{(2)}(i_{(2)})'$ if $r \neq 0$ and $y = p_{(2)}(i_{(2)})'$ otherwise; we recall that the operator $()'$ changes all 0 into $0'$ and all 1 into $1'$. By definition, y starts with a 1. We have that $\delta(s,y0x)\cap Q_k = \delta(< k,r,i >,0x) =< k,u,i >$, which is a final state. Consequently, $y0x$ is accepted by N_p.

Our aim is to show that N'_p does not recognize $y0x$, proving that N_p and N'_p are not equivalent. Indeed, notice that after the merging of $< k,r,i >$ into s, the transition from $< k,r,i >$ labeled with 0 is lost, and s has no output transitions labeled 0 ($\delta'(s,0) = \emptyset$), i.e., $\delta'(s,y0x) = \{< l,u,i >|\ l \neq k\}$. Since $A_l[u,i] = 0$, for all $l \neq k$, it follows $y0x \notin L'$, hence N_p and N'_p are not equivalent.

`Case` $q =< k,r,i >,\ \ q' =< k,r',j >,\ \ 0 \le k,r,i,j < p$ and $i \neq j$ or $r \neq r'$.

Without loss of generality, we assume that $i \neq j$ (the other case, when $r \neq r'$, is proved symmetrically).

Let u be such that $A_k[u,i] = 1$ and v such that $A_k[v,j] = 1$. Using the definition of A_k, $u \neq v$. We apply Lemma 1 (and Remark 1.1 if $r = r'$), for r,r',u,v and we find an $x \in \mathbb{N}$ such that $2^{|x_{(2)}|}r + x \equiv u(\mod p)$ and $2^{|x_{(2)}|}r' + x \not\equiv v(\mod p)$.

Let $y = r_{(2)}(i_{(2)})'$ if $r \neq 0$, and $y = p_{(2)}(i_{(2)})'$ otherwise. We have that $\delta(s,yx_{(2)})\cap Q_k = \delta_k(< k,r,i >,x_{(2)}) =< k,u,i >$ which is final in N_p. Therefore, $yx_{(2)}$ is accepted by N_p. Since $x_{(2)} \in \{0,1\}^*$, we have that $\delta_k(< k,r',j >,x_{(2)}) =< k,z,j >$ for some $z \in \{0,1,\dots,p-1\}$. After the state merge, this situation remains unchanged: $\delta'(< k,r',j >,x_{(2)}) =< k,z,j >$.

Now, $\delta'(s,yx_{(2)}) = \{< l,u,i >|\ l \neq k\} \cup \{< k,z,j >\}$. Using the fact that $2^{|x_{(2)}|}r' + x \not\equiv v(\mod p)$, it follows that $\delta'(s,yx_{(2)})\cap F' = \emptyset$ ($A_l[u,i] = 0$ for all $l \neq k$ and $A_k[z,j] = 0$ since $z \neq v$).

`Case` $q =< k,r,i >,\ \ q' =< l,r',j >,\ \ 0 \le r,r',i,j,k,l < p$ and $k \neq l$.

The situations when $r \neq r'$ or $i \neq j$ are proved exactly as in Case 3. Assume that $r = r'$ and $i = j$, therefore we merge $< k,r,i >$ into $< l,r,i >$.

Let u be such that $A_k[u,i] = 1$ and v such that $A_l[v,i] = 1$. Using the definition of $\{A_k\}_k$, we have $u \neq v$.

By Remark 1.2, we know that there exists $x \in \mathbb{N}$ such that $2^{|x_{(2)}|}r + x \equiv u(\mod p)$ and $2^{|x_{(2)}|}r + x \not\equiv v(\mod p)$. Let $y = r_{(2)}(i_{(2)})'$ if $r \neq 0$, and $y = p_{(2)}(i_{(2)})'$ otherwise.

We have that $\delta(s,yx_{(2)})\cap Q_k = \delta(< k,r,i >,x) =< k,u,i >$. But $< k,u,i >$ is final because $A_k[u,i] = 1$, therefore $yx_{(2)}$ is accepted by N_p.

Since $x_{(2)} \in \{0,1\}^*$, we have that $\delta(< l,r,i >, x) =< l,z,i >$ for some $z \in \mathbb{Z}_p$. After the state merge, we will still have $\delta'(< l,r,i >, x) =< l,z,i >$. Since $2^{|x_{(2)}|}r + x \not\equiv v(\mod p)$, it follows that $z \neq v$.

We have that $\delta'(s, yx_{(2)}) = \{< m,r,i >| \ m \neq k\} \cup \{l,z,i\}$. Using $A_m[r,i] = 0$ for all $m \neq k$ and $A_l[z,i] = 0$ (since $z \neq v$), it follows $\delta'(s, yx_{(2)})$ contains no states. Hence, $yx_{(2)}$ is not accepted by N'_p.

In the previous cases we did not make explicit reference to the situation when one of the states involved in merging is final. It can be easily seen that the proof holds for these cases, as well. This concludes the proof.

□

4. Conclusion

In this paper we have completed a study on the mergibility of states in NFA, started in [1]. We have distinguished two main ways of merging states: (1) a `weak` method, where two states are merged by simply collapsing one into the other and consolidate all their input and output transitions, and (2) a `strong` method, where one state is merged into another one by redirecting its input transitions toward the other state and completely deleting it and all its output transitions. In [1] we have proven the following result:

`(the weak mergibility)` Let L be an arbitrary regular language and $k \geq 2$ an arbitrary integer. It does exist (effectively) a constant $E_{L,k}$ such that any *ε-NFA* of size at least $E_{L,k}$ has at least k weakly mergible states.

In the present paper we have supplemented the above result with the following:

`(the strong mergibility)` There exist regular languages L for which one may have an infinite number of NFA (consequently, of arbitrary large size) having no pairs of strongly mergible states.

Specifically, we have given a construction of equivalent automata $\{N_p\}_{p:prim}$ where no pair of states are mergible. Left for future work is to define the notion of groups of weak mergible states and to study the above properties for groups of a parameterized size. Notice carefully that the non-existence of pairs of mergible states in an NFA does not imply the non-existence of groups of mergible states with more than two states, in general.

References

[1] C. CÂMPEANU, N. SÂNTEAN, AND S. YU, Mergible States in Large NFA. *Theoretical Computer Science*, **330** (1) (2005) 23–34.

[2] J.-M. CHAMPARNAUD AND F. COULON. NFA reduction algorithms by means of regular inequalities. *Theoretical Computer Science*, **327** (2004) 241-253.

[3] J.E. HOPCROFT, J.D. ULLMAN, *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley, Reading Mass, 1979.

[4] L. ILIE AND S. YU, Algorithm for computing small NFAs. *Lecture Notes in Computer Science*, **2420** Springer-Verlag, 2002, 328-340.

[5] L. ILIE AND S. YU, Reducing NFA by invariant equivalences. *Theoretical Computer Science*, **306** (1-3) (2003) 373–390.

[6] S. YU, Regular Languages. In: A. SALOMAA AND G. ROZENBERG (ed.), *Handbook of Formal Languages*, Springer Verlag, 1997, 41–110.