Theoretical Computer Science 410 (2009) 2972–2981

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

On the limits of the communication complexity technique for proving lower bounds on the size of minimal NFA's☆

Juraj Hromkovič[a,*], Holger Petersen[b], Georg Schnitger[c]

[a] Department of Computer Science, ETH Zurich, ETH Zentrum, CH-8022 Zurich, Switzerland

[b] FMI, Universität Stuttgart, Universitätsstraße 38, D-70569 Stuttgart, Germany

[c] Department of Computer Science, Johann-Wolfgang-Goethe Universität, Robert Mayer-Strasse 11–15, D-60325 Frankfurt a. M., Germany

ARTICLE INFO

Keywords:
Communication complexity
Descriptional complexity
Finite automata
Nondeterminism

ABSTRACT

In contrast to the minimization of deterministic finite automata (DFA's), the task of constructing a minimal nondeterministic finite automaton (NFA) for a given NFA is PSPACE-complete. Moreover, there are no polynomial approximation algorithms with a constant approximation ratio for estimating the number of states of minimal NFA's.

Since one is unable to efficiently estimate the size of a minimal NFA in an efficient way, one should ask at least for developing mathematical proof methods that help to prove good lower bounds on the size of a minimal NFA for a given regular language. Here we consider the robust and most successful lower bound proof technique that is based on communication complexity. In this paper it is proved that even a strong generalization of this method fails for some concrete regular languages.

"To fail" is considered here in a very strong sense. There is an exponential gap between the size of a minimal NFA and the achievable lower bound for a specific sequence of regular languages.

The generalization of the concept of communication protocols is also strong here. It is shown that cutting the input word into $2^{O(n^{1/4})}$ pieces for a size n of a minimal nondeterministic finite automaton and investigating the necessary communication transfer between these pieces as parties of a multiparty protocol does not suffice to get good lower bounds on the size of minimal nondeterministic automata. It seems that for some regular languages one cannot really abstract from the automata model that cuts the input words into particular symbols of the alphabet and reads them one by one using its input head.

1. Introduction

The minimization of nondeterministic finite automata [30] is a hard computational problem [25]. The same is true even if one strives only to approximately estimate the size of a minimal NFA [12,15,13,14]. Hence, searching for small NFA's cannot be automated in an efficient way and so one has to consider estimating the size of minimal NFA's as a research problem for each particular regular language. This raises the question whether there exists a robust mathematical method that could be used to prove at least some tight lower bounds on the sizes of minimal NFA's, if its potential is explored in the right way

☆ The work on this paper was supported by SNF-grant 200020-120073/1, DFG-grant SCHN 503/4-1 and was done during the stay of the second author at ETH Zurich.

* Corresponding author.
E-mail address: juraj.hromkovic@inf.ethz.ch (J. Hromkovič).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.03.020

[14,1,2,31–33,16,19,8,3]. The first goal of this paper is to give an overview about some combinatorial methods for proving lower bounds on the descriptional complexity measures of minimal NFA's, and to show that they are covered by the robust approach [16,19] based on communication complexity of two-party protocols [29,34]. More precisely, we show that the fooling set proposed by Birget [5,6] and Glaister and Shallit [11] is covered as a special case of the communication complexity technique and that the geometrical approach of Courcelle, Niwinski and Podelski [8] corresponds to the communication complexity of two-party protocols.

The communication complexity of two-party protocols became a powerful instrument in proving lower bounds on different complexity measures [16,27]. The possibility to apply communication complexity for proving lower bounds on the size of finite automata was considered for the first time in [19] in 1986. Generalizing two-party protocols to a uniform computing model in [24,10] the usefulness for proving lower bounds on the size of finite automata grew and resulted in proving an at most quadratic relation between the sizes of Las Vegas finite automata and the deterministic ones. The communication complexity technique became especially successful in proving exponential gaps between nondeterministic finite automata with restricted ambiguity. First, communication complexity was used to simplify the arguments that provide an exponential gap between polynomial ambiguity and exponential ambiguity [20,21], and to prove an exponential gap between any fixed constant ambiguity and polynomial ambiguity. Recently, Hromkovič and Schnitger [22] proved an exponential gap between $O(1)$-ambiguity and linear ambiguity as well as exponential gaps between $O(n^d)$-ambiguity and $O(n^{d+1})$ ambiguity, and solved long-standing open problems in this way.

In spite of the fact that no other technique for proving lower bounds on the size of nondeterministic automata was comparably successful as the communication complexity of two-party protocols, Adorna [1] showed that for some regular languages this technique is unable to provide good lower bounds on the size of nondeterministic finite automata. Adorna presented a language whose two-party communication complexity is exponential in the size of minimal nondeterministic automata. This drawback of the two-party protocols can be overcome by taking the three-party protocols. But Adorna showed in [2] that, for any fixed positive integer $k \geq 3$, the communication complexity of the k-party protocols can be exponential in the size of the corresponding minimal nondeterministic finite automata.

The second goal of this paper is to show that the situation is still worse. Let, for any regular language L, $ns(L)$ denote the number of states of minimal nondeterministic finite automata accepting L. We prove that there is a sequence $\{L_n\}_{n=1}^{\infty}$ of regular languages such that the communication complexity of $2^{o(ns(L)^{1/4})}$-party protocols is essentially smaller than $ns(L_n)$. This means that exponentially many parties in the size of minimal nondeterministic automata do not help to get tight lower bounds on $ns(L)$.

This paper is organized as follows. In Section 2 we introduce two-party communication protocols of Yao [34] and show that the one-way version of them is related to finite automata [23,19]. Then we extend one-way two-party protocols to a uniform computing model [24], and show that their communication complexity provide lower bounds that are at least as good as the lower bounds provided by the fooling method [11,6] or the geometrical approach [8].

In Section 3 it is shown that for some regular languages the communication complexity of one-way two-party protocols is not helpful for proving lower bounds on $ns(L)$ [1] and the concept of multi-party protocols is introduced. Then we present the result of Adorna that, for any integer $k \geq 3$, k-party protocols provide exponentially higher lower bounds than $(k-1)$-party protocols for some regular languages. We give a more transparent proof of this fact than the proof presented in the original paper [2].

Section 4 is devoted to the original contribution of this paper. We show that there exists a sequence $\{L_n\}_{n=1}^{\infty}$ of regular languages such that, for any polynomial p, $p(ns(L_n))$-party protocols provide only a lower bound of size $O(\log_2(ns(L))$ on $ns(L)$. Finally, we discuss the consequences of this result for the possibility of developing combinatorial methods for proving lower bounds on $ns(L)$.

2. Communication complexity and proving lower bounds on the size of NFA's

Since there does not exist any efficient algorithm for estimating the minimal number of states of nondeterministic finite automata accepting a given regular language, one should ask for methods investigating the size of minimal NFA's. In 1986 communication complexity was proposed for this aim in [19]. Two-party protocols and their communication complexity were introduced by Yao [34] in order to measure the amount of information exchange between different parts of distributed computing systems. The original two-party protocol is a non-uniform computing model for computing Boolean functions from $\{0, 1\}^{2n}$ to $\{0, 1\}$. It consists of two computers C_1 and C_2 [Fig. 1]. At the beginning C_1 gets the first half of the input bits and C_2 gets the second half. The computers are required to cooperate in order to compute the value of the function corresponding to their common input. They are allowed to communicate by exchanging binary messages and the complexity of their work is measured as the number of bits exchanged. The communication complexity of two-party protocols became one of the most powerful instruments for proving lower bounds on the complexity of various computing models computing concrete tasks (see surveys in [16,23,27]) as well as for investigating the relative power of determinism, nondeterminism and randomness [29,16,27].

A special version of protocols, called one-way two-party protocols can be used to prove lower bounds on the number of states of finite automata. One-way protocols are restricted in the sense, that C_1 is allowed to send only one binary message to C_2 and after that C_2 is required to compute the correct answer. Formally, the work of C_1 can be described by

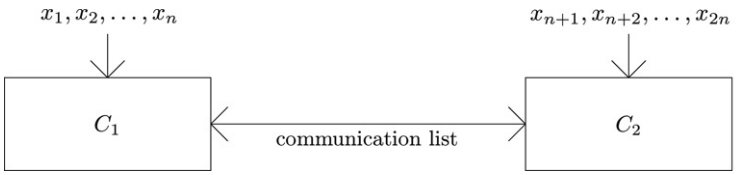

Fig. 1.

a function $C_1 : \{0, 1\}^n \to \{0, 1\}^*$, where $C_1(\alpha)$ is the binary message sent to C_2. The work of C_2 can be described by a function $C_2 : \{0, 1\}^n \times \{0, 1\}^* \to \{0, 1\}$. The arguments of C_2 are its n input bits and the message received, the output has to be the value of the computed Boolean function. If a one-way two-party protocol (C_1, C_2) computes a Boolean function $f : \{0, 1\}^{2n} \to \{0, 1\}$, then, for each argument $x_1, x_2, \ldots, x_{2n}$ of f, the computer C_1 gets the first half $x_1, \ldots, x_n$ of the input and the second computer C_2 gets the second half $x_{n+1}, \ldots, x_{2n}$ of the input. The computation of the protocol on the input $x_1, \ldots, x_{2n}$ is

$$D = C_1(x_1, \ldots, x_n)\#C_2(x_{n+1}, \ldots, x_{2n}, C_1(x_1, \ldots, x_n)).$$

The communication complexity of D is

$$cc_1(D) = |C_1(x_1, \ldots, x_n)|,$$

i.e., the length of the binary message $C_1(x_1, \ldots, x_n)$ sent from C_1 to C_2. The **one-way communication complexity** of (C_1, C_2) is

$$\begin{aligned} cc_1(C_1, C_2) &= \max\{cc_1(D) | D \text{ is a computation of } (C_1, C_2)\} \\ &= \max\{|C_1(x_1, \ldots, x_n)| x_1, \ldots, x_n \in \{0, 1\}^n\}. \end{aligned}$$

The **one-way communication complexity** of a Boolean function f, $cc_1(f)$, is the **communication complexity** of the best one-way protocol computing f.

One-way protocols can be considered for any computation mode such as determinism, nondeterminism or different kinds of randomization. Since each language can be viewed as an infinite sequence of finite functions, one can easily construct a sequence of one-way protocols that simulate the work of a given finite automaton as follows. For each input length m we have one separate one-way protocol $P_m = (C_{1,m}, C_{2,m})$. The computer $C_{1,m}$ with input α of length $\lfloor \frac{m}{2} \rfloor$ sends the binary code of the state q reached by the finite automaton after reading α from its initial state. Then, $C_{2,m}$ with its input β simulates the work of the finite automaton on the word β when starting in state q. If and only if the simulation finishes in an accepting state, $C_{2,m}$ outputs the value 1. One can easily observe that this way of simulating finite automata by one-way protocols works for any mode of computation. Since the communication complexity of all protocols simulating an automaton with a state set Q is at most $\lceil \log_2 |Q| \rceil$, one-way communication complexity provides a lower bound on the number of states of finite automata.

In other words, measuring the complexity of an one-way protocol (C_1, C_2) as the number $mc_1(n) = |\{C_1(x_1, \ldots, x_n) | x_1 \ldots x_n \in \{0, 1\}^n\}|$ of different messages used in all computations on all inputs of length n, the message complexity $mc_1(n)$ is a lower bound on the number of states of finite automata. There is one essential drawback of this lower bound technique however. The two-party protocol model is non-uniform, while automata are a uniform computing model. Due to this difference in modeling, the gap between the message complexity of one-way protocols and the number of states of finite automata can be arbitrarily large for some languages. For instance, the message complexity of regular languages over one-letter alphabets is only 1. Another example is the finite language $L_n = \{0^{2n}xx \mid x \in \{0, 1\}^n\}$, whose deterministic message complexity is 2, but the size of the minimal NFA's is at least 2^n.

In order to overcome this drawback we have introduced uniform one-way protocols in [24]. A uniform one-way protocol (C_1, C_2) consists again of two computers C_1 and C_2, but inputs $x = x_1, x_2, \ldots, x_n \in \Sigma^*$ for an alphabet Σ are arbitrarily divided into a prefix $x_1, \ldots, x_k$ as the input of C_1 and a suffix $x_{k+1}, \ldots, x_n$ as the input of C_2. Thus, C_1 is a function from Σ^* to $\{0, 1\}^*$ and C_2 is a function from $\Sigma^* \times \{0, 1\}^*$ to $\{0, 1\}$ where 0 means *reject* and 1 means *accept*. A uniform protocol is required to provide the correct answer for each of the possible $n + 1$ partitions ($k \in \{0, 1, \ldots, n\}$) of the word $x \in \Sigma^n$. This means that, for every word $x = x_1, \ldots, x_n \in \Sigma^*$,

$$\begin{aligned} C_2(x_1, \ldots, x_n, C_1(\lambda)) &= C_2(x_2, \ldots, x_n, C_1(x_1)) \\ &= C_2(x_3, \ldots, x_n, C_1(x_1, x_2)) \\ &\vdots \\ &= C_2(x_n, C_1(x_1, \ldots, x_{n-1})) \\ &= C_2(\lambda, C_1(x_1, \ldots, x_n)). \end{aligned}$$

The message complexity of (C_1, C_2) is

$$mc_1(C_1, C_2) = |\{C_1(w) | w \in \Sigma^*\}|.$$

For a concrete example of a uniform one-way protocol one can look at [24,1].

	λ	0	1	00	01	10	11	...	β	...
λ	a_λ	a_0	a_1	a_{00}	a_{01}					
0	a_0	a_{00}	a_{01}	a_{000}						
1	a_1	a_{10}	a_{11}	a_{100}						
00	a_{00}	a_{000}	a_{001}							
01	a_{01}									
10										
$\vdots$										
α									$a_{\alpha\beta}$	
$\vdots$										$\ddots$

Fig. 2.

Interestingly, for each regular language L, the message complexity of deterministic uniform one-way protocols accepting L equals the size of the minimal deterministic finite automaton for L. To see this fact one has to represent the task of accepting L as the infinite communication matrix M_L for L (Fig. 2).

Rows as well as columns of M_L are labeled by words from Σ^* in the canonical order starting with λ. M_L is a 0/1-matrix. The element $a_{\beta\alpha}$ of M_L in the intersection of the row α and the column β is 1 iff the word $\alpha\beta$ belongs to L. Observe, that, for each word $x \in \Sigma^n$, there are exactly $n+1$ elements in M_L corresponding to x. Hence, we cannot assign a language to any such 0/1-matrix.

Now, we are ready to argue that deterministic message complexity is equal to the size of the minimal deterministic FA for any regular language L. The computer C_1 is required to send a message to C_2. We claim that the number of messages needed is the number of different rows of M_L. Let $R_i = (r_{i1}, r_{i2}, \ldots)$ and $R_j = (r_{j1}, r_{j2}, \ldots)$ be different rows of M_L. For sure, there is a column k labeled by β_k in which they differ, i.e., $r_{ik} \neq r_{jk}$. If C_1 sends the same message m to C_2 for its inputs α_i and α_j corresponding to the rows R_i and R_j, then C_2 either has to accept both $\alpha_i\beta_k$ and $\alpha_j\beta_k$ or to reject both words $\alpha_i\beta_k$ and $\alpha_j\beta_k$ (The arguments of C_2 are the message m and the word β_k and these arguments are the same for inputs $\alpha_i\beta_k$ and $\alpha_j\beta_k$). Hence, the number of messages used is at least the number of different rows of M_L. Now, one can easily observe that the number of different rows of M_L is nothing else than the number of the equivalence classes of the Nerode relation for L and we are done (for a detailed argumentation see [24]).

Unfortunately, the situation for the nondeterministic mode of computation is completely different. A nondeterministic uniform one-way protocol (C_1, C_2) is a natural generalization of the deterministic one. The first computer C_1 has, for any input prefix x, a nondeterministic choice from a finite set of possible messages. The second computer C_2 can nondeterministically decide, whether it accepts or rejects. For any input $w \in L$ and any partition of w into $w = \alpha\beta$ a nondeterministic protocol for L must have at least one accepting computation on the input (α, β). Formally, C_1 can be viewed as a relation

$$C_1 \subseteq \Sigma^* \times \{0, 1\}^*,$$

and C_2 can be viewed as a relation

$$C_2 \subseteq (\Sigma^* \times \{0, 1\}^*) \times \{accept, reject\}.$$

Thus,

$$u\#a$$

is a computation on the partition (α, β) of an input $x = \alpha\beta$ iff

$$(\alpha, u) \in C_1 \text{ and } ((\beta, u), a) \in C_2.$$

If $a = accept$, then the computation is called an accepting one. The nondeterministic message complexity $nmc(C_1, C_2)$ of the protocol (C_1, C_2) is the number of all possible messages, i.e.,

$$nmc(C_1, C_2) = |\{u \in \{0, 1\}^* | (\alpha, u) \in C_1 \text{ for an } \alpha \in \Sigma^*\}|.$$

The nondeterministic message complexity $nmc(L)$ of a regular language L is the minimum of $nmc(C_1, C_2)$ over all nondeterministic protocols (C_1, C_2) accepting L. If one has an NFA A accepting L, then a nondeterministic one-way protocol (C_1, C_2) can simulate the work of A as follows. If (α, β) is a partition of an input $w = \alpha\beta$, then the set of states reachable by A by reading α from its initial state is exactly the set of messages (in a binary coding) C_1 is allowed to send to C_2 when getting the input part α. If the message sent corresponds to a state q and A can reach an accepting state working from q on β, then C_2 "answers" accept. Therefore

$$nmc(L) \leq ns(L).$$

		β_1		β_2		β_3	...	β_i	...
α_1		1		1		1		1	
α_2		1		1		1		1	
α_3		1		1		1		1	
$\vdots$									
α_i		1		1		1		1	
$\vdots$									

Fig. 3.

In what follows we omit the term "one-way" for nondeterministic protocols because one can prove [29] that allowing two-way communication does not change the nondeterministic message complexity of regular languages. The nondeterministic message complexity $nmc(L)$ can essentially differ from the size $ns(L)$ of the minimal NFA's for L. Additionally, it is not so easy to estimate $nmc(L)$ for a given regular language L. From communication complexity theory [16,27] we know that $nmc(L)$ is the minimal number of 1-monochromatic submatrices that cover all 1's in M_L.

One can visualize the argument as follows. The computations of a (nondeterministic) one-way protocol can be viewed as $m\#a$ for $a \in \{\text{accept}, \text{reject}\}$, where m is the message sent from C_1 to C_2. Consider a concrete accepting computation

$$m\#\text{accept}.$$

Let

$$(\alpha_1, \beta_1), (\alpha_2, \beta_2), (\alpha_3, \beta_3), \ldots, (\alpha_i, \beta_i), \ldots$$

be inputs with corresponding partitions (C_1 has α_i and C_2 has β_i) for which $m\#$accept is an accepting computation. Then this accepting computation accepts all words of L corresponding to the words on the intersections of

the rows $\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots$ and the columns $\beta_1, \beta_2, \ldots, \beta_i, \ldots$.

In Fig. 3 one immediately sees that this intersection of rows and columns determines unambiguously a 1-monochromatic submatrix of M_L.

To solve the combinatorial problem of covering all 1's of a matrix by the minimal number of potentially overlapping 1-monochromatic submatrices is not easy. One possibility to use this fact is to restrict M_L to a finite submatrix M'_L and then to estimate the largest 1-monochromatic submatrix S of M'_L. The number of 1's in M'_L divided by the number of 1's in S is a lower bound on the message complexity of L. Hence, we see that the geometrical approach [8] for proving lower bounds on $ns(L)$ corresponds to the communication complexity method based on two-party protocols.

Another lower bound technique is based on the so-called 1-fooling sets and this technique covers the approach proposed independently by [6] and Glaister and Shallit [11], who directly strived to prove lower bounds on the size of NFA's for concrete languages without using the concept of communication complexity. A one-way 1-fooling set for a language L is a finite subset A_L of the set of pairs

$$\{(\alpha, \beta) \mid \alpha\beta \in L\}$$

as introduced and discussed in [16,29], such that

$$\text{if } (\alpha, \beta) \in A_L \text{ and } (\gamma, \delta) \in A_L \text{ (i.e., } \alpha\beta \in L \text{ and } \gamma\delta \in L),$$

then

$$\alpha\delta \notin L \text{ or } \gamma\beta \notin L.$$

If A_L has this property, then for any two different elements (α, β) and (γ, δ) from A_L, each protocol accepting L has to send another message $C_1(\alpha)$ from C_1 to C_2 for α than the message $C_1(\alpha)$ for γ. One can argue for this fact as follows. Let, for any $w, v \in \Sigma^*$, $S(w, v)$ be the set of all messages submitted from C_1 to C_2 in all possible accepting computations on wv, where the input part of C_1 is w. If $S(\alpha, \beta) \cap S(\gamma, \delta) \neq \emptyset$ and $\alpha\beta$ is accepted, then unavoidably $\gamma\beta$ is accepted by the protocol as well. Analogously, if $S(\alpha, \beta) \cap S(\gamma, \delta) \neq \emptyset$ and $\gamma\delta$ is accepted, then $\alpha\delta$ is accepted too. If one wants to argue directly for finite automata, one can say that for any two elements (α, β) and (γ, δ) of A_L, and any NFA accepting L, each state q reached after reading α in an accepting computation on $\alpha\beta$ differs from any state p reached after reading δ in an accepting computation on $\delta\gamma$. Hence, the number of different messages (states) of a protocol (an NFA) accepting L must be at least $|A_L|$. For more details and a survey about methods for proving lower bounds on communication complexity we recommend [23,16,27,9,28,18,17].

		β		δ		
α		1		0		
γ		0		1		

Fig. 4.

To visualize the argument above one can consider Fig. 4. We see that the intersection of the rows α and γ and the columns β and δ does not build a 1-monochromatic matrix (because both (α, β) and (δ, γ) are elements of a one-way 1-fooling set). Hence, the 1's corresponding to (α, δ) and to (β, γ) cannot be covered by any 1-monochromatic submatrix because at least one of the elements $a_{\alpha\delta}$ and $a_{\gamma\beta}$ is zero. Therefore, the number of all monochromatic submatrices covering all 1's corresponding to the elements of the one-way 1-fooling sets must be at least the cardinality of the one-way 1-fooling set.

Note that the minimization algorithm of [26] is not directly related to the communication complexity approach. The rows and the columns of finite matrices in [26] correspond to overlapping sets of words instead of particular words and this changes the game of searching for a minimal cover of 1's.

3. Multiparty protocols for proving lower bounds on $ns(L)$

For most regular languages it is not easy to investigate their nondeterministic communication complexity in order to prove a lower bound on the size of minimal NFA's. The situation is still worse. A good lower bound on $nmc(L)$ need not to be a good lower bound for $ns(L)$. Consider the language

$$L_{(3,n)} = \{xyz \mid x, y, z \in \{0, 1\}^n, x = y \vee x \neq z\}.$$

In [21] an exponential difference between nondeterministic message complexity and the actual size of a minimal NFA is shown, namely

$$nmc(L_{(3,n)}) \in O(n^2) \text{ and } ns(L_{(3,n)}) \in 2^{\Omega(n)}.$$

Why is the message complexity $nmc(L_{(3,n)})$ small? If C_1 gets at least xy (i.e., the cut is inside of z), then C_1 can check whether $x = y$. If so, C_1 knows that the input word is in $L_{(3,n)}$. If $x \neq y$, C_1 guesses the position in which x and z differ and verifies it by sending to C_2 the order (index) and the value of this position. Additionally, C_1 sends the length of its input part to C_2. Hence, for this input partition, the protocol needs at most $O(n^2)$ messages.

If C_2 obtains at least the whole suffix yz (i.e., the cut is inside of x), then C_2 checks whether $y = z$. The main point is that if $y = z$, C_2 knows that the input xyz is in the language and accepts independently of the message received from C_1. If $y \neq z$, then the words outside the language $L_{(3,n)}$ must have the form

$$zyz$$

for $y \neq z$. To check the opposite for xyz it is again sufficient to check whether $x \neq z$, which is easy for nondeterministic computation models. Therefore, one needs again at most $O(n^2)$ messages in this case.

If the cut of xyz is somewhere inside y, one can check the property $x = y$ or $x \neq z$ in a similar way as described above for the other two cases. Observe, that one has to accept all words except xyx for $x \neq y$. To check $x \neq z$ in xyz with a cutpoint in y in a nondeterministic way is easy. To get xxx accepted the protocol accepts if C_1 sees that its part of y is a prefix of x and C_2 sees that its part of y is a suffix of z for an input xyz with a cutpoint in y. If $x = z$ then consequently the input is $xyz = xxx$. If $x \neq z$, we are allowed to accept, because in that case the input cannot be of the forbidden form xyx for $y \neq x$.

In that case C_1 sends the length of its input part, and the index of a position in x with the value on this position, and one bit more telling whether the possessed prefix of y is a prefix of x. Again, the number of possible messages is in $O(n^2)$. Altogether, $O(n^2)$ messages are enough to implement the protocol described above.

The main point is, that independently of the cutpoint, it is sufficient to verify the inequality of two strings and this is an existence task and all existence tasks are convenient for nondeterminism.

Now, let us argue that $ns(L_{(3,n)})$ is large. Observe, that $L_{(3,n)}$ contains all words $xxx \in \{0, 1\}^{3n}$. For each xxx fix an accepting computation $Com(xxx)$. Let $T\text{-}Com(x, x) = (p, q)$ be the trace of $Com(xxx)$ consisting of the state p reached after reading x and the state q read after reaching xx. Assume, for $x \neq y$,

$$T\text{-}Com(xxx) = T\text{-}Com(yyy) = (p, q).$$

One can easily observe that there exists an accepting computation $Com(xyx)$ on xyx with

$$T\text{-}Com(xyx) = T\text{-}Com(xxx) = T\text{-}Com(yyy),$$

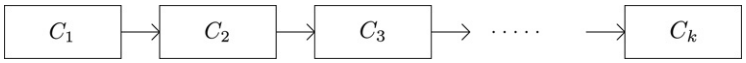

Fig. 5.

because q can be reached from p by reading x as well as y. Hence, the number of different tracks must be at least 2^n, which is the cardinality of $\{xxx \mid x \in \{0,1\}^n\}$. Hence, if Q is the state set of a NFA accepting L, then $|Q|^2 \geq 2^n$, i.e., $ns(L_{(3,n)}) \geq 2^{\frac{n}{2}}$.

The proof above shows that cutting the words into 3 parts may be helpful.

In other words we see that the size of the minimum cover of 1's in M_L can be essentially smaller than $ns(L)$. But considering a three-dimensional matrix corresponding to $\Sigma^* \times \Sigma^* \times \Sigma^*$ and trying to cover all 1's in this matrix by 1-monochromatic three-dimensional submatrices can help to get a better lower bound on $ns(L)$.

Motivated by this observation we proposed to introduce a generalization of two-party one-way communication protocols (see Fig. 5).

The uniform k-party communication protocol consists of k agents $A_1, \ldots, A_k$. Inputs $x = x_1 \cdots x_k \in \Sigma^*$ are arbitrarily divided into (possibly empty) substrings such that agent A_i receives substring x_i. The one-way communication starts with agent A_1 and ends with agent A_k who has to decide whether to accept or reject. Agent A_i, upon receiving a message m from its left neighbor A_{i-1}, determines a message m' based on m and its substring x_i and sends m' to its right neighbor A_{i+1}. Formally, A_1 is a function from Σ^* to $\{0,1\}^*$, and so $A_1(x_1) = m_1$ is the message sent from A_1 to A_2. For $i = 2, \ldots, k-1$, A_i is a function from $\Sigma^* \times \{0,1\}^*$ to $\{0,1\}^*$. If m_{i-1} is the message sent from A_{i-1} to A_i, then $A_i(x_i, m_{i-1}) \in \{0,1\}^*$ is the message m_i sent from A_i to A_{i+1}. Finally A_k is a function from $\Sigma^* \times \{0,1\}^* \to \{accept, reject\}$, and so $A_k(x_k, m_{k-1})$ is the result of the computation. The whole computation on the partition $(x_1, x_2, \ldots, x_k)$ of an input $x = x_1, x_2, \ldots, x_k$ can be written as

$$m_1\#m_2\#\cdots\#m_{k-1}\#A_k(x_k, m_{k-1}).$$

Again, we require that for each partition of each input, the k-party protocol computes the correct answer.

We see that a k-party protocol can simulate the work of finite automata by taking the messages m_i as states reached after reading the prefix $x_1, x_2, \ldots, x_i$.

The message complexity of a protocol is the maximum, over all agents A_i, of the number of different messages sent in all computations. The different communication modes such as deterministic, probabilistic or nondeterministic communication are defined in the canonical way. For formal definitions we recommend reading Adorna [1,2].

Let, for any regular language L, $nmc_k(L)$ denote the message complexity of the best nondeterministic uniform k-party protocol. One can show that $nmc_k(L)$ is exactly the size of the minimum cover of all 1's of the k-dimensional matrix M_L^k labeled by $(\Sigma^*)^k$ by 1-monochromatic k-dimensional submatrices.

In [1] Adorna established an exponential gap between the message complexities $nmc_2(L_{(3,n)})$ and $nmc_3(L_{(3,n)})$, and in his PhD thesis [2] he showed an exponential gap between nmc_k and nmc_{k+1} for any $k \geq 2$. To do so he considered the language

$$L_{(k,n)} = \{x_1x_2\ldots x_k \mid x_i \in \{0,1\}^n \text{ for } i = 1, 2, \ldots, n \text{and } \exists i \in \{1, 2, \ldots, k-2\} \text{ such that } x_i = x_{i+1} \vee x_i \neq x_{i+2}\}$$

for any $k \geq 3$. To prove that

$$nmc_k(L_{(k,n)}) \geq 2^{\frac{n}{k-1}} \text{ (i.e., } ns(L_{(k,n)} \geq 2^{\frac{n}{k-1}})$$

one can argue similarly as for $L_{(3,n)}$. The set

$$\{x^k \mid x \in \{0,1\}^n\}$$

is a subset of $L_{(3,n)}$. If one partitions x^k into k pieces x and each agent A_i gets an x, then one can fix an accepting computation

$$Com(x) = m_1\#m_2\#\ldots\#m_{k-1}\#\text{accept}$$

of a nondeterministic uniform k-party protocol accepting $L_{(k,n)}$ on the input x^k, where m_i is the message sent by the agent A_i. If, for two different inputs $x^k \neq y^k$,

$$Com(x) = Com(y)$$

then $Com(x)$ is also an accepting computation on the word

$$xyxy\ldots x \text{ (if } k \text{ is odd) or } xyxy\ldots xy \text{ (if } k \text{ is even).}$$

But none of these two words belongs to $L_{(k,n)}$. Hence, the number of different accepting computations must be at least 2^n and consequently at least $(2^n)^{\frac{1}{k+1}} = 2^{\frac{n}{k+1}}$ different messages are needed.

To understand that $L_{(k,n)}$ is easy for nmc_{k-1} one has to observe that if a word does not belong to $L_{(k,n)}$ then it has the form

$$vuvuvu\ldots$$

for $v, u \in \{0,1\}^n$, and $v \neq u$. Since a nondeterministic protocol can verify a difference of x_i and x_{i+2} for any i easily, one needs only to think how to accept words x^k. If one agent gets two consecutive xx, then it immediately knows that the input is in $L_{(k,n)}$ and we are done. If x^k is partitioned into $k-1$ parts and none contains xx, then each agent getting wh or wxh for

a suffix w of x and a prefix h of x, compares all the positions i of $x = a_1a_2 \dots a_n$ for which it has a_i of two consecutive x's. One can prove by induction that all positions $i = 1, 2, \dots, n$ will be checked at least once by the agents during the whole computation. The protocol accepts x^k in the computations in which all attempts to prove $x_i \neq x_{i+2}$ failed and all internal comparisons of bits in x_i and x_{i+1} succeeded. This approach works because none of such computations can accept words of the form $uvuvuv \dots$ for $u \neq v$ and these words are the only ones not in $L_{(k,n)}$.

To manage all these comparisons the message of an agent A_j must on one side provide the exact information of the length of the prefix processed up until now and the order of the position to be compared in the test $x_i \neq x_{i+2}$ for some i. Hence, $O(n^2)$ messages suffice.

4. Limits of multiparty message complexity for proving lower bounds of the size of minimal NFA's

The result of Adorna shows, that there is no fixed k such that $nmc_k(L)$ is polynomially related to $ns(L)$ for each regular language. This result need not to be viewed as a failure in proving lower bounds on the size of NFA's for concrete regular languages. One can simply propose searching for a suitable k and then prove a lower bound on $nmc_k(L)$ for a given language L. Here we prove that the situation is still worse and that the idea above does not work. We strengthen the result by showing that there is a sequence of languages $\{L_n\}_{n\geq 1}$ such that values of k exponential in $ns(L_n)$ are not sufficient to get $nmc_k(L_n)$ tight to $ns(L_k)$.

To do so, for every positive integer n, we consider the unary language

$$L_n = \{1^\ell \mid \ell \neq n\}.$$

We work with k parties $A_1, \dots, A_k$. Agent A_1 nondeterministically selects a prime number $p \leq P_k$ for a parameter P_k to be determined later.

If agent A_i receives the substring 1^{m_i} and if $m_i > n$ or $m_i = n$ and the previous agents did not have all input λ, it sends a "too long" message to its right neighbor A_{i+1}, resp. passes a received "too long" message on to A_{i+1}. Otherwise, assuming that A_i has received a message $(p, m_1 + \dots + m_{i-1} \mod p)$ from its left neighbor A_{i-1}, it sends the message $(p, m_1 + \dots + m_i \mod p)$ to its right neighbor. A_k accepts the joint input if its suffix is too long or if it has received a "too long" message. In this way all words with length at least $k(n-1)+1$ are accepted, because for each partition of such long inputs at least one of the agents gets an input part of length n. Additionally it accepts, if

$$m_1 + \dots + m_k \not\equiv n \mod p.$$

The protocol accepts only strings from L_n, since it requires a proof that the joint input is different from 1^n.

When will all inputs 1^m with $1^m \neq 1^n$ be accepted? If

$$m = m_1 + \dots + m_k \text{ for } m_1, \dots, m_k < n,$$

then $m \leq k \cdot (n-1)$ and consequently

$$|m - n| \leq (k-1) \cdot n.$$

Thus if we require

$$(k-1) \cdot n < \Pi_{p \leq P_k} p, \tag{1}$$

then $m \equiv n \mod \Pi_{p\leq P_k} p$ implies that $\Pi_{p\leq P_k} p$ divides $m - n$ which is only possible if $m = n$. Thus the protocol is correct, since all strings in L_n are indeed accepted.

In summary, L_n can be accepted by the uniform k-party communication model even if each agent sends only $O(P_k^2)$ messages. Now a combination of the prime number theorem and Stirling's formula establishes that $P_k = O(\log_2(k \cdot n))$ suffices. A detailed argument of this fact is given in the proof below. As a consequence it turns out that at least $2^{\Omega(n^{1/4})}$ agents are required to increase message complexity up to $\Omega(\sqrt{n})$:

Theorem 1. *Let $L_n = \{1^\ell \mid \ell \neq n\}$. Then*

$$ns(L_n) = \Theta(\sqrt{n}),$$

and

$$L_n \text{ has } k\text{-party protocols with message complexity } O(\log_2^2(k \cdot n)).$$

In particular, even for $k = 2^{c \cdot n^{1/4}}$ agents, message complexity is smaller than state complexity, provided that c is sufficiently small.

Proof. We first give the proof of the lower bound for $ns(L_n)$. Let N_n be some unary NFA recognizing L_n with s states. We apply the Chrobak normal form for unary NFA's [7] and can assume that there is an equivalent NFA N_n' which consists of an initial path of length at most s^2 and subsequent cycles with at most s states altogether. But, if $s = o(\sqrt{n})$, then inputs 1^n and $1^{n+n!}$ are treated alike on each cycle although they have to be separated.

The upper bound is shown by describing automata M_n accepting the languages L_n. For $n \leq 3$ an automaton for L_n can be defined in a straightforward manner.

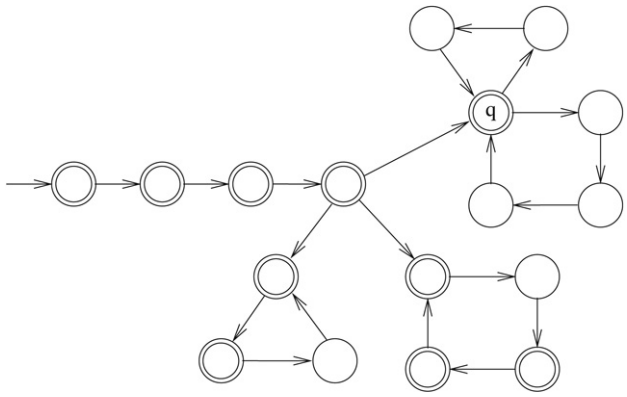

Fig. 6. M_9.

For $n \geq 4$ let $r = \lfloor \sqrt{n} \rfloor$. The main idea of the construction is to accept the complement $\bar{L}$ of a finite language L with a component of M_n having $\Theta(r)$ states, such that the word w of maximum length in L contains $r^2 - \Theta(r) = n - \Theta(\sqrt{n})$ symbols. In $\bar{L}$ some additional words are missing. These are accepted by two additional disjoint cycles of states. Another problem is, that w does not reach length n. We therefore include a path from the initial state to the three components outlined above.

More precisely, a path of $n - r^2 + r + 1$ accepting states starting from the initial state of M_n is connected to three other components. The first component consists of two loops of lengths r and $r + 1$ sharing exactly one final state q, while the other states of these loops are not final. The longest word not accepted starting from q is $w = 1^{r^2-r-1}$. To see this, we prove by induction on i that a word $1^{i(r+1)+j}$ with $0 \leq j \leq r$ is accepted if and only if $j = 0$ or $(r - i) \leq j$. This is true for $i = 0$, because the initial state q is accepting and by following the shorter loop 1^r is accepted. Now assume the claim holds for $i \geq 0$. All words of the form $1^{(i+1)(r+1)+j}$ with $j = 0$ or $(r - i) \leq j \leq r$ are accepted by following the longer loop from q after reading the prefix $1^{i(r+1)+j}$. The word $1^{(i+1)(r+1)+(r-i-1)}$ can be accepted by following the loop of length r after an accepting computation on $1^{i(r+1)+(r-i)}$. None of the words $1^{(i+1)(r+1)+j}$ with $1 \leq j < r - i - 1$ is in the language, since the words $1^{i(r+1)+j}$ are not accepted by the induction hypothesis.

There are in general words shorter than w not accepted by the first component. These can be covered with the help of two disjoint loops. The first loop has length r and accepts all words of the form 1^{ir+j} with $0 \leq j < r - 1$ from its state connected to the initial path. The second loop of length $r + 1$ accepts all words $1^{i(r+1)+j}$ with $j \in \{0, 2, \ldots, r\}$. Since $r^2 - r - 1 \equiv -1 \pmod{r}$ and $r^2 - r - 1 = (r + 1)(r - 2) + 1 \equiv 1 \pmod{r + 1}$, the word w is accepted by none of the loops. Moreover it is the shortest such word, because r and $r + 1$ are relatively prime. Therefore all shorter words are accepted and the resulting components accept all words except w. The initial path extends $w = 1^{r^2-r-1}$ to 1^n.

The construction is illustrated for $n = 9$ in Fig. 6.

We have $\sqrt{n} \geq r > \sqrt{n} - 1$ and $r^2 - r - 1 > n - 2\sqrt{n} + 1 - \sqrt{n} - 1 = n - 3\sqrt{n}$. Thus the path has length $O(\sqrt{n})$. The same holds for the loops described above.

We have already shown how to achieve message complexity $O(\log_2^2(k \cdot n))$ with k parties. Hence for $k = 2^{c \cdot n^{1/4}}$ parties, message complexity is bounded by $O(c^2 \cdot \sqrt{n})$ and hence smaller than the state complexity, if c is sufficiently small.

Our next goal is to show that $P_k = O(\log_2(k \cdot n))$ holds. If true, then, as claimed, our protocol achieves message complexity $O(\log_2^2(k \cdot n))$ for k parties. Now consider $\vartheta(x) = \sum_{p \leq x} \ln p$, where we sum over all primes $p \leq x$. Then $\vartheta(x) \sim x$ [4] and hence $\ln(\Pi_{p \leq x} p) = \vartheta(x) \geq x/2$. In particular, $\Pi_{p \leq x} p \geq e^{x/2}$ and the requirement (1) is indeed satisfied for $P_k = O(\log_2(k \cdot n))$. □

5. Conclusion

What is the consequence of our result? A nondeterministic finite automaton working on an input of length n can be viewed as an n-party nondeterministic protocol where each party has exactly one symbol of the input word. Theorem 1 shows that we are unable to derive good lower bounds on $ns(L)$ using a reasonable abstraction of the automata model. We cannot partition the inputs into reasonably many blocks of reasonable sizes and then use some combinatorial arguments about the necessary information transfer. Theorem 1 says that for input words of polynomial length in $ns(L)$ one is required to take at least as many parties as the input length, i.e. one has to work properly with the nondeterministic finite automaton. This is very bad news because they say that the most common proof techniques based on information transfer do not work.

On the other hand the growing number of parties makes the argument for proving lower bounds on their message complexity extremely hard. How to develop successful and transparent techniques for searching for a minimum cover of 1's in a d-dimensional infinite matrix for d exponentially large in $ns(L)$? A reasonable lower bound has to be of logarithmic size in the number of dimensions. We see that Theorem 1 provides us an insight into the difficulty of the minimization of NFA's and of the estimation of the sizes of minimal NFA's.

We see that estimating $ns(L)$ is not only hard from the algorithmic point of view, but even from the mathematical one. Therefore, we propose the following research problem of main interest:

> *Find a mathematical method (instrument), that, when explored optimally, provides tight lower bounds on $ns(L)$ for each regular language L.*

Our result may be an indication that no abstract and universal technique for proving lower bounds on $ns(L)$ exists.

References

[1] H.N. Adorna, 3-party message complexity is better than 2-party ones for proving lower bounds on the size of minimal nondeterministic finite state automata, in: Proc. 3rd Int. Workshop on Descriptional Complexity of Automata, Grammars and Related Structures, Preprint No. 16, Univ. Magdeburg, 2001, pp. 23–34. See also Journal of Automata, Languages and Combinatorics 7 (4) (2002) 419–432.
[2] H.N. Adorna, On the separation between k-party and $(k+1)$-party nondeterministic message complexity, in: Proc. DLT'2002, in: Lecture Notes in Computer Science, vol. 2450, 2002, pp. 152–161.
[3] A. Arnold, A. Dicky, M. Nivat, A note about minimal non-deterministic automata, Bulletin of the EATCS 47 (1992) 166–169.
[4] E. Bach, J. Shallit, Algorithmic Number Theory 1, MIT Press, 1996.
[5] J.-C. Birget, Intersection and union of regular languages and state complexity, Information Processing Letters 43 (14) (1992) 185–190.
[6] J.C. Birget, Partial orders on words, minimal elements of regular languages and state complexity, Theoretical Computer Science 119 (1993) 267–291.
[7] M. Chrobak, Finite automata and unary languages, Theoretical Computer Science 47 (3) (1986) 149–158.
[8] B. Courcelle, D. Niwinaki, A. Podelski, A geometrical view of the determinization and minimization of finite state automata, Mathematical System Theory 24 (2) (1991) 117–146.
[9] M. Dietzfelbinger, J. Hromkovič, G. Schnitger, A comparison of two lower bound methods for communication complexity, Theoretical Computer Science 168 (1996) 39–51.
[10] P. Ďuriš, J. Hromkovič, J.D.P. Rolim, G. Schnitger, Las Vegas versus determinism for one-way communication complexity, finite automata, and polynomial-time computations, in: Proc. STACS'97, in: Lecture Notes in Computer Science, vol. 1200, pp. 117–128.
[11] I. Glaister, J. Shallit, A lower bound technique for the size of nondeterministic finite automata, Information Processing Letters 59 (1996) 75–77.
[12] G. Gramlich, G. Schnitger, Minimizing NFA's and regular expressions, Journal of Computer and System Sciences 73 (2007) 909–923.
[13] H. Gruber, M. Holzer, Computational complexity of NFA minimization for finite and unary languages, in: Proc. 1st LATA, 2007, pp. 261–272.
[14] H. Gruber, M. Holzer, Finding lower bounds for nondeterministic state complexity is hard, in: Proc. DLT 2006, in: Lecture Notes in Computer Science, vol. 4036, pp. 363–374.
[15] H. Gruber, M. Holzer, Inapproximability of nondeterministic state and transition complexity assuming $P \neq NP$, in: Proc. of the 11th DLT, in: Lecture Notes in Computer Science, vol. 4588, 2007, pp. 205–216.
[16] J. Hromkovič, Communication Complexity and Parallel Computing, Springer, 1997.
[17] J. Hromkovič, Communicatoin protocols — an exemplary study of the power of randomness, in: Sanguthevar Rajasekharan, Panos M. Pardalos, John H. Reif, José Rolim (Eds.), Handbook of Randomized Computing, vol II, pp. 533–596.
[18] J. Hromkovič, Randomized communication protocols (a survey), in: Stochastic Algorithms: Foundations and Applications, in: Lecture Notes in Computer Science, vol. 2264, 2001, pp. 1–32.
[19] J. Hromkovič, Relation between Chomsky hierarchy and communication complexity hierarchy, Acta Math. Univ. Com. 48–49 (1986) 311–317.
[20] J. Hromkovič, J. Karhumäki, H. Klauck, G. Schnitger, S. Seibert, Measures of nondeterminism in finite automata, in: Proc. ICALP 2000, in: Lecture Notes in Computer Science, vol. 1853, pp. 199–210.
[21] J. Hromkovič, J. Karhumäki, H. Klauck, S. Seibert, G. Schnitger, Communication complexity method for measuring nondeterminism in finite automata, Information and Computation 172 (2) (2002) 202–217.
[22] J. Hromkovič, G. Schnitger, Ambiguity and communication, in: Dagstuhl Seminar Proceedings 09001, pp. 553–564.
[23] J. Hromkovič, G. Schnitger, Communication complexity and sequential computation, in: Príara, P. Ružička (Eds.), Proc. of Mathematical Foundations of Computer Science, in: Lecture Notes in Computer Science, vol. 1295, Springer Verlag, 1997, pp. 71–84.
[24] J. Hromkovič, G. Schnitger, On the power of Las Vegas for one-way communication complexity, OBDD's, and finite automata, Information and Computation 169 (2001) 284–296.
[25] T. Jiang, B. Ravikumar, Minimal NFA problems are hard, SIAM Journal on Computing 22 (6) (1993) 1117–1141.
[26] T. Kameda, P. Weiner, On the state minimization of nondeterministic finite automata, IEEE Transactions on Computers C-19 (1970) 617–627.
[27] E. Kushilevitz, N. Nisan, Communication Complexity, Cambridge University Press, 1997.
[28] L. Lovász, Communication Complexity. A survey, in: Korte, Lovász, Promel, Schrijver (Eds.), Paths, Flows, and VLSI Layout, Springer-Verlag, Berlin, New York, 1990.
[29] C. Papadimitriou, M. Sipser, Communication complexity, in: Proc. 14th ACM STOC, 1982, pp. 196–200.
[30] M.O. Rabin, D. Scott, Finite automata and their decision problems, IBM Journal of Research and Development 3 (1959) 114–125.
[31] K. Salomaa, Descriptional complexity of nondeterministic finite automata, in: Proc. DLT 2007, in: Lecture Notes in Computer Science, vol. 3572, pp. 31–35.
[32] K. Salomaa, P. Schofield, State complexity of additive weighted finite automata, International Journal of Foundations of Computer Science 18 (2007) 1407–1416.
[33] A. Salomaa, K. Salomaa, S. Yu, State complexity of combined operations, Theoretical Computer Science 303 (2007) 140–152.
[34] A.C. Yao, Some complexity questions related to distributed computing, in: Proc. 11th ACM STOC, 1979, pp. 209–213.