PRACE IPI PAN • ICS PAS REPORTS

Piotr Berman, Andrzej Lingas

On complexity of regular languages in terms of finite automata

304

1977

WARSZAWA

INSTYTUT PODSTAW INFORMATYKI POLSKIEJ AKADEMII NAUK
INSTITUTE OF COMPUTER SCIENCE POLISH ACADEMY OF SCIENCES
00-901 WARSAW, P.O. Box 22, POLAND

Piotr Berman, Andrzej Lingas

ON COMPLEXITY OF REGULAR LANGUAGES IN TERMS OF FINITE AUTOMATA

304

Warsaw 1977

Pracę zgłosił Zdzisław Pawlak

Mailing addresses: Piotr Berman
ul. Nowowiejska 10 m.50
00-643 Warszawa

Andrzej Lingas
ul. Nowogrodzka 76 m.11
02-018 Warszawa

Printed as a manuscript

Na prawach rękopisu

Nakład 450 egz. Ark. wyd. 0,85; ark. druk. 1,25. Papier offset. kl. III, 70 g, 70 x 100. Oddano do druku we wrześniu 1977 r. W. D. N. Zam. nr 600/0/77

F - 15

Abstract . Содержание . Streszczenie

In our paper 4 measures of complexity of regular languages are defined. These measures are related with complexity of finite automaton recognizing the given language (automaton may be 1 or 2 way, deterministic or not). We construct families of regular languages for which these measures are substantially different. It is also shown that the hipothesis L = NL implies the polinomial relation between measures corresponding to 2-way automata

Характеристика регулярных языков при помощи конечных автоматов

Определяются /четырьмя способами/ меры сложности регулярных языков посредством сложения опознающих эти языки конечных автоматов /с одним или двумя направлениями движения головки, детерминированных или недерминированных/. Конструируются семейства регулярных языков, для которых эти меры действительно различны. Показывается, что из гипотезы L=NL вытекает выражаемая многочленами зависимость между мерами, соответствующими автоматам с двумя направлениями движения головки.

Charakteryzacja złożoności regularnych języków w terminach skończonych automatów

Określamy (na 4 sposoby) miarę złożoności języka regularnego poprzez złożoność rozpoznającego go automatu skończonego (jedno - lub dwustronnego) deterministycznego lub nie). Konstruujemy rodziny języków regularnych, dla których miary te są istotnie różne. Pokazujemy też, że z hipotezy L=NL wynika wielomianowa zależność pomiędzy miarami odpowiadającymi automatom dwustronnym.

Main notions and results

DFA(NDFA) - one-way (non-) deterministic finite automaton

2DFA(2NDFA) - two-way (non-) deterministic finite automaton

$D_1(L)(N_1(L))$ - minimal number of states for DFA(NDFA) which recognizes language L

$D_2(L)(N_2(L))$ - minimal number of states for 2DFA(2NDFA) which recognizes language L

L(NL) - the class of languages which are recognized by deterministic (or non-deterministic) TM supplied by a logarithmicaly bounded work tape.

There is a constant c and families of languages P_n and B_n given by construction such that

$$\forall n \in N \quad D_2(P_n) \leqslant cn \ \& \ D_1(P_n) \geqslant n^n = 2^{n\log n} \qquad (1)$$

$$\forall n \in N \quad N_2(B_n) \leqslant cn \ \& \ D_1(B_n) \geqslant 2^{n^2} \qquad (2)$$

Corollary from theorem of the second part shows, that L = NL implies that there is a polynomial p_o such that

$$D_2(L) \leqslant p_o(\max(N_2(L), m_L)) \qquad (3)$$

where constant m_L means the length of the longest word of language L. This inequality is shown by construction.

Introduction

Recent investigations of computional complexity of languages become the main subject in computer science. Numerous results are obtained but principal hypotheses still remain open. One of the most important subjects in this field is seeking how complexity of a language changes when we replace the deterministic form of an algorithm by the non-deterministic one. Unsolved hypotheses P=NP and L=NL [2,3] are related to this problem. In studying these problems the need emerges of a complexity measure of finite subsets of the languages under investigation.

When a machine works with a finite set of words it only uses a finite portion of its memory. Thus it is possible to identify the state of control of this machine and the configuration of its memory together with the state of a certain two-way finite automaton (which is deterministic if that machine is also deterministic). The minimal number of states for a finite automaton recognizing the mentioned part of a language may be its measure of computational complexity.

Rabin [1] showed that 2DFA and 2NDFA recognize the same family of languages as DFA, i.e. regular languages. Its proof is effective and implies following estimations:

$$D_1(L) \leqslant D_2(L)^{D_2(L)} \qquad (4)$$

$$D_2(L) \leqslant 2^{[N_2(L)]^2} \qquad (5)$$

hese inequalities and results of the first part of this work imply

$$D_2(B_n) > N_2(B_n) \qquad (6)$$

for almost every natural n. Hence non-deterministic automata have an advantage over deterministic ones. We do not know how essential this advantage is. Corollary associated with a theorem from the second part of this work shows that this question is related to the answer to the hypothesis L=NL. Roughly speaking, if this hypothesis holds the complexity of 2DFA accepting finite language depends polynomially on the complexity equivalent 2NDFA and the length of the longest word from this language.

Aknowledgements. The authors thank dr A. Skowron for his help in preparing this paper and sponsoring during their research.

I. Definition. A two-way finite automaton is a 5-tuple $M = (Q_M, \Sigma_M, \delta_M, q_{oM}, q_{FM})$ where

1. Q_M is the finite set of states
2. Σ_M is the finite input alphabet, $\{¢ , \$\} \cap \Sigma_M = \emptyset$
3. $\delta_M : (Q_M \times (\Sigma_M \cup \{¢, \$\})) \to P(Q_M \times \{-1, 1, 0\})$ is the transition partial function of M and P(X) denotes the family of all subsets of X.
4. $q_{OM} \in Q_M$ is the initial state of M.
5. $q_{FM} \in Q_M$ is the final or accepting state of M.

We say that M is deterministic if for each $q \in Q_M$ and $a \in \Sigma_M \cup \{¢, \$\}$ $\delta_M(q, a)$ is a singleton. Otherwise, it is called nondeterministic.

L(M) is a language accepted by M. A word $a_1, \dots, a_n$ from Σ_M^* belongs to L(M) iff there are configurations $C_0, C_1, \dots C_k$ in form

$$C_i = (a_0\ a_1 \dots a_n\ a_{n+1}, q_i, l_i)$$

where $a_0 = ¢$, $a_{n+1} = \$$, $a_1 \dots a_n$ as mentioned above, $q_i \in Q_M$, $l_i \in \{0, 1, -, n+1\}$ and $C_0 = (a_0\ a_1 \dots a_n\ a_{n+1}, q_0, 1)$ (initial condition)

$(q_{i+1}, l_{i+1} - l_i) \in \delta(q_i, a_{l_i})$ (condition of proper transition)

$q_k = q_F$ (final condition)

For other details see Rabin and Scott [1].

Now our purpose will be proving the statements (1) and (2) mentioned among the main results. First the idea of Rabin and Scott's proof of (4) and (5) will be reminded.

Let a DFA, called **A**, recognize the language accepted by 2DFA, called B, with the set of states Q_B consisting of n elements. Automaton **A**, after reading an initial part of a certain word, ought to be able to reconstruct the action of B when it comes back into the subword read. Thus, the state of **A** may be a function $f: Q_B \rightarrow Q_B$, where $f(q_1)=q_2$ means that if B comes into the subword read in state q_1, it will leave this subword in state q_2 or will accept the whole word, if q_2 is the accepting state. Hence **A** has so many states as there are functions $f: Q_B \rightarrow Q_B$ i.e. n^n. Because B might be the minimal 2DFA for the given language, this construction implies the following inequality:

$$D_1(L) \leqslant \overline{\overline{Q_A}} = n^n = D_2(L)^{D_2(L)} \qquad (4')$$

Similarly when automaton B is non-deterministic, then the state of A may be a function $f: Q_B \rightarrow P(Q_B)$, where $f(q)$ consists of these members of Q_B, in which B can leave the subword read after coming into it on the state q. There are 2^{n^2} of similar functions, hence

$$D_1(L) \leqslant \overline{\overline{Q_A}} = 2^{n^2} = 2^{[N_2(L)]^2} \qquad (5')$$

One can pose the question, whether **A** really ought to have so many states. In other words, will all of these functions be needed during testing words whether or not they are members of **L**. We will build families of languages, P_n and B_n, such that every word of one of these languages will reveal one of the mentioned functions explicitly, coding it as a set of pairs (argument, value).

Definition $\quad 1^{*n} = \varepsilon \cup 1 \cup 1\,1 \cup \ldots \cup 1^n$

Definition $\quad B_n$ is subset of $A_n = (\$1^{*n}\text{¢}1^{*n})^{*} \# (1^{*n}\text{¢}1^{*n}\$)^{*}$.
If s is a member of A_n it is in form $s = \$1^{i_1}\text{¢}1^{j_1}\$ \ldots \$1^{i_k}\text{¢}1^{j_k}\# 1^{i_{k+1}}\text{¢}1^{j_{k+1}} \ldots \$1^{i_{k+l}}\text{¢}1^{j_{k+l}}\$$. $s \in B_n$ iff there are subscripts $\alpha_1, \ldots, \alpha_r$, $\beta_1, \ldots, \beta_r$ such that

$$\forall s\in\{1,\dots,r\}\quad \alpha_s \leqslant k < \beta_s \leqslant k-1 \qquad \text{(i)}$$

(it means that α_s signs a pair from the left side of # and β_s signs a pair from the right side)

$$i_{\alpha_1} = 0 \;\&\; \forall s\in\{1,\dots,r\}\; j_{\alpha_s} = i_{\beta_s} \;\& \;\&\; \forall s\in\{1,\dots,r-1\}\; j_{\beta_s} = i_{\alpha_{s+1}} \;\&\; j_{\beta_r} = 0 \qquad \text{(ii)}$$

Lemma 1. There is a constant c such that

$$\forall n \quad N_2(B_n) \leqslant cn$$

Proof. Automaton which accepts B_n consists of a controlling part having c states, common for every n, and a counter which counts up to n. First, the automaton checks whether the given word is in A_n or not. Second, the automaton takes nondeterministically a pair as the left side of the marker and checks that $i_{\alpha_1} = 0$. Then the automaton loads j_{α_1} into the counter and goes as the right side of the marker. It takes a pair nondeterministically and checks if $i_{\beta_1} = j_{\alpha_r}$. Next, it loads j_{β_1} into the counter and seeks a pair i_{α_2} nondeterministically. At the last step of computation checks if $j_{\beta_s} = 0$ and accepts the word.

Lemma 2. The number of quotients of B_n is non less then 2^{n^2}. (It means that the DFA recognizing B_n ought to have at least 2^{n^2} states).

Proof. Let X be a subset of $\{1,\dots,n\} \times \{1,\dots,n\}$, and let $X = \{(i_1, j_1) \dots (i_k, j_k)\}$. Then the word s_X will be in form $s_X = \$\, \varepsilon\, \not{c}\, \varepsilon\, \not{c}\, 1^{i_1}\, \not{c}\, 1^{j_1}\, \not{c} \dots \not{c}\, 1^{i_k}\, \not{c}\, 1^{j_k}\, \#$. It can be checked that if X_1 and X_2, the subsets of $\{1,\dots,n\}\times\{1,\dots,n\}$, are different then the quotients $s_{X_1} \backslash\backslash B_n$ and $s_{X_2} \backslash\backslash B_n$ are different too.

Assume, that the pair (i_o, j_o) belongs to x_1 but not to x_2. Then, it is easy to see that the word t_{i_o,j_o}

$$t_{i_o,j_o} = \mathcal{E} \not\# \, 1^{i_o} \not\# \, 1^{j_o} \not\# \, \mathcal{E} \not\#$$

belongs to the first quotient, but not to the second. We remind to the reader that $\mathcal{E}$, the empty word, codes 0. There are 2^{n^2} different subsets of $\{1, \ldots, n\} \times \{1, \ldots, n\}$ hence there are 2^{n^2} different quotients.

The above two lemmas complete the proof of the statetemt (1).

Definition. P_n is a subset of the language B_n. If s is a member of P_n then it in addition to the conditions (i) and (ii) fulfils

$$\forall s \in \{1, \ldots, r\} \forall p \forall q \; [\alpha_s < p \leqslant k < q < \beta_s] \rightarrow [i_{\alpha_s} \neq i_p \;\&\; i_q \neq i_{\beta_s}] \quad \text{(iii)}$$

Proof of the following lemma reveals why this condition was introduced.

Lemma [illegible]. There is a constant c such that

$$\forall n \quad N_2(B_n) \leqslant cn$$

Proof. The automaton, which accepts P_n, is a 2DFA consisting of a controlling part having c states, common for every n, and a counter, which counts up to n. First, the automaton checks if the given word is in A_n or not. Second, the head of the automaton stands on the marker $\not\#$ and checks step by step, if the left elements of the subsequent scanned pairs from the left of $\not\#$ are equal to 1^0. If it has found such a pair, then it loads the right element of this pair to the counter and comes back onto the marker. Next, the automaton checks step by step if the left elements of the subsequent pairs from the right are equal to a number contained in the counter.

The reader may find how it can be done deterministically. If the automaton has detected such a pair, then it loads the right element into the counter and comes back onto the marker. At the last step of computation the automaton loads 1^0 into the counter on the right side of the marker, and accepts the word.

Lemma 4. The number of quotients of P_n is non less then n^n (it follows $D_2(P_n) \geqslant n^n$).

Proof. Let f bee a function $f: \{1,\dots,n\} \to \{1,\dots,n\}$. Then the word s_f will be in the form $s_f = \not{c}\, \varepsilon \not{c}\, \varepsilon \not{c}\; 1^1 \not{c}\; 1^{f(1)} \not{c} \dots \not{c}\; 1^n \not{c}\; 1^{f(n)} \#$

It can be checked that if functions f_1 and f_2 are different, then quotients $s_{f_1} \backslash\backslash P_n$ and $s_{f_2} \backslash\backslash P_n$ are different too.

Assume that $f_1(i) \neq f_2(i)$. Then it is easy to see that the word $u_{i,f_1(i)} = \varepsilon \not{c}\; 1^i \not{c}\; 1^{f(i)} \not{c}\; \varepsilon \not{c}$

belongs to the first quotient but not to the second. There are n^n different functions from $\{1,\dots,n\}$ to $\{1,\dots,n\}$ hence there are n^n different quotients.

The last two lemmas complete the proof of (2).

Now we will use the obtained results to compare the measures N_2 and D_2 in respect to the languages B_n.

Lemma 5. $D_2(B_n) \geqslant 0.5n^2/\ln n$

Proof. This is a conclusion of lemma 2 and (5):

$$D_1(B_n) \leqslant D_2(B_n)^{D_2(B_n)}$$

$$2^{n^2} \leqslant D_1(B_n)$$

Hence $2^{n^2} \leqslant 2^{D_2(B_n).\ln D_2(B_n)}$, so $n^2 \leqslant D_2(B_n).\ln D_2(B_n)$

It is obvious, that $(0.5n^2/\ln n).\ln(0.5n^2/\ln n) \leqslant n^2$, thus

$(0.5n^2/\ln n).\ln(0.5n^2/\ln n) \leqslant D_2(B_n).\ln D_2(B_n)$

Because x.ln x is a monotonic function, we have $0.5n^2/\ln n \leqslant D_2(B_n)$.

Lemma 5 and lemma 1 imply that for some constant c':

$$D_2(B_n) \geqslant c' \left[N_2(B_n)\right]^2 / \ln N_2(B_n), \qquad (6')$$

and this implies

$$D_2(B_n) > N_2(B_n) \qquad (6)$$

for almost every natural n.

II.

Informally, by a Turing machine (TM for short) we mean a device with a finite-state control, a two-way read-only input tape with the endmarkers (¢ , $) and a single two-way read-write work tape. Additionally, the machine may be equipped with a one-way write-only output tape. If the Turing machine is deterministic (nondeterministic) we write DTM (NDTM respectively) for short. For the details see [1] .
For any NDTM or DTM , M , L(M) denotes the language accepted by M like for 2NDFA or 2DFA.
Let $f: N \longrightarrow N$. A Turing machine is $f(.)$ tape-bounded iff, whenever it is started with any input word x on its input tape, it will never scan more than $f(|x|)$ symbols on its work tape. $|x|$ denotes the lenght of x . If $f(\cdot) \leq \log(\cdot)$ we also say , that M is logaritmically tape-bounded.

The hypothesis L = NL seems not to hold, when one compare languages about which we know that they belong respectively to L and NL . Thus the first result we will prove in this part of our paper can offer another possibility to reject it.

__Theorem 6.__ If L = NL then exists a polynomial p_0 such that for each $m \in N$ and for each 2NDFA , A , one can construct [1] 2DFA , B_A, which satisfies the following conditions :

1/ $\{ x \mid |x| \leqslant m \cdot \overline{\overline{Q}}_A \ \& \ x \in L(A) \} \subseteq L(B_A) \subseteq L(A)$

2/ $\bar{\bar{Q}}_{B_A} \leqslant p_0 (m \cdot \bar{\bar{Q}}_A)$

Proof. Let cod : $\{ c \mid c$ is a 2NDFA over the input alphabet $\Sigma\} \rightarrow (\Sigma \cup \Sigma_1)^*$ be the given one-to-one encode-function. For any 2NDFA , C , the word cod(C) consists of three parts encoding respectively : q_{oc} , δ_c, q_{FC} . The states of C are represented in cod(C) by the numbers from $\{n \in N \mid 1 \leqslant n \leqslant Q_c\}$ in the dyadic notion. Σ_1 is the auxiliary alphabet , $\{0,1\} \subseteq \Sigma_1$, $\{¢, \$, \#\} \cap (\Sigma \cup \Sigma_1) = \emptyset$. The details of this encoding are left to the reader.

We may assume without loss of generality, that

$$|cod\ (C)| \leqslant k \cdot \bar{\bar{Q}}_c^2 , \qquad (7)$$

where k is an integer constant independent of C .

Let us define a language L_1 :

$$L_1 = \{x \# y \mid x \in \Sigma^* \;\&\; y \in (\Sigma \cup \Sigma_1)^* \;\&\; \exists \text{ 2NDFA } C \; [y = cod\ (C) \;\&\; x \in L\ (C)]\}$$

It is easy to construct a logaritmically tape-bounded NDTM, M , accepting L_1 . When M verifies that a input word is of the form $x \# cod(C)$, where $x \in \Sigma^*$ and C is 2NDFA , then M will simulate the action of C with the input word x , using the cod (C) . For this purpose M will keep the last state and the last head position of simulated C on its work tape. Since $\bar{\bar{Q}}_c \leqslant |cod(C)|$ so M will only need $\lceil \log |cod\ (C)| + \log |x| + 2 \rceil \leqslant$ $\leqslant \lceil 2 \log |x \# cod\ (C)| + 2 \rceil$ bits of its work tape to do it. M accepts $x \# cod\ (C)$ if and only if the simulated 2DFA , C , reaches the final state for x .

Hence, matching a suitably numberous work-tape alphabet for M we will obtain that $L_1 (L_1 = L(M))$ belongs to NL. One may also observe that L is NL-complete, see [3] .

Let us suppose that L = NL . Then we may assume that L_1 is recognized by a logaritmically tape-bounded DTM , T .

p_1 denotes the polynomial , $p_1(n) = k \cdot n^2 + n + 1$ for $n \in N$.

Let A be a 2NDFA , $m \in N$.

If we bound length of the work tape of T by $\lceil \log(p_1(m \cdot \bar{\bar{Q}}_A)) \rceil$, we will obtain certain device which one may identify with the 2DFA , T_A , as it was mentioned in the introduction.

Notice , that by (7) the following inequalities hold :

$$|x \# \operatorname{cod}(A)| \leqslant m \cdot \bar{\bar{Q}}_A + |\operatorname{cod}(A)| + 1 \leqslant m \cdot \bar{\bar{Q}}_A + k \cdot \bar{\bar{Q}}_A^{\,2} + 1 \leqslant p_1(m \cdot \bar{\bar{Q}}_A) \text{ , where}$$

$x \in \sum^*$, $|x| \leqslant m \cdot \bar{\bar{Q}}_A$.

Since T is a logaritmically tape bounded so

$$\{ x \# \operatorname{cod}(A) \,|\, |x| \leqslant m \cdot \bar{\bar{Q}}_A, \ x \# \operatorname{cod}(A) \in L_1 \} \subseteq L(T_A) \subseteq \subseteq L(T) = L_1 \qquad (8)$$

As in the introduction, under the state of T_A we mean a pair consisting of a configuration of the bounded work tape of T and of a state of T . Thus exists an integer constant l assigned by T that :

$\bar{\bar{Q}}_A \leqslant l^{\log(p_1(m\, Q_A))}$ what implies

$$\bar{\bar{Q}}_{T_A} \leqslant p_o(m \ \bar{\bar{Q}}_A) \text{ ,} \qquad (9)$$

where p_o is the fixed polynomial independent of m and A .

Each $y \in (\sum \cup \sum_1)^*$ appoints the partial function f_y from Q_{T_A} to Q_{T_A} as follows :

for any $q, r \in Q_{T_A}$

a1/ if $\delta_{T_A}(q, \#) \in Q_{T_A} \times \{-1\}$ then $f_y(q) = q$

a2/ suppose that : if T_A starts to scan $\# y \not{c}$ in the state q with its head over $\#$ then (without leaving $\# y \not{c}$)/

i/ T_A will reach the state r , which is the final state of T_A

or

ii/ T_A will reach the state r and its head will scan $\#$ simultaneously, where $\delta_{T_A}(r, \#) \in$ $\in Q_{T_A} \times \{-1\}$

In the above cases $f_y(q) = r$

Now, we will modify T_A to 2DFA , B_A, as below :

$Q_{B_A} = Q_{T_A}$, $q_{oB_A} = q_{oT_A}$, $q_{FB_A} = q_{FT_A}$,

$\delta_{B_A} \mid Q_{B_A} \times (\{\not{c}\} \cup \Sigma) = \delta_{T_A} \mid Q_{B_A} \times (\{\not{c}\} \cup \Sigma)$

$$\delta_{B_A}(q, \not{c}) = \begin{cases} (f_{cod(A)}(q), 0) & \text{if } f_{cod(A)}(q) = q_{FB_A} \\ \delta_{T_A}(f_{cod(A)}(q), \#) & \text{if } f_{cod(A)}(q) \in Q_{B_A} - \{q_{FB_A}\} \\ \text{indefined} & \text{if } f_{cod(A)} \text{ is indefined for } q \end{cases}$$

It is quite obvious that

$L(B_A) = x\ \{x \mid \#\ \mathrm{cod}(A) \in L(T_A)\}$

By (8) and by the definitions of L, T_A, B_A we obtain that :

$$\{x \mid |x| \leqslant m \cdot \bar{\bar{Q}}_A,\ x \in L(A)\} \subseteq L(B_A) \subseteq L(A)$$

The statement (9) and the equality $Q_{B_A} = Q_{T_A}$ imply

$$\bar{\bar{Q}}_{B_A} \leqslant p_o(m \cdot \bar{\bar{Q}}_A)$$

__Corollary 7.__ If $L = NL$ then exists a polynomial p_o, where for each finite language L

$$D_2(L) \leqslant p_o(\max\{N_2(L), m_L\}) \qquad (3)$$

m_L denotes the length of the longest word from L.

1) __Remark.__ The construction used in the proof of the theorem 6 can be done by a logaritmically tape - bounded DTM for an input word encoding any constant and any 2NDFA. This DTM prints on its output tape the encoding of the wanted 2DFA. If we add the above observation to the thesis of the theorem 6 we can proof the reversal theorem, too.

Without making the assumption that $L = NL$ we can present some worse estimations as that of the theorem 6 and corollary 7. In this purpose we will use the result of Savitch [4], who showed that any language accepted by $f(n)$ tape-bounded NDTM can be recognized by $O(f^2(n))$ tape-bounded DTM, where $f(n) \geqslant \log n$, $n \in N$.

Lemma 8 . There is a constant d such that, for each $m \in N$ and for each 2NDFA A one can construct [2] 2DFA, D_A, which satisfies the following conditions :

1/ $\{ x \mid |x| \leqslant m \cdot \bar{\bar{Q}}_A \ \& \ x \in L(A) \} \subseteq L(D_A) \subseteq L(A)$

2/ $\bar{\bar{Q}}_{D_A} \leqslant (m \bar{\bar{Q}}_A)^{d \log (m \bar{\bar{Q}}_A)}$.

Sketch. The proof is like one of the theorem 6 . It suffices to replace the machine T (from the theorem 6) recognizing L_1 on the log n - bounded work tape by a DTM, T_1, which does it on a $O(\log^2(n))$ - bounded work tape. Such machine T_1 exists by the mentioned result of Savitch.

Corollary 9. There is a constant d that for each finite language L

$D_2(L) \leqslant FN_2(L)^{d \log FN_2(L)}$

where $FN_2(L) = \max \{ N_2(L), m_L \}$

2) Remark. The construction from the lemma 8 can be made by $\log^2(n)$ tape-bounded DTM for an input word encoding any constant and any 2NDFA.

REFERENCES

[1] Rabin, M.O. , Scott, D., Finite automata and their decision problems , IBM Journal of Res. Development, 3(1959) , page 114-125.

[2] Cook, S.A., The complexity proving procedures, proceedings of 3rd Annal ACM Symposium of Theory of Computing.

[3] Jones, N.D., Lien, Y.E., Laaser, W.T., New Problem complete for nondeterministic log - space , Mathematical Systems Theory rd 10 no 1 (1976)

[4] Savitch, Relations between nondeterministic and deterministic tape complexities. Journal of Comp. and Syst. Sc 4 (1970) .

Ehrenfeucht and Zeigler present in [5] and [6] another approach to the problem of complexity of finite languages:

[5] Ehrenfeucht A., Practical Decidability, Journal of Comp. and Syst. Sc. 11(1975).

[6] Ehrenfeucht A., Zeigler P., Complexity Measure for regular Expresions, Journal of Comp. and Syst. Sc. 12(1976).

Ostatnio w serii tej ukazały się:

Nr 285 - T. Traczyk: On consecutive retrieval from drum-type storage

Nr 286 - J. Winkowski: Game of life as a synchronous process that can be realized asynchronously

Nr 287 - J. Winkowski: Algebras of arrays - a tool to deal with concurrency

Nr 288 - W. Lipski: One more polynomial complete consecutive retrieval problem

Nr 289 - T. Skrzypkowski: Homomorfizmy maszyn ze sterowaniem

Nr 290 - J. Górski: Suboptymalny algorytm wyznaczania organizacji zwartej zbioru zapisów

Nr 291 - K. Subieta, W. Balcerzak: Automat skończony jako model zawartości bazy danych

Nr 292 - Z. Raś: A mathematical model of a diagnostic experiment

Nr 293 - P. Dembiński and S. Budkowski, I. Hansen, J. Leszczyłowski, A. Pupliński: Verification oriented microprogramming language

Nr 294 - B. Muchlado-Marońska: Problemy translacji w systemach konwersacyjnych

Nr 295 - R. Błaszczyk: Analiza możliwości implementacji metody list inwersyjnych dla systemu informacyjnego STUDENT

Nr 296 - E. Turska: Podstawy teoretyczne i metody maszynowe związane z określeniem stref numeracyjnych w sieci łączności

Nr 297 - S. Teluk: Pewien sposób organizacji banku informacji z zastosowaniem metod matematycznych

Nr 298 - M. Karpiński: The equivalence problem of binary EOL-systems is decidable

Nr 299 - M. Karpiński: The structural equivalence problem for EOL-systems is recursively solvable

Nr 300 - W. Lipski: On the logic of incomplete information

Nr 301 - M. Michalewicz: Optimizacya po koefficientam liniejnoj formy liniejnogo zamykanija koniecznoj sistiemy raspoznanija ałgoritmow

Nr 302 - E. Jurkiewicz: Optymalizacja dwukryterialna podsystemu komunikacyjnego sieci teleinformatycznej

Nr 303 - B. Kopeć, K. Subieta: Konstrukcje integrujące w lingwistycznym modelu bazy danych

Adt