

ANDRZEJ SZEPIETOWSKI

Lower space bounds for accepting shuffle languages

RAIRO. Theoretical Informatics and Applications, tome 33, n° 3
(1999), p. 303-307

http://www.numdam.org/item?id=ITA_1999__33_3_303_0

© AFCET, 1999, tous droits réservés.

L'accès aux archives de la revue « RAIRO. Theoretical Informatics and Applications » implique l'accord avec les conditions générales d'utilisation (<http://www.numdam.org/conditions>). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques

<http://www.numdam.org/>

LOWER SPACE BOUNDS FOR ACCEPTING SHUFFLE LANGUAGES

ANDRZEJ SZEPIETOWSKI¹

Abstract. In [6] it was shown that shuffle languages are contained in **one-way-NSPACE**($\log n$) and in **P**. In this paper we show that nondeterministic one-way logarithmic space is in some sense the lower bound for accepting shuffle languages. Namely, we show that there exists a shuffle language which is not accepted by any deterministic one-way Turing machine with space bounded by a sublinear function, and that there exists a shuffle language which is not accepted with less than logarithmic space even if we allow two-way nondeterministic Turing machines.

AMS Subject Classification. 68Q15, 68Q45.

1. INTRODUCTION

The operations shuffle and shuffle closure have been introduced to describe sequentialized execution histories of concurrent processes [7, 8]. Together with other operations they describe various classes of languages which have been extensively studied (see [1, 3–5, 10]). Here, we consider the class of shuffle languages which emerges from the class of finite languages through regular operations (union, concatenation, Kleene star) and shuffle operations (shuffle and shuffle closure). In [6] it was shown that shuffle languages are contained in the class **one-way-NSPACE**($\log n$) and thus in the class **P** (*i.e.* they are accepted in polynomial time by deterministic Turing machines). For every shuffle expression E , a shuffle automaton was constructed which accepts the language generated by E and it was shown that the computations of the automaton can be simulated by a one-way nondeterministic Turing machine in logarithmic space.

In this paper we show that nondeterministic one-way logarithmic space is in some sense the lower bound for accepting shuffle languages. Namely, we show that there exists a shuffle language which is not accepted by any deterministic one-way

¹ Institute of Mathematics, University of Gdańsk, ul Wita Stwosza 57, 80952 Gdańsk, Poland;
e-mail: matszp@paula.univ.gda.pl

Turing machine with space bounded by a sublinear function, and that there exists a shuffle language which is not accepted with less than logarithmic space even if we allow two-way nondeterministic Turing machines.

2. SHUFFLE LANGUAGES

Let Σ be any fixed alphabet and λ the empty word. The shuffle operation \odot is defined inductively as follows:

- $u \odot \lambda = \lambda \odot u = \{u\}$, for $u \in \Sigma^*$ and
- $au \odot bv = a(u \odot bv) \cup b(au \odot v)$, for $u, v \in \Sigma^*$ and $a, b \in \Sigma$.

For any languages $L_1, L_2 \subset \Sigma^*$ the shuffle $L_1 \odot L_2$ is defined as

$$L_1 \odot L_2 = \bigcup_{u \in L_1, v \in L_2} u \odot v.$$

For any language L , the shuffle closure operator is defined by:

$$L^\otimes = \bigcup_{i=0}^{\infty} L^{\odot i}, \quad \text{where } L^{\odot 0} = \{\lambda\} \text{ and } L^{\odot i} = L^{\odot i-1} \odot L.$$

Definition 1. Each $a \in \Sigma$, λ and \emptyset are shuffle expressions. Besides, if S_1, S_2 are shuffle expressions, then $(S_1 \cdot S_2)$, S_1^* , $(S_1 + S_2)$, $(S_1 \odot S_2)$ and S_1^\otimes are shuffle expressions, and nothing else is a shuffle expression.

The language $L(S)$ generated by a shuffle expression S is defined as follows. $L(a) = \{a\}$, $L(\lambda) = \{\lambda\}$, $L(\emptyset) = \emptyset$. If $L(S_1) = L_1$ and $L(S_2) = L_2$, then $L((S_1 \cdot S_2)) = L_1 \cdot L_2$, $L((S_1 + S_2)) = L_1 \cup L_2$, $L(S_1^*) = L_1^*$, $L((S_1 \odot S_2)) = L_1 \odot L_2$, and $L(S_1^\otimes) = L_1^\otimes$.

A language L is a shuffle language if there exists a shuffle expression E such that $L = L(E)$. We shall also use the following notation, for arbitrary string z : $|z|$ denotes the length of z , $|z|_e$ the number of occurrences of a symbol e in z , z_i the i -th symbol of z , and z^R the reverse of z (z written backwards).

3. TURING MACHINES

We consider the Turing machine model with a read-only input tape and a separate two-way, read-write work tape. The number of tape cells used on the work tape, called space, is our measure of computational complexity. A Turing machine is called one-way if its input head cannot move to the left.

We use so called weak mode of space complexity. Let $L(n)$ be a function on natural numbers. A Turing machine is said to be weakly $L(n)$ space-bounded if for every accepted input of length n , at least one accepting computation uses no more than $L(n)$ space. But our results are also valid for strong mode of space complexity, which requires that for every input of length n , all computations are $L(n)$ space bounded. We shall use the following notation: $DSPACE[L(n)]$

or $NSPACE[L(n)]$ denotes the class of languages accepted by deterministic or nondeterministic $L(n)$ space-bounded Turing machines, respectively. We add the prefix *one-way* if we consider classes of languages accepted by one-way Turing machines.

By a configuration of a Turing machine M we shall mean a tuple (q, γ, j) , where q is the current state of M , γ are the contents of the non-blank sector of the work tape, and j is the position of the work head, $1 \leq j \leq |\gamma| + 1$ (we assume that M cannot write the blank symbol on its work tape). The space used by the configuration (q, γ, j) is equal to $|\gamma|$ —the number of non-blank cells on the work tape. It is easy to see that the number of all configurations with space bounded by k is less than r^k , for some constant $r > 1$ (for more details see [9] or [2]).

4. LOWER BOUND FOR ONE-WAY TURING MACHINES

In this section we show that there exists a shuffle language which is not accepted by any deterministic one-way Turing machine in space bounded by a sublinear function.

Theorem 2. *There exists a shuffle language L such that $L \notin \text{one-way-DSPACE}[S(n)]$, for any $S(n) = o(n)$.*

Proof. Consider the shuffle language

$$L = (a + b)^* a(ac + bd)^{\otimes} d(c + d)^* + (a + b)^* b(ac + bd)^{\otimes} c(c + d)^*$$

and let $h : \{a, b\}^* \rightarrow \{c, d\}^*$ be the isomorphism described by $h(a) = c$ and $h(b) = d$. First we shall prove the following.

Lemma 3. *Let k be a positive number. For every $u, v : u \in (a + b)^k$ and $v \in (c + d)^k$, the concatenation uv belongs to L if and only if $h(u) \neq v^R$ (v^R denotes the reverse of v).*

Proof. If $uv \in L$ then uv can be decomposed into

$$uv = u'au''v'dv' \quad \text{or} \quad uv = u'bu''v'cv'$$

with $u', u'' \in (a + b)^*$, $v', v'' \in (c + d)^*$, and $u''v'' \in (ac + bd)^{\otimes}$. We shall only deal with the first case. Note that in this case $u = u'au''$ and $v = v''dv'$.

Since $u''v'' \in (ac + bd)^{\otimes}$, we have

$$|u''v''|_a = |u''v''|_c \quad \text{and} \quad |u''v''|_b = |u''v''|_d$$

(where $|z|_e$ denotes the number of occurrences of a symbol e in a string z).

And because

$$|u''v''|_a + |u''v''|_b = |u''| \quad \text{and} \quad |u''v''|_c + |u''v''|_d = |v''|$$

we have

$$|u''| = |v''|$$

and hence

$$|u'| = |v'|.$$

Let $i = |u'| + 1 = |v'| + 1$. Then the words $h(u)$ and v^R disagree on the i -th symbol, $(h(u))_i = h(u_i) = h(a) = c$ and $(v^R)_i = d$ (where z_i denotes the i -th symbol of a word z). Thus $h(u) \neq v^R$.

Suppose now that $h(u) \neq v^R$ and that i is the last index, where $h(u)$ and v^R disagree. We can assume that $u_i = a$ and $(v^R)_i = d$. Then u and v can be decomposed in the following way: $u = u'au''$, $v = v''dv'$, and $h(u'') = (v'')^R$. (It is possible that $u'' = v'' = \lambda$.) In this situation $u''v'' \in (ac + bd)^\otimes$ and thus $uv \in L$. This ends the proof of the lemma. \square

Suppose now, for a contradiction, that L is accepted by a one-way deterministic Turing machine M with space weakly bounded by $S(n)$.

Let k be a positive number. For every $u \in (a + b)^k$, let $conf(u)$ be the configuration reached by M after reading u . Because there exists $v \in \{c, d\}^k$ such that the word $uv \in L$, then $conf(u)$ uses at most $S(2k)$ cells on the work tape. There are 2^k different words in $(a + b)^k$, and at most $r^{S(2k)}$ configurations with space bounded by $S(2k)$, for some constant $r > 1$. Since $\lim_{n \rightarrow \infty} \frac{S(n)}{n} = 0$, there exists k such that $r^{S(2k)} < 2^k$, and there exist two different words $x, y \in (a + b)^k$, such that $conf(x) = conf(y) = \alpha$.

Consider now the accepting computation of M on the word $x(h(y))^R$. By Lemma 3, $x(h(y))^R \in L$, because $h(x) \neq (h(y)^R)^R = h(y)$. In this computation M reaches the configuration α just after reading x . This means that M also accepts the word $y(h(y))^R$ because M reaches α after reading y and afterwards it proceeds exactly like for $x(h(y))^R$ and accepts at the end. But, by Lemma 3, $y(h(y))^R$ does not belong to L , a contradiction. \square

5. LOWER BOUND FOR TWO-WAY TURING MACHINES

In this section we show that there exists a shuffle language which is not accepted by any nondeterministic two-way Turing machine in space bounded by a sublogarithmic function.

Theorem 4. *There exists a shuffle language L_1 such that $L_1 \notin NSPACE[S(n)]$ for any $S(n) = o(\log n)$.*

Proof. Consider the shuffle language

$$L_1 = (ab)^\otimes.$$

The theorem follows from the fact that the class $NSPACE[S(n)]$ is closed under intersections with regular languages, and that the language

$$L_1 \cap a^*b^* = \{a^n b^n \mid n \geq 0\}$$

is not accepted by any nondeterministic Turing machine with space bounded by $S(n) = o(\log n)$ (see [9]). \square

REFERENCES

- [1] J.L. Gischer, Shuffle languages, Petri nets and context sensitive grammars. *Comm. ACM* **24** (1981) 597–605.
- [2] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading MA (1979).
- [3] M. Jantzen, Extending regular operations with iterated shuffle. *TCS* **38** (1985) 223–247.
- [4] J. Jędrzejowicz, On the enlargement of the class of regular languages by shuffle closure. *IPL* **16** (1983) 51–54.
- [5] J. Jędrzejowicz, Nesting of shuffle closure is important. *IPL* **25** (1987) 363–367.
- [6] J. Jędrzejowicz and A. Szepietowski, *Shuffle languages are in P*, Preprint No. 124, Mathematical Institute, University of Gdańsk, ul. Wita Stwosza 57, 80-952 Gdańsk, Poland, March 1997, *Theoret. Comput. Sci.*, accepted.
- [7] W.E. Riddle, Software system modelling and analysis, Tech. Report, Dept. of Computer and Communication Sciences, University of Michigan **RSSM25** (1976).
- [8] A.C. Shaw, Software descriptions with flow expressions. *IEEE Trans. Software Engrg* **SE-4** (1978) 242–254.
- [9] A. Szepietowski, *Turing Machines with Sublogarithmic Space*, LNCS 843, Springer-Verlag, Berlin (1994).
- [10] M.K. Warmuth and D. Haussler, On the complexity of iterated shuffle. *J. CSS* **28** (1984) 345–358.

Communicated by Ch. Choffrut.

Received March 15, 1999. Accepted July 18, 1999.