Information Processing Letters 18 (1984) 239–242
North-Holland

18 June 1984

DETERMINISTIC TWO-WAY ONE-HEAD PUSHDOWN AUTOMATA ARE VERY POWERFUL

B. MONIEN

University of Paderborn, 4790 Paderborn, Fed. Rep. Germany

Communicated by K. Mehlhorn
Received 2 January 1984

We show that if a language over a two-letter alphabet belongs to $\mathbb{P}$, then its unary encoding belongs to 2DPDA(1).

Keywords: Complexity classes, pushdown automata, one-letter input alphabet

1. Introduction

A large number of classes of languages occurring in complexity theory and formal language theory are defined by automata of different types. We distinguish deterministic and nondeterministic automata, one-way and two-way automata (according to the way the input string is scanned) and a further essential distinction is given by the nature of the storage access. For proving classes of languages not to be equal there exist mainly two methods. In the case of powerful automata (and these are usually the two-way automata) we have the method of diagonalization which has been refined by padding, transformation and recursive padding [5,6,8]. In the case of not so powerful automata (and these are usually the one-way automata) there is constructed some witness language belonging to one class but not to the other [4,9]. Recently this scheme of proof techniques has lost some of its general validity. Duris and Galil [2] showed by constructing a witness language that a deterministic two-way pushdown automaton is more powerful than a deterministic two-way counter automaton.

The straightforward generalization of the counter is the pushdown tape. Restricting the input alphabet to only one symbol makes the device even simpler. Can we find a natural language over a one-letter alphabet not acceptable by a two-way pushdown automaton with one head? Apparently there has been done work in this area over some time and there was a rumour that the language $\{a^{n^2} \mid n \in \mathbb{N}\}$ is such a witness language. In this paper we will show that deterministic two-way one-head automata are very powerful even when restricted to a one-letter alphabet. Of course, as a general statement this was known before [3,7]. We want to mention especially the results of [6], where it is shown that for every $k \in \mathbb{N}$ the class 2DPDA(k) is 'nearly equal' to the class of languages acceptable by some random access machine within time $O(n^k)$. The result of the present paper has a different flavour. We define explicitly a large class of languages which is contained in 2DPDA(1).

We show that if the 'binary version' of a language belongs to $\mathbb{P}$, the class of languages acceptable by deterministic Turing machines in polynomial time, then its 'unary version' belongs to 2DPDA(1). Here 2DPDA(k), $k \in \mathbb{N}$, denotes the class of languages acceptable by deterministic two-way k-head pushdown automata.

For words $w \in \{1, 2\}^*$, $w = a_1 a_2 \ldots a_n$, we use the bijective encoding $f(w) = \sum_{i=1}^{n} a_i 2^i$ and for a language $L \subset \{1, 2\}^*$ we set $un(L) = \{f(w) \mid w \in L\}$.

Theorem. *Let* $L \subset \{1, 2\}^*$. *Then*

$$L \in \mathbb{P} \Rightarrow \mathrm{un}(L) \in \mathrm{2DPDA}(1).$$

This theorem can be generalized in two directions. First it is clear that it holds also for languages $L \subset \{1, 2, \ldots, p\}^*$, $p \in \mathbb{N}$ provided the mapping 'un' is defined in an appropriate way. Furthermore using the same proof technique the theorem can be generalized such that $\mathrm{un}(L) \in \mathrm{2DPDA}(1)$ holds also if L is accepted by a deterministic Turing machine within the time bound t(n), where t is some nice subexponential function, i.e., t is 'simple computable' and $\lim(f(n))^k/2^n \to 0$ holds for $n \to \infty$ and for every $k \in \mathbb{N}$.

It is clear that the theorem annuls the rumour mentioned above. The language containing all binary encodings of quadratic numbers is computable in polynomial time and therefore $\{a^{n^2} \mid n \in \mathbb{N}\}$ belongs to 2DPDA(1).

The theorem is known to the author since many years and it is implicitly contained in [7]. The theorem is published now since the result and the proof technique seem not to be known to everybody working in the field. The proof uses no really deep idea. It is a little bit tricky since the limited access of the automaton to its pushdown tape has to be overcome by recomputing some of the information again and again.

2. Proof of the theorem

$L \in \mathbb{P}$ implies that there exists a $k \in \mathbb{N}$ such that $L \in \mathrm{2DPDA}(k)$ [1]. Furthermore it is well known that in this case the language

$$L_k = \left\{ \underbrace{wa \ldots a}_{|w|^k} ; w \in L \right\}$$

belongs to 2DPDA(1) [7]. Let M be the deterministic two-way one-head pushdown automaton accepting L_k.

We have to define a pushdown automaton $\hat{M}$ accepting un(L). $\hat{M}$ will simulate M step by step. A configuration of such an automaton is described as a 4-tuple (internal state, inscription input tape, head position, inscription pushdown tape).

Let M ($\hat{M}$, respectively) start in the configuration

$$\left(s_0, -|\underbrace{wa \ldots a}_{|w|^k}|-, 0, \epsilon\right) \quad \text{and} \quad (s_0, -|a^m|-, 0, \epsilon), \quad m = \mathrm{un}(w),$$

respectively. Whenever M reaches a configuration

$$\left(s, -|\underbrace{wa \ldots a}_{|w|^k}|-, h, u\right),$$

then $\hat{M}$ reaches the configuration $(s, -|a^m|-, h, u)$. In order to simulate the next step of M the automaton $\hat{M}$ has to compute the h*th* symbol of

$$-|\underbrace{wa \ldots a}_{|w|^k}|-.$$

Note that $\hat{M}$ knows the actual internal state of M and the actual upmost symbol on the pushdown tape of M. First $\hat{M}$ computes $n = |w|$. If $h > n$ holds, then M reads the symbol a and therefore $\hat{M}$ can simulate the next step of M. If $h \leqslant n$ holds, then $\hat{M}$ computes the h*th* symbol of w by dividing m successively by two. We shall show in the following that a pushdown automaton can perform these computations.

(i) $\hat{M}$ writes its head position on the pushdown tape and computes $n = |w| = \lfloor \log_2(m+1) \rfloor$, i.e.,

$$(s, -|a^m|-, h, u) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_1, -|a^m|-, 0, u \# 1^h) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_2, -|a^m|-, 0, u \# 1^h \# 1^n).$$

The last computation is done by moving the head first to position $(m+1)$ and then dividing the head position successively by two. The number of divisions is stored on the pushdown tape.

(ii) Now $\hat{M}$ compares h with n (we have written down this computation for the case $h \leqslant n$).

$$\ldots \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_3, -|a^m|-, n, u \# 1^h) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_4, -|a^m|-, n-h, u).$$

If $h > n$ holds, then M reads the symbol a in its actual configuration and therefore $\hat{M}$ can simulate the next step of M. Now let us assume that $h \leqslant n$ holds. $\hat{M}$ has to compute the h*th* symbol of w. First $\hat{M}$ has to recompute h,

$$\ldots \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_5, -|a^m|-, 0, u \# 1^{n-h}) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_6, -|a^m|-, 0, u \# 1^{n-h} \# 1^n) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_7, -|a^m|-, h, u).$$

(iii) In order to compute the h*th* symbol of w we have to divide m exactly h times by two. The remainder of the last division determines the h*th* symbol of w. This computation destroys the head position h. Therefore we have to compute first a copy of h. In order to do this we multiply h with $n+2$ (i.e., $h \to h(n+2)$) and afterwards we divide by $n+1$ and store the number of subtractions and the last remainder. Note that we are doing this computation only if $h \leqslant n$ holds.

The multiplication is done by:

$$\begin{aligned}
&\ldots \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_*, -|a^m|-, 0, u \# 1^h \# 1^0 \# 1^n) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_{**}, -|a^m|-, n+1, u \# 1^h) \\
&\overset{*}{\underset{\hat{M}}{\rightarrow}} (s_{***}, -|a^m|-, 0, u \# 1^{h-1} \# 1^{n+2}) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_*, -|a^m|-, 0, u \# 1^{h-1} \# 1^{n+2} \# 1^n) \\
&\overset{*}{\underset{\hat{M}}{\rightarrow}} \ldots \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_8, -|a^m|-, h(n+2), u).
\end{aligned}$$

The division is done by

$$\begin{aligned}
&\ldots \overset{*}{\underset{\hat{M}}{\rightarrow}} (s^*, -|a^m|-, 0, u \# 1^0 \# 1^{h(n+2)} \# 1^n) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s^{**}, -|a^m|-, 1^{(h-1)(n+2)+1}, u \# 1^1) \\
&\overset{*}{\underset{\hat{M}}{\rightarrow}} (s^{***}, -|a^m|-, 0, u \# 1^1 \# 1^{(h-1)(n+2)+1}) \overset{*}{\underset{\hat{M}}{\rightarrow}} (s^*, -|a^m|-, 0, u \# 1^1 \# 1^{(h-1)(n+2)+1} \# 1^n) \\
&\overset{*}{\underset{\hat{M}}{\rightarrow}} \ldots \overset{*}{\underset{\hat{M}}{\rightarrow}} (s_9, -|a^m|-, h, u \# 1^h).
\end{aligned}$$

(iv) Now $\hat{M}$ can compute the h*th* symbol of w and can simulate the next step of M.

In this way $\hat{M}$ accepts a^m, $m = un(w)$, if and only if M accepts

$$\underbrace{wa \ldots a}_{|w|^k}.$$

Acknowledgment

I want to thank Zvi Galil who draw my attention to the subject of this paper.

References

[1] S.A. Cook, The characterization of pushdown machines in terms of time-bounded computers, J. Assoc. Comput. Mach. 18 (1971) 4–18.

[2] P. Duris and Z. Galil, Fooling a two-way automaton or one pushdown store is better than one counter for two way machines, Theoret. Comput. Sci. 21 (1982) 39–53.

[3] Z. Galil, Some open problems in the theory of computation as questions about two way deterministic pushdown automaton languages, Math. Systems Theory 10 (1977) 211–228.

[4] J. Hromković, One-way multihead deterministic finite automata, Acta Informatica 19 (1983) 377–384.

[5] O.H. Ibarra, On two-way multihead automata, J. Comput. System. Sci. 7 (1973) 28–36.

[6] B. Monien, Characterizations of time-bounded computations by limited primitive recursion, in: Automata, Languages and Programming, 2nd Colloquium 1974, Lecture Notes in Comput. Sci. 14 (Springer, Berlin, 1974) pp. 280–293.

[7] B. Monien, Transformational methods and their application to complexity problems, Acta Informatica 6 (1976) 95–108; also: Corrigenda in: Acta Informatica 8 (1977) 383–384.

[8] J.I. Seiferas, M.J. Fischer and A.R. Meyer, Separating nondeterministic time complexity classes, J. Assoc. Comput. Mach. 25 (1978) 146–167.

[9] A.C. Yao and R.L. Rivest, $k + 1$ heads are better than k, J. Assoc. Comput. Mach. 25 (1978) 337–340.