ACHIEVABLE HIGH SCORES OF ε-MOVES AND RUNNING TIMES IN DPDA COMPUTATIONS *

Paul M.B. VITÁNYI
Mathematisch Centrum, Amsterdam, The Netherlands

Received 7 August 1979

Automata theory, parsing, deterministic pushdown automata computations, maximal number of ε-moves, largest running times, highest inefficiency

1. Introduction

Large scores in the number of ε-moves a DPDA can make without entering a loop or decreasing its stack below the original stack height are investigated. The achieved scores are very near to an upper bound in the general case and are the upper bound for one-state DPDA's. Upper and lower bounds are derived for the worst case running times of accepting DPDA computations. Hence, given an arbitrary (non looping) DPDA, we have a priori tight upper and lower bounds on how inefficient its computations can be in the worst case. As will appear, these bounds do not follow straightaway from the largest amount of consecutive ε-moves a DPDA with given parameters can make in the worst case, since it may use a stacking and popping regime of ε-moves and read moves in an ingeneous way.

Deterministic pushdown automata (*DPDA's*) accept the so-called deterministic context free languages and constitute an important device in the theory of parsing and compiling [1]. Given a DPDA acceptor for some language (the device tells us whether an input word is in the language) we can convert it to a recognizer (the device tells us whether or not the input word is in the language) by eliminating *loops*, i.e., infinite sequences of consecutive *ε-moves* (non-reading machine steps). Schützenberger [5] showed how one can do so. Later proofs analyzed the amount of work involved in bringing a DPDA in loop-free form, which involved giving an upper bound on the number of consecutive ε-moves a DPDA can make without entering a loop or decreasing its stack below its original height.

In [3, Lemma 12.1] it is shown that for a DPDA with n_1 states, n_2 stack symbols and ℓ the maximal length of a string with which the topmost stack symbol can be replaced in a single move, $n_1(n_2+1)^{n_1 n_2 \ell}$ is such an upper bound. symbol can be replaced in a single move, $n_1(n_2+1)^{n_1 n_2 \ell}$ is such an upper bound. In [1, Algorithm 2.16] the slightly better upper bound of $n_1(n_2^{n_1 n_2 \ell} - n_2)/(n_2 - 1)$ (or n_1 if $n_2 = 1$) is given. Using a different approach, in [4] the upper bound of $(\ell^{n_1 n_2} - 1)/(\ell - 1)$ (or $n_1 n_2$ if $\ell = 1$) is given. This latter bound is achieved by using techniques already appearing in [6], where it is proven that we can test for looping configurations in DPDA's in time linear in the parameters. Hence the problem of determining the maximal number of consecutive ε-moves a DPDA can make without looping or decreasing the stack below the original stack height merits interest primarily as a combinatorial problem. In the present note we investigate how high a score a DPDA can actually achieve. It is shown that for DPDA's which read input

$$\frac{(\ell-1)^{n_1}\ell^{(n_1+1)(n_2-2)} - \ell^{n_2-2})}{(\ell-1)\ell^{n_2-2} - 1}$$

is an achievable lower bound on this maximal number of ε-moves for $n_1 \geq 1$, $n_2 \geq 3$ and $\ell \geq 2$. For $n_1 = 1$ (one-state DPDA's) this is also an upper bound, and the above score is very near to an upper bound in the general case. Finally, we give upper and lower bounds

* Registered at the Mathematical Centre as Report IW 70/76.

on the worst case running times of DPDA computations in which all input is read.

2. Results

Definitions and terminology closely follow [1]. We assume familiarity with the way of looking at DPDA computations of [4] and [6].

Let M be a DPDA with n_1, n_2 and ℓ as in the introduction. Denote the maximal number of consecutive ϵ-moves a DPDA M with these parameters can make, without entering a loop or decreasing its stack below the original stack height, by $f(n_1, n_2, \ell)$ where we assume that there is at least one (state, stack symbol) pair for which M reads input. When we do not impose the latter requirement we denote the corresponding function by $f'(n_1, n_2, \ell)$ and observe that DPDA's with parameters n_1, n_2, ℓ which score between $f(n_1, n_2, \ell)$ and $f'(n_1, n_2, \ell)$ accept the language $\emptyset$ or $\{\epsilon\}$.

Theorem 1. $f(n_1, n_2, \ell) \geqslant g(n_1, n_2, \ell)$, where

$$g(n_1, n_2, \ell) = \frac{(\ell - 1)^{n_1} \ell^{(n_1+1)(n_2-2)} - \ell^{n_2-2}}{(\ell - 1)\ell^{n_2-2} - 1}$$

for $n_1 \geqslant 1$, $n_2 \geqslant 3$ and $\ell \geqslant 1$.

Proof. Let the state set of M be $\varphi = \{1, 2, ..., n_1\}$ and let the set of stack symbols be $\Gamma = \{1, 2, ..., n_2\}$. The following *canonical scheme* (see [4]) for (1, 1) with respect to M will achieve the claimed lower bound. The canonical scheme is the context free grammar

$G = (\varphi \times \Gamma \cup \varphi, \varphi \cup \{(1, n_2)\}, (1, 1), P)$,

where P is defined by

(i) $(1, 1) \xrightarrow{\epsilon} (1, 2)(1, 2) \cdots (1, 2)(1, n_2)$,

(ii) $(i, n_2 - 1) \xrightarrow{\epsilon} (i + 1, 1)(i + 1, 1) \cdots (i + 1, 1)(i + 1, n_2)$ for $1 \leqslant i < n_1$,

(iii) $(i, j) \xrightarrow{\epsilon} (i, j + 1)(i, j + 1) \cdots (i, j + 1)$ for $1 \leqslant i \leqslant n_1$, $1 \leqslant j < n_2 - 1$ and $(i, j) \neq (1, 1)$,

(iv) $(i, n_2) \xrightarrow{\epsilon} i - 1$ for $1 < i \leqslant n_1$,

(v) $(n_1, n_2 - 1) \xrightarrow{\epsilon} n_1$,

where the lengths of the right-hand sides of rules (i)–(iii) is ℓ.

The unique leftmost derivation of the unique terminal word $i_1 i_2 \cdots i_k(1, n_2)$ produced by G represents the sequence of ϵ-moves of the corresponding DPDA M starting in state 1 with stack symbol 1 as its stack contents and ending in state 1 with stack symbol n_2 as its stack contents, i.e., the only (state, stack symbol) pair which reads input. Every direct production of the leftmost derivation corresponds to an ϵ-move of M and vice-versa. For an intermediate sentential form

$i_1 i_2 \cdots i_m(i_{m+1}, j_{m+1})(i_{m+2}, j_{m+2}) \cdots (i_s, j_s)(1, n_2)$

$i_1, i_2, ..., i_m$ are the return states (states resulting from) of all popmoves executed up to the present stage (and in historical order from left to right); i_{m+1} is the present state of the finite control and $j_{m+1} j_{m+2} \cdots j_s n_2$ is the present stack contents. i_{m+p}, $2 \leqslant p \leqslant s - m$, represents the state of the finite control when it accesses

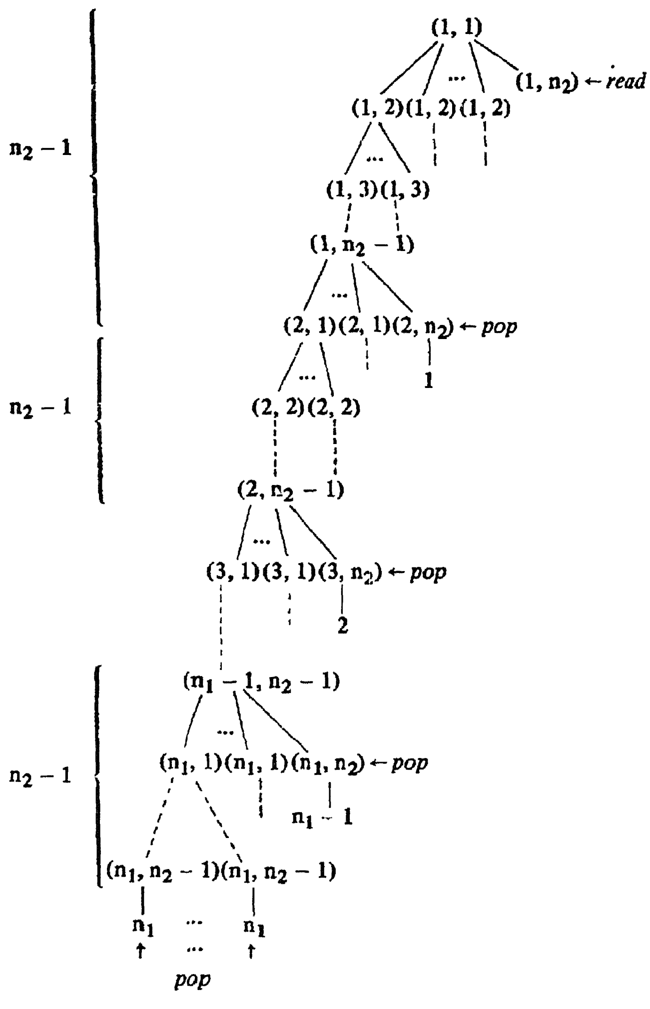

Fig. 1.

for the first time stack symbol j_{m+p}. (i)–(iii) correspond to pushmoves and (iv)–(v) to popmoves. The constraints on such a context-free grammar representing a nonlooping ϵ-computation are therefore:

(a) there are no circular nonterminals,

(b) there is a unique production for all nonterminals,

(c) if $(i, j) \xrightarrow{\epsilon} i' \in P$ (a popmove), then (i, j) can only occur in a right-hand side followed by (i'', j') for some $j' \in \Gamma$ if $i'' = i'$.

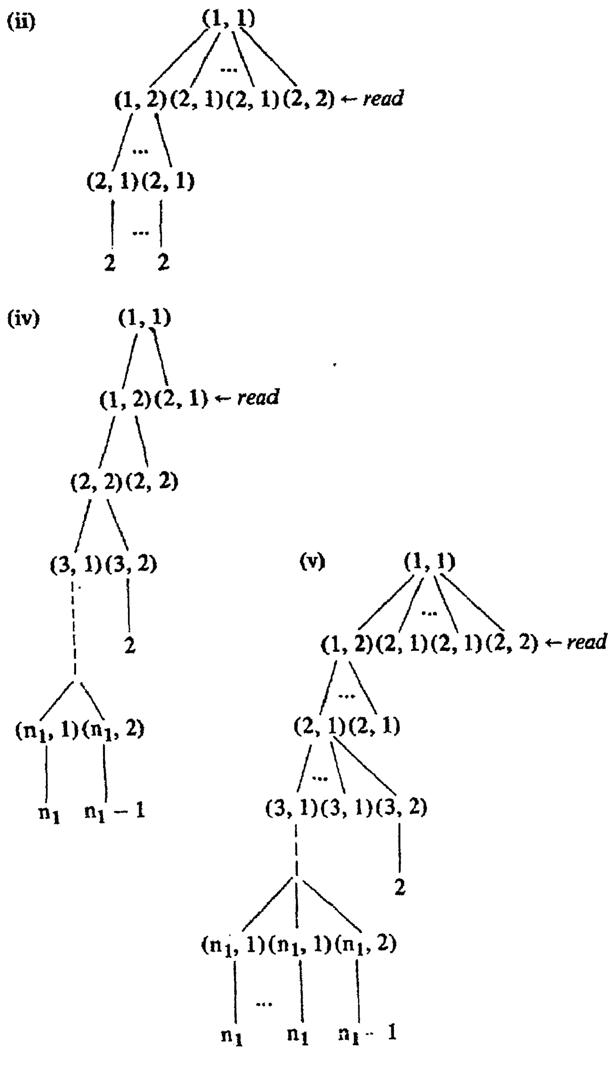

Fig. 2.

(a) and (b) garanty determinacy and absence of loops, while (c) garanties that the nonterminal right of a nonterminal which is rewritten according to (iv) or (v) will indeed represent by its first coordinate the retunr state of the executed pop. We display the derivation tree of the unique derivation in G in Fig. 1, where it is clear that identically labelled nodes are the roots of identical subtrees in the derivation tree. The internal nodes in the tree correspond to ϵ-moves of M and counting their number yields $g(n_1, n_2, \ell)$.

Corollary 2. If we do not insist on M having a (state, stack symbol) pair for a read move we achieve a score of consecutive ϵ-moves of

$$g'(n_1, n_2, \ell) = \frac{\ell^2}{\ell - 1}(g(n_1, n_2, \ell) - 1) + \ell + 1 ,$$

in the obvious way.

Corollary 3. For one-state DPDA's it is easily verified that $g(1, n_2, \ell)$ (and $g'(1, n_2, \ell)$) are also upper bounds, and indeed $g'(1, n_2, \ell)$ is equal to the bound in [4] for $n_1 = 1$. Therefore, $f(1, n_2, \ell) = g(1, n_2, \ell)$ for $n_2 \geqslant 3$ and $\ell \geqslant 2$.

For lower values of the parameters n_1, n_2, ℓ we can similarly to Theorem 1 derive $f(n_1, n_2, \ell) \geqslant g(n_1, n_2, \ell)$, where for $n_2 < 3$ or $\ell < 2$ $g(n_1, n_2, \ell)$ is given by:

(i) $g(1, 2, 2) = 1$,

(ii) $g(2, 2, \ell) = 2\ell$ for $\ell \geqslant 2$,

(iii) $g(n_1, 1, \ell) = n_1 - 1$,

(iv) $g(n_1, n_2, 1) = n_1 n_2 - 1$,

(v) $g(n_1, 2, 2) = 4n_1 - 4$ for $n_1 \geqslant 2$,

(vi) $g(n_1, 2, \ell) = 4((\ell - 1)^{n_1} - 1)/(\ell - 2) - 2(\ell - 1)^{n_1 - 1} - 2$ for $\ell \geqslant 3$ and $n_1 \geqslant 2$;

as we leave for the reader to verify, from Fig. 2.

That $g(n_1, n_2, \ell)$ is very near an upper bound on f is argued as follows. Since M needs at least one read move and n_1 popmoves to access all elements of $\varphi \times \Gamma$ (necessary for a balanced derivation tree), the number of pushmoves is less than $n_1(n_2 - 1)$ and $\ell^{n_1(n_2 - 1)}$ is surely an upper bound on the number of ϵ-moves. More detailed reasoning gets f close to g, and it seems very likely that f = g (and f' = g').

We now take a look at the running time of DPDA computations. The following fact belongs to the folklore in the field and is implicit in [2].

Lemma 4. DPDA's accept in linear time.

Proof. We can distinguish sequences of consecutive ϵ-moves, which from start to finish do not decrement the stack height below its starting height except possibly at the last move, in

(i) *popping* sequences, i.e., the last move decrements the stack height to 1 below its starting height,

(ii) *reading* sequences, i.e., those which end with a read move,

(iii) *looping* sequences.

Only reading sequences can increase the height of the stack and then by not more than $n_1 n_2(\ell - 1)$. Hence if M accepts a word $a_1 a_2 \cdots a_n$, the total number of symbols pushed on the stack (by sequences) is less than $n\, n_1 n_2(\ell - 1)$ and therefore the total running time is less than $n(n_1 n_2(\ell - 1) + 1)\, f(n_1, n_2, \ell)$, i.e., the combined length of popping and reading sequences.

It is clear that there is a trade-off between the fact that anything is stacked in a read sequence and whether a large sequence in the order of $f(n_1, n_2, \ell)$ is reached.

Let $T(n)$ be the longest running time of a computation by a DPDA M with parameters n_1, n_2, ℓ up to reading the n^{th} letter of an input $a_1 a_2 \cdots a_n$.

Theorem 5

$$(2n - 1)g(n_1, n_2, \ell) \leqslant T(n) \leqslant (n - 1)\frac{\ell^{n_1 n_2} - 1}{\ell - 1}.$$

Proof. $(2n - 1)\, g(n_1, n_2, \ell) \leqslant T(n)$. The lower bound on $T(n)$ is achieved by adding, in the proof of Theorem 1, the read move $(1, n_2) \xrightarrow{a} (1, 2)(1, 2) \cdots (1, 2)(1, 1)$ for each input letter a.

$T(n) \leqslant (n - 1)(\ell^{n_1 n_2} - 1)/(\ell - 1)$. In the proof of Lemma 4 we introduced sequences of ϵ-moves. If, starting from starting (state, stack symbol) pair the sequence of ϵ-moves leads to a read move and the stack height has been increased by $x(\ell - 1)$, then a popping or reading sequence has a length of less than $(\ell^{n_1 n_2 - x} - 1)/(\ell - 1)$ since there are at least x (state, stack symbol) pairs which lead to a premature read move. Hence the total number of ϵ-moves up to reading the n^{th} letter of input is less than $(n - 1)$ $((\ell - 1)x + 1)(\ell^{n_1 n_2 - x} - 1)/(\ell - 1)$ which is largest for $x = 0$.

Another, easier, subject is how large a stack a DPDA can accumulate up to reading the n^{th} letter of input. It is easy to show that

$$(n - 1)\, n_1 n_2(\ell - 1) + (n_1 n_2 - 2)(\ell - 1) + 1$$

can be reached, which seems to be the maximum. Notice, that the machine cannot achieve both a large score in stack height and running time.

References

[1] A.V. Aho and J.D. Ullman, The Theory of Parsing, Translation and Compiling, Vols. 1 and 2 (Prentice Hall, Englewood Cliffs, NJ, 1972).

[2] S. Ginsburg and S.A. Greibach, Deterministic context-free languages, Information and Control 9 (1966) 620–648.

[3] J.E. Hofcroft and J.D. Ullman, Formal Languages and their Relation to Automata (Addison-Wesley, Reading, MA, 1969).

[4] J. van Leeuwen and C. Smith, An improved bound for looping configurations in deterministic PDA's, Information Processing Lett. 3 (1974) 22–24.

[5] M.P. Schützenberger, On contextfree languages and pushdown automata, Information and Control 6 (1963) 246–264.

[6] L.G. Valiant, Decision procedures for families of deterministic pushdown automata, Ph.D. Thesis, Dept. Comp. Sci., Univ. of Warwick, England (1973).