# ECONOMY OF DESCRIPTION BY PARSERS, DPDA'S, AND PDA'S* 

Matthew M. GELLER'<br>University of Michigan. Ann Arbor. MI 48105. U.S.A.

Harry B. HUNT, III ${ }^{2}$
Harward University: Cambridge, MA 02I.38. U.S.A.
Thomas G. SZYMANSKI ${ }^{2}$ and Jeffiey D. ULLMA ${ }^{3}$
Princeton Liniversity, Princeton. N.'. . 540, U.S.A.

Communicated by A. Meyer
Received May 1976
Revised Scptember 1976


#### Abstract

It is shown that there is a sequence of languages $E_{1}, E_{i}, \ldots$ such that every correct prefix ${ }^{1}$ arser (one which detects errors at the earliest possible moment. e. C . IR or I L parsers) for $E_{n}$ has size $2^{2 n}$, yet a deterministic PDA recognizing $E_{n}$ exists and has size $O\left(n^{2}\right)$. There is another easily described sequence of languages $N_{1}, N_{2} \ldots$ for whic $\mathcal{N}_{5}$ hds a nondeterministic PDA of size $\mathrm{O}\left(n^{2}\right)$, but no deterministic PDA of size less than $2^{2 n}$. It is shown moreover, that this latter gap can be made arbitrarily large for different sequences of languages.


## 1. Introduction

Meyer and Fischer [10] attempted to analyze with respect to size certain systems for expressing languages. They obtained various results showing how specification of certain languages was far more economical in one system of specification than another. For example, they exhibit a family of languages $\left\{I_{n} \mid n \geqslant 1\right\}$ for which the size of finite automata needed to recognize languages of this family grows doubly exponentially in $n$. whereas the size of DPDA's recognizing $I_{n}$ grow as $\mathrm{O}\left(n^{3}\right)$.

In this paper, we exhibit several results of this flavor th t relate to pushdown automata (PDA's), deterministic pushdown automata (DPDA's), and parsers. The results relatirg to parsers are particularly interesting, as we show there can be an

[^0]exponential difference between the size of a minimal DPDA for a language and the size of any DPDA which tehaves as an $\operatorname{LR}(k)$ parser for the sume language. The technique invoived in the proof makes use of an unusual "closure property" which LR parsers possess but general DPDA's do not.

Geller and Harrison [3] present a model for comparing the size of the tables required by different bottom-up parsing algorithms for a given language. In [4] it is shown that a family of grammars, $\left\{G_{n} \mid n \geqslant 1\right\}$, exists, such that the size of production prefix parsers for $G_{n}$ grows as $O\left(n^{2}\right)$, yet the size of LR(0) parsers grows as $O\left(2^{n}\right)$. The following question, however, still remained. Can we transform each grammar $G_{n}$ to another grammar $G_{n}^{\prime}$, generating the same language, such that the size of $\mathrm{LR}(0)$ parsers recognizing $G_{n}^{\prime}$ grows polynomially in ..? In this paper, we answer that question in the negative. That is, a family of languages, $L_{n}$, is given for which there exists a family of grammars $G_{n}$, with the size of production prefix parsers (or precedence parsers or strict deterministic parsers) growing as $O\left(n^{2}\right)$. However, for any family of grammars generating $L_{n}$, the size of $\mathrm{LR}(0)$ parsers (in fact, the size of any correct prefix parser) grows as $O\left(2^{c n}\right)$ for some $c>0$. The correct prefix parsers include all parsers that halt as soon as an error has provably occurred. These include $\operatorname{LL}(k), \operatorname{SLR}(k), \operatorname{LALR}(k)$ and $\operatorname{LR}(k)$ parsers for all $k$. The correct prefix property and its relation to error detection and recovery in compilers is discussed in $[1,6]$.

We obtain several other results that relate the economy of description of certain families of languages to PDA's and DPDA's. A simple sequence of languages with an exponential size difference between PDA's and DPDA's recognizing them will be exhibiied, and a result of [10] is generalized to show that there is, for example, no recursive bound relating the size of DPDA's and PDA's for the same regolar set.

## 2. A family of languages that need exponentially growing PDA's for recognition

Definition 2.1. A scanning PDA is the standard PDA of Ginsburg [5] with the following modifications:
(1) To each input string we add an endmarker, $\$$.
(2) Acceptance occurs with only $Z_{0}$, the botiom of stack marker, on the pushdown store, the machine in a final state and the irsput tape empty.
(3) The stack can grow at most one symbol at a time. If it grows a symbol, the previous top stack symbol is not changed. That is, in one move, stack symbol $X$ can be replaced by the empty string, by some other symbol $Y$ or by $X Z$ for some symbol $Z$. We refer to the language accepted by $A$ as $I^{\prime}(A)$.

A configuration of a PDA will be denoted by a triple ( $q, \alpha, w$ ), where $q$ is the current state of the PDA, $\alpha$ is the contents of the pushdown store (with the top of the store on the right) and $w$ is the "unprocessed" portion of the input tape. The empty string will be denoted by $A$. By (2) above, a string $w$ is accepted by a scanning PDA if and only if

$$
\left(q_{0}, Z_{0}, w \$\right) \stackrel{*}{\vdash}\left(q_{f}, Z_{0}, \Lambda\right)
$$

where $q_{0}$ is the intitial state and $q_{t}$ is a final state of the rachine.

By the size of an automaton or grammar, we mean the number of symbols used to specify it. However, when dealing with PDA's, the use of the state-symbol product will be far more convenient. Since we are dealing $w$ th exponential gaps in this paper, we will be able to interchange these measures, as the following lemma indicates.

Lemma 2.2. For a given input alphabet there are constants $c_{1}>0$ and $c_{2}$ such that for any scanning PDA with standard description of length $n$ and $:$ tate -symbol product $m$, we have $c_{1} \sqrt{m} \leqslant n \leqslant c_{2} m^{2}$.

Proof. Let there be $s$ states and $t$ stack symbols, so $m=i t$. The standard list of alternates for each state, input and stack symbol of a scan ing PDA can have at most $s(2 t+1)$ entries, since there are, by condition (3) of Dt finition 2.1 , only $2 t+1$ stack moves performable in any situation. Thus, $n=\mathbf{O}((s i)[s(2 t+1)])=O\left(m^{2}\right)$.

For the lower bound on $m$ simply observe that $s$ and $t$ are each no greater than $n$, since each state and symbol must be mentioned in the standard representation of the PDA.

Lemma 2.3. There is a constant $c$ such that for any PDA with standard description of length $n$ there is an equivalent scanning PDA with desciption of length at most $c n^{2}$.

Proof. The standard constructions of Ginsburg [5] suffice.

From here on, we shall use "PDA" to mean scanning PI)A and "size" to mean state-symbo! product.

We next introduce the notion of a scan. A scan is a special kind of sequence of moves in which the symbols below that symbol which was on top of the pushdown store initially, have no effect on the behavior of the mactine.

Definition 2.4. Let $A$ be a (scanning) PDA with state set $Q$, input alphäbet $\Sigma$, pushdown alphabet $\Gamma$, initial state $q_{0}$, final state set $F$ and bottom of stack marker $\mathcal{Z}_{0}$. If $p$ and $q$ are in $Q, X$ and $Y$ are in $\Gamma$ and $w$ is in $\Sigma^{*} \cup \pm \$$, we say that $A$ makes a $q X p Y$ scan on $w$ provided there exist $q_{t} \in F, \alpha \equiv \Gamma^{*}$ and $t, z \in \Sigma^{*}$ for which

$$
\left(q_{1}, Z_{1}, t w z\right) \stackrel{*}{\vdash}(q, \alpha X, w z) \stackrel{*}{\vdash}(p, \alpha Y, z) \stackrel{*}{\vdash}\left(q_{f}, Z_{1}, 1\right)
$$

and in the sequence of moves

$$
(q, \alpha X, w z) \stackrel{*}{\vdash}(p, \alpha Y, z)
$$

the stack always contains at least $|\alpha X|$ elements.

We next prove a technical lemma that allows us to infer that if some sufficiently long string causes a scan, then it has some substring of smaller length that also causes a scan. In particular, we can find in any computation of a PDA a scan whose length is within a factor of two of some desired length. This lemma will be necessary for a future combinatorial argument, and it is a generalization of a lemma originally appearing in Lewis, Stearns and Hartmanis [9].

Lemma 2.5. Let $A$ be a PDA and let $c$ be any constant between 0 and 1 . Then if $x$ is any input of length at least 4/c which is accepted by $A$, we may write $x \$=x_{1} x_{2} x_{3} \$$ such that ! $c|x| \leqslant\left|x_{2}\right| \leqslant c|x|$ and $A$ makes a scan on $x_{2}$ or $x_{2} \$$.

Proof. We constuct $x_{2}$ by the following recursive algorithm. At all times we have a substring $y$ of $x \$$, with $|y|>c|x|$. on which $A$ makes a scan. Initially $y=x \$$. Whenever the algorithm calls itself, it does so on a string with a shorter scan than the given scan for $y$.

The scan on $y$ can be of twa, types, depending on whether the first move grows the stack or not.

Case 1. The scan of $y$ uses an inpul symbol before growing its stack. Then $y=a y^{\prime}$. for some $a \in \Sigma$, and

$$
(q, \alpha, K, y)^{*}\left(q^{\prime}, \alpha X^{\prime}, y^{\prime}\right){ }^{*}(p, \alpha Y, 1)
$$

Then $A$ makes a scan on substring $y^{\prime}$ of $x$. If $\left|y^{\prime}\right| \leqslant c|x|$, then $x_{2}$ is $y^{\prime}$. Note in the case where $\left|y^{\prime}\right| \leqslant c|x|$. we must have $\left|y^{\prime}\right| \geqslant \frac{1}{2} c|x|$. For $|y|>c|x|$, so $\left|y^{\prime}\right|>$ $c|x|-1$. If $\left|y^{\prime}\right|<\frac{3}{2} c|x|$, a contradiction of the hypothesis $|x| \geqslant 4 / c$ is immediate. If. on the other hand, $\left|y^{\prime}>c\right| x \mid$, recursively apply the algorithm to $y^{\prime}$.

Case 2. (The stack grows before the first use of an input symbol). Then we may write $y=b_{1} y_{1} b_{2} y_{2}$, with $b_{1}$ and $b_{2}$ each either in $\Sigma$ or equal to .1 , and the scan of $y$ may be written

$$
(q, \alpha X, y) \stackrel{*}{+}\left(q^{\prime}, \alpha X^{\prime} Z, y_{1} b_{2} y_{2}\right) \stackrel{*}{\vdash}\left(q^{\prime \prime}, \alpha X^{\prime \prime} Z^{\prime}, b_{2} y_{2}\right)+\left(q^{\prime \prime \prime}, \alpha X^{\prime \prime}, y_{2}\right) \stackrel{*}{\vdash}(p, \alpha Y, 1)
$$

where scans are made on $b_{1} y_{1} b_{i}, y_{1}$ and $y_{2}$.
If $y_{1}|>c| x \mid$. call the algorithm recursively on $y_{1}$ if $\frac{1}{2}|x| \leqslant\left|y_{1}\right| \leqslant c|x|$, then choose $x_{2}=y_{1}$, and we are done. The only case remaining is where $\left|y_{1}\right|<!c|x|$. Suppose $\left|b_{1} y_{0} b_{2}\right| \geqslant \frac{1}{2} c|x|$. Given the hypothesis $|x| \geqslant 4 / c$, together with the assumption $\left|y_{1}\right|<\frac{1}{2}|x|$ it is easy to show $\left|b_{i} y_{1} b_{2}\right| \leqslant c|x|$. Thus, we may pick $b_{1} y_{1} b_{2}$ for $x_{2}$. Thus assume $\left.b_{1} y_{1} b_{2}\left|<\frac{1}{3} c\right| x \right\rvert\,$. Then $\left|y_{2}\right|>\frac{1}{2}|x|$. If $\left|y_{2}\right| \leqslant c|x|$ we are done; if not, apply the algorithm to $y$. Note that $y$. may be $y$, but the scan $\left(q^{\prime \prime \prime} \alpha X^{\prime \prime}, y_{2}\right)^{*}(p, \quad Y, i)$ is shorter than the given scan for $y$ so the algorithm will converge.

We now restrict ourselves to a special family of languages. We let $\Sigma_{n}=$ $\left\{a_{1}, \ldots, a_{n}\right\}$, and let $L_{n}$ be the set of permutations of $\Sigma_{n}, \imath \geqslant 1$.

We next wish to show that distinct permutations establish distinct scans when one of the languages $L_{n}$ is being recognized.

Lemma 2.6. Let $P$ be a PDA accepting $L_{n}$. If $P$ makes $p X_{q} Y$ scans on two strings $x$ and $y$, then $x$ and $y$ are permutations of one another.
proof. Otherwise, substitute $y$ for $x$ and accept a word r ot in $L_{n}$.
We are now ready to prove that PDA's accepting $L_{n} £$ row as $2^{c n}$ in size.
Theorem 2.7. There exists a constant $c>0$ such that for $n \geqslant 6$, any PDA $F$ accepting $L_{n}$ has size at least $2^{\text {cn }}$.

Proof. It suffices to show that for some $d>0$ there are $2^{a n}$ distinct strings, not permutations of one another, such that while accepting sorie word, $P$ makes scans on those strings. For then by Lemma 2.6, there must be $2^{d n}$ quadruples $q X p Y$ such that $q X p Y$ scans are performed by $P$. Therefore, the size of $P$ is at least $2^{d n / 2}$.

Construct a maximal coiiection of sets $S_{1}, S_{2}, \ldots, S_{s,}$ each included in $\mathbf{\Sigma}_{n,}$ such that
(1) for each $S_{i}$, there is some string $w_{r}$, a permutation of $S$, such that $P$ performs a scan on $w_{1}$ or $w_{1} \$$ and
(2) $n \leqslant k_{1} \leqslant \frac{2}{3} n$ for all $i$, where $h_{1}$ is the size of $S_{1}$.

The number of strings in $L_{n}$ that contain the symbols of $S_{1}$ as a substring is $k_{1}!\left(n-k_{1}+1\right)$ !. By condition (2), this number is at most $(n / 3)!((2 n / 3)+1)!$. Thus, if

$$
s<\frac{1}{n+1}\binom{n+1}{n / 3},
$$

there is some string $w$ in $L_{n}$ which contains none of $S_{1}, S_{i}, \ldots, S_{\mathrm{s}}$ as a substring. By Lemma 2.5 with $c=2 / 3, w$ causes $P$ to perform a scan on a substring $w^{\prime}$ or $w^{\prime} \$$, where $w^{\prime}$ is of length between $\frac{1}{3} n$ and $\frac{2}{3} n$. By hypothesis, the set $S$ of symbols of $w^{\prime}$ is none of $S_{1}, S_{2}, \ldots, S_{n}$. Thus $\left\{S_{1}, S_{2}, \ldots, S_{4}\right\}$ was not maximal as supposed. Hence

$$
s \geqslant \frac{1}{n+1}\binom{n+1}{n / 3} \geqslant 2^{d n}
$$

for some $d>0$.
We also wish to consider another family of languages, namely $Q_{n}=$ $\left\{x \# x \mid x \in\{0,1\}^{n}\right\}$. Note that words in $Q_{n}$ are of length $?_{n}+1$. We get a similar result, namely:

Theorem 2.8. There exists a constant $c>0$ such that for $n \geqslant 8$, any PDA recognizing $Q_{n}$ has size at least $2^{\text {cn }}$.

Proof. As in Lemma 2.6, we can show that a PDA $P$ accepting $Q_{n}$ cannot make $q X p Y$ scans on two distinct strings whose lengths do not exceed $n+1$. (Note tha: this result does not hold for strings of greater length, as the strings could contain corresponding symbols of the two copies of $x$ in $x \# x \$$ ).
Thus, consider a maximal set of strings $y_{1}, y_{2}, \ldots, y$ in $(0+1)^{*}$ that
(1) For each $y_{i}$ there is a string in $Q_{n}$ on which $P$ makes a scan, and
(2) $(n+1) / 2 \leqslant\left|y_{i}\right| \leqslant n+1$ for all $i$.

Any word in $Q_{n}$ is determined by knowing any $n+1$ consecutive positions. Each $y_{i}$ is therefore a substring of at most $2^{n+1-\left|y_{1}\right|} \leqslant 2^{(n+1) / 2}$ words of $Q_{n}$. By Lemma 2.5 with $c=(n+1) /(2 n+1)$ we may use the technique of Theorem 2.7 to show $s \geqslant 2^{(n+1) / 2}$ for $n \geqslant 8$. The balance of the argument follows Theorem 2.7.

We now need a lemma relating a closure property to the growth rate of sequences of PDA's.

Lenma 2.9. Let $\left\{M_{n} \mid n \geqslant 1\right\}$ be some family of languages and $\left\{R_{n} \mid n \geqslant 1\right\}$ a family of regular sets. Assume that there exists a constant $c>0$ such that for sufficiently large $n$, any PDA recognizing $M_{n} \cap R_{n}$ is of size greater than $2^{\mathrm{cn}}$. Also assume there exists some function $f(n)$, such that for sufficiently large $n$ some finite automaton recognizing $R_{n} \$$ has at most $f(n)$ states. Then for sufficiently large n, any PDA recognizing $M_{n}$ is of size greater than $2^{\text {cn }} / f(n)$.

Proof. If not, then the standard "cross product of states" construction [5] provides a PDA recognizing $M_{n} \cap R_{n}$ with size less than $2^{c n}$.

We now give two examples where this lemma is applied.

Example 2.10. We let $M_{n}=\left\{\right.$ the set of strings in $\Sigma_{n}^{*}$ containing at least one instance of each symbol\}. We know $L_{n}=M_{n} \cap\left(\Sigma_{n}\right)^{n}$, and $\left(\Sigma_{n}\right)^{n} \$$ is recognized by an $n+2$ state auiomaton. Therefore, it follows from Theorem 2.7 and Lemma 2.9 that there exists a constant $c>0$ such that for sufficiently large $n$. any PDA recognizing $M_{n}$ is of size greater than $2^{c n}$. This example will be useful later on in examining the applications of the results of this section to parsing.

Example 2.11. Let $P_{n}=\left\{x_{1} 2 x_{2} 2 \cdots 2 x_{k} 22 x_{i} \mid x_{i} \in\{0,1\}^{n}\right.$ for $1 \leqslant j \leqslant k, 1 \leqslant i \leqslant k$, and regarded as binaiy integers, $x_{1}<x_{1+1}$ for $\left.1 \leqslant j<k\right\}$. Consider $P_{n} \cap(0+1)^{n} 22(0+1)^{n}$, which is essentially $Q_{n}$. Therefore, it follows from Theorem 2.8 and Lemma 2.9 that there exists a constant $c>0$ such that for sufficiently large $n$, any PDA recognizing $P_{n}$ is of size greater than $2^{c n}$. This example resolves a conjecture of Meyer and Fischer [10].

## 3. The size of minimal PDA's recognizing languages compared with the size of minimal DPDA's and other deterministic devices

In this section, we first demonstrate a particular family of languages $N_{n}$ for which the size of PDA's accepting $N_{n}$ grows polynomially in $n$, yet the size of DPDA's recognizing $N_{n}$ grows exponentially in $n$. We then show that no recursive function can bound the gain in enconomy of PDA's over DPDA's.

Lemma 3.1. Let $L \subseteq \Sigma^{*}$ be a language accepted by a DPDA $P$ of size $m$. Then $\bar{L}$ is accepted by a DPDA of size 3 m .

Proof. Delete from the description of $P$ any transition $\delta(q, \Lambda, X)$ if for every $i$, there exists a state $p$ and non-empty string $\gamma$ such that $(q, X, A) \stackrel{H}{(p, \gamma, 4)}$. The test for such transitions can in fact be performed in polynomial time, although this fact is irrelevant to the present proof. The resulting DPDA has no loops. Then use the construction of [5] to complement the language accepted by the DPDA.

This result leads to the following theorem, which shows that there exists a natural sequence of languages for which small PDA's exist, yet for which DPDA's must be large.

Theorem 3.2. There exists a constant $c>0$ such that for sufficiently large $n$, any DPDA P accepting

$$
N_{n}=\left\{a_{i, 1}, a_{1}, \cdots a_{t n} b_{n} b_{12} \cdots b_{l m} \mid 1 \leqslant i_{k} \leqslant n, 1 \leqslant j_{k} \leqslant n \text { and }(\exists r)(\forall s) i_{r} \neq j_{s}\right\}
$$

has size at least $2^{\text {cn }}$.
Proof. By Lenvma 3.1, given $P$ we may construct $P^{\prime}$, of size polynomial in the size of $P$, accepting $\bar{N}_{n}$. Construct DPDA $P^{\prime \prime}$ to simulate $P^{\prime}$ on (imaginary) input $a_{1} a_{2} \cdots a_{n}$ and then read a (real) input string of $b$ 's, again simulating $P^{\prime}$. The size of $P^{\prime \prime}$ is also polynomial in the size of $P$. But $P^{\prime \prime}$ accepts the language $M_{n}$ of Example 2.10, with $a$ 's recoded as $b$ 's, which we showed requires exponentially sized PDA's.

Comparison of the size of DPDA's and PDA's for context free grammars reveals a property displayed by Meyer and Fischer [10] between finite automata and context free giammars. That is, the gain in economy can be arbitrary ${ }^{3}$. We prove a considerably stronger result, in fact.

[^1]Lemma 3.3. [10]. Let $f$ be any recursive function. Then there is a Turing machine $T_{f}$ which on any input of length $n$ accepts after a sequence of at least $f(n)$ moves. Moreover, there is a constant $k$ such that for each input $x$ of length $n$ there is a context free grammar $G_{l, x}$ of size at most $k n$. which generates the noncomputations of $T_{f}$ on input $x$.

Note that $\overline{L\left(G_{f: ~}\right)}$ is a single string of length at least $f(n)$, whenever $f(n)$ is defined.

Lemma 3.4. Let $\mathscr{D}$ be any class of language descriptors, e.g., DPDA's. Suppose that for any $D \in \mathscr{D}$ there exists $D^{\prime} \in \mathscr{D}$ defining the complement of $L(D)$, the language defined by $D$, and $c(n)$ is a total recursive function such that size $\left(D^{\prime}\right) \leqslant c(\operatorname{size}(D))$. Further, let $z(n)$ be a total recursive function such that if $L(D)$ has a word $x$ of length greater than $i(\operatorname{size}(D)$ ), then $L(D)$ has some word besides $x$, i.e., $x$ can be "pumped." Then there is no total recursive function $g$ such that for every CFG G of size $m$ generating a regular set there is a descriptor $D \in \mathscr{A}$ defining $L(G)$, with $\operatorname{size}(D) \leqslant g(m)$.

Proof. Suppose there were such a $g$, and let $f(n)=z c g\left(n^{2}\right)$. By Lemma 3.3, there is an integer $k$ and a sequence of context free grammars $G_{1}, G_{2}, \ldots$, where $G_{i}$ is of size at most $k i$ and generates a language whose complement is one string of length at least $f(i)$. By hypothesis, there is a descriptor $D_{11}$ such that $L\left(D_{1}\right)=L\left(G_{k}\right)$ and $\operatorname{size}\left(D_{1}\right) \leqslant g\left(k^{2}\right)$. Then there is a descriptor $D_{2}$ such that $L\left(D_{2}\right)=\overline{L\left(G_{k}\right)}$, and $\operatorname{size}\left(D_{2}\right) \leqslant \operatorname{cg}\left(k^{\prime}\right)$.

If $z c g\left(k^{2}\right) \leqslant f(k)$, then $\overline{L\left(G_{k}\right)}$ contains more than one string, so $f(k)<z c g\left(k^{2}\right)$. But $f(k)=z c g\left(k^{2}\right)$ by definition. Hence, $g$ does not exist.

Theorem 3.5. For regular sets there is no recursive relationship beiween the sizes of a context-free grammar (or PDA) and the smallest equivalent finite automaton, DPDA or 1 way deterministic stack automaton.

Prsof. Closure of one way deterministic stack automata under complement is easy. "Pumping" for these devices follows from [11].

Independently, Valiant [13] has obtained a related result, that there is no recursive relationship between the sizes of anambigious context free grammars and DPDA's for the same language. Close examination of Valiant's construction reveals that the unambiguous grammars involved are actually deterministic grammars, that is, grammars which are LR. He has thus shown that no recursive relationship holds between the sizes of a deterministic grammar and the smallest eguivalent DPDA.

## 4. Applications to parsing

We now wish to apply the results we have obtained to parsing. Geller, Graham and Harrison [4] snow that there exist families of grammars $\left\{G_{n} \mid n \geqslant 1\right\}$ for which the size of production prefix parsers grows as $c n^{2}$, while $\operatorname{SLR}(k)$ parsers grow as $2^{c n}$. In this section, we shall show that for this family of grammars, for any grammars generating the same language, $\operatorname{SLR}(k)$ parsers must grow as $2^{c n}$.

This result will follow from the fact that by nature of the correct prefix property (cf. Graham and Rhodes [6]) correct prefix parsers have the task of recognizing two distinct languages. That is, the parsers must halt with the correct parse after reading a correct input, and must also halt and declare error on an input as soon as the input has been found incorrect. We shall exhibit a seequence of languages which can be recognized by a sequence of DPDA's growing polynomially in size, yet any sequence of PDA's recognizing the set of input strings on which the parser first declares error grows exponentially in size. First, we need some definitions.

Definition 4.1. Let $L \subseteq \Sigma^{*}$ be a language. Then $x \in \mathbf{I}^{*}$ is a correct prefix of $L$ if and only if there exists some $z \in \mathbf{\Sigma}^{*}$ such that $x z \in L$.

Definition 4.2. If $L$ is a deterministic CFL, let $I(L)=\{x \mid x=y a$ for some symbol a, $y$ is a correct prefix of $L$, but $x$ is not $\}$.

Definition 4.3. Le: $\boldsymbol{A}$ be a DPDA acting as a parser for $L$. Then $A$ is a correct prefix parser if by changing the set of final states of $A$ we produce a recognizer for $I(L)$. Note that many types of parsers are correct prefix parsers. Important examples are $\operatorname{LR}(k), \operatorname{LL}(k) . \operatorname{SLR}(k)$ and $\operatorname{LALR}(k)$.

It follows by definition that:

Lemma 4.4. Let $L$ be a deterministic CFL. Then the smallest correct prefix parser for $L$ is no smaller than the smallest DPDA for $I(L)$.

We can apply Lemma 4.4 to exhibit a specific sequence of languages having an exponential gap between the sizes of their smallest DPDA's and smallest correct prefix parsers.

Theorem 4.5. There is a constant c such that for sufficiently large n. every correct prefix parser for the language

$$
E_{n}=\left\{a_{11} a_{t 2} \cdots a_{t n} b_{1} \mid 1 \leqslant i_{k}, j \leqslant n \text { and } j \neq i \text {, for any } r\right\}
$$

has size at least $2^{2 n}$.

Proof. We see that $I\left(E_{m}\right) \cap \Sigma_{n-1}^{*} \cdot a_{n}=M_{n-1} \cdot a_{n}$, where $M_{n-1}$ is the language of Example 2.10. By an argurient similar to that of Example 2.10, there is a constant $c$ such that for sufficiently large $n$, every DFDA recognizing $M_{n-1} \cdot a_{n}$ has size at least $2^{\text {cn }}$. But, by Lemmas 2.9 and 4.4, every correct prefix property parser for $E_{n}$ is at least as large as any DPDA recognizing $M_{n-1} \cdot a_{n}$.

There are, however, strict deterministic and production prefix parsers for each $E_{n}, n \geqslant 1$ with size $O\left(n^{2}\right)$.

Theorem 4.6. Consider the sequence of grammars

$$
\begin{aligned}
& G_{n}=\left(V_{n}, \Sigma_{n}, P_{n}, S\right) \text { where } n \geqslant 1, \Sigma_{n}=\left\{a_{1}, b_{1} \mid 1 \leqslant i \leqslant n\right\} \text { and } \\
& P_{n}=\left\{A_{1} \rightarrow a_{1} A_{1} \mid 1 \leqslant i, j \leqslant n, j \neq i\right\} \cup\left\{A_{1} \rightarrow b_{1} \mid 1 \leqslant i \leqslant n\right\} \\
& \cup\left\{S \rightarrow A_{1} \mid 1 \leqslant i \leqslant n\right\} .
\end{aligned}
$$

Then:
(1) $L\left(G_{n}\right)=E_{n}$ for all $n \geqslant 1$.
(2) There exist production prefix and strict deterministic parsers for $G_{n}$ with size $O\left(n^{2}\right)$.

Proof. (1) Is easy to show.
(2) Is found in [4].

Remark. It is easy to show the converse to Theorems 4.5 and 4.6 , that is, for every DPDA $X$ there is an equivalent correct prefix parser at most exponentialiy larger than $X$. By way of proof, consider the construction of the "predicting machine" in [8].

## 3. Comelusions

In summary, we have exhibited families of languages with the foliowing behaviors:

| size correct | minimum size | minimum size |
| :---: | :---: | :---: |
| prefix parser | DPDA | PDA or CFG |


| $L_{n} M_{n} Q_{n}$ | $2^{c n}$ | $2^{c n}$ | $2^{c n}$ |
| :--- | :--- | :--- | :--- |
| $\mathbf{N}_{n}$ | $2^{c n}$ | $2^{c n}$ | $\leqslant c n^{2}$ |
| $E_{m}$ | $2^{c n}$ | $\leqslant n^{2}$ | $\leqslant n^{2}$ |

We have shown one of the conjectures of Meyer and Fischer [10] to be true. In addition, we have shown that the gain in economy of description of a PDA over a DPDA or even over a one way deterministic stack automaton can be arbitrary.

## Acknowledgement

The authors would like oo thank the referees and Albert Meyer for their helpful comments on an early draft of this paper.

## References

[1] A.V. Aho and J.D. Ullman. The Theory of Parsing, Translation, cind Compiling, Vols. I and II (Prentice Hall, Englewood Cliffs, NJ. 1972 and 1973).
[2] M.M. Geller, Compact parsei, for deterministic langauges, Ph.D. Thesis, University of California, Berkeley (1975).
[3] M.M. Geller and M.A. Matison. Strici deterministic vetsus $\mathbb{L R}$ ( 0 ) parsing, Conf. Record ACM Symp. Principles of Programming Languages (1973) 22-.32.
[4] M.M. Geller, G.L. Graham and M.A. Harrison. Production piefix parsing, Second Colloq. Automata, Languages and Programming (1971) 232-2.41.
[5] S. Ginsburg. The Markemattcal Theory of Context-Free Language: (McGraw-Hill. NY, 1966).
[6] S.L. Graham and S.P. Rhodes, Practical syntactic error recovery, Comm. ACM 18 (11) (1475) 639-650.
[7] M.A. Harrison and I.M. Havel. Strict deterministic grammars. J. Comput. System Sci 7 (1973) 237.277.
[8] J.E. Hopcroft and J.D. Ullman, Formal Languages and their Reiation to Automata (AddisonWesley, Reajing. MA, 1969)
[9] P.M. Lewis II, R.E. Sterns and J. Hartmanis, Memory bounds for recognition of coniext-free and context-sersitive languages, IEEE Conf. Record 6th. Ann. Symp. Switching Circui: Theory and Logical Design (1966) 191-202.
[10] A.R. Meyer and M.J. Fischer, Economy of descripion by automata, grammars, and formal systems, Conf. Record 1971 Ann. Symp. Switching and Aiutomata Theory. (1971) 188-191.
[11] W. Ogden. Intercalation theorems for stack languages, Proc. ACly Symp. Theory of Computing (1969) 31-42.
[12] R.E. Stearns, A regularity test for pushdown machines. Information and Control 11 (3) (1967) 323-340.
[13] L.G. Valiant, A succinctness result for descriptions of deterministic languages, TR 70. Universit. of Leeds, (July. 1975).


[^0]:    * A preliminary version of this report appeared in the Proceedings of the IEFE 16th Annual Symposium on Foundations of Computer Science.
    ${ }^{1}$ Work supported in part hy NSF Grant MCS-76-1)774t
    ? Work supporied in part by US. Army Contract DA-31-124-ARO-[1-4ã.
    * Work supported in part by NSF Grant DCR-74-21939.

[^1]:    - A.R. Meyer points out that this result follows from Meyer and Fischer [10] and the result of Stearns [12], which showed a recursive relationship between the sizes of minimal finite automata and DPDA's for a given language.

