On Size Bounds for Deterministic Parsers

*Esko Ukkonen**

Department of Computer Science, University of Helsinki
Tukholmankatu 2, SF-00250 Helsinki 25, Finland

Abstract. Bounding the size of deterministic left and right parsers for context-free grammars is studied. It is well-known that the size of an LR(k) parser is not always polynomially bounded in the size of the grammar. A similar non-polynomial size difference occurs also in LL(k) parsers. We show that such non-polynomial size differences cannot be regarded as a weakness of the LR(k) or LL(k) parser construction methods but are a consequence of the capability of these parsers to solve inherently difficult parsing problems. This is established by proving that there exists an infinite family of LL(k) grammars where $k \geq 2$ such that the size of *every* left parser for these grammars must be $\geq 2^{cm}$ where m is the size of the grammar. Similarly, it is shown that there exists an infinite family of LR(k) grammars (as well as SLR(k) and LALR(k) grammars) where $k \geq 0$ such that the size of *every* right parser for these grammars must be $\geq 2^{c'\sqrt{m}}$.

Key Words: size complexity, left parser, right parser, LL(k) parsing, LR(k) parsing.

1. Introduction

Several classes of parsers are known for which there is at most a polynomial difference between the size of a parser and the size of the context-free grammar to be parsed. Such parsers include, among others, different families of precedence and bounded context parsers (see e.g. [1]) and strict deterministic parsers [6]. These parsing methods can be applied only on grammars that satisfy relatively strong restrictions.

The situation is changing if we consider the LL(k) and LR(k) parsing methods [8,7]. As is well-known, the LL(k) grammars contain all the context-free grammars that can be deterministically parsed top-down in a natural way using k symbol lookahead. The LR(k) grammars have the same property for bottom-up parsing. In this sense LL(k) and LR(k) parsers are as powerful as we may hope. It turns out, however, that a price for generality is that an LL(k) or LR(k) parser produced by the standard construction algorithms (as given e.g. in [1]) may be non-polynomially larger than the corresponding LL(k) or LR(k) grammar. For the LR(k) method this size gap was observed in [2]. We note that for the LL(k) method the gap exists if $k \geq 2$.

Thus we are forced to ask whether or not such non-polynomial size differences are necessary. If the answer is negative, it might, in principle, be possible to find a parser construction for LL(k) or LR(k) grammars always producing polynomially bounded

* Present address: University of California, Computer Science Division, Berkeley, Ca 94720, U.S.A.

parsers which therefore sometimes are exponentially smaller than the parsers produced by the present constructions. In fact, in some special cases such an improvement is known: in [3] a family of grammars is given for which the LR(0) parsers are non-polynomially larger than the production prefix parsers. An explanation of this result is that, unlike the production prefix parsers, the LR(0) parsers have the correct prefix property, that is, the string read by the parser is always a prefix of some correct string in the language.

In this paper we are able to show that in the general case such improvements are impossible. We prove that independently of the parsing method used, the non-polynomial gaps cannot be totally avoided for the classes of LL(k) and LR(k) grammars. This means that the correct prefix property is not the only reason for the non-polynomial size of LL(k) and LR(k) parsers.

To prove our results on LL(k) grammars we give an infinite sequence of LL(2) grammars and show that the size of *any* left parser for such a grammars must be at least an exponential function of the size of the grammar. It should be emphasized that our proof does not assume any unnecessary conditions such as the correct prefix property or the use of some fixed length lookahead. The only requirement is that a left parser is a deterministic pushdown transducer capable of producing the translation from terminal strings to left parses. (Technically, our proof is for right parsers and the original assertion follows by a simple covering argument). This result implies that if $k \geq 2$, an LL(k) or LR(k) parser must sometimes be exponentially larger than the grammar. On the other hand, since an LL(1) grammar is always strong LL(1) and thus has a left parser of polynomial size, in the LL(1) case the size difference is only polynomial.

We also analyze a sequence of LR(0) grammars mentioned already by Earley [2] (and attributed by him to John Reynolds) as an example of a grammar family for which the LR(k) construction gives non-polynomially large parsers. We show that *all* right parsers for these grammars must be at least of the same size as an LR(0) parser. A non-polynomial difference between the parser size and grammar size therefore exists for LR(k), SLR(k) and LALR(k) grammars when $k \geq 0$.

No similar results seem to be known in the literature. Most closely related is perhaps a work [4] by Geller, Hunt, Szymanski and Ullman investigating the size of different pushdown automata (but not parsers, i.e. pushdown transducers emitting a parse) in a general setting comparable to ours. For example, they generalized the result from [3] mentioned above by giving a family of languages such that there is an exponential difference between the size of a minimal deterministic pushdown automaton (DPDA) for a language and the size of any DPDA with the correct prefix property for the same language. They also gave a family of languages $\{N_n\}$ such that there is an exponential difference between the size of a minimal context-free grammar for a language and the size of any DPDA for the same language. It is easily seen that this exponential difference is not preserved if parsers in our sense are considered. This is because every LL(k) or LR(k) grammar for N_n must be exponentially larger than the minimum size context-free grammar for N_n.

2. Definitions and preliminary results

Our notation concerning strings, context-free grammars and pushdown automata is mainly as that of [1] with the exception that the *length* of a string s is denoted by $lg(s)$. Recall that e denotes the empty string and $\#W$ the number of elements of a set W.

The *size* of a (context-free) grammar $G = (N,\Sigma,P,S)$ is defined by

$$|G| = \sum_{A\to\alpha\in P} lg(A\alpha).$$

As noted in [5], the *norm* of G given by $||G|| = |G|\cdot\log_2\#(N\cup\Sigma)$ is a more realistic measure, but because $||G|| \leq |G|\cdot\log_2|G| \leq |G|^2$ and we are interested in proving larger than polynomial gaps, the more convenient measure $|G|$ can be used.

Recall that if there is in G a leftmost derivation $S \Rightarrow_L^\pi \alpha$ where π is the sequence of productions applied in the derivations, then π is called a *left parse* of α in G. Similarly, if there is a rightmost derivation $S \Rightarrow_R^\pi \alpha$ then the *reverse* of π is a *right parse* of α in G.

A *parser* for G is a deterministic pushdown automaton (DPDA) accepting language $L(G)$ and giving for each w in $L(G)$ a parse of w as an output. In general, a DPDA with output is called a *deterministic pushdown transducer* (DPDT) and defined as an 8-tuple $T = (Q,\Sigma,\Gamma,\Delta,\delta,q_0,Z_0,F)$, where Q denotes states, Σ input alphabet, Γ pushdown alphabet, Δ output alphabet, δ transition function from $Q\times(\Sigma\cup\{e\})\times\Gamma$ to $Q\times\Gamma^*\times\Delta^*$ satisfying the following determinism condition: if $\delta(q,e,t)$ is defined then $\delta(q,a,t)$ is undefined for all $a\in\Sigma$. Furthermore, q_0 denotes initial state, Z_0 initial pushdown element and $F\subset Q$ the final states. A *configuration* of T is denoted by (q,x,α,y) where $q\in Q$ is the current state, $x\in\Sigma^*$ is the unused portion of the input, $\alpha\in\Gamma^*$ is the current content of the pushdown stack, and $y\in\Delta^*$ is the output string emitted to this point. The *next move relation* $\vdash$ among configurations is defined in the usual way.

Let now $\$$ be a symbol not in Σ. A (*deterministic*) *left parser* for grammar $G = (N,\Sigma,P,S)$ is formally defined as a DPDT $T = (Q,\Sigma\cup\{\$\},\Gamma,P,\delta,q_0,Z_0,F)$ such that T accepts by final state and empty stack the language $L(G)\$$ and for each accepted input $x\$$, T outputs a left parse of x in G. Thus G has a derivation $S \Rightarrow_L^\pi x$ for some terminal string x if and only if T has a move sequence $(q_0,x\$,Z_0,e) \vdash^* (q,e,e,\pi)$ for some q in F.

Similarly, a (*deterministic*) *right parser* for grammar $G = (N,\Sigma,P,S)$ is a DPDT T which is like a left parser but outputs a right parse of x for each input $x\$$ where x is in $L(G)$.

The *size* of a DPDT T is defined by (c.f. [5])

$$|T| = \sum_{\delta(q,a,Z)=(q',\alpha,\gamma)} (3+lg(a)+lg(\alpha)+lg(\gamma))$$

Thus the size means the length of a string listing the transition function. As for grammars, $||T|| = |T|\cdot\log_2\#(Q\cup\Sigma\cup\Gamma\cup\Delta)$ is a more realistic measure, but because again $||T|| \leq |T|\cdot\log_2|T| \leq |T|^2$, the more convenient measure $|T|$ can be used without loss of generality in proving non-polynomial gaps between the sizes of T and G.

A parser or a DPDT T is called *moderate* if its transition function δ is such that whenever $\delta(q,a,Z) = (q',\alpha,\gamma)$ then $lg(\alpha) \leq 2$. If $lg(\alpha) > 2$, by adding new states we may easily replace this transition step by $lg(\alpha)-1$ equivalent moderate steps. This makes the description of the step less than 5 times as long as originally. Thus:

Lemma 1. *For each parser T, there is an equivalent moderate parser T' such that $|T'| < 5\,|T|$.* ■

3. Left parsers

Our results on left parsers and on LL(k) parsers in particular are based on properties of a sequence of grammars $G_n = (N_n,\Sigma_n,P_n,A_0)$ where

$$
\begin{array}{lll}
P_n\colon & A_i \to a_{i+1}A_{i+1}B_{i+1} \mid d_{i+1}A_{i+1}C_{i+1} & (0\leq i\leq n-1)\\
& A_n \to b_i \mid e & (1\leq i\leq n)\\
& B_i \to b_i c_i \mid e & (1\leq i\leq n)\\
& C_i \to c_i \mid e & (1\leq i\leq n)
\end{array}
$$

Grammar G_n is of size $|G_n| = 18n$. In addition:

Lemma 2. *For each $k \geq 2$, G_n is an LL(k) grammar.* ■

The proof of the lemma is left to the reader. Also note that G_n is not LL(1) or LR(1) or strong LL(2) and that language $L(G_n)$ is finite.

We will show that the size of *every* left parser for G_n must be at least an exponential function of $|G_n|$. This will be done by proving the stronger result that every right parser for G_n must be at least exponentially larger than G_n, and by noting the following lemma:

Lemma 3. *If G_n has a left parser of size t then it has a right parser of size $< 2t$.*

Proof. The Lemma is a consequence of the fact that G_n left-to-right covers itself, that is, there is a homomorphism h between production sequences of G_n such that the right parse of a string in $L(G_n)$ is a homomorphic image of the left parse of the same string. Therefore, to get a right parser we must only augment a left parser for G_n with evaluation of h. A suitable cover homomorphism h on the productions of G_n is

$$
\begin{aligned}
&h(A_n \to b_i) = (A_n \to b_i),\\
&h(A_n \to e) = (A_n \to e),\\
&h(B_i \to b_i c_i) = (B_i \to b_i c_i,\ A_{i-1} \to a_i A_i B_i),\\
&h(B_i \to e) = (B_i \to e,\ A_{i-1} \to a_i A_i B_i),\\
&h(C_i \to c_i) = (C_i \to c_i,\ A_{i-1} \to d_i A_i C_i),\\
&h(C_i \to e) = (C_i \to e,\ A_{i-1} \to d_i A_i C_i),
\end{aligned}
$$

where $i = 1,2,\ldots,n$. For the remaining productions the value of h equals e.

Let DPDT T be a left parser for G_n. Modify T as follows: Whenever $\delta(q,a,Z) = (q',\alpha,\gamma)$ in T, replace γ by $h(\gamma)$. The resulting DPDT T' is a right parser for G_n. Since always $lg(h(\gamma)) \leq 2 \cdot lg(\gamma)$, it finally follows that $|T'| < 2 \cdot |T|$. ■

Theorem 4. *There exists a constant $c > 0$ such that, when $n > 10$, any moderate right parser for G_n has size at least 2^{cm} where $m = |G_n|$.*

Proof. The (lengthy) proof is based on the following simple idea. We consider parsing strings of the form $x_1x_2 \cdots x_n b_i c_i$ in $L(G_n)$ where x_i equals a_i or d_i. The right parse of such a string begins with $A_n \to b_i$ or $A_n \to e$ depending on whether $x_i = d_i$ or $x_i = a_i$. If the current state q and the topmost stack symbol Z after reading x_n can always tell whether $x_i = d_i$ or $x_i = a_i$, then the number of different pairs (q,Z), called the modes of the parser, is immediately seen to be exponentially large in $|G_n|$, and the Theorem follows. Otherwise the parser must exponentially often consult the stack below the topmost element. But then it turns out that the information popped from the stack cannot be ignored because it is needed later in parsing. To save this information the number of modes should be exponentially large.

We now formalize the above argument. Let $T = (Q,\Sigma_n \cup \{\$\},\Gamma,P_n,\delta,q_0,Z_0,F)$ be a moderate right parser for G_n. Each pair (q,Z) in $Q \times \Gamma$ is called a *mode* of T. The mode of a configuration $(q,x,Z\alpha,u)$ is (q,Z) where Z is the topmost stack symbol. Without loss of generality we assume that each element of Q and Γ occurs at least once in the description of δ. Hence $\#Q \leq |T|$ and $\#\Gamma \leq |T|$. Therefore the number of different modes satisfies

$$\#(Q \times \Gamma) \leq |T|^2. \tag{1}$$

We will show that $\#(Q \times \Gamma) > 2^{n/10}$ when $n > 10$. This proves the Theorem since then $|T| > 2^{n/20}$ from (1), which means, noting that $|G_n| = m$ equals $18n$, that $|T| > 2^{m/360}$.

So we claim that $\#(Q \times \Gamma) > 2^{n/10}$. To derive a contradiction, assume

$$\#(Q \times \Gamma) \leq 2^{n/10}. \tag{2}$$

Denote by $C_{1/3}$ the set of those (state, stack content)-pairs that T reaches after reading a prefix of length $\lfloor n/3 \rfloor$ of some string in $L(G_n)$; note that the output of T must be empty at this moment. More formally, let $L_{1/3} = \{x = x_1x_2 \cdots x_{\lfloor n/3 \rfloor} \mid x_i = a_i \text{ or } d_i\}$. Then

$$C_{1/3} = \{(q,\alpha) \mid (q_0,x,e,e) \vdash^* (q,e,\alpha,e) \text{ where } x \in L_{1/3}\}.$$

Clearly, for different x the corresponding elements (q,α) of $C_{1/3}$ must be different because then the languages accepted starting from state q and stack content α must differ. Hence

$$\#C_{1/3} = 2^{\lfloor n/3 \rfloor}. \tag{3}$$

Let $c = (q,\alpha)$ be a fixed element of $C_{1/3}$ and let $L_{2/3} = \{y = y_{\lfloor n/3 \rfloor+1} \cdots y_n \mid y_i = a_i \text{ or } d_i\}$. Consider move sequences

$$(q,y,\alpha,e) \vdash \cdots \vdash (q',e,\beta,e) \tag{4}$$

where y is in $L_{2/3}$. Again note that the output must still remain empty in (4). Let now $(r,v,Z\gamma,e)$ be the configuration in sequence (4) whose stack $Z\gamma$ is of the lowest height; if there are more than one such configuration in (4), we let $(r,v,Z\gamma,e)$ be the first of them. The mode of this configuration, (r,Z), is called the *bottom* of (4) and denoted as *bottom* (q,y,α). Then sequence (4) can uniquely be written as

$$(q,y'v,\alpha,e) \vdash \cdots \vdash (r,v,Z\gamma,e) \vdash \cdots \vdash (q',e,\beta,e).$$

Here the sequence starting from $(r,v,Z\gamma,e)$ is uniquely determined by r, v and Z, since the height of the stack remains at least as high as $lg(Z\gamma)$. Hence we have $\beta = \beta'\gamma$ for some β'. We say that string y' is the prefix of y that *leads* to the bottom. Denote then

$$L_{(r,Z)} = \{\, y' \mid \text{for some } y \in L_{2/3},\ \textit{bottom}(q,y,\alpha) = (r,Z) \text{ and } y' \text{ is the prefix of } y \text{ that leads to the bottom}\}.$$

Lemma 5. $\#L_{(r,Z)} < 2^{n/10}$.

Proof. To derive a contradiction, assume that $\#L_{(r,Z)} \geq 2^{n/10}$. Given y' in $L_{(r,Z)}$ we may write for some v such that $y'v \in L_{2/3}$ and for some $x \in L_{1/3}$

$$\begin{aligned}(q_0,xy'v,e,e) &\vdash \cdots \vdash (q,y'v,\alpha,e)\\ &\vdash \cdots \vdash (r,v,Z\gamma,e)\\ &\vdash \cdots \vdash (q',e,\beta,e).\end{aligned} \tag{5}$$

Here y' is the prefix of $y'v$ that leads to the bottom. Since $lg(\alpha) \geq lg(Z\gamma)$, there must be in (5) before $(q,y'v,\alpha,e)$ a last configuration

$$(s,w,Y\delta,e) \tag{6}$$

such that $lg(Y\delta) = lg(Z\gamma)$. Here the assumption that T is moderate is needed because it implies that every stack height $\leq lg(\alpha)$ must occur in (5). If $\#L_{(r,Z)} \geq 2^{n/10}$ and if we recall (2), we realize that then there must be two different elements y' and y'' of $L_{(r,Z)}$ such that the corresponding modes (s,Y) of (6) are the same for y' and y''. Hence for some suffixes x', x'' of the string x occurring in (5) we get

$$(s,x'y',Y\delta') \vdash \cdots \vdash (q,y',\alpha,e) \vdash \cdots \vdash (r,e,Z\delta',e) \tag{7}$$

and

$$(s,x''y'',Y\delta'') \vdash \cdots \vdash (q,y'',\alpha,e) \vdash \cdots \vdash (r,e,Z\delta'',e) \tag{8}$$

and the height of the stack in (7) is always $\geq lg(Y\delta') = lg(Z\delta')$ and in (8) always $\geq lg(Y\delta'') = lg(Z\delta'')$. Hence we may replace $x'y'$ in (7) by $x''y''$ and still obtain the same final configuration $(r,e,Z\delta',e)$. But this is a contradiction, since $y' \neq y''$ and therefore a

replacement of $x'y'$ by $x''y''$ in (7) should also change the final configuration of (7) because the languages accepted as well as the parses omitted should change. But, as we noted, this did not occur, which completes the proof of Lemma 5.

The following lemma considers sets $L'_{(r,Z)}$ given by

$$L'_{(r,Z)} = \{ v \mid \text{for some } y',\ y = y'v \text{ is in } L_{2/3},\ \textit{bottom}(q,y,\alpha) = (r,Z) \text{ and } y' \text{ is the prefix of } y \text{ that leads to the bottom}\}.$$

Lemma 6. *If $n > 10$, there exists a mode (r,Z) such that $L'_{(r,Z)}$ contains $> 2^{n/10}$ elements having the same length.*

Proof. We may represent $L_{2/3}$ as

$$L_{2/3} = \bigcup_{(r,Z)} L''_{(r,Z)}$$

where $L''_{(r,Z)} = \{uv \mid u \in L_{(r,Z)},\ v \in L'_{(r,Z)},\ lg(uv) = \lceil 2n/3 \rceil\}$ is a subset of $L_{(r,Z)}L'_{(r,Z)}$. Noting Lemma 5 we therefore obtain

$$\#L_{2/3} \leq \sum_{(r,Z)} \#L''_{(r,Z)} < 2^{n/10} \cdot \sum_{(r,Z)} \#L'_{(r,Z)}.$$

If the Lemma were not true, then we should have $\#L'_{(r,Z)} \leq \lceil 2n/3 \rceil \cdot 2^{n/10}$ for all (r,Z), since the length of all elements in $L'_{(r,Z)}$ is $\leq \lceil 2n/3 \rceil$. But this leads to a contradiction since now, if $n > 10$,

$$\begin{aligned} \#L_{2/3} &< 2^{2n/10} \cdot \lceil 2n/3 \rceil \cdot 2^{n/10} = 2^{3n/10 + \log_2 \lceil 2n/3 \rceil} \\ &< 2^{\lceil 2n/3 \rceil} \end{aligned}$$

which by the definition of $L_{2/3}$ cannot be true. This proves Lemma 6.

All the above considerations are with respect to a fixed element $c = (q,\alpha)$ of $C_{1/3}$. Thus we may conclude from Lemma 6 that for every such $c = (q,\alpha)$, there exists a mode (r_c,Z_c) such that $L'_{(r_c,Z_c)}$ contains $> 2^{n/10}$ elements of the same length. That is, there is a string y' and more than $2^{n/10}$ disjoint strings v of equal length such that $y = y'v$ is in $L_{2/3}$ and y' is the prefix of y that leads to $\textit{bottom}(q,y,\alpha) = (r_c,Z_c)$. Denote the common length of strings v by l_c. Observe that the moves reading v do not depend on the stack content below Z_c.

Thus we have chosen for each c in $C_{1/3}$ a triple (r_c,Z_c,l_c). Since the number of different triples is at most $2^{n/10} \cdot \lceil 2n/3 \rceil < 2^{\lfloor n/3 \rfloor}$, we must have $(r_c,Z_c,l_c) = (r_{c'},Z_{c'},l_{c'})$ for some disjoint $c = (q,\alpha)$, $c' = (q',\alpha')$ in $C_{1/3}$.

By the construction, then, there are strings x, x' in $L_{1/3}$, $x \neq x'$, and strings y, y' such that, for more than $2^{n/10}$ distinct v, strings yv and $y'v$ are in $L_{2/3}$ and

$$(q_0,xyv,e,e) \vdash^* (r_c,v,Z_c\alpha,e) \vdash^* (s,e,\beta\alpha,e), \tag{9}$$

$$(q_0,x'y'v,e,e) \vdash^* (r_c,v,Z_c\alpha',e) \vdash^* (s,e,\beta\alpha',e). \tag{10}$$

Since $x = x_1 \cdots x_{\lfloor n/3 \rfloor}$ and $x' = x_1' \cdots x_{\lfloor n/3 \rfloor}'$ are different, x_i and x_i' must differ for some i. Then we may assume, by symmetry, that $x_i = a_i$ and $x_i' = d_i$.

Suppose that the input to be read after (9) or (10) is $b_i c_i$. Then the first output following (9) should be production $A_n \to e$ and the first output following (10) should be production $A_n \to b_i$. But because the last configurations of (9) and (10) can differ only in stack portions α and α', the parser must always pop β from the stack to be able to choose between $A_n \to e$ and $A_n \to b_i$. However, s or β must vary with v because the output following $A_n \to e$ or $A_n \to b_i$ varies with v. Thus after popping β the mode of T must be different for each v. This is not possible because there are $> 2^{n/10}$ different strings v but by (2), only $\leq 2^{n/10}$ different modes. We have a contradiction which completes the proof of Theorem 4. ■

The restriction to moderate right parsers can be removed from Theorem 4.

Corollary 7. *There is a constant $c' > 0$ and an integer n_0 such that, when $n > n_0$, any right parser for G_n has size at least $2^{c'm}$ where $m = |G_n|$.*

Proof. If T is a right parser for G_n then, by Lemma 1, we can find a moderate right parser T' for G_n such that $|T| > |T'|/5$. Then, by Theorem 4, $|T| > 2^{cm}/5 = 2^{\log_2(1/5)+cm} \geq 2^{c'm}$ for some $0 < c' < \log_2(1/5)/m + c$. Such a constant c' clearly exists when m, that is n, is large enough. ■

The main result of this section follows in a similar way from Corollary 7 and Lemma 3:

Theorem 8. *There is a constant $c'' > 0$ and an integer n_0' such that when $n > n_0'$, any left parser for G_n has size at least $2^{c''m}$ where $m = |G_n|$.* ■

A usual way of constructing left parsers is the LL(k) method (e.g. [1]). It is well-known that for the strong LL(k) grammars this method yields parsers with size polynomial in the size $|G|$ of the grammar: The size of a strong LL(k) parser is of the order $|G|^{k+1}$. Since every LL(1) grammar is strong LL(1), LL(1) grammars always have left parsers of polynomial size. On the other hand, Lemma 2 and Theorem 8 imply:

Theorem 9. *For each $k \geq 2$, there is an infinite family of LL(k) grammars such that the size of every left parser for these grammars increases at least exponentially in the size of the grammar.* ■

In addition to the strong LL(k) grammars we have another subclass of the LL(k) grammars having left parsers of polynomial size. This result is a consequence of a remark in [8, p. 230].

Theorem 10. *If an LL(k) grammar G has no e-productions (productions of the form $A \to e$), then G has left parser with size polynomially bounded in $|G|$.* ■

4. Right parsers

Lemma 2 implies that the grammars G_n of the previous section are LR(2). Then we immediately obtain from Corollary 7 the following counterpart of Theorem 9 for right parsers.

Theorem 11. *For each $k \geq 2$, there is an infinite family of LR(k) grammars such that the size of every right parser for these grammars increases at least exponentially in the size of the grammar.* ■

Grammars G_n are not LR(0) or LR(1). To complement our study of parser size in the LR(0) and LR(1) cases we use a sequence of grammars $Q_n = (N_n, \Sigma_n, P_n, S)$ where

$$\begin{array}{lll} P_n: & S \to A_i & (1 \leq i \leq n) \\ & A_i \to a_j A_i & (1 \leq i \neq j \leq n) \\ & A_i \to a_i B_i \mid b_i & (1 \leq i \leq n) \\ & B_i \to a_j B_i \mid b_i & (1 \leq i, j \leq n) \end{array}$$

The size $|Q_n| = m$ equals $6n^2 + 5n$. Earley [2] established grammars Q_n as an example where the size of the standard LR(0) parser grows exponentially with n. An LR(0) parser for Q_n has 2^{cn} states for some $c > 0$. The size of every LR(k) as well as SLR(k) or LALR(k) parser for Q_n is thus $\geq 2^{c'\sqrt{m}}$ for some $c' > 0$.

We will show that the size of *any* right parser for Q_n must be at least of the order $2^{\sqrt{m}}$. First a result analogous to Theorem 4 is given.

Theorem 12. *There is a constant $c > 0$ and an integer n_0 such that when $n > n_0$, any moderate right parser for Q_n has size at least $2^{c\sqrt{m}}$ where $m = |Q_n|$.*

Proof. The idea of the proof resembles to that of the proof of Theorem 4. We restrict ourselves to considering strings of the form $x_1 x_2 \cdots x_{n-1} a_n^j b_i$ in $L(Q_n)$, where each $x_1, \cdots, x_{n-1}$ is in $\{a_1, \cdots, a_{n-1}\}$, j is ≥ 0, and $1 \leq i \leq n-1$. The right parse of such a string begins with $B_i \to b_i$ or $A_i \to b_i$ depending on whether or not a_i occurs in $x_1 \cdots x_{n-1}$. Any parser cannot emit this first parse element earlier than when reading b_i. If after reading the last a_n the current mode of a parser can always tell whether or not $x_1 \cdots x_{n-1}$ contains a_i for any $1 \leq i \leq n-1$, the number of modes must be at least of the order $2^{\sqrt{m}}$ and the Theorem follows. Otherwise the parser must sometimes consult the stack below the topmost element. Then the parser should pop the stack portion corresponding to a_n^j. However, the popped information, which depends in j, cannot be ignored since it is needed later in parsing. Hence the number of modes is again at least of the order $2^{\sqrt{m}}$.

To formalize the above argument, let $T = (Q, \Sigma_n \cup \{\$\}, \Gamma, P_n, q_0, Z_0, F)$ be a moderate right parser for Q_n. It suffices to show (c.f. the proof of Theorem 4) that $\#(Q \times \Gamma) > 2^{n/2}$ when $n > 1$. To derive a contradiction, assume

$$\#(Q \times \Gamma) \leq 2^{n/2}. \tag{11}$$

Denote by X a maximal subset of $\{x = x_1 x_2 \cdots x_{n-1} \mid x_i \in \{a_1, \cdots, a_{n-1}\}, 1 \le i \le n-1\}$ such that if x and x', $x \neq x'$, are in X then some symbol a_i occurs in x or x' but not both in x and x'.

First fix a string $x \in X$. Then $x a_n^j b_i$ is in $L(Q_n)$ for each $j \ge 0$, $1 \le i \le n$. Hence T has a move sequence $(q_0, x, e, e) \vdash^* (q, e, \alpha, e)$, and a subsequent sequence

$$(q, a_n^j, \alpha, e) \vdash \cdots \vdash (q_j, e, \beta_j, e) \tag{12}$$

for each $j \ge 0$. The *bottom* of (12) is defined as in the proof of Theorem 4. Then, since a_n^j is a prefix of $a_n^{j'}$, $j < j'$, there is j_0 such that for all $j > j_0$, the bottom of (12) is the same and is achieved after reading $a_n^{j_0}$ but before reading $a_n^{j_0+1}$. Denote this particular bottom, which is unique for each x, by (r_x, Z_x).

Now, since $\#X > 2^{n/2}$, set X must contain distinct strings x and x' such that $(r_x, Z_x) = (r_{x'}, Z_{x'})$. Then, by the construction, there are for each $k \ge 0$ move sequences

$$(q_0, x a_n^{j_0} a^k, e, e) \vdash^* (r_x, a^k, Z_x \alpha, e) \vdash^* (s, e, \beta\alpha, e) \tag{13}$$

$$(q_0, x' a_n^{j_0'} a^k, e, e) \vdash^* (r_x, a^k, Z_x \alpha', e) \vdash^* (s, e, \beta\alpha', e) \tag{14}$$

Suppose that symbol a_i occurs in only one of strings x, x', say in string x, and assume that the input to be read after (13) and (14) is b_i.

Then the first output following (13) should be production $B_i \to b_i$ and the first output following (14) should be production $A_i \to b_i$. But because the last configurations of (13) and (14) can differ only in stack portions α and α', the parser must always pop β from the stack to be able to to choose between $B_i \to b_i$ and $A_i \to b_i$. However, s or β must vary with k because the output following $B_i \to b_i$ or $A_i \to b_i$ varies with k. Thus after popping β the mode of T must be different for each $k = 0, 1, \cdots$. This is not possible because by (11), there are only $\le 2^{n/2}$ different modes. This contradiction proves the Theorem. ∎

Using Lemma 1 we again get:

Corollary 13. *There is a constant $c' > 0$ and an integer n_0' such that when $n > n_0'$, any right parser for Q_n has size at least $2^{c'\sqrt{m}}$ where $m = |Q_n|$.* ∎

Since each Q_n is an LR(0) grammar, the next theorem follows.

Theorem 14. *For each $k \ge 0$, there is an infinite family of LR(k) grammars (as well as SLR(k) and LALR(k) grammars) such that the size of every right parser for these grammars is larger than polynomial in the size of the grammar.* ∎

Finally note that the above results remain true if productions $S \to A_i$ of Q_n are replaced by productions $S \to aA_i$ where a is a new nonterminal. The new grammars, which are right regular and right parsable, therefore do not have right parsers of polynomial size.

5. Further research

We have shown, using certain example families of LL(k) and LR(k) grammars, that no matter what parsing method is used, the size of a right or a left parser cannot always be polynomially bounded in the size of the grammar. It is obvious that the grammatical structures essential for our examples unlikely occur in context-free grammars for real programming languages. To establish this formally it would be interesting to find precise grammatical characterizations of those structures that need parsers of non-polynomial size.

Another problem for further research is to give strict worst case lower bounds for parser size. We have only shown that the bound for left parsers should be at least $O(2^m)$ when the grammars are left parsable with lookahead k = 2. For right parsers we have shown that the bound should be at least $O(2^m)$ when the grammars are right parsable with lookahead k = 2 ,and $O(2^{\sqrt{m}})$ when the grammars are right parsable with lookahead k = 0. We leave open the question whether or not these actually are strict lower bounds and how the bounds depend on the length of the lookahead.

Acknowledgment. The author is indebted to Seppo Sippu for pointing out the problem considered in this work. Comments by the referees were helpful in clarifying the presentation. The work was supported by the Academy of Finland and the Finnish Cultural Foundation.

References

[1] Aho,A.V. and J.D.Ullman: *The Theory of Parsing, Translation, and Compiling. Vol. I: Parsing*. Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[2] Earley, J.: An efficient context-free parsing algorithm. Ph.D. Thesis, Carnegie-Mellon Univ., Pittsburgh, 1968.

[3] Geller,M.M., S.L.Graham and M.A.Harrison: Production prefix parsing (extended abstract). Automata, Languages, and Programming (J.Loeckx, ed.), Lecture Notes in Computer Science 14, pp. 232-241, Springer-Verlag, 1974.

[4] Geller,M.M., H.B.Hunt, III, T.G.Szymanski and J.D.Ullman: Economy of description by parsers, DPDA's and PDA's. Theoretical Computer Science 4 (1977), 143-153.

[5] Harrison,M.A.: *Introduction to Formal Language Theory*. Addison-Wesley, Reading, Mass., 1978.

[6] Harrison,M.A. and I.M.Havel: On the parsing of deterministic languages. Journal of the ACM 21 (1974), 525-548.

[7] Knuth,D.E.: On the translation of languages from left to right. Information and Control 8 (1965), 607-639.

[8] Rosenkrantz,D.J. and R.E.Stearns: Properties of deterministic top-down grammars. Information and Control 17 (1970), 226-256.