Size/lookahead tradeoff for LL(k)-grammars

E. Bertsch [a] and M.-J. Nederhof [b,1]

[a] *Ruhr University, Faculty of Mathematics, Universitätsstraße 150, D-44780 Bochum, Germany*

[b] *University of Groningen, Faculty of Arts, P.O. Box 716, NL-9700 AS Groningen, The Netherlands*

Abstract

For a family of languages a precise tradeoff relationship between the size of LL(k) grammars and the length k of lookahead is demonstrated.

Keywords: Formal languages, parsing theory.

1 Introduction

This paper provides a solution to an open problem posed in [1]. One of the main results of that paper was that for certain LR(k) languages a linear decrease of lookahead length must be paid for by an exponential increase of grammar size. On a very high level of discussion, this may be seen as an invariance result for overall algorithmic complexity because lookaheads of k symbols are assumed to require parsing tables growing exponentially with k [2].

In the final section of [1] the corresponding problem with LL(k) instead of LR(k) grammars is formulated as a challenge for further studies of similar languages. The present article contains a comprehensive solution to that problem. The general structure of the argument displays some similarities to the proof strategy in [1]. Due to the inherent differences between LL(k) and LR(k) parsing our reasoning is substantially new, however. In fact, no case distinctions even remotely resembling those in the proof of the final theorem in [1] are needed here.

[1] Supported by the Royal Netherlands Academy of Arts and Sciences. His secondary affiliation is the German Research Center for Artificial Intelligence (DFKI).

2 Preliminaries

We assume the reader is familiar with LL(k) parsing. For thorough treatment we refer to [3,2].

For a given context-free grammar, let $\rightarrow^*$ denote the *derives*-relation (using zero or more nonterminal expansions), and let $\rightarrow_l^*$ denote its sub-relation for left-most derivations.

The size of a production of a context-free grammar is defined to be 1 plus the number of symbols in the right-hand side. The size $|G|$ of a grammar G is defined to be the sum of the sizes of all productions.

3 Upper bounds

Given a natural number $n \geq 1$, we define the language $L_n \subseteq \{0,1\}^*$ as:

$$L_n = \{a_1 \dots a_n a_n \dots a_1 \mid a_1, \dots, a_n \in \{0,1\}\} \cup \\ \{a_1 \dots a_{2n} a_{2n} \dots a_1 \mid a_1, \dots, a_{2n} \in \{0,1\}\}$$

A language L_n thus contains all palindromes over $\{0,1\}$ that are of length $2n$ or of length $4n$. Informally, the difficulty of obtaining LL(k) grammars for such a language consists in allowing a provision in the parser for deterministically handling the input positions from $n+1$ to $2n-k+1$. The string of symbols beginning at position $n+1$ may be either the reverse of the string up to position n, or it may be the reverse of some string yet to be seen, preceding position $3n+1$, and the parser must allow for both possibilities. This uncertainty is resolved if the input is found not to be a $2n$ palindrome because of a mismatch between two individual symbols at either side of positions n and $n+1$, or at the latest after reading the symbol at position $2n-k+1$, since then the parser may look ahead far enough to see whether the string is too long to be a $2n$ palindrome.

Below we demonstrate that we may construct LL(k) and strong LL(k) grammars for the language L_n in such a way that the choice of a larger k corresponds to a smaller grammar size.

Theorem 1 *For $1 \leq k \leq n$, there exists a (strong) LL(k) grammar $G_{n,k}$ generating L_n with the number of productions being $2^{n-k} \cdot (6n - 6k + 20) + 2n + 2k - 3$ and the longest production having length 4.*

Proof. Let $G_{n,k}$ be defined as the grammar with start symbol A_0 and nonterminals A_i, for $0 \leq i \leq k-1$, B_i^x, for $0 \leq i \leq n-k+1$ and $x \in \{0,1\}^i$, C_i^{x,xyy^R}, for $0 \leq i \leq n-k+1$ and $x \in \{0,1\}^{n-k+1-i}$ and $y \in \{0,1\}^i$, D_i, for $1 \leq i \leq n$, and E^y, for y a prefix of a string of the form xx^R, where $x \in \{0,1\}^{n-k+1}$, and all productions of the following types that can be formed by using the nonterminals just introduced:

1. $A_i \rightarrow a\, A_{i+1}\, a$, for $0 \leq i \leq k-2$ and $a \in \{0,1\}$,
2. $A_{k-1} \rightarrow B_0^\epsilon$,
3. $B_i^x \rightarrow a\, B_{i+1}^{xa}$, for $0 \leq i \leq n-k$ and $a \in \{0,1\}$,
4. $B_{n-k+1}^x \rightarrow C_0^{x,x}$,
5. $C_i^{xa,y} \rightarrow a\, C_{i+1}^{x,ya}$, for $0 \leq i \leq n-k$ and $a \in \{0,1\}$,
6. $C_i^{xa,y} \rightarrow b\, D_{i+1}\, b\, E^y$, for $0 \leq i \leq n-k$ and $a, b \in \{0,1\}$ such that $a \neq b$,
7. $C_{n-k+1}^{\epsilon,y} \rightarrow \epsilon$,
8. $C_{n-k+1}^{\epsilon,y} \rightarrow D_{n-k+1}\, E^y$,
9. $D_i \rightarrow a\, D_{i+1}\, a$, for $1 \leq i \leq n-1$ and $a \in \{0,1\}$,
10. $D_n \rightarrow \epsilon$,
11. $E^{ya} \rightarrow a\, E^y$,
12. $E^\epsilon \rightarrow \epsilon$.

The intuition behind these grammars can best be understood by considering the behaviour of a top-down parser. Consider input of the form $a_1 \cdots a_{2n}$ or $a_1 \cdots a_{4n}$. While reading the input from a_1 to a_{k-1}, using nonterminals A_i, the parser pushes the symbols it reads, for future matching at the opposite side of a $2n$ or $4n$ palindrome. From a_k to a_n, the nonterminals B_i^x encode the symbols that are read into the nonterminal name. Starting from a_{n+1}, using the nonterminals $C_i^{x,y}$, the parser at the same time treats the string as a possible $2n$ palindrome, popping symbols from the stack encoded in x, and as a possible $4n$ palindrome, pushing symbols on the stack encoded in y. This ends after a_{2n-k+1} has been read (7th or 8th clause above), or when, before reaching a_{2n-k+1}, a mismatch occurs that excludes the possibility of a $2n$ palindrome (6th clause).

If a_{2n-k+2} is reached without any mismatch, the parser may thereupon expand $C_{n-k+1}^{\epsilon,y}$ according to the 7th clause, which may lead to recognition of a $2n$ palindrome: the $k-1$ symbols that were pushed due to nonterminals A_i are matched in reverse to the next $k-1$ symbols, which should then also be the last symbols in the input. If however the parser expands $C_{n-k+1}^{\epsilon,y}$ according to the 8th clause, this may lead to recognition of a $4n$ palindrome.

By the productions from the 6th or 8th clause, the nonterminals D_i are introduced, which lead to recognition of a nested palindrome centered around $a_{2n}a_{2n+1}$, and then the string that was stacked by means of nonterminals B_i^x and $C_i^{x,y}$ is read in reverse by means of the nonterminals E^y. Finally, the $k-1$ symbols that were pushed due to nonterminals A_i are matched in reverse to the final $k-1$ symbols of the $4n$ palindrome.

A grammar of the above form is LL(k): for all nonterminals, with the exception of $C^{\epsilon,y}_{n-k+1}$, expansion with at most one production is consistent with the next symbol of the terminal string to be derived. In the case of $C^{\epsilon,y}_{n-k+1}$, any derivation of the form $A_0 \rightarrow^*_l vC^{\epsilon,y}_{n-k+1}\alpha$ is such that $\alpha \in \{0,1\}^{k-1}$, as can be easily verified. If the production from clause 7 is chosen, exactly $k-1$ symbols remain until the end of the string. If the production from clause 8 is chosen, at least k symbols remain. Since the potential end of the input after $k-1$ symbols can be detected within the window of k symbols of lookahead, a deterministic choice can be made.

The number of productions represented by the 12 clauses is respectively: $2 \cdot (k-1)$, 1, 2^{n-k+2}, 2^{n-k+1}, $2^{n-k+1} \cdot (n-k+1)$, $2^{n-k+1} \cdot (n-k+1)$, 2^{n-k+1}, 2^{n-k+1}, $2n-2$, 1, $2^{n-k+1} \cdot (n-k+3) - 2$, 1, the sum of which is $2^{n-k} \cdot (6n - 6k + 20) + 2n + 2k - 3$. □

4 Lower bounds

In this section we determine a lower bound on the size of LL(k) and strong LL(k) grammars that generate the languages L_n.

We will need the following lemma, which formalizes the intuition that a top-down parser with k symbols of lookahead will not be influenced in its actions by input that lies ahead of the reach of its lookahead; given two distinct strings, a stack that is obtained for one will be identical to a stack obtained for the other, until the difference between the two strings can be detected by the lookahead.

Lemma 2 *Assume we have an alphabet Σ, a number $k \geq 1$, a (strong) LL(k) grammar over the alphabet that generates a language L, and a pair of strings of the form $xyz, xyz' \in L$, such that $x \neq \epsilon$ and $y \in \Sigma^{k-1}$. There is a unique string of grammar symbols α such that for some $u, u', A, A', \beta, \beta'$:*

$$S \rightarrow^*_l uA\beta \rightarrow_l x\alpha \rightarrow^* xyz \;\wedge\; |u| < |x|$$

$$S \rightarrow^*_l u'A'\beta' \rightarrow_l x\alpha \rightarrow^* xyz' \;\wedge\; |u'| < |x|$$

Proof. We know that (strong) LL(k) grammars are unambiguous, and therefore each string in the language has exactly one left-most derivation. In the left-most derivations for xyz and xyz', consider the last expansion of a production before the last symbol of x becomes part of the longest prefix of the sentential form that consists only of terminals. We have:

$$S \to_l^* uA\beta \to_l x\alpha \to^* xyz \ \wedge \ |u| < |x|$$
$$S \to_l^* u'A'\beta' \to_l x\alpha' \to^* xyz' \ \wedge \ |u'| < |x|$$

By induction on the length of the derivations, and making use of the assumption that the grammar is (strong) LL(k), we can show that the derivations are identical up to the point where the last symbol of x becomes part of the longest prefix of the sentential form that consists only of terminals, which implies that $u = u'$, $A = A'$, $\beta = \beta'$, and $\alpha = \alpha'$. □

Theorem 3 *For $1 \leq k \leq n$, any (strong) LL(k) grammar that generates L_n has at least 2^{n-k+1} nonterminals.*

Proof. For given k and n, assume we have a LL(k) or strong LL(k) grammar G that generates L_n. Let S be the start symbol.

Choose a string $v \in \{0,1\}^{n-k+1}$, and consider the $2n$ palindrome $0^{k-1}vv^R0^{k-1}$ and the $4n$ palindrome $0^{k-1}vv^R0^{k-1}0^{k-1}vv^R0^{k-1}$. Given these two strings, Lemma 2 allows us to choose a string of grammar symbols α in a unique way; x as in the lemma is chosen to be $0^{k-1}vv^R$ and y is chosen to be 0^{k-1}. For this α we have:

$$S \to^* 0^{k-1}vv^R\alpha \to^* 0^{k-1}vv^R0^{k-1}$$
$$S \to^* 0^{k-1}vv^R\alpha \to^* 0^{k-1}vv^R0^{k-1}0^{k-1}vv^R0^{k-1}$$

This implies that $\alpha \to^* 0^{k-1}$ and $\alpha \to^* 0^{k-1}0^{k-1}vv^R0^{k-1}$, and therefore α must contain a nonterminal A that derives terminal strings of two different lengths l_1 and l_2; assume without loss of generality that $l_1 < l_2$. If A could also derive a terminal string of a third length, distinct from l_1 and l_2, then the grammar would generate a terminal string of a length different from $2n$ and $4n$, which is in contradiction with the assumption that the grammar generates L_n. Similarly, if α were to contain another occurrence of a nonterminal, call it B, that also derives terminal strings of different lengths, say l_3 and l_4, where $l_3 \neq l_4$, then α could derive terminal strings of all lengths from $\{l+l_1+l_3, l+l_2+l_3, l+l_1+l_4, l+l_2+l_4\}$, where l is the length of a terminal string derived from the string β, which is constructed from α by omitting A and B. Since this set of lengths must contain at least 3 elements, this again contradicts the assumption that G generates L_n.

Thus, A is uniquely determined in α, and must solely account for the difference in length between $2n$ and $4n$ palindromes, which means that l_2 must be at least $2n$, and in $\alpha \to^* 0^{k-1}0^{k-1}vv^R0^{k-1}$ A must derive a substring of

$0^{k-1}0^{k-1}vv^R0^{k-1}$ that covers at least $0^{k-1}vv^R$, and possibly additional occurrences of the symbol 0 on either side. Let us rename A to A_v, motivated by the fact that A was uniquely determined by v.

The above argument can be repeated for a string $w \in \{0,1\}^{n-k+1}$ distinct from v, which allows us to determine a nonterminal A_w in a unique way. For some α' and some numbers $p_v, p_w, q_v, q_w \leq k-1$ we now have:

$$S \rightarrow^* 0^{k-1}vv^R\alpha \rightarrow^* 0^{k-1}vv^R0^{p_v}A_v0^{q_v} \rightarrow^* 0^{k-1}vv^R0^{k-1}0^{k-1}vv^R0^{k-1}$$

$$S \rightarrow^* 0^{k-1}ww^R\alpha' \rightarrow^* 0^{k-1}ww^R0^{p_w}A_w0^{q_w} \rightarrow^* 0^{k-1}ww^R0^{k-1}0^{k-1}ww^R0^{k-1}$$

Assume that A_v and A_w are identical. A third string can now be derived:

$$S \rightarrow^* 0^{k-1}vv^R0^{p_v}A_v0^{q_v} \rightarrow^* 0^{k-1}vv^R0^{p}ww^R0^{q}$$

where $p \geq 2k-2-p_w \geq k-1$. Since this third string has length greater than $2n$ and since it is in L_n, it must have length $4n$. We can therefore write it as $0^{k-1}vv^R0^{k-1}0^{p'}ww^R0^q$, where $p' = p-k+1$, and divide it into two halves $0^{k-1}vv^R0^{k-1}$ and $0^{p'}ww^R0^q$, which must be mirror images of each other since the language contains only palindromes, or in other words, $0^{k-1}vv^R0^{k-1} = 0^{p'}ww^R0^q$.

If $0^{k-1}vv^R0^{k-1} = 0^{p'}ww^R0^q$ consists of only occurrences of 0, then since v and w have the same length, they must be identical, contrary to the assumption. If $0^{k-1}vv^R0^{k-1} = 0^{p'}ww^R0^q$ contains two or more occurrences of 1, then there is a unique centre around which these occurrences are arranged; since v and w have the same length, it follows that v and w must be identical, again contrary to the assumption. Thereby we have contradicted that A_v and A_w are identical.

Thus we have shown that, given two different strings v and w of length $n-k+1$, the nonterminals A_v and A_w are distinct, and therefore the grammar must contain at least as many nonterminals as there are strings in the set $\{0,1\}^{n-k+1}$, viz. 2^{n-k+1}. □

Together with the theorem from the previous section, this leads to an accurate estimate of the size of smallest (strong) LL(k) grammars for L_n:

Corollary 4 *Let c be a positive number. For $n \geq 2$ and $1 \leq k \leq n - c\lg n$, the smallest (strong) LL(k) grammar for L_n has size $2^{\Theta(m)}$, where $m = n-k$.*

Proof. Given that $k \leq n - c\lg n$, we have $\lg n \leq \frac{n-k}{c}$. Furthermore, since c is positive and $n \geq 2$, $k \leq n = 2^{\lg n} \leq 2^{\frac{n-k}{c}}$. Theorem 1 showed that the size of the smallest grammar is at most

$$
\begin{aligned}
&4 \cdot (2^{n-k} \cdot (6n - 6k + 20) + 2n + 2k - 3) \ = \\
&4 \cdot (2^{n-k} \cdot (6 \cdot (n - k) + 20) + 2 \cdot (n - k) + 4k - 3) \ \leq \\
&4 \cdot (2^{n-k} \cdot (6 \cdot (n - k) + 20) + 2 \cdot (n - k) + 4 \cdot 2^{\frac{n-k}{c}}) \ = \\
&\mathcal{O}(2^m) \cdot \mathcal{O}(m) + \mathcal{O}(m) + 2^{\mathcal{O}(m)} \ = \ 2^{\mathcal{O}(m)}
\end{aligned}
$$

Theorem 3 showed that the size of the smallest grammar is $\Omega(2^{n-k+1}) = \Omega(2^{n-k}) = 2^{\Omega(m)}$. □

Note that if we simplify the condition in the corollary by fixing $c = 1$, we restrict the possible combinations of the parameters n and k, but we may then benefit from a more precise expression for the upper bound, which becomes $\mathcal{O}(m \cdot 2^m)$, whereas the lower bound remains $\Omega(2^m)$ as before. This shows that under these more narrow conditions on n and k, the lower and upper bounds are very close.

Our theorems are about the finite languages L_n, but they can be trivially extended to infinite languages such as $(L_n\#)^*$, where # is a new symbol.

Since for a given language the minimal size of LC(k) and PLR(k) grammars [4,2] is polynomially related to the minimal size of LL(k) grammars, the above result of exponential increase in grammar size for decreasing k carries over to these classes of grammar as well.

5 Conclusion

In this paper, we have presented a tradeoff result concerning economy of description of languages using LL(k) grammars when k varies. Our results complement earlier findings of a very similar nature for LR(k) grammars.

Acknowledgments

We thank Hing Leung for correspondence about [1]. One anonymous reviewer made helpful comments which led to some improvements of the presentation.

References

[1] H. Leung, D. Wotschke, On the size of parsers and LR(k)-grammars, Theoretical Computer Science 242 (2000) 59–69.

[2] S. Sippu, E. Soisalon-Soininen, Parsing Theory, Vol. II: LR(k) and LL(k) Parsing, Vol. 20 of EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1990.

[3] S. Sippu, E. Soisalon-Soininen, Parsing Theory, Vol. I: Languages and Parsing, Vol. 15 of EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1988.

[4] D. Rosenkrantz, P. Lewis II, Deterministic left corner parsing, in: IEEE Conference Record of the 11th Annual Symposium on Switching and Automata Theory, 1970, pp. 139–152.