Ch(k) GRAMMARS: A CHARACTERIZATION OF LL(k) LANGUAGES

Anton Nijholt
Department of Mathematics
Vrije Universiteit, Amsterdam
The Netherlands.

Eljas Soisalon-Soininen
Department of Computer Science
University of Helsinki
Finland.

1. INTRODUCTION

From the point of view of parsing the LL(k) grammars constitute a very attractive class of context-free grammars. For each LL(k) grammar a top-down parsing algorithm can be devised which is essentially a one-state deterministic push-down automaton. From a more theoretic point of view LL(k) grammars are attractive as well. It is well-known, for example, that it is decidable whether two LL(k) grammars are equivalent. Also the hierarchy of LL(k) languages with regard to the length k of the look-ahead is a characteristic property.

The class of LL(k) grammars is properly contained in the class of LR(k) grammars, and even the family of LL(k) languages is properly contained in the family of LR(k) languages. If we focus on the "gap" between LL(k) and LR(k) grammars the following points are of interest.

(i) There is the obvious difference in grammar definition.

(ii) The generating capacities are different.

(iii) Apart from the difference between LR(0) and LR(1) languages the length k of the look-ahead does not play a role for LR(k) languages.

(iv) Every LL(k) grammar is both left parsable and right parsable but there are LR(k) grammars which are not left parsable [1].

We consider the present paper as a contribution to the research which tries to clarify the differences between LL(k) and LR(k) grammars. Research in this area has been reported e.g. in Rosenkrantz and Lewis [7], Brosgol [2], Hammer [4], Soisalon-Soininen and Ukkonen [9], Demers [3] and Soisalon-Soininen [8]. In this paper we introduce the class of so called Ch(k) grammars (pronounced "chain k grammars"). This class of grammars is properly contained in the class of LR(k) grammars and it properly contains the LL(k) grammars. However, the family of Ch(k) languages coincides with the family of LL(k) languages. Nevertheless, the parsing properties of Ch(k) grammars are quite different from the parsing properties of LL(k) grammars. The class of Ch(k) grammars can be considered as a generalization of the class of simple chain grammars [6] in the same sense as the class of LL(k) grammars is a generalization of the class of simple LL(1) grammars.

The present paper is organized as follows. In Section 2 we define the necessary

background concerning context-free grammars and parsing. The Ch(k) grammars are defined in Section 3 where also some basic properties of Ch(k) grammars are proved. In Section 4 we demonstrate that the well-known transformation process of left factoring the given grammar will always produce an LL(k) grammar from a Ch(k) grammar and that, in fact, this process cannot produce an LL(k) grammar from a non-Ch(k) grammar. This result implies the equality of the classes of LL(k) and Ch(k) languages, and it also clarifies the relationship of Ch(k) grammars with some other classes of grammars.

2. BACKGROUND

In this section we review various commonly known definitions (cf. [1]) and give some notations. A quadruple $G = (N,\Sigma,P,S)$ is a *context-free grammar* (*grammar* for short) if N and Σ are finite disjoint sets, P is a finite subset of the product $N \times (N \cup \Sigma)^*$ and S is an element of N. Elements of the set N are called *nonterminals* and denoted by capital Latin letters from the beginning of the alphabet $A,B,C,\ldots,S$. Elements of the set Σ are called *terminals* and denoted by small Latin letters from the beginning of the alphabet $a,b,c,\ldots,s$. By X, Y and Z we denote elements which are either in N or in Σ. The elements (A,ω) of P are called *productions* and denoted by $A \to \omega$. The symbol S is called the *start symbol* of the grammar.

Terminal strings, i.e. strings in Σ^*, are denoted by small Latin letters from the end of the alphabet $t,u,v,\ldots,z$, whereas small Greek letters $\alpha,\beta,\gamma,\ldots,\omega$ denote strings in $(N \cup \Sigma)^*$. The *empty string* is denoted by ε. The *derives* relation $\Rightarrow$ of G on the set $(N \cup \Sigma)^*$ is defined by the condition $\alpha A\beta \Rightarrow \alpha\omega\beta$ if α and β are strings in $(N \cup \Sigma)^*$ and $A \to \omega$ is a production in P. If here α is required to be a terminal string, then we get the definition of the *leftmost* derives relation of G, denoted by $\underset{L}{\Rightarrow}$, and if β is required to be a terminal string, then we get the definition of the *rightmost* derives relation of G, denoted by $\underset{R}{\Rightarrow}$.

A sequence $Q_1,Q_2,\ldots,Q_n$ of strings Q_i in $(N \cup \Sigma)^*$ is called a *leftmost derivation* (respectively *rightmost derivation*) of Q_n from Q_1 in the grammar G if $Q_i \underset{L}{\Rightarrow} Q_{i+1}$ (respectively $Q_i \underset{R}{\Rightarrow} Q_{i+1}$) holds in G for each $i = 1,\ldots,n-1$ whenever $n > 1$. The sequence of productions used in a leftmost derivation of a string Q from the start symbol S is a *left parse* of Q in G, and the reverse of the sequence of productions in a rightmost derivation of Q from S is a *right parse* of Q in G.

Let $G = (N,\Sigma,P,S)$ and $G' = (N',\Sigma,P',S')$ be grammars and let $h: P'^* \to P^*$ be a homomorphism. We say that G' *left-to-right covers* G with respect to the homomorphism h, if the following two conditions hold:

(i) if π' is a left parse of a terminal string w in the grammar G', then $h(\pi')$ is a right parse of w in the grammar G;

(ii) if π is a right parse of a terminal string w in the grammar G, then there is a left parse π' of w in G' such that $h(\pi') = \pi$.

If, in these conditions, "left" (resp. "right") is replaced by "right" (resp. "left"), then, under the conditions, G' *right-to-right covers* (resp. *left-to-left covers*) the grammar G with respect to the homomorphism h. The shorthand "right cover" is often used for "right-to-right cover", as well as the shorthand "left cover" is used for "left-to-left cover". If the grammar G' left-to-right covers, right covers or left covers the grammar G, then we say that G' *covers* G. Observe that the *language* $L(G') = \{x \in \Sigma^* \mid S' \Rightarrow^* x\}$ generated by the grammar G' equals the language L(G) if G' covers G.

Let $G = (N,\Sigma,P,S)$ be a grammar and let k be a nonnegative integer. If α is string in $(N \cup \Sigma)^*$ then we denote by $k{:}\alpha$ the first k symbols of α whenever the length $|\alpha|$ of α is greater than or equal to k, and α otherwise. By $FIRST_k(\alpha)$ we denote the set of all strings $k{:}u$ such that u is a terminal string and α derives u. The grammar G is said to be LL(k) if, for a terminal string w, a nonterminal A and strings γ, δ_1 and δ_2 in $(N \cup \Sigma)^*$ such that $A \to \delta_1$ and $A \to \delta_2$ are distinct productions of G, the condition

$$S \underset{L}{\Rightarrow}^* wA\gamma$$

implies that

$$FIRST_k(\delta_1\gamma) \cap FIRST_k(\delta_2\gamma) = \emptyset.$$

The definition of an LL(k) grammar immediately implies some properties of LL(k) grammars. For example, each LL(k) grammar is *unambiguous*, i.e. each terminal string in the language has exactly one left parse. In addition, if an LL(k) grammar is *reduced*, i.e. every production is used in a left (or right) parse of some terminal string, then it is not *left-recursive*, i.e. it has no nonterminal A such that $A \Rightarrow^+ A\alpha$ for a general string α.

3. DEFINITION OF Ch(k) GRAMMARS

In order to intuitively characterize the class of grammars to be defined and its relationship with other classes of grammars, we first illustrate the deterministic top-down, bottom-up and left-corner parsing algorithms; i.e. the parsing algorithms that apply to LL(k), LR(k) and LC(k) [7,8] grammars, respectively. Consider the derivation tree shown in Figure 1.

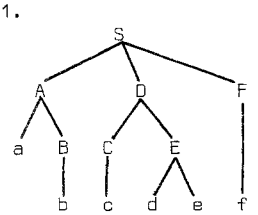

Figure 1. Derivation tree

In the top-down parsing algorithm for LL(k) grammars the productions for the nonterminals in the tree are recognized in the order S,A,B,D,C,E,F. Each production in the tree is recognized before its descendants and its right siblings and their descendants. In the bottom-up parsing algorithm for LR(k) grammars the productions for the nonterminals in the tree are recognized in the order B,A,C,E,D,F,S. Each production in the tree is recognized after its descendants but before its ancestors and its right siblings and their descendants. In the left-corner parsing algorithm for LC(k) grammars the productions are recognized in the order A,B,S,C,D, E,F. Each production is recognized after its left corner but before any of the siblings of the left corner (or their descendants).

The Ch(k) grammars can now be characterized as LR(k) grammars for which the left-hand sides of the productions can be recognized in the same order as the whole productions in top-down parsing, but the right-hand sides are recognized in the order of the bottom-up parse. This method for constructing the derivation tree "node by node" corresponds to the way in which the well-known recursive descent-parser constructs the tree. For example, when the top-down parsing algorithm has recognized the productions for S,A,B and D in the derivation tree of Figure 1, then in the case of Ch(k) grammars the left-hand sides S,A,B and D are determined but the whole productions only for B and A.

In a sense, the Ch(k) grammars constitute a dual of the PLR(k) grammars [8,9] in a similar way as the LL(k) grammars constitute a dual of LR(k) grammars as regards the construction of the derivation tree. The PLR(k) grammars are LR(k) grammars for which the left-hand sides of the productions can be recognized in the same order as the whole productions in left-corner parsing, but the right-hand sides are recognized in the order of the bottom-up parse. Thus the PLR(k) grammars are those for which deterministic "node by node" parsing bottom-up is possible, whereas the Ch(k) grammars are those for which deterministic "node by node" parsing top-down is possible.

DEFINITION 3.1. Let k be a non-negative integer. A grammar $G = (N,\Sigma,P,S)$ is said to be a Ch(k) *grammar* if, for a terminal string w, a nonterminal A and strings γ, α, δ_1 and δ_2 in $(N \cup \Sigma)^*$ such that $A \to \alpha\delta_1$ and $A \to \alpha\delta_2$ are distinct productions of G and α is the longest common prefix of $\alpha\delta_1$ and $\alpha\delta_2$, the condition

$$S \overset{*}{\underset{L}{\Rightarrow}} wA\gamma$$

implies that

$$FIRST_k(\delta_1\gamma) \cap FIRST_k(\delta_2\gamma) = \emptyset. \qquad \square$$

Observe the obvious difference with the definition of LL(k) grammars. In that case the implication $FIRST_k(\alpha\delta_1\gamma) \cap FIRST_k(\alpha\delta_2\gamma) = \emptyset$ is used. Thus in the case of Ch(k) grammars it is not necessary to consider the terminal strings which can be derived from the longest common prefix α of the right-hand sides of two distinct

productions $A \to \alpha\delta_1$ and $A \to \alpha\delta_2$. These observations imply immediately

THEOREM 3.2. Every LL(k) grammar is a Ch(k) grammar.

Proof. Assume that a grammar $G = (N,\Sigma,P,S)$ is not a Ch(k) grammar. Then there exist a terminal string w, a nonterminal A and strings γ, α, δ_1 and δ_2 in $(N \cup \Sigma)^*$ such that $A \to \alpha\delta_1$ and $A \to \alpha\delta_2$, where α is the longest common prefix of $\alpha\delta_1$ and $\alpha\delta_2$, are two distinct productions of G, $S \underset{L}{\Rightarrow}^* wA\gamma$ and $FIRST_k(\delta_1\gamma) \cap FIRST_k(\delta_2\gamma) \neq \emptyset$. But then also $FIRST_k(\alpha\delta_1\gamma) \cap FIRST_k(\alpha\delta_2\gamma) \neq \emptyset$, which means that the grammar G is not LL(k), as desired. □

As an informal description of the definition of Ch(k) grammars and of its relationship with the definitions of other classes of grammars consider the following situation. There exist terminal strings w, x, y and z, a nonterminal A, a symbol X which is a nonterminal or a terminal, and a general string α such that $A \to X\alpha$ is a production and

$$S \Rightarrow^* wAz, \quad X \Rightarrow^* x, \quad \alpha \Rightarrow^* y.$$

Consider then the terminal string wxyz. The production $A \to X\alpha$ in question in the derivation tree of wxyz can be recognized with certainty after scanning

(i) w and k:xyz if the grammar is LL(k),

(ii) wx and k:yz if the grammar is LC(k),

(iii) wxy and k:z if the grammar is LR(k), PLR(k) or Ch(k).

However, if the grammar is PLR(k) then the left-hand side A of the production $A \to X\alpha$ is recognized after scanning wx and k:yz, and if the grammar is Ch(k) then the left-hand side A is recognized after scanning w and k:xyz.

As we remarked in the introduction the Ch(k) grammars can be considered as a generalization of simple chain grammars [6]. A grammar $G = (N,\Sigma,P,S)$ is said to be a simple chain grammar if G is ε-free (i.e. P contains no production of the form $A \to \varepsilon$), P is prefix-free (i.e. there are no two productions $A \to \alpha\beta$ and $A \to \alpha$ in P) and for any pair of productions $A \to \alpha X\beta$ and $A \to \alpha Y\gamma$, X and Y are in $N \cup \Sigma$, such that $X \neq Y$, we have $FIRST_1(X) \cap FIRST_1(Y) = \emptyset$. For left part grammars [5] the requirement that P is prefix-free is dropped. The following theorem is an immediate consequence of the above discussion.

THEOREM 3.3. A grammar $G = (N,\Sigma,P,S)$ is a simple chain grammar if and only if P is prefix-free and G is an ε-free Ch(1) grammar.

For example, the grammar with productions

$$S \to a, \quad S \to ab$$

is not a simple chain grammar, because these two productions do not constitute a prefix-free set of productions. However, this grammar is Ch(1). Thus we conclude by Theorem 3.3 that the class of simple chain grammars is properly contained in

the class of Ch(k) grammars whenever k > 0. Further, since there exist simple chain grammars which are not LL(k) for any k [6], we conclude by Theorems 3.2 and 3.3 that the class of LL(k) grammars is properly contained in the class of Ch(k) grammars whenever k > 0. In the case k = 0 both classes contain only the grammars that derive at most one terminal string.
The following two theorems are also immediate consequences of the Ch(k) definition. The proofs are analogous to corresponding ones in the LL(k) case and therefore omitted.

THEOREM 3.4. Each Ch(k) grammar is unambiguous.

THEOREM 3.5. A reduced Ch(k) grammar is not left-recursive.

4. PROPERTIES OF Ch(k) GRAMMARS

In this section our primary interest is to relate the Ch(k) grammars and LL(k) grammars by a grammatical transformation. In fact, we shall show that the Ch(k) grammars are exactly those which can be transformed into LL(k) grammars by *left-factoring* the grammar until it has no two productions of the form $A \to \alpha\beta$ and $A \to \alpha\gamma$ where $\alpha \neq \varepsilon$. This implies, in particular, that the language generated by a Ch(k) grammar is always an LL(k) language, and thus, by Theorem 3.2, the family of Ch(k) languages equals the family of LL(k) languages. Furthermore, this result implies the interesting property that Ch(k) grammars are PLR(k) grammars [8,9]. Since the left factoring process yields an LL(k) grammar if and only if the given grammar is Ch(k), we can perform the test whether a grammar is Ch(k) by left-factoring the grammar and then testing the LL(k)-property. However, the LL(k) parser of the resulting grammar cannot be used to produce left or right parses in the original grammar. That is, left-factoring can distort the structure of the grammar such that no left-to-left nor left-to-right cover is obtained [6]. Nevertheless, we can give a simple modification of the left-factoring process such that the above mentioned properties are preserved except that, as an additional bonus, the transformed grammar left-to-right covers the original grammar.
We begin by defining that a grammar is in the *left-factored* form, if it has no two productions $A \to \alpha\beta$ and $A \to \alpha\gamma$ such that α is not the empty string. The definitions of LL(k) and Ch(k) grammars imply immediately

THEOREM 4.1. A grammar in the left-factored form is LL(k) if and only if it is Ch(k).

The process of left-factoring can be regarded as a transformation which is composed

by consecutive steps of "factoring" two distinct productions $A \to \alpha\beta$ and $A \to \alpha\gamma$, $\alpha \neq \varepsilon$, into productions $A \to \alpha A'$, $A' \to \beta$ and $A' \to \gamma$, where A' is a new nonterminal. These steps are performed, in an arbitrary way, until the grammar is in the left-factored form. It should be noticed that the above specification of the process does not define the "left-factored" grammar uniquely. However, our results are independent of the particular way in which the individual steps and their order in the left-factoring process are chosen.

THEOREM 4.2. The grammar obtained by the left-factoring process is LL(k) if and only if the original grammar is Ch(k).

Sketch of proof. By Theorem 4.1 it is enough to show that the process of left-factoring does not affect the Ch(k)-ness of a grammar and that the process of left-factoring cannot produce a Ch(k) grammar from a non-Ch(k) grammar. It is clear by the definition that this is true as regards one individual step in the left-factoring process. Since the whole process is just a consecutive sequence of these individual steps, we thus conclude the theorem. □

COROLLARY 4.3. The families of Ch(k) and LL(k) languages are identical.

The PLR(k) grammars [8,9] are exactly those grammars which can be transformed into LC(k) grammars by the left-factoring process [8]. Thus, since the class of LL(k) grammars is properly contained in the class of LC(k) grammars [8], we conclude by Theorem 4.2 that the class of Ch(k) grammars is properly contained in the class of PLR(k) grammars. This implies further that Ch(k) grammars are LR(k) grammars, since PLR(k) grammars are LR(k) grammars [8].

The inclusion of the Ch(k) grammars in the class of PLR(k) grammars is an interesting property because PLR(k) grammars can be transformed into LL(k) grammars such that the transformed grammar left-to-right covers the original grammar [8]. This is thus true also for Ch(k) grammars. However, the transformation involved is rather complicated, and it is thus desirable to find out easier possibilities.

Let therefore $G_1 = (N_1,\Sigma,P_1,S)$ be a grammar and let $G_1' = (N_1',\Sigma,P_1',S)$ where $N_1' = N_1 \cup \{[A\alpha] \mid A \to \alpha \text{ is in } P\}$ and $P_1' = \{A \to \alpha[A\alpha] \mid A \to \alpha \text{ is in } P_1\} \cup \{[A\alpha] \to \varepsilon \mid A \to \alpha \text{ is in } P_1\}$. Further let $G_2 = (N_2,\Sigma,P_2,S)$ be a grammar obtained by the left-factoring process from G_1', and let $h: P_2^* \to P_1^*$ be a homomorphism defined by the conditions

$h([A\alpha] \to \varepsilon) = A \to \alpha$, and

$h(A \to \alpha) = \varepsilon$.

LEMMA 4.4. The grammar G_2 left-to-right covers the grammar G_1 with respect to the homomorphism h.

Sketch of proof. The appearance of a production of the form $[A\alpha] \to \varepsilon$ in a left parse of a terminal string x in G_1' means that a substring y of x for which $A \Rightarrow \alpha \Rightarrow^* y$ in G_1 has been analyzed. Thus it can be shown by any easy induction that the sequence of these productions in a left parse of x in G_1' defines a right parse of x in G_1. Further, this is also true when G_1' is replaced by the grammar G_2, since the process of left-factoring does not affect the order in which the productions of the form $[A\alpha] \to \varepsilon$ are recognized. By formalizing the above discussion we can conclude that the grammar G_2 left-to-right covers the grammar G_1 with respect to the homomorphism h. □

Since the grammar G_2 obtained by the left-factoring process from G_1' is LL(k) if and only if any grammar obtained by the left-factoring process from G_1 is LL(k), we conclude by Theorem 4.2 and Lemma 4.4 that the following theorem holds.

THEOREM 4.5. Each Ch(k) grammar of size*) n can be left-to-right covered by an LL(k) grammar of size O(n).

REFERENCES

1. A.V. Aho and J.D. Ullman, The Theory of Parsing, Translation and Compiling, Vols 1 and 2, Prentice Hall, Englewood Cliffs, N.J., 1972 and 1973.
2. B.M. Brosgol, Deterministic translation grammars, Proc. 8th Princeton Conf. on Information Sciences and Systems 1974, pp. 300-306.
3. A.J. Demers, Generalized left corner parsing, Conf. Record of the 4th ACM Sympos. on Principles of Programming Languages 1977, pp. 170-182.
4. M. Hammer, A new grammatical transformation into LL(k) form, Conf. Record of the 6th Annual ACM Sympos. on Theory of Computing 1974, pp. 266-275.
5. A. Nijholt, A left part theorem for grammatical trees, Discrete Mathematics 25 (1979), pp. 51-63.
6. A. Nijholt, Simple chain grammars and languages, to appear in Theoretical Computer Science.
7. D.J. Rosenkrantz and P.M. Lewis, Deterministic left-corner parsing, IEEE Conf. Record of the 11th Annual Sympos. on Switching and Automata Theory 1970, pp. 139-152.
8. E. Soisalon-Soininen, Characterization of LL(k) languages by restricted LR(k) grammars, Ph.D. Thesis, Report A-1977-3, Department of Computer Science, University of Helsinki.
9. E. Soisalon-Soininen and E. Ukkonen, A characterization of LL(k) languages, in: Automata, Languages and Programming (eds. S. Michaelson and R. Milner), Third Colloquium, Edinburgh University Press, Edinburgh, 1976, pp. 20-30.

*) The size $|G|$ of a grammar G is defined by $|G| = \sum_{A \to \alpha \in P} |A\alpha|$.