ON THE PARSING OF LL-REGULAR GRAMMARS

Anton Nijholt
Department of Mathematics
Free University, Amsterdam, The Netherlands*

1. INTRODUCTION

Culik II and Cohen [1] introduced the class of LR-regular grammars, an extension of the LR(k) grammars. In [2] and [3] the same idea is applied to the class of LL(k) grammars and the LL-regular grammars were introduced. The idea is that the parsing is done with a two-scan parsing algorithm. The first scan of a sentence w to be parsed, called the pre-scan, is done by a Moore machine (reading w from right to left) and yields a string of symbols which is the input for a deterministic push-down transducer (dpdt). In the case of an LR-regular grammar G the result of the pre-scan is a sentence of an LR(0) grammar G' which can be constructed from G, and the parsing can be done with regard to this LR(0) grammar. In the case of an LL-regular grammar it is possible to construct a strict deterministic grammar [8] and after the pre-scan has been performed the parsing can be done with regard to this grammar. However a more efficient method can be given since it can be shown that the parsing can be done with a 1-predictive parsing algorithm or even with a simple LL(1) parsing method (see section 3 and [2]).
The classes of LR-regular and LL-regular grammars have some similar properties as the classes of LR(k) and LL(k) grammars. Moreover, sometimes the proofs of these properties need only slight adaptions. In this paper the proofs are omitted. In [2] proofs, and some properties and examples not given here, can be found. In the remainder of this section we give some notations and definitions. In section 2 we list some properties and the main part of this paper is in section 3 where we consider the parsing of LL-regular grammars.

A (reduced) context-free grammar (cfg) is denoted by $G = (N,T,P,S)$, $V = N \cup T$; we will denote elements of N by $A,B,C,\ldots$; elements of T by $a,b,c,\ldots$; elements of T^* by $\ldots w,x,y,z$; elements of V^* by $\alpha,\beta,\gamma,\delta,\ldots$; ε denotes the empty string. A regular partition of T^* is a partition of T^* of finite index and such that each block is a regular set. The states of a Moore machine (with input alphabet T) define a regular partition of T^* [4].

* The research reported in this paper has been carried out at the Department of Applied Mathematics of the Twente University of Tehnology, Enschede.

The following definition can be found in [6] and in a generalized form in [5].

DEFINITION 1. (Left-cover)

Let G and G' be cfgs, G = (N,T,P,S), G' = (N',T,P',S') and L(G) = L(G'). G' left-covers G if there is a homomorphism h from P' to P^* (extended to P'^*) such that

(1) if $S' \overset{\rho'}{\underset{\ell}{\Longrightarrow}} w$, then $S \overset{h(\rho')}{\underset{\ell}{\Longrightarrow}} w$, and

(2) for all ρ such that $S \overset{\rho}{\underset{\ell}{\Longrightarrow}} w$, there exists ρ' such that $S' \overset{\rho'}{\underset{\ell}{\Longrightarrow}} w$ and $h(\rho') = \rho$.

In this definition ρ and ρ' denote the concatenations of the productions used in the left-most derivations.

2. LL-REGULAR GRAMMARS, AN EXTENSION OF LL(k) GRAMMARS

DEFINITION 2. (LL-regular grammar)

Let G = (N,T,P,S) be a cfg, π a regular partition of T . G is said to be an LL(π) grammar if, for any two left-most derivations of the forms

(i) $S \overset{*}{\underset{\ell}{\Longrightarrow}} wA\alpha \underset{\ell}{\Longrightarrow} w\gamma\alpha \overset{*}{\Longrightarrow} wx$,

(ii) $S \overset{*}{\underset{\ell}{\Longrightarrow}} wA\alpha \underset{\ell}{\Longrightarrow} w\delta\alpha \overset{*}{\Longrightarrow} wy$,

where $x \equiv y \pmod{\pi}$, then we may conclude $\gamma = \delta$. A cfg G is said to be LL-regular or LLR if there exists such a partition π of T^*.

The class of grammars introduced in [3] is in fact a subclass of our class of LLR grammars. We prefer to call those grammars strong LLR grammars to obtain a framework analogous to the LL(k) and strong LL(k) grammars. If we replace in definition 2 each occurrence of w and α in (i) by w_1 and α_1 and in (ii) by w_2 and α_2 respectively, then we obtain the definition of a strong LL(π) grammar. It will be clear that every strong LLR grammar is LLR and easily can be verified that every LL(k) grammar is LLR.

Example 1. Cfg G with only productions $S \to aAaa|bAbaa|bAbab$ and $A \to bA|b$ is neither LL nor strong LLR. However G is LLR. A regular partition for G is given in section 3.

THEOREM 1.

a. Every LLR grammar is unambiguous

b. No LLR grammar is left-recursive

c. It is decidable whether a cfg is LL(π) for a given regular partition π.

Since every left-recursive grammar can be covered by a non-left-recursive grammar [7] in some cases it may be useful to see if elimination of left recursion yields an LL(π) grammar for some regular partition π. Theorem 1c. can be proved in a way such that it amounts to the construction of the parsing algorithm. This algorithm will be

discussed in the following section.
The following two theorems have proofs which differ only in details of proofs for LL(k) and LR(k) grammars as given in [6].

THEOREM 2.
Every LL(π) grammar, where π is a left congruence, is an LR(π) grammar.

Since a left congruence can always be found by refining of the partition we may say that every LLR grammar is also an LRR grammar. This inclusion is proper.
Example 2. Cfg G with only production $S \to Cc$, $C \to Cb|b$ is LR(0) and hence LRR, but G is not LLR.

THEOREM 3.
Every LLR grammar G, such that $\varepsilon \notin L(G)$, has an equivalent LLR grammar G' in Greibach normal form (GNF). Moreover G' left-covers G.

Like the equivalent theorem for LL(k) grammars this theorem is useful in showing that a language may be non-deterministic. The LLR languages are properly contained in the LRR languages. For example, the language $L = \{c^n d^n,\ c^{n+l} d^n \mid n > 1,\ l \geq 1\}$ is a deterministic language, and therefore LRR, but it has no LLR grammar in GNF.

3. PARSING OF LL-REGUALR GRAMMARS

An LL-regular grammar can be parsed, after a regular pre-scan from right to left has been performed, by using a strict deterministic parsing method [2]. This section however is devoted to a generalization of the LL(k)-parsing method. This generalization is such that any LL(π) grammar can be parsed, after a regular pre-scan from right to left has been performed, with a 1-predictive parsing algorithm.
First we need the following definition, in which π is a regular partition of T^*, $\pi = \{B_0, B_1, \ldots, B_n\}$ and $\alpha \in V^*$.

DEFINITION 3.
$\mathrm{BLOCK}(\alpha) = \{B_k \in \pi \mid L(\alpha) \cap B_k \neq \emptyset\}$. If $B_i, B_j \in \pi$, then $B_i \,\square\, B_j = \{B_k \in \pi \mid B_k \cap (B_i.B_j) \neq \emptyset\}$, where $B_i.B_j$ denotes the usual concatenation of sets of strings.
Let $L_1, L_2 \subseteq \pi$, then $L_1 \,\square\, L_2 = \{B_k \in \pi \mid B_k \in B_i \,\square\, B_j,\ B_i \in L_1 \text{ and } B_j \in L_2\}$.

Notice that $L(\alpha)$ is a context-free language (cfl), B_k is a regular set and therefore $L(\alpha) \cap B_k$ is a cfl. Hence it is decidable whether $L(\alpha) \cap B_k$ is non-empty [4].
This definition, together with lemma 1 we will give below, enables us to introduce the generalized parsing method.

LEMMA 1.
a. $\mathrm{BLOCK}(\alpha\beta) = \mathrm{BLOCK}(\alpha) \,\square\, \mathrm{BLOCK}(\beta)$.
b. Let $G = (N,T,P,S)$ be a cfg and suppose $A \to \beta$ and $A \to \gamma$ are in P, $\beta \neq \gamma$. G is not

$LL(\pi)$ iff there is a derivation $S \overset{*}{\underset{\ell}{\Longrightarrow}} wA\alpha$ and
$(BLOCK(\beta) \boxdot BLOCK(\alpha)) \cap (BLOCK(\gamma) \boxdot BLOCK(\alpha)) \neq \emptyset$.

Analogous to the theory of LL(k) parsing we define functions $T_{A,L}$ on partition π (these functions are called the $LL(\pi)$-tables), where A is a nonterminal and L is a set of blocks. These functions satisfy the following conditions.

(1) $T_{A,L}(B_k)$ = error, if there is no production $A \to \alpha$ in P such that $BLOCK(\alpha) \boxdot L$ contains B_k.

(2) $T_{A,L}(B_k) = (A \to \alpha, [L_1,L_2,\ldots,L_m])$, if $A \to \alpha$ is the unique production in P such that $BLOCK(\alpha) \boxdot L$ contains B_k. If $\alpha = x_0C_1x_1C_2\ldots C_mx_m$, $m \geq 0$, $C_i \in N$ and $x_i \in T^*$, then $L_i = BLOCK(x_iC_{i+1}\ldots C_mx_m) \boxdot L$, $(0 \leq i \leq m)$.

(3) $T_{A,L}(B_k)$ = undefined if there are two or more productions $A \to \alpha_1$ and $A \to \alpha_2$, $\alpha_1 \neq \alpha_2$, such that $(BLOCK(\alpha_1) \boxdot L) \cap (BLOCK(\alpha_2) \boxdot L)$ contains B_k.

Now it will be clear that if cfg G is $LL(\pi)$ and there is a derivation $S \overset{*}{\underset{\ell}{\Longrightarrow}} wA\alpha \overset{*}{\Longrightarrow} wx$, then $T_{A,L}(B_k)$, where $x \in B_k$ and $L = BLOCK(\alpha)$, will uniquely determine which production is to be used to expand A.
Starting with $LL(\pi)$-table $T_0 = T_{S,\{B_0\}}$, where $B_0 = \{\varepsilon\}$, it is possible to determine the set T(G) of all relevant $LL(\pi)$-tables of G. In the example at the end of this section T(G) is given for the cfg of example 1.
With the $LL(\pi)$-tables as input the following algorithm constructs a 1-predictive parsing table.
In this algorithm we use the partition $\pi_0 = \{aT^* \mid a \in T\} \cup \{\varepsilon\}$ and we require that partition π for which the parsing table is constructed is a refinement of π_0. We let $\pi = \{B_0,B_1,\ldots,B_n\}$, where $B_0 = \{\varepsilon\}$. It is always possible to obtain such a partition π if G is LLR. The condition $\pi \subseteq \pi_0$ is introduced to prevent the parsing algorithm (see algoritm 2) from giving left parses for sentences which do not belong to L(G).
To each block in π we assign a unique number $(0,1,2,\ldots,n)$, and we let π also denote the set of these numbers. These numbers will be the output alphabet of the Moore machine in the parsing algorithm.
To each production in P we also assign a unique number and we let P also denote the set of these numbers.

ALGORITHM 1. (construction of a 1-predictive parsing table)
Input: $LL(\pi)$ grammar $G = (N,T,P,S)$, $\pi \subseteq \pi_0$ and the set T(G).
Output: a parsing table Q for G,

$$Q: (T(G) \cup T \cup \{\$\}) \times \pi \to ((T(G) \cup T)^* \times P) \cup \{pop, accept, error\}$$

Method:

(1) if $A \to x_0C_1x_1C_2x_2\ldots C_mx_m$ is the i-th production in P and $T_{A,L}$ is in T(G), then for every B_j such that $T_{A,L}(B_j) = (A \to x_0C_1x_1C_2x_2\ldots C_mx_m, [L_1,L_2,\ldots,L_m])$ we have $Q(T_{A,L},j) = (x_0T_{C_1,L_1}x_1T_{C_2,L_2}x_2\ldots T_{C_m,L_m}x_m,i)$.

(2) $Q(a,j)$ = pop, if $w \in B_j$ implies that the first symbol of w is a.

(3) $Q(\$,0)$ = accept

(4) otherwise $Q(X,j)$ = error, for X in $T(G) \cup T \cup \{\$\}$ and block B_j.

Now we are prepared to give the parsing algorithm. We let w^R denote the string w in a reversed order, $B_j^R = \{w^R \mid w \in B_j\}$ and $\pi^R = \{B_j^R \mid B_j \in \pi\}$. For convenience we assume that G is $LL(\pi)$, where π is a left congruence. We assume the reader is familiar with the construction of a Moore machine M_π which defines by its states the right congruence π^R. M_π will perform the pre-scan from right to left.

ALGORITHM 2. (1-predictive parsing algorithm)

Input: $LL(\pi)$ grammar $G = (N,T,P,S)$, parsing table Q and Moore machine M_π. The string $w = a_0a_1 \ldots a_ia_{i+1} \ldots a_m \in T^*$ has to be parsed.

Output: The left parse for w if $w \in L(G)$, otherwise 'error'.

Method:

(1) Apply M_π to w^R such that if $a_ia_{i+1} \ldots a_m$ is in block B_j, then the to B_j^R corresponding state of M_π gives output j. The result is a string $w_\pi = j_0j_1 \ldots j_m \in \pi^*$.

(2) A configuration is a triple $(x, X\alpha, \psi)$, where

i. x represents the unused portion of the original input string w_π.

ii. $X\alpha$ represents the string on the pushdown list (with X on top), $X\alpha \in (T(G) \cup T)^*\$$.

iii. ψ is the string on the output tape.

The initial configuration is $(w_\pi, T_0\$, \varepsilon)$, where $T_0 = T_{S,\{B_0\}}$, the accept configuration is $(\varepsilon, \$, \rho)$ where ρ is the left parse of w with respect to G.

(3) A move $\vdash$ is defined on the configurations as follows:

i. $(jx, T_k\alpha, \psi) \vdash (jx, \beta\alpha, \psi i)$, $T_k \in T(G)$ and $Q(T_k,j) = (\beta,i)$.

ii. $(jx, a\alpha, \psi) \vdash (x, \alpha, \psi)$, $a \in T$ and $Q(a,j)$ = pop.

If none of these moves can be done, hence $Q(X,j)$ = error, then the parsing ceases.

Example 3. Cfg G with only productions 1. $S \to aAaa$, 2. $S \to bAbaa$, 3. $S \to bAbab$, 4. $A \to bA$ and 5. $A \to b$. The table below gives a regular partition for G which satisfies the conditions of the two algorithms.

π							
B_0	$\{\varepsilon\}$	B_6	$bbbT^*b$	B_{12}	$babT^*b$	B_{18}	$\{bab\}$
B_1	$aaaT^*$	B_7	$bbaT^*a$	B_{13}	$\{b\}$	B_{19}	$\{ba\}$
B_2	$aabT^*$	B_8	$bbaT^*b$	B_{14}	$\{bb\}$	B_{20}	$\{a\}$
B_3	$abaT^*$	B_9	$baaT^*a$	B_{15}	$\{bbb\}$	B_{21}	$\{aa\}$
B_4	$abbT^*$	B_{10}	$baaT^*b$	B_{16}	$\{bba\}$	B_{22}	$\{ab\}$
B_5	$bbbT^*a$	B_{11}	$babT^*a$	B_{17}	$\{baa\}$		

In the LL(π)-tables we only display the non-error entries.

T_0	$= T_{S,\{B_0\}}$
B_3	$S \to aAaa$, $[\{B_{21}\}]$
B_4	$S \to aAaa$, $[\{B_{21}\}]$
B_5	$S \to bAbaa$, $[\{B_{17}\}]$
B_6	$S \to bAbab$, $[\{B_{18}\}]$

T_1	$= T_{A,\{B_{21}\}}$
B_5	$A \to bA$, $[\{B_{21}\}]$
B_7	$A \to bA$, $[\{B_{21}\}]$
B_{17}	$A \to b$, $[\emptyset]$

T_2	$= T_{A,\{B_{17}\}}$
B_5	$A \to bA$, $[\{B_{17}\}]$
B_7	$A \to b$, $[\emptyset]$

T_3	$= T_{A,\{B_{18}\}}$
B_6	$A \to bA$, $[\{B_{18}\}]$
B_8	$A \to b$, $[\emptyset]$

Parsing table Q. (only the entries of T_0, T_1, T_2 and T_3 are given)

Q	3	4	5	6	7	8	17
T_0	aT_1aa, 1	aT_1aa, 1	bT_2aa, 2	bT_3bab, 3	-	-	-
T_1	-	-	bT_1, 4	-	bT_1, 4	-	b, 5
T_2	-	-	bT_2, 4	-	b, 5	-	-
T_3	-	-	-	bT_3, 4	-	b, 5	-

Let us apply algorithm 2 on w = abbaa.

(1) applying M_π yields 4.7.17.21.20

(2) $(4.7.17.21.20,\ T_0\$,\ \varepsilon) \vdash (4.7.17.21.20,\ aT_1aa\$,\ 1) \vdash (7.17.21.20,\ T_1aa\$,\ 1)$
$\vdash (7.17.21.20,\ bT_1aa\$,\ 14) \vdash (17.21.20,\ T_1aa\$,\ 14) \vdash (17.21.20,\ baa\$,\ 145)$
$\overset{*}{\vdash} (\varepsilon,\ \$,\ 145)$, and hence 145 is the left parse for abbaa.

Note. It is possible to show that if G is in GNF then we can construct from parsing table Q a simple LL(1) grammar G_π with properties:

(i) $\{[M_\pi(w^R)]^R \mid w \in L(G)\} \subseteq L(G_\pi)$,

(ii) if $w \notin L(G)$ then $[M_\pi(w^R)]^R \notin L(G_\pi)$, and

(iii) there exist homomorphisms h and g such that if ρ is a left parse for $w_\pi \in L(G_\pi)$ then $h(\rho)$ is a left parse for $w = g(w_\pi) \in L(G)$.

From these properties and from theorem 3 it follows that every LLR grammar can be parsed, after a regular pre-scan has been performed, with respect to a simple LL(1) grammar.

Acknowledgements:

I am grateful to prof. L. Verbeek who gave me the opportunity to do this work.

References.

1. Čulik II K. and Cohen R., LR-regular grammars - an extension of LR(k) grammars, J. Comput. System Sci.7, (1973), No. 1, 66-96.
2. Nijholt A., Regular extensions of some classes of grammars, T.W. mem. No. 100, september 1975, Twente University of Technology.
3. Jarzabek S. and Krawczyk T., LL-regular grammars, Information Processing Letters, Vol. 4, No. 2, november 1975, 31-37.
4. Hopcroft J.E. and Ullman J.D., "Formal languages and their relation to automata", Add. Wesley, Reading, M.A., 1969.
5. Nijholt A., On the covering of parsable grammars, T.W. mem. No. 96, september 1975, Twente University of Technology.
6. Aho A.V. and Ullman J.D., "The theory of parsing, translation and compiling", Vols. I and II, Prentice Hall, Englewood Cliffs, 1972 and 1973.
7. Nijholt A., On the covering of left-recursive grammars, T.W. mem. No. 127, april 1976, Twente University of Technology.
8. Harrison M.A. and Havel I.M., Strict deterministic grammars, J. Comput. System Sci. 7, (1973), No. 3, 237-277.