RAIRO
INFORMATIQUE THÉORIQUE

ANTON NIJHOLT

From *LL*-regular to *LL*(1) grammars : transformations, covers and parsing

RAIRO – Informatique théorique, tome 16, n° 4 (1982), p. 387-406.

<http://www.numdam.org/item?id=ITA_1982__16_4_387_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 16, n° 4, 1982, p. 387 à 406)

FROM *LL*-REGULAR TO *LL(1)* GRAMMARS: TRANSFORMATIONS, COVERS AND PARSING (*)

by Anton NIJHOLT (1)

Communicated by W. BRAUER

Abstract. — *In this paper it is shown that it is possible to transform any LL-regular grammar G into an LL(1) grammar G' in such a way that parsing G' is as good as parsing G. That is, a parse of a sentence of grammar G can be obtained with a simple string homomorphism from the parse of a corresponding sentence of G'. Since any LL(k) grammar is an LL-regular grammar the results which are obtained are valid for LL(k) grammars as well. The relation between LL-regular and LL(1) grammars is expressed by means of a generalized version of the well-known cover relation between two context-free grammars.*

Résumé. — *On montre que toute grammaire LL-régulière G peut se transformer en une grammaire LL(1) G' de telle sorte que l'analyse de G' soit équivalente à celle de G, en ce sens qu'une dérivation d'un mot selon G soit image homomorphe de la dérivation du mot correspondant selon G' . Comme toute grammaire LL(k) est LL-régulière, nos résultats valent aussi bien pour les grammaires LL(k). La relation entre grammaires LL-régulières et LL(1) s'exprime au moyen d'une généralisation de la relation classique de couverture entre deux grammaires algébriques (context-free).*

1. INTRODUCTION

The class of $LL(k)$ grammars has extensively been studied. *See* Wood [12] for a bibliography and a survey of the area of top-down parsing. Since $LL(k)$ and especially $LL(1)$ grammars require a relatively simple parsing method many authors have tried to generalize the definition of $LL(k)$ grammars in such a way that the generalized class of grammars has the property that each of its grammars can be transformed to an $LL(k)$ grammar. *See* Nijholt [8] (Chapter 12), where a survey is given of definitions of grammars which can be transformed to $LL(k)$ grammars.

(*) Received July 1980, revised June 1981.
(1) Department of Computer Science, Twente University of Technology, Enschede, The Netherlands.

R.A.I.R.O. Informatique théorique/Theoretical Informatics, 0399-0540/1982/387/$ 5.00

Parsing methods for $LL(k)$ grammars use a fixed amount k of look-ahead of the input string to decide at each moment which production has been used in the generation of the input string which is under consideration. Instead of using this concept of finite look-ahead it is also possible to use "regular" look-ahead. That is, if we divide the set of possible input strings into regular disjoint subsets, then the next production in the generation of the input string is determined by recognizing the regular subset to which the look-ahead string (of unbounded length) belongs. In this way we obtain an other generalization of the class of $LL(k)$ grammars, the class of LL-regular or $LL(\pi)$ grammars. Here π denotes the set of regular and disjoint subsets of the set of possible input strings. In Jarzabek and Krawczyk [5], Nijholt [6, 7, 9] and Poplawski [10, 11] this class of grammars has been studied.

In the present paper we are concerned with the construction of an $LL(1)$ grammar G' from an LL-regular grammer G, in such a way that parsing G' is "as good" as parsing G. This notion of "as good" will be formalized by using the concept of the grammatical cover. In Gray and Harrison [2] the concept of cover has been systematically introduced. Here we use a more general definition which we take from Nijholt [8]. Since, in contrast to the definition of Gray and Harrison, we have the possibility that grammars which generate different languages can cover each other, we use two homomorphisms in the cover definition. The first homomorphism maps sentences of G' into sentences of G. The second homomorphism maps the parses of a sentence of G' into the parses of the corresponding sentence of G.

Since we are able to transform each LL-regular grammar G into an $LL(1)$ grammar G' which covers G, it is possible, by studying G', to obtain results which give us information on the parsing of G without being obliged to consider a parsing method for G. The aim of this paper is to study grammar G' and its relation to the original LL-regular grammar G. This will be done in section 2. This section is concluded with some preliminary definitions and notation. The paper is concluded with a few remarks on the parsing problem for LL-regular grammars.

Preliminaries

We assume that the reader is familiar with Aho and Ullman [1]. For notational reasons we review some concepts.

A *context-free grammar* (*CFG* for short) is denoted by the four-tuple $G=(N, \Sigma, P, S)$, where N consists of the *nonterminal symbols* (denoted by the Roman capitals $A, B, C, \ldots, S$); Σ consists of the *terminal symbols* (denoted

by the Roman smalls a, b, c and d); $N \cap \Sigma = \emptyset$ (the empty set); $N \cup \Sigma$ is denoted by V (elements of V will be denoted by X, Y and Z; elements of V^* will be denoted by $\alpha, \beta, \gamma, \delta, \varepsilon$ and ω). The elements of Σ^* will be denoted by x, y, z, w and ε. The set P of *productions* is a subset of $N \times V^*$ (notation $A \to \alpha$ if $(A, \alpha) \in P$) and $S \in N$ is called the *start symbol* of the grammar. If $A \to \alpha$ is in P, then A is called the *lefthand side* and α is called the *righthand side* of this production.

We have the usual notation $\Rightarrow$, $\underset{L}{\Rightarrow}$ and $\underset{R}{\Rightarrow}$ for *derivations*, *leftmost derivations* and *righmost derivations*, respectively. The superscripts $+$ and $*$ will be used to denote the *transitive* and the *reflexive-transitive closures* of these relations. For some α, β, γ in V^* we use the notation $\alpha\underline{\gamma} \overset{*}{\underset{L}{\Rightarrow}} \beta\underline{\gamma}$ to denote that in the specific derivation $\alpha\gamma \overset{*}{\underset{L}{\Rightarrow}} \beta\gamma$ which is considered the displayed string γ is not rewritten.

We will identify a production in P by a unique number i by writing $i.\ A \to \alpha$. The set of these *production identifiers* of a *CFG* G will be denoted by Δ_G. If $\delta = p_1 p_2 \ldots p_n$ is a sequence of production identifiers, then:

$$\alpha \overset{\delta}{\underset{L}{\Rightarrow}} \beta,$$

represents a leftmost derivation from α to β using in sequence the productions $p_1, p_2, \ldots, p_n$.

The set:

$$l_G = \{ (w, \delta) \mid S \overset{\delta}{\underset{L}{\Rightarrow}} w, w \in \Sigma^* \},$$

is called the *left parse relation* of *CFG* $G = (N, \Sigma, P, S)$. If $(w, \delta) \in l_G$, then δ is said to be a *left parse* of w (with respect to G). For any string $\alpha \in V^*$ define:

$$L(\alpha) = \{ w \in \Sigma^* \mid \alpha \overset{*}{\Rightarrow} w \}.$$

The *language* of a *CFG* G, denoted by $L(G)$, is the set $L(S)$, where $S \in N$ is the start symbol of the grammar. A *CFG* G is said to be *unambiguous* if for any $w \in L(G)$ there is exactly one element $(w, \delta) \in l_G$.

For any string $\alpha \in V^*$ we use α^R to denote the *reverse* of α. If $\alpha \in V^*$ then $|\alpha|$ denotes the *length* of α. The symbol ε is reserved to denote the *empty string*,

that is, the string with length zero. For any $\alpha \in V^*$ and non-negative integer k we use $k : \alpha$ to denote the prefix of α with length k if $|\alpha| \geqq k$ and otherwise $k : \alpha$ denotes α. We use $N(\alpha)$ to denote the number of occurrences of nonterminal symbols in a string $\alpha \in V^*$. For any non-negative integer k and for any $\alpha \in V^*$ we define:

$$\text{FIRST}_k(\alpha) = \{ v \mid \alpha \overset{*}{\Rightarrow} w, w \in \Sigma^* \text{ and } k : w = v \}.$$

Moreover, for any $A \in N$ we define:

$$\text{FOLLOW}(A) = \{ \omega \mid S \underset{L}{\overset{*}{\Rightarrow}} w A \omega, w \in \Sigma^* \text{ and } \omega \in V^* \}.$$

If Q_1 and Q_2 are sets, then $Q_1 - Q_2 = \{ x \mid x \in Q_1 \text{ and } x \notin Q_2 \}$. The *empty set* is denoted by $\emptyset$.

DEFINITION 1.1: A *CFG* $G = (N, \Sigma, P, S)$ is said to be *regular* if $P \subseteq N \times (\Sigma \cup N\Sigma)$ or if $P \subseteq N \times (\Sigma \cup \Sigma N)$. A set $L \subseteq \Sigma^*$ is said to be regular if there exists a regular grammar G such that $L(G) = L$ or if $L = \{ \varepsilon \}$.

DEFINITION 1.2: Let $\pi = (B_1, B_2, \ldots, B_n\}$ denote a partition of Σ^*, where Σ is a finite set, into a finite number of n disjoint sets B_i. The elements of a partition are called *blocks*. Partition π is said to be a *regular partition* of Σ^* if all the sets B_i are regular. If two strings x and y belong to the same block B_i, then we write $x \equiv y \pmod{\pi}$. The partition π is said to be a *left congruence* (*right congruence*) if for any strings x, y and z in Σ^*, $x \equiv y \pmod{\pi}$ implies $zx \equiv zy \pmod{\pi}$ $(xz \equiv yz \pmod{\pi})$.

A partition $\pi' = \{ B'_1, B'_2, \ldots, B'_m \}$ is a *refinement* of the partition $\pi = \{ B_1, B_2, \ldots, B_n \}$ if each B_i of π is the union of some of the blocks of π' It is well-known that every regular partition of Σ^* has a refinement of finite index which is both a left and a right congruence (which we call a *congruence* for short) (*see* Hopcroft and Ullman [4]).

Now we are sufficiently prepared to present the definition of *LL*-regular grammars.

DEFINITION 1.3: Let $\pi = \{ B_1, B_2, \ldots, B_n \}$ be a regular partition of Σ^*. A *CFG* $G = (N, \Sigma, P, S)$ is said to be an *LL*(π) *grammar* if for each $w, x, y \in \Sigma^*$; $\alpha, \gamma, \delta \in V^*$ and $A \in N$, the conditions:

(i) $$S \underset{L}{\overset{*}{\Rightarrow}} w A \alpha \underset{L}{\Rightarrow} w \gamma \alpha \underset{L}{\overset{*}{\Rightarrow}} wx;$$

(ii) $$S \underset{L}{\overset{*}{\Rightarrow}} wA\alpha \underset{L}{\Rightarrow} w\delta\alpha \underset{L}{\overset{*}{\Rightarrow}} wy;$$

(iii) $$x \equiv y \pmod{\pi},$$

always imply that $\gamma = \delta$.

A *CFG* will be called *LL-regular* if it is $LL(\pi)$ for some regular partition π. A grammar $G=(N, \Sigma, P, S)$ is said to be $LL(k)$, where k is a non-negative integer, if G is $LL(\pi_k)$ for the regular partition:

$$\pi_k = \{\{u\} \mid u \in \Sigma^* \text{ and } |u| < k\} \cup \{\{uw \mid w \in \Sigma^*\} \mid u \in \Sigma^k\}.$$

Here Σ^k denotes the k-times Cartesian product of Σ with itself. Clearly, this definition coincides with the usual definition of an $LL(k)$ grammar (*see* e. g. Aho and Ullman [1]). Notice that if $k=1$ then:

$$\pi_k = \{\{\varepsilon\}\} \cup \{\{aw \mid w \in \Sigma^*\} \mid a \in \Sigma\},$$

and the condition $x \equiv y \pmod{\pi_k}$ in Definition 1.3 amounts to the condition $1 : x = 1 : y$.

In the forthcoming sections it is assumed that the grammars under consideration are *reduced*, that is, for each $X \in V$ there exists a derivation $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$, for some $\alpha, \beta \in V^*$ and $w \in \Sigma^*$.

The following definition of a cover homomorphism is taken from Nijholt [8]. However, here we restrict ourselves to left parses and left parse relations.

DEFINITION 1.4: Let $G'=(N', \Sigma', P', S')$ and $G=(N, \Sigma, P, S)$ be two *CFG*'s. Let l' be a subset of $l_{G'}$. A *partial cover homomorphism* $g_{l'} : l_{G'} \to l_G$ is defined by two homomorphims $\varphi : \Sigma'^* \to \Sigma^*$ and $\psi : \Delta_{G'}^* \to \Delta_G^*$ such that the following two conditions are satisfied:

(i) if $(w, \delta) \in l'$, then $(\varphi(w), \psi(\delta)) \in l_G$;

(ii) for any $(w, \delta) \in l_G$ there exists $(w', \delta') \in l'$ such that $(\varphi(w'), \psi(\delta')) = (w, \delta)$.

We say that $g_{l'} = \langle \varphi, \psi \rangle$ is a *total* cover homomorphism, or simply a cover homomorphism, whenever $l' = l_{G'}$. In that case we omit index l' from $g_{l'}$. If $(w, \delta) \in l'$, then $g_{l'}(w, \delta)$ denotes $(\varphi(w), \psi(\delta))$.

We now can describe various properties of (partial) cover homomorphisms.

DEFINITION 1.5: A partial cover homomorphism $g_{l'} : l_{G'} \to l_G$ is said to be *injective* if for any $(w_1, \delta_1) \in l'$ and $(w_2, \delta_2) \in l'$, if $g_{l'}(w_1, \delta_1) = g_{l'}(w_2, \delta_2)$, then $(w_1, \delta_1) = (w_2, \delta_2)$.

DEFINITION 1.6: A partial cover homomorphism $g_{l'} : l_{G'} \to l_G$ is said to be *properly injective* if for any $(w_1, \delta_1) \in l'$ and $(w_2, \delta_2) \in l'$.

(i) $\varphi(w_1) = \varphi(w_2$ implies $w_1 = w_2$, and,

(ii) $\psi(\delta_1) = \psi(\delta_2)$ implies $\delta_1 = \delta_2$.

Notice that if a partial cover homomorphism is properly injective then it is injective. Moreover, if a grammar G is unambiguous then $g_{l'}$ is injective implies that $g_{l'}$ is properly injective. We will deal only with unambiguous grammars. In this paper a (properly) injective (partial) cover homomorphism will be called *faithful.*

2. FROM *LL*-REGULAR TO *LL*(1) GRAMMARS

We now start the preliminaries to transform an *LL*-regular grammar to a covering *LL*(1) grammar. The transformation makes use of part of the construction of a "parsing table" for *LL*-regular grammars. We use a modified version of the construction which was presented in Nijholt [7]. The following definition is the key definition of this section.

DEFINITION 2.1: Let $G = (N, \Sigma, P, S)$ be a *CFG* and let $\pi = \{B_1, B_2, \ldots, B_n\}$ be a regular partition of Σ^*. For any $\alpha \in V^*$ define:

$$\text{BLOCK}(\alpha) = \{B_k \in \pi \mid L(\alpha) \cap B_k \neq \emptyset\}.$$

With this definition we can characterize *LL*-regular grammars as follows.

LEMMA 2.1: *Let π be a regular partition of Σ^* and let $G = (N, \Sigma, P, S)$ be a CFG. Grammar G is $LL(\pi)$ if and only if for each $w \in \Sigma^*$, $\alpha, \beta, \omega \in V^*$, $A \in N$ and productions $A \to \alpha$, $A \to \beta$ in P with $\alpha \neq \beta$, if $S \underset{L}{\overset{*}{\Rightarrow}} wA\omega$ then:*

$$\text{BLOCK}(\alpha\omega) \cap \text{BLOCK}(\beta\omega) = \emptyset.$$

Proof: Straightforward from the definitions. □

We recall (*cf.* Nijholt [7]) the definitions of the concatenation of blocks and the concatenation of sets of blocks.

DEFINITION 2.2: *Concatenation of blocks.* Let $\pi = \{B_1, B_2, \ldots, B_n\}$ be a regular partition of Σ^*.

Define:

$$B_i \square B_j = \{B_k \mid B_k \cap (B_i . B_j) \neq \emptyset\},$$

where $B_i . B_j$ stands for the usual concatenation of sets of strings.

The symbol $\square$ is also used to denote concatenation of sets of blocks.

DEFINITION 2.3: *Concatenation of sets of blocks.* Let $\pi = \{B_1, B_2, \ldots, B_n\}$ be a regular partition of Σ^*. Let $L_1, L_2 \subseteq \pi$. Define:

$$L_1 \square L_2 = \{B_k \mid B_k \in B_i \square B_j,\ B_i \in L_1 \text{ and } B_j \in L_2\}.$$

LEMMA 2.2: *Let $G=(N, \Sigma, P, S)$ be a CFG. Let $\pi = \{B_1, B_2, \ldots, B_n\}$ be a regular partition of Σ^* such that π is a congruence. If $\alpha, \beta \in V^*$, then:*

$$\text{BLOCK}(\alpha\beta) = \text{BLOCK}(\alpha) \square \text{BLOCK}(\beta).$$

Proof: See Nijholt [7]. $\square$

LEMMA 2.3= *Let $G=(N, \Sigma, P, S)$ be a CFG and let $\pi = \{B_1, B_2, \ldots, B_n\}$ be a regular partition of Σ^*. If $\alpha, \beta \in V^*$ and $\alpha \overset{*}{\Rightarrow} \beta$ then* $\text{BLOCK}(\beta) \subseteq \text{BLOCK}(\alpha)$.

Proof: Straight forward from the definition of BLOCK. $\square$

Now let $G=(N, \Sigma, P, S)$ be a *CFG* and let π be a regular partition of Σ^* such that π is a congruence. Since any regular partition of Σ^* has a refinement which is a congruence, this can be assumed without loss of generality. In what follows we have a more detailed discussion on the construction of the so-called $LL(\pi)$ functions than was presented in Nijholt [7].

For any $A \in N$ and $L \subseteq \pi$ such that $L = \text{BLOCK}(\omega)$ for some $\omega \in \text{FOLLOW}(A)$ we define a *partial function* $T_{A,L}$ on π as follows: For any $B \in \pi$,

$$T_{A,L}(B) = (A \to \alpha, \langle L_1, L_2, \ldots, L_m \rangle),$$

if $A \to \alpha$ is the (unique) production in P such that $\text{BLOCK}(\alpha) \square L$ contains B. We will show shortly that if G is $LL(\pi)$ then this production is unique. If $\alpha = x_0 C_1 x_1 C_2 \ldots C_m x_m$, where $m \geqq 0$, each $C_i \in N$ and $x_i \in \Sigma^*$, then:

$$L_j = \text{BLOCK}(x_j C_{j+1} \ldots C_m x_m) \square L, \qquad 1 \leqq j < m$$

and:

$$L_m = \text{BLOCK}(x_m) \square L.$$

LEMMA 2.4: *Let $G=(N, \Sigma, P, S)$ be an $LL(\pi)$ grammar, where π is a congruence. Let $A \in N$ and let $L \subseteq \pi$ such that $L = \text{BLOCK}(\omega)$ for some $\omega \in \text{FOLLOW}(A)$. For any $B \in \pi$ there is at most one element $(A \to \alpha, \langle L_1, L_2, \ldots, L_m \rangle)$ such that:*

$$T_{A,L}(B) = (A \to \alpha, \langle L_1, L_2, \ldots, L_m \rangle).$$

Proof: If $T_{A,L}(B)=(A\to\alpha, \langle L_1, L_2, \ldots, L_m\rangle)$ then $\text{BLOCK}(\alpha)\square L$ contains B.

Since $L=\text{BLOCK}(\omega)$ for some $\omega\in\text{FOLLOW}(A)$ it follows that there exists a derivation:

$$S \underset{L}{\overset{*}{\Rightarrow}} wA\omega,$$

and a production $A\to\alpha$ such that:

$$B\in\text{BLOCK}(\alpha)\square L=\text{BLOCK}(\alpha\omega).$$

(Notice that we assume that π is a congruence).

Suppose that there exists a production $A\to\beta$ in P such that $B\in\text{BLOCK}(\beta)\square L$. It follows that $B\in\text{BLOCK}(\beta\omega)$ and from Lemma 2.1 we must conclude that $\alpha=\beta$. Notice that we can not have:

$$(A\to\alpha, \langle L_1, L_2, \ldots, L_m\rangle)\neq(A\to\alpha, \langle L_1', L_2', \ldots, L_m'\rangle)$$

since the sets L_i, $1\leqq i\leqq m$ are uniquely determined by α and L. Hence, the lemma is satisfied. □

It follows that $T_{A,L}$ is well-defined if G is an LL-regular grammar. $T_{A,L}$ will be called an $LL(\pi)$ function. In the following algorithm the set of relevant $LL(\pi)$ functions is computed.

ALGORITHM 2.1: *The construction of $LL(\pi)$ functions.*

Input: an $LL(\pi)$ grammar $G=(N, \Sigma, P, S)$, where π is a congruence.

Output: the set $\mathcal{T}$ of $LL(\pi)$ functions for G.

Method:

(i) First construct $T_0=T_{S,\,\text{BLOCK}(\varepsilon)}$ and set $\mathcal{T}=\{T_0\}$.

(ii) For each function T in $\mathcal{T}$ and each $B\in\pi$ such that:

$$T(B)=(A\to x_0 C_1 x_1 C_2 \ldots C_m x_m, \langle L_1, L_2, \ldots, L_m\rangle)$$

add to $\mathcal{T}$ the functions T_{C_j, L_j}, $1\leqq j\leqq m$, if T_{C_j, L_j} is not already in $\mathcal{T}$.

(iii) Repeat step (ii) until no new functions can be added to $\mathcal{T}$.

Clearly, since N and π are finite sets the algorithm terminates.

LEMMA 2.5: *Let $G=(N, \Sigma, P, S)$ be an $LL(\pi)$ grammar, where π is a congruence. Let $\mathcal{T}$ be the set of $LL(\pi)$ functions which is constructed with Algorithm 2.1. For any $A\in N$ and $L\subseteq\pi$, function $T_{A,L}$ is in $\mathcal{T}$ if and only if there exists a derivation:*

$$S \underset{L}{\overset{*}{\Rightarrow}} w A \omega,$$

for some $w \in \Sigma^*$, $\omega \in V^*$ *and* $L = \text{BLOCK}(\omega)$.

Proof: Let $S \underset{L}{\overset{\delta}{\Rightarrow}} w A \omega$ for some $\delta \in \Delta_G^*$. The proof is by induction on $|\delta|$. If $|\delta| = 0$, then $A = S$, $\omega = \varepsilon$ and since $T_{S, \text{BLOCK}(\varepsilon)}$ is in $\mathscr{T}$ we conclude that the lemma is satisfied. Now assume that $|\delta| = n$ and the lemma holds for the derivations of length less than n. For any derivation $S \underset{L}{\overset{\delta}{\Rightarrow}} w A \omega$ there exist $u, v \in \Sigma^*$, $C \in N$, $\alpha_1, \alpha_2, \beta \in V^*$ and a production $C \to \alpha_1 A \alpha_2$ in P such that there exists a leftmost derivation:

$$S \underset{L}{\overset{*}{\Rightarrow}} u C \beta \underset{L}{\Rightarrow} u \alpha_1 \underline{A \alpha_2 \beta} \underset{L}{\overset{*}{\Rightarrow}} uv \underline{A \alpha_2 \beta} = w A \omega.$$

From the induction hypothesis we may conclude that there exists a set $L' \subseteq \pi$ and a function $T_{C, L'}$ in $\mathscr{T}$ with $L' = \text{BLOCK}(\beta)$. Since:

$$\text{BLOCK}(\alpha_1 A \alpha_2) \square L' = \text{BLOCK}(\alpha_1 A \alpha_2 \beta) \neq \emptyset,$$

there exists $B \in \pi$ such that:

$$T_{C, L'}(B) = (C \to \alpha_1 A \alpha_2, \langle L_1, \ldots, L_{N(\alpha_1)}, L_{N(\alpha_1 A)}, \ldots, L_{N(\alpha_1 A \alpha_2)} \rangle),$$

where:

$$L_{N(\alpha_1 A)} = \text{BLOCK}(\alpha_2) \square L' = \text{BLOCK}(\alpha_2 \beta).$$

It follows from step (ii) of Algorithm 2.1 that a function $T_{A, L}$ has been added to $\mathscr{T}$, with:

$$L = L_{N(\alpha_1 A)} = \text{BLOCK}(\alpha_2 \beta) = \text{BLOCK}(\omega),$$

which had to be proved.

If $T_{A, L} \in \mathscr{T}$ then we can use a straightforward induction on the order in which the $LL(\pi)$ functions are introduced in Algorithm 2.1 in order to prove that there exists a derivation $S \underset{L}{\overset{*}{\Rightarrow}} w A \omega$, with $L = \text{BLOCK}(\omega)$. This concludes the proof of Lemma 2.5. □

At this point we introduce a restriction on the set of productions which does not affect the generality of our observations. We assume that $LL(\pi)$ grammar $G=(N, \Sigma, P, S)$ is such that $P \subseteq N \times (\Sigma \cup \{\varepsilon\})\, N^*$. This "*normal form*" can be obtained as follows. Consider a production $i.\ A \to \alpha a \beta$ in P with $\alpha \neq \varepsilon$. We can replace this production by the productions:

$$i'.\ A \to \alpha H_a \beta,$$
$$i''.\ H_a \to a.$$

In this way a grammar $G'=(N', \Sigma, P', S)$ is obtained with $N'=N \cup \{H_a\}$ and $P'=(P-\{A \to \alpha a \beta\}) \cup \{A \to \alpha H_a \beta,\ H_a \to a\}$. It is not difficult to see that G' is also $LL(\pi)$ (for the same regular partition π). Moreover, there exists a cover homomorphism (*cf.* Definition 1.4) $g : l_{G'} \to l_{G'}$ where $\varphi : \Sigma^* \to \Sigma^*$ is the identity homomorphism and $\psi : \Delta_{G'}^* \to \Delta_G^*$ is defined by $\psi(k)=k$ for each production $k \in \Delta G$, $\psi(i')=i$ and $\psi(i'')=\varepsilon$. This process can be repeated until the normal form is obtained. Due to the transitivity of the cover relation it is possible to define a faithful cover homomorphism between the grammar in normal form and the original grammer. In the following algorithm the grammar in normal form will be used as input grammar.

ALGORITHM 2.2:

Input: an $LL(\pi)$ grammar $G=(N, \Sigma, P, S)$, where $\pi=\{B_1, B_2, \ldots, B_n\}$ is a congruence and $P \subseteq N \times (\Sigma \cup \{\varepsilon\})\, N^*$.

Output: an $LL(1)$ grammar $G'=(N', \Sigma', P', S')$.

Method: first construct with Algorithm 2.1 the set $\mathscr{T}$ of relevant $LL(\pi)$ functions. The symbols which denote these functions will be used as nonterminal symbols for G'. That is $N'=\mathscr{T}$ and $S'=T_{S,\,\mathrm{BLOCK}\,(\varepsilon)}$. Furthermore, $\Sigma'=\{[ai] \mid a \in \Sigma \text{ and } 1 \leqq i \leqq n\}$ and set P' is defined below. Initially, set $P'=\emptyset$. For each $T_{A,L} \in \mathscr{T}$ and $B_j \in \pi$, if:

$$T_{A,L}(B_j)=(A \to a\, C_1\, C_2 \ldots C_m, \langle L_1, L_2, \ldots, L_m \rangle),$$

then add:

$$T_{A,L} \to a'\, T_{C_1, L_1}\, T_{C_2, L_2} \ldots T_{C_m, L_m}$$

to P'. Here, $a'=\varepsilon$ if $a=\varepsilon$ and $a'=[aj]$ if $a \in \Sigma$.

The remainder of this section is devoted to the investigation of the properties of the grammar G' which is obtained in Algorithm 2.2. We introduce two homomorphisms:

(i) $\varphi : \Sigma'^* \to \Sigma^*$, where $\varphi([ai])=a$ for each $[ai] \in \Sigma'$, and,

(ii) $\psi : \Delta_{G'}^* \to \Delta_G^*$ where $\psi(i') = i$ for each production:

$$i' \,.\, T_{A,L} \to a' \, T_{C_1, L_1} \, T_{C_2, L_2} \ldots T_{C_m, L_m}$$

in P' which is obtained, with Algorithm 2.2, from a production $i \,.\, A \to a C_1 C_2 \ldots C_m$ in P.

It will turn out to be convenient to extend the domain of φ from Σ'^* to $(\Sigma' \cup N')^*$ by defining $\varphi(T_{A,L}) = A$ for each $T_{A,L} \in N'$. In the following lemmas we will refer to the grammars G and G' which are mentioned in Algorithm 2.2.

LEMMA 2.6: *Let* $T_{A,L} \in N'$, $\delta' \in \Delta_{G'}^*$ *and* $\alpha' \in V'^*$. *If* $T_{A,L} \underset{L}{\overset{\delta'}{\Rightarrow}} \alpha'$ *in* G', *then* $\varphi(T_{A,L}) \underset{L}{\overset{\delta}{\Rightarrow}} \varphi(\alpha')$ *in* G, *with* $\delta = \psi(\delta')$.

Proof: The proof is by induction on $|\delta'|$. If $|\delta'| = 0$ then $T_{A,L} = \alpha'$, $|\delta| = 0$ and $\varphi(T_{A,L}) = \varphi(\alpha') = A$. Now assume that the lemma is satisfied for all derivations with length less than n, $n \geqq 0$. We show that the lemma is also satisfied for derivations of length n. If $|\delta'| = n$, then we can factor the derivation into:

$$T_{A,L} \underset{L}{\overset{i'}{\Rightarrow}} a' \, T_{C_1, L_1} \, T_{C_2, L_2} \ldots T_{C_m, L_m} \underset{L}{\overset{\delta''}{\Rightarrow}} a' \, \alpha'_1 \, \alpha'_2 \ldots \alpha'_m,$$

where:

$$T_{C_j, L_j} \underset{L}{\overset{\delta'_j}{\Rightarrow}} \alpha'_j,$$

$1 \leqq j \leqq m$, $i' \, \delta'' = \delta'$, $\delta'' = \delta'_1 \, \delta'_2 \ldots \delta'_m$, $a' \in \Sigma' \cup \{\varepsilon\}$, $a' \, \alpha'_1 \, \alpha'_2 \ldots \alpha'_m = \alpha'$ and $i' \,.\, T_{A,L} \to a' \, T_{C_1, L_1} \, T_{C_2, L_2} \ldots T_{C_m, L_m}$ is the first production which is used in the leftmost derivation denoted by δ'. Since $|\delta'_j| < n$, $1 \leqq j \leqq m$, there exist derivations:

$$C_j \underset{L}{\overset{\delta_j}{\Rightarrow}} \varphi(\alpha'_j),$$

where $\delta_j = \psi(\delta'_j)$, $1 \leqq j \leqq m$ and there exists a production:

$$\psi(i') \,.\, A \to \varphi(a') \, C_1 \, C_2 \ldots C_m.$$

Therefore we have a derivation:

$$A \underset{L}{\overset{\delta}{\Rightarrow}} \varphi(\alpha'),$$

with $\delta = \psi(\delta')$, which had to be proved. □

An immediate consequence of this lemma is that if α' and β' are in V'^* and $\alpha' \underset{L}{\overset{*}{\Rightarrow}} \beta'$ in G' then $\varphi(\alpha') \underset{L}{\overset{*}{\Rightarrow}} \varphi(\beta')$ in G.

LEMMA 2.7: *Let* $A \in N$, $\alpha \in V^*$, $\omega \in \text{FOLLOW}(A)$ *and* $\delta \in \Delta_G^*$. *If* $A \underset{L}{\overset{\delta}{\Rightarrow}} \alpha$ *in* G *then there exists a derivation* $T_{A,L} \underset{L}{\overset{\delta'}{\Rightarrow}} \alpha'$ *in* G', *with* $\alpha' \in V'^*$, $\delta' \in \Delta_{G'}^*$ *and* $L = \text{BLOCK}(\omega)$ *such that* $\psi(\delta') = \delta$ *and* $\varphi(\alpha') = \alpha$.

Proof: The proof is by induction on $|\delta|$. If $|\delta| = 0$ then $\delta = \varepsilon$ and $A = \alpha$. It follows from Lemma 2.5 that there exists $T_{A,L}$ in N' with $L = \text{BLOCK}(\omega)$ and there exists a derivation $T_{A,L} \underset{L}{\overset{\delta'}{\Rightarrow}} \alpha'$ with $\alpha' = T_{A,L}$, $\varphi(\alpha') = \alpha = A$, $\delta' = \varepsilon$ and $\psi(\delta') = \delta = \varepsilon$. Now let $A \underset{L}{\overset{\delta}{\Rightarrow}} \alpha$ in G with $|\delta| = n$. We assume that the lemma is satisfied for derivations with length less than n. Factor $A \underset{L}{\overset{\delta}{\Rightarrow}} \alpha$ into:

$$A \overset{i}{\Rightarrow} a C_1 C_2 \ldots C_m \overset{*}{\Rightarrow} a \gamma_1 \gamma_2 \ldots \gamma_m = \alpha,$$

with:

$$C_j \underset{L}{\overset{\delta_j}{\Rightarrow}} \gamma_j,$$

$1 \leqq j \leqq m$. We may again (Lemma 2.5) assume the existence of a nonterminal $T_{A,L}$ with $L = \text{BLOCK}(\omega)$ and there exists $B \in \text{BLOCK}(a C_1 C_2 \ldots C_m \omega)$ such that:

$$T_{A,L}(B) = (A \to a C_1 C_2 \ldots C_m, \langle L_1, L_2, \ldots, L_m \rangle).$$

Therefore we have a production:

$$i' . T_{A,L} \to a' T_{C_1,L_1} T_{C_2,L_2} \ldots T_{C_m,L_m},$$

with $\psi(i')=i$, $\varphi(a')=a$ and:

$$L_j=\text{BLOCK}(C_{j+1}\ldots C_m)\,\square\, L=\text{BLOCK}(C_{j+1}\ldots C_m\,\omega).$$

From the induction hypothesis it follows that there exist derivations:

$$T_{C_j,\,L_j}\underset{L}{\overset{\delta'_j}{\Rightarrow}}\gamma'_j,$$

with $\psi(\delta'_j)=\delta_j$ and $\varphi(\gamma'_j)=\gamma_j$, $1\leqq j\leqq m$.

Hence, there exists a derivation:

$$T_{A,\,L}\underset{L}{\overset{\delta'}{\Rightarrow}}\alpha',$$

with:

$$\delta'=i'\,\delta'_1\,\delta'_2\ldots\delta'_m,\ \psi(\delta')=i\,\delta_1\,\delta_2\ldots\delta_m=\delta,$$

$$\alpha'=a'\,\gamma'_1\,\gamma'_2\ldots\gamma'_m \qquad \text{and} \qquad \varphi(\alpha')=a\,\gamma_1\,\gamma_2\ldots\gamma_m=\alpha.$$

This concludes the proof of Lemma 2.7. □

Now we are sufficiently prepared to show that grammar G' covers grammar G.

LEMMA 2.8: *Let G and G' be as in Algorithm 2.2. There exists a cover homomorphism $g: l_{G'}\to l_G$.*

Proof: If $(w',\delta')\in\lambda_{G'}$ then it follows from Lemma 2.6 that $(\varphi(w'),\psi(\delta'))\in l_G$. Conversely, if $(w,\delta)\in l_G$ then it follows from Lemma 2.7 that there exists $(w',\delta')\in\lambda_{G'}$ such that $(\varphi(w'),\psi(\delta'))=(w,\delta)$. Therefore, $g=\langle\varphi,\psi\rangle$ is a (total) cover homomorphism. □

We now want to show that G' of Algorithm 2.2 is an $LL(1)$ grammar. In the following lemma we have a rather obvious but useful observation.

LEMMA 2.9: *If $S'\underset{L}{\overset{*}{\Rightarrow}}w'\,T_{A,\,L}\,\omega'$ in G', then $L=\text{BLOCK}(\varphi(\omega'))$.*

Proof: Consider a derivation $S'\underset{L}{\overset{\delta}{\Rightarrow}}w'\,T_{A,\,L}\,\omega'$ in G'. The proof of the lemma is by induction on $|\delta|$. If $|\delta|=0$, then $w'=\omega'=\varepsilon$, $T_{A,\,L}=T_{S,\,\text{BLOCK}(\varepsilon)}$ and we have indeed that $L=\text{BLOCK}(\varphi(\omega'))=\text{BLOCK}(\varepsilon)$. Now consider a derivation $S'\underset{L}{\overset{\delta}{\Rightarrow}}w'\,T_{A,\,L}\,\omega'$ with $|\delta|=n$. Assume the lemma is satisfied for derivations

with length less than n. We can factor the derivation into:

$$S' \underset{L}{\overset{*}{\Rightarrow}} u' T_{C,L'} \gamma' \underset{L}{\Rightarrow} u' \alpha' \underline{T_{A,L} \beta' \gamma'} \underset{L}{\overset{*}{\Rightarrow}} u' v' \underline{T_{A,L} \beta' \gamma'} = w' T_{A,L} \omega',$$

for some $u', v' \in \Sigma'^*$, $T_{C,L'} \in N'$ and $\alpha', \beta', \gamma' \in V'^*$.

From the induction hypothesis it follows that $L' = \text{BLOCK}(\varphi(\gamma'))$. From the construction of G' it follows that $L = \text{BLOCK}(\varphi(\beta')) \square L'$. That is, $L = \text{BLOCK}(\varphi(\beta'\gamma')) = \text{BLOCK}(\varphi(\omega'))$, which had to be proved. □

LEMMA 2.10: *Let* $w' \in \Sigma'^*$, $T_{A,L} \in N'$, $\alpha, \omega' \in V'^*$ *and* $T_{A,L} \to \alpha'$ *in* P' *such that:*

$$S' \underset{L}{\overset{*}{\Rightarrow}} w' T_{A,L} \omega' \underset{L}{\Rightarrow} w' \alpha' \omega',$$

in G'. *If* $[ai] \in \text{FIRST}_1(\alpha'\omega')$ *then* $B_i \in \text{BLOCK}(\varphi(\alpha'\omega'))$.

Proof: We distinguish between two cases. First assume that $[ai] = 1 : \alpha'\omega'$. Notice that due to the normal form of the grammar we can not have $\alpha' = \varepsilon$ and $[ai] = 1 : \omega'$. Therefore, production $T_{A,L} \to \alpha'$ is of the form $T_{A,L} \to [ai]\alpha''$ for some $\alpha \in V'^*$. From the construction of P' it follows that $B_i \in \text{BLOCK}(\varphi(\alpha')) \square L$. Since $L = \text{BLOCK}(\varphi(\omega'))$ (*see* Lemma 2.9) we have that $B_i \in \text{BLOCK}(\varphi(\alpha'\omega'))$. Now consider the case that $1 : \alpha'\omega'$ is a non-terminal symbol. Then there exists a derivation:

$$\alpha'\omega' \underset{L}{\overset{*}{\Rightarrow}} T_{C,L'}\omega'' \underset{L}{\Rightarrow} [ai]\gamma'\omega'',$$

where $T_{C,L'} \to [ai]\gamma'$ is a production in P' and $\omega'', \gamma' \in V'^*$. It follows from the construction of P' that $B_i \in \text{BLOCK}(\varphi([ai]\gamma')) \square L'$ and since $L' = \text{BLOCK}(\varphi(\omega''))$ (*see* Lemma 2.9) we have that $B_i \in \text{BLOCK}(\varphi([ai]\gamma'\omega''))$. Since $\alpha'\omega' \underset{L}{\overset{*}{\Rightarrow}} [ai]\gamma'\omega''$ we have also (*cf.* the remark which follows Lemma 2.6) $\varphi(\alpha'\omega') \underset{L}{\overset{*}{\Rightarrow}} \varphi([ai]\gamma'\omega'')$ and from Lemma 2.3 it follows that $\text{BLOCK}(\varphi([ai]\gamma'\omega'')) \subseteq \text{BLOCK}(\varphi(\alpha'\omega'))$. Since $B_i \in \text{BLOCK}(\varphi([ai]\gamma'\omega''))$ it follows that $B_i \in \text{BLOCK}(\varphi(\alpha'\omega'))$, which had to be proved. □

LEMMA 2.11: *Let* G' *be the CFG which is obtained in Algorithm* 2.2. *Grammar* G' *is* $LL(1)$.

Proof: For any derivation $S' \underset{L}{\overset{*}{\Rightarrow}} w' T_{A,L} \omega'$ in G' with $w' \in \Sigma'^*$, $\omega' \in V'^*$ and $T_{A,L} \in N'$, we have to show that if $T_{A,L} \to \alpha'$ and $T_{A,L} \to \beta'$ are in P' then $\text{FIRST}_1(\alpha'\omega') \cap \text{FIRST}_1(\beta'\omega') \neq \emptyset$ implies $\alpha' = \beta'$. Suppose that $[ai] \in \text{FIRST}_1(\alpha'\omega') \cap \text{FIRST}_1(\beta\omega')$. Then there exists $B_i \in \pi$ (the regular partition of input grammar G) such that $B_i \in \text{BLOCK}(\varphi(\alpha'\omega'))$ and $B_i \in \text{BLOCK}(\varphi(\beta'\omega'))$ (*cf.* Lemma 2.10). Since $L = \text{BLOCK}(\varphi(\omega'))$ we have also $B_i \in \text{BLOCK}(\varphi(\alpha')) \square L$ and $B_i \in \text{BLOCK}(\varphi(\beta')) \square L$. From the construction of P' it follows that:

$$T_{A,L}(B_i) = (A \to \varphi(\alpha'), \langle L_1, L_2, \ldots, L_m \rangle)$$

and:

$$T_{A,L}(B_i) = (A \to \varphi(\beta'), \langle L'_1, L'_2, \ldots, L'_{m'} \rangle).$$

Since G is $LL(\pi)$ we must conclude that $\varphi(\alpha') = \varphi(\beta')$ and

$$\langle L_1, L_2, \ldots, L_m \rangle = \langle L'_1, L'_2, \ldots, L'_{m'} \rangle$$

It follows that $\alpha' = \beta'$ which had to be proved. $\square$

We mention in passing that although the above obtained results may clarify the properties of LL-regular grammars, we do not need them to be able to parse LL-regular languages. The parsing method which is in Nijholt [7] is, after a "regular pre-scan" on the input string has been performed, an $LL(1)$ parsing method. In fact, $L(G')$, where G' is obtained with Algorithm 2.2, is a superset of the set of strings which are obtained with this regular pres-can. In view of these remarks we will further investigate G' and cover homomorphism $g = \langle \varphi, \psi \rangle$. See Section 3 for further details on the parsing of LL-regular languages.

Unfortunately, cover homomorphism $g : l_{G'} \to l_G$ for which G' covers G (*cf.* Lemma 2.9) is not a faithful cover homomorphism. That is, it is possible that for some (w_1, δ_1) and (w_2, δ_2) in $l_{G'}$ with $(w_1, \delta_1) \neq (w_2, \delta_2)$, the homomorphisms φ and ψ are such that $(\varphi(w_1), \psi(\delta_1)) = (\varphi(w_2, \psi(\delta_2))$. However, there exists a natural subset of $l_{G'}$ such that the restriction of $g = \langle \varphi, \psi \rangle$ to this subset is properly injective. In what follows we shall characterize this subset. The following lemma is necessary.

LEMMA 2.12: *Let* $S' \underset{L}{\overset{*}{\Rightarrow}} w' T_{A,L} \omega'$. *For each* $v' \in L(\omega')$, *if we have a derivation* $T_{A,L} \overset{*}{\Rightarrow} u'$ *then there exists a string* $u'' \in \Sigma'^*$, *with* $u'' = \varepsilon$ *or:*

$$u'' = [a_1 i_1][a_2 i_2] \ldots [a_p i_p]$$

for some $p \geqq 1$, *which satisfies* $T_{A,L} \overset{*}{\Rightarrow} u''$, $\varphi(u'') = \varphi(u')$ *and*

$$\varphi([a_j i_j] \ldots [a_p i_p] v') \in B_{i_j},\ 1 \leqq j \leqq p.$$

Proof: The proof is by induction on the length of the derivation $T_{A,L} \overset{*}{\Rightarrow} u'$. If this length is 1 then $u' = \varepsilon$, in which case the lemma is trivially satisfied, or $u' = [a_1 k]$ for some $[a_1 k] \in \Sigma'$. In the latter case, consider a string $v' \in L(\omega')$ and suppose that $\varphi([a_1 k] v') \in B_{i_1}$ where B_{i_1} is a block of regular partition π. It follows (*cf.* Lemma 2.3) that $B_{i_1} \in \text{BLOCK}(\varphi([a_1 k]\, \omega')) = \text{BLOCK}(a_1) \square L$. From the construction of P' in Algorithm 2.2 it follows that $T_{A,L} \to [a_1 i_1]$ is in P'. Therefore, with $u'' = [a_1 i_1]$ we have that $\varphi(u'') = \varphi(u') = a_1$ and $\varphi([a_1 i_1] v')$ is in B_{i_1}, which had to be proved.

Now consider the case that the length of the derivation is greater than 1. If $u' = \varepsilon$ then the lemma is trivially satisfied. Assume $|u'| > 0$. We can factor the derivation $T_{A,L} \overset{*}{\Rightarrow} u'$ into:

$$T_{A,L} \Rightarrow a' T_{C_1,L_1} T_{C_2,L_2} \ldots T_{C_m,L_m} \overset{*}{\Rightarrow} a' u'_1 u'_2 \ldots u'_m.$$

Now consider a string $v' \in L(\omega')$. We can write:

$$S' \underset{L}{\overset{*}{\Rightarrow}} w' a' u'_1 \ldots u'_{k-1} T_{C_k,L_k} \ldots T_{C_m,L_m} \omega',$$

with $T_{C_k,L_k} \overset{*}{\Rightarrow} u'_k$, $1 \leqq k \leqq m$. Hence, for $u''_{k+1} \ldots u''_m v'$ in $L(T_{C_{k+1},L_{k+1}} \ldots T_{C_m,L_m} \omega')$, where k descends from $k = m-1$ until $k = 1$, we can obtain by using the induction hypothesis a string u''_k such that u''_k satisfies the lemma. If $a' = \varepsilon$ then we are done. Otherwise, if $\varphi(a' u''_1 u''_2 \ldots u''_m v')$ is in B_{i_1}, then $B_{i_1} \in \text{BLOCK}(\varphi(a' u''_1 u''_2 \ldots u''_m v'))$ and, because of Lemma 2.3,

$$B_{i_1} \in \text{BLOCK}(\varphi(a' T_{C_1,L_1} \ldots T_{C_m,L_m} \omega'))$$

$$= \text{BLOCK}(\varphi(a' T_{C_1,L_1} \ldots T_{C_m,L_m})) \square L.$$

From the construction of P' it follows that:

$$T_{A,L} \to [a_1 i_1]\, T_{C_1,L_1} \ldots T_{C_m,L_m},$$

is a production in P'. Hence,

$$T_{A,L} \overset{*}{\Rightarrow} a' u''_1 u''_2 \ldots u''_m$$

and $a' u_1'' u_2'' \ldots u_m''$ satisfies the lemma. This concludes the induction proof. □

LEMMA 2.13: *Let G and G' be as in Algorithm 2.2. For each string $u \in L(G)$ there exists a string $u' \in L(G')$ such that $\varphi(u') = u$, and if:*

$$u' = [a_1 i_1][a_2 i_2] \ldots [a_m i_m],$$

then:

$$\varphi([a_j i_j] \ldots [a_m i_m]) \in B_{i_j}, \qquad 1 \leqq j \leqq m.$$

Proof: If $u \in L(G)$, then there exists (*cf.* Lemma 2.7) a derivation $T_{S,\,\text{BLOCK}\,(\varepsilon)} \overset{*}{\Rightarrow} u'$ in G', such that $\varphi(u') = u$. From Lemma 2.12 it follows (with $T_{A,\,L} = T_{S,\,\text{BLOCK}\,(\varepsilon)}$ and $w' = \omega' = \varepsilon$) that we can choose u' in such a way that the lemma is satisfied. □

From Lemma 2.13 we may now conclude the existence of a subset L_π of $L(G')$ which is defined as follows:

$$L_\pi = \{ w' \in L(G') \mid w' = \varepsilon \text{ or } w' = [a_1 i_1] \ldots [a_m i_m]$$

for some $m \geqq 1$, and:

$$\varphi([a_j i_j] \ldots [a_m i_m]) \in B_{i_j},\ 1 \leqq j \leqq m \}$$

and we can define:

$$l_\pi = \{(w', \delta') \in l_{G'} \mid w' \in L_\pi \}.$$

Notice, that if $(w_1', \delta_1') \in l_\pi$ and $(w_2', \delta_2') \in l_\pi$, then $w_1' \neq w_2'$ implies $\varphi(w_1') \neq \varphi(w_2')$. Grammer G' is an $LL(1)$ grammar. Therefore G' is unambiguous and we can conclude that $g_{l_\pi} : l_{G'} \to l_G$ is a faithful (partial) cover homomorphism. Here, as before, g is defined by the homomorphisms φ and ψ which were introduced immediately following Algorithm 2.2.

3. ON THE PARSING OF LL-REGULAR GRAMMARS

In Nijholt [7] and Poplawski [11] $LL(1)$-type parsing methods for LL-regular grammars have been developed. The first step of such a parsing method consists of a pre-scan of a potential sentence with a generalized sequential machine. Then an $LL(1)$ table driven parser can be used. That is, suppose we have an LL-regular grammar G with a regular partition π. The pre-scan

performed on the sentences of G yields the set of strings L_π, which is defined immediately after the proof of Lemma 2.13. Then, for L_π it is possible to use an $LL(1)$ table driven parser. This means that there exists a practical parsing method for LL-regular grammars.

However, from a more theoretical point of view several questions remain. For example, is the set L_π an $LL(1)$ language ? Does there exist a transformation from G an $LL(1)$ grammar G_π such that $L(G_\pi) = L_\pi$? Moreover, there exists a theory of grammatical covers. In this theory we often consider transformations between grammars which give rise to a simple relation between the parses of the sentences of the grammars. Therefore, parsing with respect to the newly obtained grammar is "as good" as parsing with respect to the original grammar. In [7, 9] we have the situation that instead of an LL-regular parsing method we use, by changing the language, an $LL(1)$ parsing method. Now it is interesting to investigate how in this case we can use the cover formalism in order to describe the relations between the sentences and the parses of the original LL-regular grammar G and the strings and parses wich are dealt with by the $LL(1)$ parsing method. Hence, what is the relation, in terms of grammatical covers, between the $LL(1)$-type parser for L_π and the original grammar G and the language $L(G)$? This relation has been studied in detail in Section 2. It should be mentioned that there exists a simple construction which for any context-free grammar G and sequential machine yields a context-free grammar for the intersection of the context-free language $L(G)$ and the regular set which is defined by the sequential machine.

Unfortunately, this construction does not necessarily yield an $LL(1)$ grammar if it is applied to an LL-regular grammar and the (generalized) sequential machine which performs the pre-scan. In Nijholt [9] some counter-examples can be found.

Now, what we have shown in the previous section is that we are able to give an algorithm which transforms an LL-regular grammar G into an $LL(1)$ grammar G' and, moreover, we are able to describe the relation between G and G' with a cover-homomorphism. In order to give a complete description of the relations between G and G' and between $L(G)$, $L(G')$ and L_π, we have to use the general framework for grammatical covers as it has been developed in [8]. The results of this paper are illustrated with Figure 1.

In Figure 1 we have the situation where $LL(\pi)$ grammar G is transformed into an $LL(1)$ grammar G'. Each sentence w' of $L(G')$ can be parsed with an $LL(1)$ parsing method for grammar G'. The parse δ' can be mapped on the corresponding parse $\psi(\delta')$ of $\varphi(w')$.

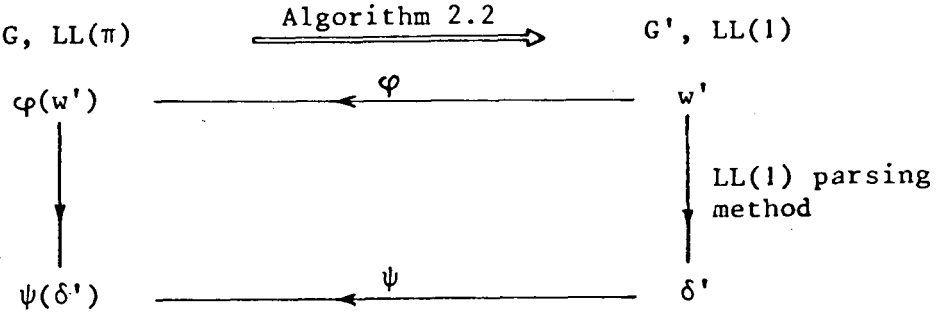

Figure 1. – Covers for LL-regular grammars.

Obviously, if we start with a potential sentence w of grammar G we need to have a device which converts w into w' before we can start parsing. Analogous to what has been done in [7] we can perform a regular pre-scan of w with a generalized sequential machine which reads w from right to left. This machine attaches to each symbol of w the index of the block of the partition π to which the string read so far, belongs. That is, we obtain the elements of the set L_π which has been defined above. Therefore it should be noted that the results of this paper are not to be understood as a new parsing method for LL-regular grammars but as an attempt to understand and to describe the relation between the original LL-regular grammar and the $LL(1)$-type parsing method which can be used.

REFERENCES

1. A. V. AHO and J. D. ULLMAN, *The Theory of Parsing, Translation and Compiling*, Vols. I and II, Prentice Hall, Inc., Englewood Cliffs, N. J., 1972 and 1973.
2. J. N. GRAY and M. A. HARRISON, *On the Covering and Reduction Problems for Context-Free Grammars*, J. Assoc. Comput. Mach., Vol. 19, 1972, pp. 675-697.
3. M. A. HARRISON, *Introduction to Formal Language Theory*, Addison-Wesley, Reading, Mass., 1978.
4. J. E. HOPCROFT and J. D. ULLMAN, *Formal Languages and their Relation to Automata*, Addison-Wesley, Reading, Mass., 1969.
5. S. JARZABEK and T. KRAWCZYK, *LL-Regular Grammars*, Information Processing Letters, Vol. 4, 1975, pp. 31-37.
6. A. NIJHOLT, *On the Parsing of LL-Regular Grammars*, Proc. of the 5th Sympos. On the *Mathematical Foundations of Computer Science*, A. MAZURKIEWICZ, Ed., Lect. Notes in Comput. Science, Vol. 45, Springer, Berlin, 1976, pp. 446-452.
7. A. NIJHOLT, *LL-Regular Grammars*, Int. J. of Computer Mathematics, Vol. 8, 1980, pp. 303-318.
8. A. NIJHOLT, *Context-Free Grammars: Covers, Normal Forms, and Parsing*, Lect. Notes in Comput. Science, Vol. 93, Springer, Berlin, 1980.

9. A. NIJHOLT, *The Equivalence Problem for LL- and LR-Regular Grammars*, Proc. of the 3rd Sympos. On *Fundamentals of Computation Theory*, Lect. Notes in Comput. Science 117, M. CHYTIL and J. GRUSKA, Ed., Springer, Berlin, 1981, pp. 291-300.
10. D. A. POPLAWSKI, *Error Recovery for Extended LL-Regular Parsers, Ph. D. Thesis*, Purdue University, August 1978.
11. D. A. POPLAWSKI, *On LL-Regular Grammars*, J. Comput. System Sc., Vol. 18, 1979, pp. 218-227.
12. D. WOOD, *Lecture Notes on Top-Down Syntax Analysis*, J. of the Computer Society of India, Vol. 8, 1978, pp. 1-22.