On Covers and Left-Corner Parses

Ricks op den Akker

Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

A transformation is defined which is a modification of a classic transformation on context-free grammars. By means of this transformation, a proof is presented of the fact that any cycle-free context-free grammar can be left-to-left-corner covered by a non-left-recursive grammar. The proof method is based on the idea to transform the characteristic grammar associated with the simple syntax-directed translation scheme which defines the left-corner parse of the strings generated by the input grammar of the scheme. It is shown that the transformation yields an $LL(k)$ grammar if and only if it is applied to an $LC(k)$ grammar. Finally, some ideas are presented to extend the theory of covers to the semantical covering of attribute grammars.

1. Introduction

Let $G_1$ and $G_2$ be context-free grammars. $G_1$ and $G_2$ are called weakly equivalent if they generate the same context-free language. Let $x$ denote a string in the language $L = L(G_1) = L(G_2)$. Weak equivalence of $G_1$ and $G_2$ does not imply that there is a structural similarity between the parse tree or parse trees of $x$ with respect to $G_1$ and the parse tree or parse trees of $x$ with respect to $G_2$. Here we will consider a stronger form of equivalence between context-free grammars. We say that $G_2$ covers $G_1$ if we can transform $G_1$ into the weakly equivalent grammar $G_2$ and we can systematically find the parse tree of each string $x$ in $L$ with respect to $G_1$ from the parse tree of $x$ with respect to $G_2$.

A class $Y$ of context-free grammars covers the class $X$ of context-free grammars if there is a transformation defined on all grammars in $X$ such that the transformed grammar is a grammar in class $Y$ and covers the original grammar. If a class $Y$ covers a class $X$ then we can use a compiler writing system (CWS) based on a parsing method for grammars of class $Y$ also for grammars in class $X$; see Figure 1.

First, we transform a grammar $G$ from class $X$ into the covering grammar $G'$ and then built a parser for $G'$. The output of this parser is a parse tree with respect to $G'$. This tree is transformed into the corresponding parse tree with respect to the original grammar $G$. This last transformation makes it possible to define the semantics of $x$ with respect to the semantical definition based on grammar $G$. 

107
For this reason the original grammar on which the syntax-directed semantics is based is called *semantic grammar* in [7] and the grammar obtained after transformation(s) is called the *parsing grammar*. The reason for using two grammars may be that the parsing grammar is not convenient for expressing semantics. Sometimes it is possible and also useful to do semantic actions or — in terms of the formalism of attribute grammars [5] — to evaluate attributes during the parsing of the input string. This allows for the use of semantic properties (attribute values) of the already analyzed part of the input string in making parsing decisions. Then it becomes practically interesting to adapt the transformations on context-free grammars in such a way that they preserve the semantics defined by the original (attribute) grammar.

We will define a transformation on context-free grammars. If this transformation is applied to a context-free grammar $G$, then the resulting grammar $G'$ covers the original grammar $G$. If the transformation is applied to an $LC(k)$ grammar, then the resulting grammar is $LL(k)$. The transformation is a modification of the one given by Rosenkrantz and Lewis II [9]. The modification is due to the fact that our definition of the $LC(k)$ grammars is that of Soisalon-Soininen [11], which is a modification of the original definition of Rosenkrantz and Lewis II. The equivalence of this modified definition and the definition of $LC(k)$ grammars of Soisalon-Soininen and Ukkonen is proved in [2].
This paper is organized as follows. In Section 2 we give some preliminary definitions concerning parse relations and cover relations between classes of context-free grammars. In Section 3 we define the left-corner parse relation. In Section 4 we define a transformation on context-free grammars. In Section 5 we consider general cover properties of the transformation. Section 6 presents definitions of $LL(k)$ grammars and left-corner or $LC(k)$ grammars. In Section 7 we show that the transformation yields an $LL(k)$ grammar if and only if it is applied to an $LC(k)$ grammar. Finally, we present in Section 8 some ideas to extend the notion of cover to the semantical covering of attribute grammars.

2. Preliminaries on Covers

The first results on cover relations between context-free grammars were obtained by workers in the area of compiler writing. One of the first theoretical studies in this field is that of Gray and Harrison [6]. A general theory of covers has been developed by Nijholt [8].

We recall some preliminary notions from this theory of covers. Since we will only consider cover relations between grammars that have the same terminal alphabets, we do not need all notions in the general formulation presented by Nijholt.

$<w,G>$ denotes the degree of ambiguity of $w$ with respect to $G$. Let $G$ be a cfg (context-free grammar) and $\Delta_G$ a set of unique labeling symbols for the productions of $G$.

**Definition 2.1.** A relation $f_G \subseteq \Sigma^* \times \Delta_G^*$ is a parse relation for $G$ if it satisfies the following conditions.

i. For each string $w \in L(G)$ there exists at least one element $(w, \pi) \in f_G$.

ii. For each $w \in \Sigma^*$, $|\{\pi \mid (w, \pi) \in f_G\}| \leq <w,G>$.

**Definition 2.2.** A relation $f_G \subseteq \Sigma^* \times \Delta_G^*$ is a proper parse relation for $G$ if it satisfies the following conditions.

i. If $(w, \pi) \in f_G$ and $(w', \pi) \in f_G$ then $w = w'$.

ii. For each $w \in \Sigma^*$, $|\{\pi \mid (w, \pi) \in f_G\}| = <w,G>$.

Thus a proper parse relation for $G$ is a parse relation for $G$.

**Definition 2.3.** Let $G = (N, \Sigma, P, S)$ and $G' = (N', \Sigma, P', S')$ be context-free grammars with labeling sets for the productions $\Delta_G$ and $\Delta_{G'}$. Let $f_G$ and $f_{G'}$ be parse relations for $G$ and $G'$, respectively. A homomorphism $\phi: \Delta_G \rightarrow \Delta_{G'}$ is a parse homomorphism if $(w, \pi) \in f_{G'}$ implies $(w, \phi(\pi)) \in f_G$.

**Definition 2.4.** A parse homomorphism is a cover homomorphism if for all $(w, \pi) \in f_G$, there exists $(w, \pi') \in f_{G'}$ such that $\phi(\pi') = \pi$. 


Definition 2.5. Let $G = (N, \Sigma, P, S)$ and $G' = (N', \Sigma, P', S')$ be cfgs. Let $f_G$ and $f_{G'}$ be parse relations for $G$ and $G'$ respectively. Grammar $G'$ is $f_{G'}$-to-$f_G$ covers $G$ if there exists a cover homomorphism $\phi : \Delta_{G'} \rightarrow \Delta_G$. Notation: $G' \models f_{G'} f_G \models G$. □

Two well-known parse relations are the following. The left parse relation for $G$ is $I_\ell = \{(w, \pi) \mid S \Rightarrow^* w \}$. The right parse relation for $G$ is $I_r = \{(w, \pi^R) \mid S \Rightarrow^* w \}$, in which $\pi^R$ denotes the reverse of $\pi$.

The cover relation satisfies the transitivity property. Let $f, g, h$ be parse relations for cfgs $F, G, H$, respectively. If $F[f/g]G$ with respect to cover homomorphism $\phi_1$, and $G[g/h]H$ with respect to cover homomorphism $\phi_2$ then $F[f/h]H$ with respect to cover homomorphism $\phi_2 \circ \phi_1$.

Most results on covers between cfgs concern left-to-left (or simply left), right-to-right (right), left-to-right and right-to-left covers by grammars in some normal form, as for example Greibach Normal Form or Chomsky Normal Form, by grammars without $e$-rules or by non-left-recursive grammars. All the results obtained upto 1980 can be found in Nijholt [8].

3. The Left-Corner Parse

Before we come to the definition of the left-corner parse of a string with respect to a given context-free grammar, we define a useful homomorphism. Let $A$ be a set of symbols and $\Sigma \subseteq A$. The $\Sigma$-erasing homomorphism on $A$, $h_\Sigma : A^* \rightarrow A^*$ is defined by $h_\Sigma(a) = a$ if $a \notin \Sigma$ and $h_\Sigma(a) = \epsilon$ if $a \in \Sigma$. For a language $L$ we define $h_\Sigma(L) = \{ h_\Sigma(x) \mid x \in L \}$.

Let $G = (N, \Sigma, P, S)$ be a cfg, $|P| = m$, $\Delta$ a set $\{p_1, \ldots, p_m\}$ such that $\Sigma \cap \Delta = \emptyset$ and $\lambda_G : P \rightarrow \Delta$ a labeling function associating with each production in $P$ a unique symbol in $\Delta$. We will omit the subscript $G$ and simply write $\lambda$ instead of $\lambda_G$.

With each cfg $G$ and label set $\Delta$ we associate the cfg $G_{lc} = (N, \Sigma \cup \Delta, P_{lc}, S)$ in which $P_{lc}$ is defined as follows.

$P_{lc} = \{ A \rightarrow p_i \mid A \rightarrow \epsilon \text{ in } P \text{ and } \lambda(A \rightarrow \epsilon) = p_i \} \cup \{ A \rightarrow Xp_i \alpha \mid A \rightarrow X\alpha \text{ in } P \text{ and } \lambda(A \rightarrow X\alpha) = p_i \}$

Clearly, $h_\Delta(L(G_{lc})) = L(G)$.

We use the grammar $G_{lc}$ in order to define the left-corner parse of a string $x \in L(G)$ with respect to $G$.

Definition 3.1. Let $G$ be a cfg, $x \in L(G)$ and $\Delta, G_{lc}$ as defined above. $\pi \in \Delta^*$ is a left-corner parse of $x$ with respect to $G$ if there is a string $y \in L(G_{lc})$, such that $h_\Sigma(y) = \pi$ and $h_\Delta(y) = x$. □
The left-corner parse relation for $G = \{(h_A(y), h_B(y)) | y \in L(G_{lc})\}$ is the same as the one defined by a simple syntax directed translation scheme (SDTS) in [1] or [8]. In fact the grammar $G_{lc}$ is the characteristic grammar [1] associated with the simple SDTS defining the left-corner parse relation. The left-corner parse relation is a production directed parse relation as defined by Nijholt [8].

**Example 3.2.** Let $G$ be the cfg given by the productions in the leftmost table of Figure 2. The right-most table shows the productions of the cfg $G_{lc}$ associated with $G$ and the production label set $\Delta = \{p_1, p_2, p_3, p_4\}$.

1. $S \rightarrow aSa$  
2. $S \rightarrow Ab$  
3. $S \rightarrow c$  
4. $A \rightarrow S$  

Figure 2. The productions of grammars $G$ and $G_{lc}$.

Let $x_1 = ap_1ap_1cp_3p_4p_5b$ and $x_2 = ap_1ap_1cp_3ap_4p_5b$. Since $x_1$ and $x_2$ are both sentences in $L(G_{lc})$ and $h_2(x_1) = h_2(x_2)$ although $h_\Delta(x_1) \neq h_\Delta(x_2)$, the left-corner parse relation is not proper for $G$. The sentences $aaceba$ and $aacaab$ both have left-corner parse $p_1p_1p_3p_4p_2$. The derivation trees of the sentences $h_\Delta(x_1)$ and $h_\Delta(x_2)$ are shown in Figure 3.

![Derivation trees](image)

Figure 3. Derivation trees of $aaceba$ and $aacaab$.

**4. The Transformation $\tau$**

We describe a transformation — we call it $\tau$ — which, when applied to a cfg yields a cfg that is equivalent to the original one. $\tau$ is a modification of a transformation described by Rosenkrantz and Lewis II in [9] for transforming an $LC(k)$ grammar into an $LL(k)$ grammar. Also Griffith and Petrick have used this transformation. The modification is because we prefer the definition of $LC(k)$ grammars by Soisalon-Soininen [11] which is equivalent to a slightly modified
version of the original $LC(k)$ definition in [9]. In Section 5 we show that transformation $\tau$ when applied to a cfg $G$ yields a cfg $\tau(G)$ which left-to-left-corner covers $G$. In Section 7 we will prove that $\tau$ yields an $LL(k)$ grammar if and only if it is applied to an $LC(k)$ grammar. $LL(k)$ grammars and left-corner grammars are defined in Section 6.

As we already noticed in Section 2, most of the cover results concern left, right, left-to-right or right-to-left covers. The reason for this is simply that the parse relations involved in these covers correspond with the canonical left-most and right-most derivations in a cfg. For the left-corner parse relation there is not such a smooth canonical derivation. A left-corner parser (see for instance [1] or [9] for a description) jumps through the parse tree: the left-corner parsing method is a method which combines top-down and bottom-up recognition of parts of the parse tree. Proofs of theorems on left-corner grammars or related concepts tend to be long and tedious and are therefore mostly omitted.

Before we come to a description of $\tau$ we introduce some notation and one definition.

Let $G = (N, \Sigma, P, S)$ be a cfg. Context-free grammars are always assumed to be reduced, that is they do not contain useless symbols. We write $\epsilon$ for the empty string, $V$ will denote $N \cup \Sigma, V_\epsilon$ will denote the set $V \cup \{\epsilon\}$ and $\Sigma_\epsilon$ the set $\Sigma \cup \{\epsilon\}$. If $\alpha \in V^*$ then $|\alpha|$ denotes the length of $\alpha$. Furthermore, for an integer $k > 0$, $k: \alpha$ denotes $\alpha$ if $|\alpha| \leq k$ and $k: \alpha$ denotes the prefix of $\alpha$ of length $k$ if $|\alpha| > k$ (notice that $k: \epsilon = \epsilon$ for any $k$). The left-corner of a production $A \rightarrow \alpha$ is the symbol $1: \alpha$.

**Definition 4.1.** We define the relation $\succ^G$ with respect to a cfg $G$ as follows:

i. $\succ^G \subseteq N \times V_\epsilon$.

ii. $(X,Y) \in \succ^G$ if and only if $X \rightarrow \alpha$ is a production of $G$ and $Y = 1: \alpha$. 

We will write $X \succ^G_{lc} Y$ instead of $(X,Y) \in \succ^G$. $\succ^G_{lc}$ will denote the transitive closure of $\succ^G_{lc}$.

Let $G = (N, \Sigma, P, S)$ be a context-free grammar and let $\overline{N}$ be the set $\{A \in N \mid A = S$ or there is a production in $P$ of the form $B \rightarrow \alpha A \beta$, where $\alpha \neq \epsilon\}$. (Thus $A \in \overline{N}$ if $A$ is the start symbol of $G$ or $A$ occurs in the right-hand side of a production of $G$ of which it is not the left-corner). Let $\overline{N}$ be ordered: $\overline{N} = \{A_1, A_2, ..., A_n\}$. The transformed grammar $\tau(G)$ of $G$ is the context-free grammar $(\overline{N}, \Sigma, \overline{P}, S)$. $\overline{N}$ is a superset of $\overline{N}$ and contains all symbols of the form $[A, Y]$, with $A \in \overline{N}$ and $Y \in V_\epsilon$, which appear in the productions of $\tau(G)$.

$P'$ is defined as follows. Start with $P' = \emptyset$. $P'$ will contain only those productions added to $P'$ in one of the following three steps.
1. For all $i$, $1 \leq i \leq n$, for all $a \in \Sigma_e$ add to $P'$ the production $A_i \rightarrow a[A_i,a]$ if $A_i \not \rightarrow B \in \Sigma_e$.

2. For all $[A_i,Y]$, where $Y \in V_e$, which occur in the right-hand side of a production in $P'$, for all productions in $P$ of the form $B \rightarrow Y \beta$, where $\beta \in V^*$ such that $A_i \not \rightarrow \beta B$, add the production $[A_i,Y] \rightarrow \beta[A_i,B]$ to $P'$ if it is not already in $P'$.

3. Add $[A_i,A_i] \rightarrow \epsilon$ to $P'$.

Grammar $\tau(G)$ does not contain useless symbols.

We give two examples of the transformation.

**Example 4.2.** Consider the context-free grammar $G$ given by the following productions.

1. $S \rightarrow S + T$
2. $S \rightarrow T$
3. $T \rightarrow T \times id$
4. $T \rightarrow id$

The symbols $id, +$ and $\times$ are terminal symbols. The transformed grammar $G'$ has the following productions; cf. Figure 4.

<table>
<thead>
<tr>
<th>Production</th>
<th>$S' \rightarrow \text{id }[S,\text{id}]$</th>
<th>$[S,\text{id}] \rightarrow [S,T]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3'$</td>
<td>$[S,T] \rightarrow [S,S]$</td>
<td>$4'$. $[S,T] \rightarrow \times id$ $[S,T]$</td>
</tr>
<tr>
<td>$5'$</td>
<td>$[S,S] \rightarrow + T[S,S]$</td>
<td>$6'$. $[S,S] \rightarrow \epsilon$</td>
</tr>
<tr>
<td>$7'$</td>
<td>$T \rightarrow \text{id }[T,\text{id}]$</td>
<td>$8'$. $[T,\text{id}] \rightarrow [T,T]$</td>
</tr>
<tr>
<td>$9'$</td>
<td>$[T,T] \rightarrow \epsilon$</td>
<td>$10'$. $[T,T] \rightarrow \times id$ $[T,T]$</td>
</tr>
</tbody>
</table>

**Figure 4.**

**Example 4.3.** If we apply $\tau$ to the cfg $G$ of Example 3.2, we obtain the grammar $H$ given by the productions shown in Figure 5. Do not pay attention yet to the last column. We will later return to this example.

<table>
<thead>
<tr>
<th>Production</th>
<th>$S \rightarrow a[S,a]$</th>
<th>$S \rightarrow c[S,c]$</th>
<th>$[S,a] \rightarrow \text{sa}[S,S]$</th>
<th>$[S,c] \rightarrow [S,S]$</th>
<th>$[S,S] \rightarrow [S,A]$</th>
<th>$[S,A] \rightarrow b[S,S]$</th>
<th>$[S,S] \rightarrow \epsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1'$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$1$</td>
<td>$3$</td>
<td>$4$</td>
<td>$2$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$2'$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$[S,S]$</td>
<td>$[S,A]$</td>
<td>$[S,S]$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$3'$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$[S,S]$</td>
<td>$[S,A]$</td>
<td>$[S,S]$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$4'$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$[S,S]$</td>
<td>$[S,A]$</td>
<td>$[S,S]$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$5'$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$[S,S]$</td>
<td>$[S,A]$</td>
<td>$[S,S]$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$6'$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$[S,S]$</td>
<td>$[S,A]$</td>
<td>$[S,S]$</td>
<td>$\epsilon$</td>
</tr>
<tr>
<td>$7'$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$\epsilon$</td>
<td>$[S,S]$</td>
<td>$[S,A]$</td>
<td>$[S,S]$</td>
<td>$\epsilon$</td>
</tr>
</tbody>
</table>

**Figure 5.**

5. **General Properties of $\tau$**

In this section we will show that if transformation $\tau$ is applied to a cfg $G$ we obtain a cfg $G'$ which left-to-left-corner covers $G$. Therefore we have to show that there exists a homomorphism which maps left parses of a string $x \in L(G)$ with respect to $G'$ onto left-corner parses
of \( x \) with respect to \( G \). The method of proof is inspired by Soisalon-Soininen [10]. The diagram of Figure 6 shows the grammars involved in the proof and the steps we will take.

![Figure 6.](image)

We apply transformation \( \tau \) to the grammar \( G_{ic} \) associated with \( G \) to obtain the cfg \( H \). \( L(H) \) should be \( L(G_{ic}) \). Then we define the grammar \( H' \) and a homomorphism \( \phi_r \) such that \( L(H') = L(G) \) and moreover \( H'[l/lc]G \) with respect to \( \phi_r \). Finally, we construct the cfg \( G' \) from \( H' \) in such a way that \( G' = \tau(G) \) and define a homomorphism \( \psi \) such that \( G'[l/l]H' \). By the transitivity property of the cover relation we obtain the desired result.

We first show that \( \tau \) preserves the language generated by the cfg to which it is applied.

**Lemma 5.1.** Let \( G = (N, \Sigma, P, S) \) be a context-free grammar. Then \( L(G) = L(\tau(G)) \).

**Proof:** From the construction of \( \tau(G) \) from \( G \), it follows that for all \( A \in \bar{N} \) (for the meaning of \( \bar{N} \) see the transformation):

\[
A \Rightarrow_l A_1 \alpha_1 \Rightarrow_l A_2 \alpha_2 \alpha_1 \Rightarrow_l \cdots \\
\Rightarrow_l A_n \alpha_{n-1} \cdots \alpha_2 \alpha_1 \Rightarrow_r a \alpha_n \alpha_{n-1} \cdots \alpha_2 \alpha_1. \quad (5.1.1)
\]

\( a \in \Sigma \), is a derivation in \( G \) if and only if

\[
A \Rightarrow_r a[A,a] \Rightarrow_r a \alpha_n [A,A_{n-1}] \Rightarrow_r a \alpha_n \alpha_{n-1}[A,A_{n-2}] \\
\Rightarrow_r a \alpha_n \alpha_{n-1} \cdots \alpha_2 \alpha_1[A,A] \\
\Rightarrow_r a \alpha_n \alpha_{n-1} \cdots \alpha_2 \alpha_1 \quad (5.1.2)
\]

is a derivation in \( \tau(G) \). Notice that each symbol that occurs in \( \alpha_n \cdots \alpha_2 \alpha_1 \), is either a terminal symbol or a symbol in \( \bar{N} \).

Any derivation \( A \Rightarrow_l^* x \) in \( G \) can be seen to be constructed from derivations of the form (5.1.1). And with these derivations there are corresponding derivations in \( \tau(G) \) of the form (5.1.2). As a special case we have \( S \Rightarrow_l^* x \) in \( G \) if and only if \( S \Rightarrow_r^* x \) in \( \tau(G) \). Thus \( L(G) = L(\tau(G)) \). So we can proof the Lemma by induction on this construction. We do not give the complete proof here.

As a basis for the induction consider the following observations. Let \( A \in \bar{N} \) and let \( A \to x \) be a production of \( G \), with \( x \in \Sigma^* \). First suppose that \( x \neq \varepsilon \). Let \( x = ax' \) for some \( a \in \Sigma \) and \( x' \in \Sigma^* \). By the
definition of the transformation, it follows that in \( \tau(G) \) there is a
derivation \( A \Rightarrow a[A,a] \Rightarrow ax'[A,A] \Rightarrow ax = x \). On the other
hand this last derivation exists in \( \tau(G) \) only if \( A \rightarrow x \) is a production
in \( G \). Now suppose that \( x = e \). Then in \( \tau(G) \) we have the derivation
\( A \Rightarrow [A,e] \Rightarrow [A,A] \Rightarrow e \). And also this last derivation only
exists in \( \tau(G) \) if \( A \rightarrow e \) is a production in \( G \).

Let \( G = (N, \Sigma, P, S) \) be a cfg, \( \Delta \) a set of labeling
symbols for the productions of \( G \) (\( \lambda \) denotes the corresponding
labeling function).

\( G'_{lc} = (N, \Sigma \cup \Delta, P'_{lc}, S) \) the grammar associated with \( G \) and \( \Delta \)
defined in Section 3 and \( H \) the grammar \( \tau(G_{lc}) \), i.e., the grammar obtained
from \( G_{lc} \) by the transformation \( \tau \). \( H = (N_{H}, \Sigma \cup \Delta, P_{H}, S) \).
Furthermore, let \( H^{*} = (N_{H} \cup \Delta, \Sigma, P^{*}_{H}, S) \), where \( P_{H} = P_{H} \cup \{ p_{i} \rightarrow e \mid p_{i} \in \Delta \} \).

**Lemma 5.2.** There is a leftmost derivation \( A \Rightarrow \cdad x \) for a terminal
string \( x \) in \( L(H) \) if and only if there is a string \( y \) in \( L(H') \)
such that \( h_{A}(y) = x \), \( h_{2}(y) = \pi_{1}\pi_{2}...\pi_{n} \) and \( \pi_{1}\pi_{1}\pi_{2}\pi_{2}...\pi_{n}\pi_{n} = \pi \),
where \( \pi_{1}...\pi_{n} \) is a left parse of \( y \) from \( A \) in \( H' \).

**Proof:** By induction on the length of \( \pi \).

We now define a homomorphism \( \phi_{\tau} \) which should map left parses
of a string \( x \) with respect to \( H' \) onto left-corner parses of \( x \) with
respect to the original grammar \( G = (N, \Sigma, P, S) \). Let \( \Delta'_{H} \)
be a set of labeling symbols for the productions in \( P'_{H} \).

Let \( \lambda' \) be a one-to-one labeling function from \( \Sigma' \) onto \( \Delta'_{H} \). Define the homomorphism \( \phi_{\tau} \)
from \( \Delta'_{H} \) to \( \Delta' \) as follows. For all productions in \( P_{H} \) of the form
\( [A,X] \rightarrow \beta[A,B], \) where \( \beta \in (V \cup \Delta)^{*} \), introduced in step 2 of
the transformation applied to \( G_{lc} \) we have:

\[
\phi_{\tau}(\lambda'([A,X] \rightarrow \beta[A,B])) = \lambda'(B \rightarrow X \beta).
\]

For all other elements \( q \) of \( \Delta'_{H} \): \( \phi_{\tau}(q) = e \).

**Lemma 5.3.** \( H' \upharpoonright_{lc} G \) with respect to homomorphism \( \phi_{\tau} \).

**Proof:** By definition of a left-corner parse of a string \( x \in L(G) \) with
respect to \( G \) we have to show that \( S \Rightarrow \cdad x \) in \( G_{lc} \) if and only if
\( S \Rightarrow \cdad y \) in \( G_{lc} \), where \( h_{A}(y) = x \) and \( h_{2}(y) = \phi_{\tau}(\pi) \).

By the previous Lemma there is a left parse \( \pi \) of \( x \in L(H') \) if and
only if there is a string \( y \) in \( L(H) \) such that \( h_{A}(y) = x \) and \( h_{2}(y) = \pi_{1}\pi_{2}...\pi_{n} \) and \( \pi_{1}\pi_{1}\pi_{2}\pi_{2}...\pi_{n}\pi_{n} = \pi \),
where \( \pi_{1}...\pi_{n} \) is a left parse of \( y \) in \( H \). Since \( L(H) = L(G_{lc}) \) by Lemma 5.1, we are
done if we could prove the following Claim.

**Claim.** \( \phi_{\tau}(\pi) = \pi_{1}\pi_{2}...\pi_{n} \)

**Proof of the Claim:** Recall that \( \pi = \pi_{1}\pi_{1}\pi_{2}\pi_{2}...\pi_{n}\pi_{n} \) and notice
that \( |\pi_{i}| \geq 0 \) and \( |\pi_{i}| = 1 \) for all \( i, 1 \leq i \leq n \). Since \( \phi_{\tau}(\pi_{i}) = e \), we
have to show that \( \phi_{\tau}(\pi_{1}\pi_{1}\pi_{2}\pi_{2}...\pi_{n}\pi_{n}) = \pi_{1}...\pi_{n} \).
If \( G \) does not contain \( \epsilon \)-productions in \( P \) and let \( \lambda(B \rightarrow \epsilon) = p \).
Then \( B \rightarrow \epsilon \) is an \( \epsilon \)-production in \( P \) and let \( \lambda(B \rightarrow \epsilon) = p \).
Then \( B \rightarrow \epsilon \) is a production in \( P \).
Suppose that \( A \rightarrow p[A,p] \) is a production of \( H = \tau(G_{lc}) \) introduced
in step 1 of the construction of \( H \). Then \( A,p \rightarrow [A,B] \) is a
production of \( H \) introduced in step 2 of the construction of \( H \).
definition of \( \phi_r, \phi_r(\lambda'(A \rightarrow p [A,p])) = \epsilon \) and \( \phi_r(\lambda'([A,p] \rightarrow [A,B])) = p \). Let \( \pi_j \) in \( \pi \) denote the production \( p \rightarrow \epsilon \) in \( P_H' \). Then \( \pi_j^{(j+1)} \) equals \( \lambda'([A,p] \rightarrow [A,B]) \). Thus \( \phi_r(\pi_j^{(j+1)}) = \pi_r \).

End of the proof of the Claim.

Now we should obtain \( \tau(G) \) from \( H' \). Define \( G' = (N_H, \Sigma, P_G, S) \) where \( P_G = \{ A \rightarrow h_A(\alpha) \mid A \rightarrow \alpha \text{ in } P_H \} \). Let \( \Lambda' \) be the production label set for \( G' \) and let \( \lambda'_A \) denote a label function which satisfies the following: for all productions in \( P_G \), \( \lambda'_A(A \rightarrow h_A(\alpha)) = \lambda'(A \rightarrow \alpha) \).

It is not difficult to see that \( G' = \tau(G) \) (Equality is meant here up to renaming of some nonterminal symbols). Thus we have: \( L(G') = L(\tau(G)) = L(G) = L(H') \).

Define the homomorphism \( \psi \) from \( \Lambda'_G \) to \( \Lambda'_H \) as follows:

\[ \psi(\lambda'_A(A \rightarrow h_A(\alpha))) = \lambda'(A \rightarrow \alpha), \text{ if } \alpha = h_A(\alpha). \]

\[ \psi(\lambda'_A(A \rightarrow h_A(\alpha))) = \lambda'(A \rightarrow \alpha) \lambda'(p \rightarrow \epsilon), \text{ if } 1: \alpha = p, \epsilon \in \Delta. \]

Lemma 5.4. \( G'[\ell / \ell c]G \) with respect to \( \psi \).

Proof: This follows immediately from the construction of grammar \( G' \) and the definition of \( \psi \).

Theorem 5.5. \( G'[\ell / \ell c]G \).

Proof: Use Lemma 5.3 and Lemma 5.4 and the transitivity property of the cover relation.

Example 5.6. See Example 4.2 in the previous section. Let \( \Delta' \) be the set \{1', 2', ..., 10'\} of unique labeling symbols of the productions in \( G' \) and let \( \Delta \) be the set \{1, 2, 3, 4, 5\} of unique labeling symbols of the productions in \( G \). Define the homomorphism \( \phi \) from \( \Delta' \) into \( \Delta' \) as follows: \( \phi(1') = \epsilon, \phi(2') = 4, \phi(3') = 2, \phi(4') = 3, \phi(5') = 1, \phi(6') = \epsilon, \phi(7') = \epsilon, \phi(8') = 4, \phi(9') = \epsilon \text{ and } \phi(10') = 3 \).

The sentence \( id \times id \times id \) has left-parse \( 1'2'4'3'5'6'7'8'9' \) with respect to \( G' \). This left-parse is mapped by \( \phi \) on the left-corner-parse 4 3 2 1 4 of \( id \times id \times id \) with respect to \( G \).

Example 5.7. See Example 4.3 in the previous section. With respect to grammar \( H \) the sentences \( aacbaa \) and \( aacaab \) have left-parsees \( 1'3'1'3'2'4'5'6'7'7'7' \) and \( 1'3'1'3'2'4'7'5'6'7' \). The homomorphism \( \phi_r \), given by the second column in Figure 5, maps both left-parsees onto the left-corner parse 1 1 3 4 2 with respect to \( G \).

Before we can present the following result we recall the definition of cycle-freeness of a context-free grammar.

A cfg \( G = (N, \Sigma, P, S) \) is called cycle-free if for no \( A \in N \) there is a derivation \( A \Rightarrow^* A \) in \( G \).

Theorem 5.8. Any cycle-free context-free grammar is left-to-left-corner covered by a non-left-recursive grammar.
Proof: The only thing left to show is that cycle-freeness of a cfg $G$ implies non-left-recursiveness of $\tau(G)$.

Let $G = (N, \Sigma, P, S)$. First notice that a symbol in $N$ cannot be left-recursive in $\tau(G)$. We show that there is in $G$ a derivation

$$Y \Rightarrow^+ Z,$$

where $Y, Z \in N$, if there is in $\tau(G)$ a derivation of the form

$$[A,Z] \Rightarrow^+_r [A,Y]x,$$

where $x \in \Sigma^*$. We use induction on the length of derivation (5.8.2). Let $[A,Z] \Rightarrow [A,Y]x$ be a derivation in $\tau(G)$. It follows from the construction of $\tau(G)$, that $Y \rightarrow Z$ is a production in $P$ (and $x = \epsilon$).

Suppose that if there is a derivation in $\tau(G)$ of the form (5.8.2) with length less than or equal to $n$, then there is a derivation (5.8.1) in $G$. Consider a derivation of the form (5.5.2) with length $n+1$. This derivation has the form:


By the induction hypothesis we may conclude that $Y \Rightarrow^+ X$ and $X \Rightarrow^+ Z$ are derivations in $G$. Thus $Y \Rightarrow^+ Z$ in $G$. $\square$

6. Two Classes of Grammars

In this section we give definitions of $LL(k)$ grammars and left-corner or $LC(k)$ grammars.

Definition 6.1. Let $G$ be a cfg. Let $A \in N$, $\alpha, \beta, \gamma \in V^*$, $w \in \Sigma^*$ and let $A \rightarrow \alpha$ and $A \rightarrow \beta$ be two distinct productions of $G$. $G$ is an $LL(k)$ grammar if the conditions

(i) $S \Rightarrow^+_r wA \delta \Rightarrow^+_r w \alpha \delta \Rightarrow^+ wz_1$

(ii) $S \Rightarrow^+_r wA \delta \Rightarrow^+_r w \beta \delta \Rightarrow^+ wz_2$

(iii) $k:x_1 = k:x_2$

always imply that $\alpha = \beta$. $\square$

In the following definition of the class of $LC(k)$ grammars, the notion of a left-corner sentential form ($lcsf$) is used. Informally, a left-corner sentential form is a left sentential form $u\gamma\alpha$ such that the nonterminal or terminal symbol $Y$ is not the left-corner of the production that introduced $Y$ in the leftmost derivation $S \Rightarrow^+_r u\gamma\alpha$. If $u\gamma\alpha$ is a left-corner sentential form, we write $S \Rightarrow^+_r u\gamma\alpha$. Formally, $S \Rightarrow^+_r u\gamma\alpha$ if and only if either this derivation has the form:

$$S \Rightarrow^+_r u'B\alpha'i \Rightarrow^+_r u'\gamma_1\gamma_2\alpha'i \Rightarrow^+_r u'u''Y\gamma_2\alpha'i = u\gamma\alpha,$$

where $B \rightarrow \gamma_1\gamma_2$ is a production rule in which $\gamma_i \neq \epsilon$, or $u\gamma\alpha = S$ (The first left sentential form in any derivation). We write $A \Rightarrow^+_r B\gamma$ if for some integer $n \geq 0$ there are $B_i$ in $N$, $\gamma_i$ in $V^*$, with $B_0 = A$, $B_n = B$ and $\gamma_n \ldots \gamma_1 = \gamma$, such that
\[ A \Rightarrow_1 B_1 \gamma_1 \Rightarrow_1 B_2 \gamma_2 \gamma_1 \Rightarrow_1 \cdot \cdot \cdot \Rightarrow_1 B_n \gamma_n \ldots \gamma_1. \]

**Definition 6.2.** Let \( G = (N, \Sigma, P, S) \) be a cfg and let \( k \) be an integer \((k > 0)\). \( G \) is an LC\((k)\) grammar if and only if

1. the conditions

   (i) \( S \Rightarrow^* \gamma_1 \Rightarrow_1 u_1 B_1 \gamma_1 \delta_1 \Rightarrow_1 u_1 X \beta_1 \gamma_1 \delta_1 \Rightarrow_1 u_1 x_1 \beta_1 \gamma_1 \delta_1 \Rightarrow_1 u_1 x_1 z_1 \)

   (ii) \( S \Rightarrow^* \gamma_2 \Rightarrow_1 u_2 B_2 \gamma_2 \delta_2 \Rightarrow_1 u_2 X \beta_2 \gamma_2 \delta_2 \Rightarrow_1 u_2 x_2 \beta_2 \gamma_2 \delta_2 \Rightarrow_1 u_2 x_2 z_2 \)

   (iii) \( u_1 x_1 = u_2 x_2 \) and \( k : z_1 = k : z_2 \)

   imply \( B_1 = B_2 \) and \( \beta_1 = \beta_2 \).

   Notice that if \( X \) is a terminal symbol then \( X = x_1 = x_2 \) and condition (iii) implies that \( u_1 = u_2 \). If \( X \beta_1 = \epsilon \) then \( X \beta_2 = \epsilon \), \( u_1 = u_2 \) and \( x_1 = x_2 = \epsilon \).

2. (condition for \( \epsilon \)-rules) If

   \( S \Rightarrow^* \gamma_1 \Rightarrow_1 u_1 B_1 \gamma_1 \delta_1 \Rightarrow_1 u_1 \gamma_1 \delta_1 \Rightarrow_1 u x \gamma_1 \)

   is a derivation in \( G \), then there is no derivation of the form:

   \( S \Rightarrow^* \gamma_2 \Rightarrow_1 u_2 B_2 \gamma_2 \delta_2 \Rightarrow_1 u a \beta_2 \gamma_2 \delta_2 \Rightarrow_1 u a z_2 \)

   in \( G \), such that \( k : z_1 = k : z_2 \).

3. (condition for left recursive nonterminal symbols) If

   \( S \Rightarrow^* \gamma_1 \Rightarrow_1 u_1 B_1 \gamma_1 \delta_1 \Rightarrow_1 u_1 A \beta_1 \gamma_1 \delta_1 \Rightarrow_1 u_1 x_1 \beta_1 \gamma_1 \delta_1 \Rightarrow_1 u_1 x_1 z_1 \)

   is a derivation in \( G \), then there is no derivation of the form

   \( S \Rightarrow^* \gamma_2 \Rightarrow_1 u A \gamma_2 \delta_2 \Rightarrow_1 u x \delta \Rightarrow_1 u x z \)

   such that \( u x = u_1 x_1 \) and \( k : z_1 = k : z \).

   \( \square \)

This definition is equivalent with the definition of LC\((k)\) grammars in terms of right-most derivations given by Soisalon-Soininen [11]. For a proof of this equivalence see [2]. Any LL\((k)\) grammar is LC\((k)\) and any LC\((k)\) grammar is LR\((k)\) [11].

**Example 6.3.** Grammar \( G \) in Example 4.2 is an LC\((1)\) grammar. \( \square \)
7. Special Properties of Transformation $\tau$

In this section we show that the transformation $\tau$ yields an $LL(k)$ grammar if and only if it is applied to an $LC(k)$ grammar.

**Lemma 7.1.** Let $G = (N, \Sigma, P, S)$ be a context-free grammar. For all $A_0 \in \bar{N}$ (For the meaning of $\bar{N}$ see Section 4.), there exists in $G$ a derivation

$$A_0 \Rightarrow \ ae_1 \Rightarrow \ a\alpha_1A_1\gamma_1\delta_1 \Rightarrow \ a\delta_1$$

$$\Rightarrow \ a\alpha_1A_1\gamma_1\delta_1 \Rightarrow \ a\alpha_2A_2\gamma_2\delta_2\gamma_1\delta_1$$

$$\Rightarrow \ a\alpha_2A_2\gamma_2\delta_2\gamma_1\delta_1 \Rightarrow \ a\delta_2\gamma_1\delta_1$$

$$\Rightarrow \ a\delta_2\gamma_1\delta_1 \Rightarrow \ a\delta_1,$$

if and only if in grammar $\tau(G)$ the derivation

$$A_0 \Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,B_2]y_1[A_0,B_1]$$

$$\Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,B_2]y_1[A_0,B_1]$$

exists, such that for all $i$, if $1 \leq i \leq n$ then

$$[A_{i-1},B_i] \Rightarrow \ a\gamma_i[A_{i-1},A_i] \Rightarrow \ a\delta_i.$$

**Proof:** By induction on the length of the derivations. \hfill \Box

**Lemma 7.2.** For any $k > 0$, if $G$ is an $LC(k)$ grammar, then $\tau(G)$ is an $LL(k)$ grammar.

**Proof:** Let $G = (N, \Sigma, P, S)$ be an $LC(k)$ grammar for some $k > 0$. $G'$ denotes the grammar $\tau(G)$. Suppose that $G'$ is not an $LL(k)$ grammar. Then for some $Z \in N'$,

$$S \Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,B_2]y_1[A_0,B_1]$$

$$\Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,B_2]y_1[A_0,B_1]$$

are derivations in $G'$, where $\omega_1 \neq \omega_2$, although $k \gamma_1 = k \gamma_2$.

We distinguish three cases: I) $Z$ is a symbol in $N$, II) $Z$ is of the form $[A,Y]$, where $A \in N$, $Y \in V - \{A\}$ and III) $Z$ is of the form $[A,A]$.

Case I. It follows from the construction of the grammar $G'$, that the productions $Z \Rightarrow \omega_1$ and $Z \Rightarrow \omega_2$ both have the form $A \Rightarrow a[A,a]$, where $a \in \Sigma_\gamma$. Since $k > 0$, $k \gamma_1 = k \gamma_2$ and $\gamma_1 \neq \gamma_2$, the following derivations exist in $G'$.

$$S \Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,B_2]y_1[A_0,B_1] \Rightarrow \ a\gamma_2[y_1 = \omega_1]$$

$$S \Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,B_2]y_1[A_0,B_1] \Rightarrow \ a\gamma_2[y_2 = \omega_2]$$

Because of the first part of these derivation in $G'$ we may conclude, using Lemma 7.1, that $S \Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,\epsilon]$. Thus $S \Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,\epsilon]$. Further, using Lemma 7.1, that $S \Rightarrow \ a\gamma_1[A_0,B_1] \Rightarrow \ a\gamma_2[A_1,\epsilon]$.
Notice that in general we may not conclude from \( \delta \Rightarrow^* \delta' \) and \( \delta \Rightarrow^* \nu \) that \( \delta' \Rightarrow^* \nu \). However, by Lemma 7.1 the derivation \( \delta \Rightarrow^* \delta' \) has a special form.

If in Lemma 7.1 \( [A_{i-1}, B_i] \Rightarrow^* y_i \), then \( [A_{i-1}, B_i] \Rightarrow^* y_1 \) and thus \( [A_{i-1}, B_i] \Rightarrow ^* \delta_i [A_{i-1}, A_{i-1}] \Rightarrow ^* \delta_i \Rightarrow ^* y_1 \). Therefore we may conclude here that \( \delta' \Rightarrow^* \nu_1 \) and \( \delta' \Rightarrow^* \nu_2 \) in \( G \). Since \( A \Rightarrow a [A, a] \) is a production in \( G' \), it follows from the construction of \( G' \) that \( \Rightarrow ^* B_1 \gamma_1 \Rightarrow a \beta_1 \gamma_1 \) is a derivation in \( G \). In the same way, since \( A \Rightarrow [A, e] \) is a production of \( G' \), there is a derivation \( \Rightarrow ^* B_2 \gamma_2 \Rightarrow \gamma_2 \) in \( G \). Furthermore we know (See the proof of Lemma 5.1.) that \( a \beta_1 \gamma_1 \Rightarrow^* a \nu_1 \) and \( \gamma_2 \Rightarrow^* \nu_2 \) in \( G \). Thus derivations

\[
S \Rightarrow^* w A \delta' \Rightarrow^* w B_2 \gamma_2 \delta' \Rightarrow^* w \gamma_2 \delta' \Rightarrow^* w u_2 \nu_2 = w z_2
\]

and

\[
S \Rightarrow^* w A \delta' \Rightarrow^* w B_1 \gamma_1 \delta' \Rightarrow^* w \beta_1 \gamma_1 \delta' \Rightarrow^* w a \nu_1 \nu_1 = w z_1
\]

exist in \( G \). Since \( k \vdash z_1 = k \vdash z_2 \), we conclude that \( G \) does not satisfy the condition for \( e \)-productions in Definition 6.2 and so we have shown that the assumption that \( G \) is \( LC(k) \) leads to a contradiction.

Case II. In this case the productions \( Z \Rightarrow \omega_1 \) and \( Z \Rightarrow \omega_2 \) in the introduction of this proof have the form \( [A, Y] \Rightarrow \gamma_1 [A, Y_1] \) and \( [A, X] \Rightarrow \gamma_2 [A, Y_2] \). Suppose that

\[
S \Rightarrow^* w [A, X] \delta \Rightarrow^* w \gamma_1 [A, Y_1] \delta \Rightarrow^* w u_1 [A, Y_1] \delta
\]

and

\[
S \Rightarrow^* w [A, X] \delta \Rightarrow^* w \gamma_1 [A, Y_1] \delta \Rightarrow^* w u_1 [A, Y_1] \delta
\]

are derivations in \( G' \), where \( \gamma_1 [A, Y_1] \neq \gamma_2 [A, Y_2] \), although \( k \vdash z_1 = k \vdash z_2 \). From the construction of \( G' \) it follows that the first part of these derivations has the form: \( S \Rightarrow^* w A \delta \Rightarrow^* w w'' [A, Y] \delta \), where \( w = w w'' \) and \( Y \Rightarrow^* w'' \) is a derivation in \( G \). By Lemma 7.1 we know that \( S \Rightarrow^* w A \delta' \) is a derivation in \( G \), such that \( \delta \Rightarrow^* \delta' \) in \( G' \). It will be clear that in \( G \) the derivations

\[
A \Rightarrow^* \gamma_1 \beta_1 \Rightarrow \gamma_1 \beta_1 \Rightarrow^* w'' \gamma_1 \beta_1
\]

and

\[
A \Rightarrow^* \gamma_2 \beta_2 \Rightarrow \gamma_2 \beta_2 \Rightarrow^* w'' \gamma_2 \beta_2
\]

exist, such that \( [A, Y_1] \Rightarrow^* \gamma_1 \beta_1 \) and \( [A, Y_2] \Rightarrow^* \gamma_2 \beta_2 \) in \( G' \). Notice that we may conclude that \( \beta_1 \Rightarrow^* \gamma_1 \) and also that \( \beta_2 \Rightarrow^* \gamma_2 \) (see Case I for the justification of this). Thus we know that

\[
S \Rightarrow^* w A \delta' \Rightarrow^* w Y \gamma_1 \beta_1 \delta' \Rightarrow w Y \gamma_1 \beta_1 \delta'
\]
\[ \Rightarrow w'w'' \gamma_1 \beta_1 \delta' \Rightarrow *w'w''u_1y_1v_1 = w'w''z_1 \]  
(7.2.1)

and

\[ S \Rightarrow \bar{w}wA \delta' \Rightarrow \bar{w}Y_2B_2 \delta' \Rightarrow w'Y \gamma_2 \beta_2 \delta' \Rightarrow w'w'' \gamma_2 \beta_2 \delta' \Rightarrow *w'w''u_2Y_2v_2 = w'w''z_2 \]  
(7.2.2)

are derivations in \( G \). Since \( k : z_1 = k : z_2 \) we have \( k : w''z_1 = k : w''z_2 \).

From this last equality and derivations (7.2.1) and (7.2.2) we conclude that clause 1 of Definition 6.2 is not satisfied. This, however, contradicts the assumption that \( G \) is \( LC(k) \).

Case III. We consider the case in which the productions \( Z \rightarrow \omega_1 \) and \( Z \rightarrow \omega_2 \) in the introduction of this proof have the form \( [A,A] \rightarrow \epsilon \) and \( [A,A] \rightarrow \beta [A,B] \). Suppose that in \( G' \) derivations

\[ S \Rightarrow \bar{w}[A,A] \delta \Rightarrow \bar{w} \delta \Rightarrow \bar{w}z_2 \]  
(7.2.3)

and

\[ \Rightarrow \bar{w}[A,A] \delta \Rightarrow \bar{w} \beta [A,B] \delta \Rightarrow \bar{w}y_1[A,B] \delta \]  
(7.2.4)

exists, such that \( k : z_2 = k : y_1v_1z_1 \).

From the construction of \( G' \) it follows that the first part of derivations (7.2.3) and (7.2.4) has the form

\[ S \Rightarrow \bar{w}A \delta \Rightarrow \bar{w}w''[A,A] \delta \]

and \( A \Rightarrow *w'' \) in \( G \). By Lemma 7.1 we know that in \( G \) derivation \( S \Rightarrow \bar{w}A \delta' \) exists, such that \( \delta \Rightarrow *\delta' \) in \( G' \). From derivation (7.2.4) we conclude that in \( G \) the derivation

\[ A \Rightarrow \bar{w}A \gamma \Rightarrow \bar{w}A \beta \gamma \]

exist, where \( [A,B] \Rightarrow \bar{w} \gamma \) in \( G' \). We may conclude that \( \gamma \Rightarrow *v_1 \).

Thus in \( G \) derivations

\[ S \Rightarrow \bar{w}A \delta' \Rightarrow \bar{w}w'' \delta' \Rightarrow *w'w''z_2 \]

and

\[ S \Rightarrow \bar{w}A \delta' \Rightarrow \bar{w}A \beta \gamma \delta' \Rightarrow \bar{w}w'' \beta \gamma \delta' \]

\[ \Rightarrow *w'w''y_1 \gamma \delta' \Rightarrow *w'w''y_1v_1z_1 \]

exist. Since \( k : z_2 = k : y_1v_1z_1 \) we conclude from these derivations that \( G \) doesn't satisfy clause 3 in Definition 6.2. This contradicts the assumption that \( G \) is \( LC(k) \).

We finally conclude that \( G' \) must be \( LL(k) \).

We now show the converse of Lemma 7.2.

**Lemma 7.3.** Let \( G \) be a context-free grammar. For any \( k > 0 \), if \( G \) is not an \( LC(k) \) grammar, then \( \tau(G) \) is not an \( LL(k) \) grammar.
Proof: Let $G = (N, \Sigma, P, S)$ be a context-free grammar which is not LC$(k)$. $G'$ will denote $\tau(G)$.

Case I. Suppose that $G$ does not satisfy clause 1 in the definition of LC$(k)$ grammars. Then there exist derivations

$$ S \Rightarrow^* u_1A\delta_1 \Rightarrow^* u_1B_1\gamma_1\delta_1 \Rightarrow^* u_1X\beta_1\gamma_1\delta_1 $$

$$ \Rightarrow^* u_1x_1\beta_1\gamma_1\delta_1 \Rightarrow^* u_1x_1y_1\nu_1z_1 \quad (7.3.1) $$

and

$$ S \Rightarrow^* u_2A\delta_2 \Rightarrow^* u_2B_2\gamma_2\delta_2 \Rightarrow^* u_2X\beta_2\gamma_2\delta_2 $$

$$ \Rightarrow^* u_2x_2\beta_2\gamma_2\delta_2 \Rightarrow^* u_2x_2y_2\gamma_2z_2 \quad (7.3.2) $$

in $G$, where $B_1 \neq B_2$ or $\beta_1 \neq \beta_2$, although $u_1x_1 = u_2x_2$ and $k:y_1\nu_1z_1 = k:y_2\nu_2z_2$.

Consider derivation $(7.3.1)$. By Lemma 7.1 we conclude from the first part of this derivation that in $G'$ the derivation $S \Rightarrow^* u_1A\delta_1'$ exists, such that $\delta_1' \Rightarrow^* \delta_1$. From the second and third part of the derivation we may conclude that $[A,X] \Rightarrow \beta[A,B_1]$ is a production of $G'$ and $[A,B_1] \Rightarrow^* \gamma_1$ in $G'$. Moreover, we may conclude that $A \Rightarrow^* x_1[A,X]$ in $G'$. Similar conclusions can be derived from derivation $(7.3.2)$ in $G$. Thus in $G'$ the derivations

$$ S \Rightarrow^* u_1A\delta_1' \Rightarrow^* u_1x_1[A,X]\delta_1' \Rightarrow^* u_1x_1\beta_1[A,B_1]\delta_1' $$

$$ \Rightarrow^* u_1x_1y_1[A,B_1]\delta_1' \Rightarrow^* u_1x_1y_1\nu_1z_1 $$

and

$$ S \Rightarrow^* u_2A\delta_2' \Rightarrow^* u_2x_2[A,X]\delta_2' \Rightarrow^* u_2x_2\beta_2[A,B_2]\delta_2' $$

$$ \Rightarrow^* u_2x_2y_2[A,B_2]\delta_2' \Rightarrow^* u_2x_2y_2\nu_2z_2 $$

exist. Since $B_1 \neq B_2$ or $\beta_1 \neq \beta_2$, the productions $[A,X] \Rightarrow \beta_1[A,B_1]$ and $[A,X] \Rightarrow \beta_2[A,B_2]$, used in these derivations, are not the same. Because of the equalities $u_1x_1 = u_2x_2$ and $k:y_1\nu_1z_1 = k:y_2\nu_2z_2$, we conclude that $G'$ is not LL$(k)$.

Case II. Suppose that $G$ does not satisfy clause 2 in the definition of LC$(k)$ grammars. Then there exist derivations

$$ S \Rightarrow^* uA\delta_1 \Rightarrow^* uB_1\gamma_1\delta_1 \Rightarrow^* u\gamma_1\delta_1 $$

$$ \Rightarrow^* uy_1\delta_1 \Rightarrow^* uy_1z_1 \quad (7.3.3) $$

and

$$ S \Rightarrow^* uA\delta_2 \Rightarrow^* uB_2\gamma_2\delta_2 \Rightarrow^* u\beta_2\gamma_2\delta_2 $$

$$ \Rightarrow^* u\beta_2\gamma_2\delta_2 \Rightarrow^* u\gamma_2z_2 \quad (7.3.4) $$

in $G$, such that $k:y_1z_1 = k:y_2z_2$. 


From derivation (7.3.3) we conclude that in $G'$ derivation
\[ S \Rightarrow^* uA \delta_1 \] exists, such that \( \delta_1 \Rightarrow^* \gamma_1 \). In addition, \( A \not\rightarrow [A, \epsilon] \)
and \( [A, \epsilon] \not\rightarrow [A, B_1] \) are productions of \( G' \) and \( [A, B_1] \Rightarrow^* \gamma_1 \) in \( G' \).
From derivation (7.3.4) we conclude that in \( G' \) derivation
\[ S \Rightarrow^* uA \delta_2 \] exists, such that \( \delta_2 \Rightarrow^* \gamma_2 \). Moreover, \( A \not\rightarrow a[A, a] \)
and \( [A, a] \rightarrow \beta[A, B_2] \) are productions of \( G' \) and \( [A, B_2] \Rightarrow^* \gamma_2 \) in \( G' \).
Thus in \( G' \) derivations
\[ S \Rightarrow^* uA \delta_1 \Rightarrow^* i[A, \epsilon] \delta_1' \Rightarrow^* u[A, B_1] \delta_1' \Rightarrow^* u \gamma_1 \delta_1' \Rightarrow^* uy_1 z_1 \]
and
\[
\begin{align*}
S &\Rightarrow^* uA \delta_2 \Rightarrow^* i u[a[A, a] \delta_2' \Rightarrow^* i uA \beta[A, B_2] \delta_2' \\
&\Rightarrow^* i uAv[A, B_2] \delta_2' \Rightarrow^* i uavy_2 \delta_2' \Rightarrow^* i uavy_2 z_2
\end{align*}
\]
exist. Since \( k : avy_2 z_2 = k : y_1 z_1 \) we conclude that \( G' \) is not \( LL(k) \).
Case III. Suppose that in \( G \) the derivations
\[
S \Rightarrow^* i u_1 A \delta_1 \Rightarrow^* i u_1 B_1 y_1 \delta_1 \Rightarrow^* i u_1 A \beta_1 y_1 \delta_1 \\
\Rightarrow^* i u_1 x_1 \delta_1 \beta_1 y_1 \delta_1 \Rightarrow^* i u_1 x_1 y_1 v_1 z_1
\]
(7.3.5)
and
\[
S \Rightarrow^* i u_1 A \delta \Rightarrow^* i u_1 x \delta \Rightarrow^* i u x z\]
exist, such that \( u_1 x = u_1 x_1 \) and \( k : u_1 y_1 z_1 = k : z \). In a way similar as in
the other two cases we may conclude from these derivations that the derivations
\[
S \Rightarrow^* i u_1 A \delta_1' \Rightarrow^* i u_1 x_1 [A, A] \delta_1 \Rightarrow u_1 x_1 \beta_1 [A, B_1] \delta_1' \\
\Rightarrow^* i u_1 x_1 y_1 [A, B_1] \delta_1' \Rightarrow^* i u_1 x_1 y_1 v_1 z_1
\]
and
\[
S \Rightarrow^* i uA \delta' \Rightarrow^* i u x [A, A] \delta' \Rightarrow^* i u x \delta' \Rightarrow^* u x z
\]
exist in \( G' \). Since \( u x = u x_1 \) and \( k : z = k : y_1 v_1 z_1 \) we conclude that \( G' \)
is not \( LL(k) \).
From Lemma 7.2 and Lemma 7.3 we may conclude the following result.

Theorem 7.4. The transformation \( \tau \) yields an \( LL(k) \) grammar \( (k > 0) \)
if and only if it is applied to an \( LC(k) \) grammar.

8. Semantical Covering of Attribute Grammars

In the introduction we already noticed that it is sometimes possible to
evaluate attributes of the nodes of the parse tree during parsing.
Therefore it makes some sense to consider transformations on attribute
grammars which yield a semantically equivalent attribute grammar
based on a cfg which covers the original cfg. It lies at hand to call the
result of such a transformation a semantical covering grammar of the original attribute grammar. Here we consider the question whether all translations definable by a class \( X \) of attribute grammars can also be defined by attribute grammars in a class \( Y \) which semantically covers class \( X \). A similar question is posed by Aho and Ullman for syntax-directed translation schemes [1]: Suppose that \( G_2 \) left or right covers \( G_1 \). Is every SDTS with \( G_1 \) as underlying grammar equivalent to an SDTS with \( G_2 \) as underlying grammar? A partial answer to this question is given by Shyamasundar in [14]. For special classes of SDTS this question has been studied by Rosenkrantz and Lewis II [9] and by Soisalon–Soininen [11]. These studies consider the semantical cover relation between classes of simple syntax directed translation schemes (simple SDTS) [1]. SDTS can be viewed as a special class of attribute grammars (See Filé [13] for a precise description of SDTS in the formalism of attribute grammars).

The semantic equivalence of covering attribute grammars is also studied by Bochmann [4]. However, the notion of semantical covering introduced by Bochmann is quite different from the semantical cover relation we have in mind. Let us first introduce some notions and notation.

Let \( G \) be an attribute grammar (AG). The reader is referred to Filé [13] for a definition. Notice that the specification of the semantic domain, that is a set of sets of values of the attributes together with the set of functions denoted by the evaluation rules of the attributes, is a part of the definition of an AG. Let \( \delta \) denote the distinguished synthesized attribute of the start symbol of the AG \( G \). In an evaluated complete grammatical tree (parse tree), \( \delta \) is a special attribute of the root of the tree of which the value represents the meaning or the translation of the yield of the tree. Let \( D \) denote the value set of \( \delta \). Let \( Tr \) denote the set of complete grammatical trees of the underlying CFG \( G_0 \) of \( G \). Let \( \delta(t) \in D \) denote the value of \( \delta \) of tree \( t \in Tr \) and let the translation of \( x \in L(G) \) be \( \psi(x) = \{ \delta(t) \mid t \in Tr \text{ and } \text{yield}(t) = x \} \). \( \psi \) is called the translation function of \( G \). Even if \( G_0 \) is (syntactically) ambiguous \( \psi(x) \) may contain only one element (We assume that the AG is non-circular so \( \delta(t) \) is always defined). The translation \( TRANS(G) \) of the AG \( G \) is:

\[
TRANS(G) = \{ (x, \psi(x)) \mid x \in L(G) \}.
\]

Let \( 2^D \) denote the power set of \( D \). The following definition is from Bochmann [4].

**Definition 8.1.** Let \( G_1 \) and \( G_2 \) be AG's over the same terminal alphabet, \( \delta_1 \) and \( \delta_2 \) the distinguished attributes of \( G_1 \) and \( G_2 \), respectively, \( D_1 \) and \( D_2 \) their value sets and \( \psi_1 \) and \( \psi_2 \) the translation functions. \( G_2 \) is semantically finer then \( G_1 \) if there exists a mapping \( \phi : 2^{D_2} \rightarrow 2^{D_1} \), such that: for all \( x \in L(G_1) \cap L(G_2) \), \( \psi_1(x) \subseteq \phi(\psi_2(x)) \). \( \square \)
Thus $G_2$ is semantically finer then $G_1$ implies that for all $x$ in both languages the translation according to $G_1$ can be obtained by applying the mapping $\phi$ to the translation according to $G_2$. Because of the similarity between this definition and the definition of cover Bochmann uses the phrase "semantical covering". In order to explain our idea of semantical covering a little more, we consider for a start the following definition.

**Definition 8.2.** An AG $G_2$ semantically covers an AG $G_1$ if

i. the underlying cfg of $G_2$ covers the underlying cfg of $G_1$.

ii. $\text{TRANS}(G_1) = \text{TRANS}(G_2)$.

If an attribute grammar has a deterministically parsable underlying cfg and all attributes of all parse trees are evaluable during parsing following a parsing method suitable for the cfg, then we call the AG a one-pass AG. We are specially interested in one-pass AG's. All one-pass AG's are $L$-attributed, at least if we adopt the strict one-pass evaluation strategy as defined in [3]. $L$-attributed $LL(k)$ grammars are one-pass AG's since all attributes are evaluable during top-down parsing. For other classes of one-pass AG we refer to [3] where also the class of $LC$-attributed grammars is defined. Let $X$-AG and $Y$-AG be classes of one-pass AG's over a specific semantic domain. If we want to compare different classes of AG with respect to their ability to define translations (or their "formal power", cf. [12]) we must explicitly mention the semantic domain because this ability not only depends on the number of attributes and the kind of attribute dependencies in the AG but also on the types of attributes and the function types in the semantic domain. Knuth already showed in [5] that any translation defined by an AG can be defined by an AG which has only synthesized attributes. The question whether a class $X$-AG semantically covers a class $Y$-AG asks for a transformation which yields a cfg in class $X$ when applied to a cfg in $Y$ and a redefinition of attributes and attribute-rules such that the translation is preserved and the AG obtained is in $X$-AG. By Definition 8.2 semantically covering of attribute grammars does not imply any correspondence between attribute values of internal nodes of corresponding parse trees. However, if we want to show that an AG $G_1$ semantically covers an AG $G_2$, we need some inductive argument on the construction of corresponding derivation trees of the involved underlying cfgs. Therefore we can use a stronger form of semantical covering which implies the equality of attribute values of the roots of corresponding subtrees of corresponding parse trees. What "corresponding subtrees" are should follow from a particular transformation by means of which the covering grammar is obtained. For example the left factoring transformation applied to a grammar which is not left factored yields a left factored grammar that right covers the original grammar. If a grammar is not left factored then there exist productions $A \rightarrow \alpha \beta$ and $A \rightarrow \alpha \gamma$, with $\alpha \neq \epsilon$ and
$\beta \neq \gamma$. A step in the process of left factoring consists of replacing the productions $A \rightarrow \alpha\beta$ and $A \rightarrow \alpha\gamma$ by the productions $A \rightarrow \alpha H$, $H \rightarrow \beta$ and $H \rightarrow \gamma$, where $H$ is a newly introduced nonterminal symbol. In [3] an informal algorithm is given for transforming AG's in the class $LP$-$AG$ based on left-part grammars into the class $LL$-$AG$, the class of $L$-attributed $LL(k)$ grammars. This transformation is an attributed variant of the left factoring transformation. Here it is immediately clear what the corresponding subtrees of corresponding parse trees are.

If we want to consider semantical cover relations between AG's with semantical conditions or disambiguating predicates which play a role in making parsing decisions based on attribute values [3] — Definition 8.2 is not suitable. In this case the language generated by the AG is a subset of the language generated by the underlying CFG so we cannot say anything about the existence of a cover relation between the underlying CFGs without considering attribute values of internal corresponding nodes.

References

2. R. op den Akker: A left-corner property for context-free grammars, Memorandum INF-86-8 (1986), Twente University of Technology, Department of Informatics, Enschede, The Netherlands.


