FORMAL MODELING BY A BI-PARALLEL GRAMMAR

ADALBERT GOLOMETY, EMIL M. POPA
Department of Computer Science
Lucian Blaga University of Sibiu
4 Emil Cioran Street, 550025 Sibiu
ROMANIA

Abstract: In this paper we propose a compound grammar with horizontal and vertical parallelism. This grammar combines horizontal parallelism from compound grammars introduced by Abraham and vertical parallelism from matrix grammars introduced by Greibach and Abraham. We named this hierarchy of grammars bi-parallel grammar. This combination permits a unitary formal model of hierarchy processes with parallel actions. Also strings generating process may be done on a parallel-computing environment.

Key-Words: formal languages, Chomsky hierarchy, context-sensitive grammar, context-free grammar, compound grammars, matrix grammars.

1 Introduction
Classical grammars Chomsky present horizontal parallelism. Matrix grammars present vertical parallelism.

1.1 Chomsky grammars
One use the definition of Chomsky grammars: a grammar is a construct $G = (N,T,S,P)$, where $N$ is the nonterminal alphabet, $T$ is the terminal alphabet, $S$ the initial letter or axiom and $P$ the set of rewriting rules or productions. The rewriting rules are on the form $A \rightarrow w$, $A \in N$, $w \in (N \cup T)^*$ for free-context grammars.

Given $w,v \in (N \cup T)^*$, an immediate or direct derivation (in 1 step) denoted $w \Rightarrow_G v$ holds if and only if there exist $u_i \in (N \cup T)^*$ such that $w = u_1 a u_2$ and $v = u_1 b u_2$ and there exist $\alpha \rightarrow \beta \in P$.

$\Rightarrow^+ = \Rightarrow^* \cup \Rightarrow^+$ denotes the reflexive transitive closure and $\Rightarrow^*$ the transitive closure, respectively of $\Rightarrow_G$.

The language generated by a grammar is defined by:

$$L(G) = \{ w : S \Rightarrow^*_G w \text{ and } w \in T^* \}$$

In other words $L(G)$ is the set of terminal strings generated by sequential derivations from $S$.

Example 1.1
Let $G = (N,T,S,P)$ be a grammar such that:

$N = \{ S, A, C \}$
$T = \{ a, b, c \}$
$P = \{ S \rightarrow a b c, S \rightarrow a A b C, A \rightarrow a A b, A \rightarrow a b, C \rightarrow c C, C \rightarrow c \}$

The language generated by $G$ is the language:

$$L = \{ a^n b^n c^n : n \geq 1 \}$$

(1)

The languages (1) is of type $L_2 = \text{context free language}$. From Chomsky hierarchy of languages:

$L_2 \subseteq L_1$ with $L_1 = \text{context sensitive language}$.

For the context-free grammar $G$ of example 1.1 the derivation tree for the string $a^2 b^2 c$ is shown below:

![Derivation tree for $a^2 b^2 c$ in $G$ of example 1.1](image)

Fig.1 Derivation tree for $a^2 b^2 c$ in $G$ of example 1.1

The dash-line shows the horizontal parallelism. Generally in the strings generating process by rewriting rules the context is inherited. The strings generated beginning from $S$ continues with derivation of $A$ and $B$. The string $aAbC$ inherit the context $a$, $b$ for $A$ and then carry that context forward.

Abraham gives in [1] a method in a top-down manner for horizontal parallel dividing of a context-free grammar.

1.2 Abraham Compound Grammars
Compound grammar in Abraham [1] sense is a horizontal hierarchy of generalized grammars. A Chomsky grammar $G = (N,T,S,P)$ with $S \in (N \cup T)^*$ is named generalized grammar ($S$ is a set of strings).

Strongly formal, the grammar from example 1.1 may be divided in two generalized grammar $G_1$ and $G_2$ like that:

$$G_1 = (N^1, T^1, S^1, P^1) \quad \text{with:}$$

$$N^1 = \{ S \}$$

$$L_2 \subseteq L_1$$ with $L_1 = \text{context sensitive language}$.

For the context-free grammar $G$ of example 1.1 the derivation tree for the string $a^2 b^2 c$ is shown below:

![Derivation tree for $a^2 b^2 c$ in $G$ of example 1.1](image)

Fig.1 Derivation tree for $a^2 b^2 c$ in $G$ of example 1.1

The dash-line shows the horizontal parallelism. Generally in the strings generating process by rewriting rules the context is inherited. The strings generated beginning from $S$ continues with derivation of $A$ and $B$. The string $aAbC$ inherit the context $a$, $b$ for $A$ and then carry that context forward.

Abraham gives in [1] a method in a top-down manner for horizontal parallel dividing of a context-free grammar.
\[ S^1 = \{ S \} \]
\[ T^1 = \{ a, b, c, A, C \} \]
\[ P^1 = \{ S \rightarrow abc, S \rightarrow aAbC \} \]

and
\[ G_1 = (N^1, T^1, S^1, P^1) \] with:

1. \[ N^1 = \{ A, C \} \]
2. \[ S^1 = \{ aAbC \} \]
3. \[ T^1 = \{ a, b, c \} \]
4. \[ P^1 = \{ a \rightarrow aAb, A \rightarrow ab, C \rightarrow cC, C \rightarrow C \} \]

The language \( L(G^1) \) is the same context-free language (1). However this strongly horizontal division makes the grammar \( G_2 \) more complex than the initial grammar and almost a context sensitive grammar.

1.2.1. Our point of view
According to figure 1 our point of view is that the axiom \( \{ aAbC \} \) of \( G_2 \) (7) may be reduced to \( \{ A, C \} \). The context strings \( a \) and \( bC \) of \( A \) has no role in the first rewriting rules \( aAbC \rightarrow aaAb \) and \( aAbC \rightarrow aabbcC \). So the grammar \( G_2 \) become more simply:
\[ G_2 = (N^2, T^2, S^2, P^2) \] with:

1. \[ N^2 = \{ A, C \} \]
2. \[ S^2 = \{ aAbC \} \]
3. \[ T^2 = \{ a, b, c \} \]
4. \[ P^2 = \{ A \rightarrow aAb, A \rightarrow ab, C \rightarrow cC, C \rightarrow C \} \]

In the strings generating process the context is inherited. In our case strings generation begin from \( G_1 \) and continue with \( G_2 \). Grammar \( G_2 \) inherit the context from \( G_1 \), transport that context forward and give the final language \( L(G^2) = L(G) \).

1.3 Rus Compound Grammars Hierarchy
For context-free languages, Th. Rus in [5] constructs a hierarchy of grammars for a given grammar in a bottom-up manner. The start level is the terminals set of the start grammar and the first grammar contains all the rules that generate strings of terminals.

The axiom of a Th.Rus grammar is \( N \) the entire set of nonterminals. Formal construction for types hierarchy (nonterminals hierarchy) over \( N \) is based on the operator GTC (Grammar Types Constructor). An example is detailed in [10].

The main lack of Th.Rus grammars hierarchy is that a nonterminal may be present in more than one grammar. We think that a nonterminal has a unique semantic and must belong to one grammar.

1.4 Matrix grammars
In the previous grammars types (Chomsky grammars) derivation steps are made sequential (one occurrence of a nonterminal is rewritten) and in leftmost/rightmost fashion (extreme left/right derivation rule: the leftmost/rightmost nonterminal of string is rewritten first). In matrix grammars the derivations are made sequential or parallel, in one step many occurrences of the nonterminals are rewritten. A (simple) matrix grammar is a construct \( G_M = (N, T, S, M) \) where \( N, T, S \) are the same as in Chomsky grammars and \( M \) is a finite set of nonempty sequences(matrices)

\[ m_i : \{ r_1, r_2, \ldots, r_m \}, m_i \geq 1 \]

with context free rewriting rules:
\[ A_k \rightarrow w_k, A_k \in N, w_k \in (N \cup T)^* \]

A derivation in a matrix grammar is as follows: for every \( x, y \in (N \cup T)^* \), \( x \Rightarrow_{GM} y \) if and only if there exist strings \( x_0, x_1, \ldots, x_n \in (N \cup T)^* \) (intermediate sentential forms) such that \( x_0 = x, x_n = y \), and for \( 1 \leq i \leq n \):
\[ x_i = u_i \Rightarrow_{G_M} A_i \rightarrow^{r_i} u_i, 1 \leq i \leq n \]

In a derivation step all the rules from matrix \( m \) are done and in a sequential manner, each rule has to be performed in leftmost fashion. The language generated by the matrix grammar \( G_M \) is defined usual as:
\[ L(G_M) = \{ w : S \Rightarrow^* GM w \text{ and } w \in T^* \} \]

Despite of context-free rewriting rules the matrix grammars may generate context-sensitive languages like in example 1.2.

One notes with \( L_M \) the languages generated by matrix grammars with context-free rules without \( \lambda \) productions. \( L_M \) family includes context-free (CF) languages (every CF grammar is a matrix grammar with one single rule in every matrix \( m \)) and it is included in context-sensitive languages:
\[ L_2 \subseteq L_M \subseteq L_1 \]

Example 1.2
\[ G_M = (N, T, S, M) \] is a matrix grammar such that:
\[ N = \{ S, A, B, C \} \]
\[ T = \{ a, b, c \} \]
\[ S = \{ S \} \]
\[ M = \{ m_1, m_2, m_3 \} \]
\[ m_1 : \{ S \rightarrow ABC \} \]
\[ m_2 : \{ A \rightarrow aA, B \rightarrow bB, C \rightarrow cC \} \]
\[ m_3 : \{ A \rightarrow a, B \rightarrow b, C \rightarrow c \} \]

\( G_M \) generates the language:
\[ L(G_M) = \{ a^n b^n c^n : n \geq 1 \} \]

1.4.1 Vertical parallelism
For the grammar \( G_M \) the derivation tree for the string \( a^n b^n c^n \) is shown in figure 2. The vertical dash lines show the vertical parallelism from the matrix grammar \( G_M \)
2 Bi-Parallel Grammar

If context-free grammars have horizontal parallelism (section 1.1) and matrix grammars have vertical parallelism (section 1.4.1) my proposition is to combine the two parallelisms and to obtain a bi-parallel grammar with horizontal and vertical parallelism.

For the matrix grammar of example 1.6 the derivation tree for the string \( a^2b^2c^2 \), tree with bi-parallel properties is shown in figure 3.

![Derivation tree for \( a^2b^2c^2 \) in \( G_M \) of example 1.2](image)

Fig 2. Derivation tree for \( a^2b^2c^2 \) in \( G_M \) of example 1.2

The dash lines show horizontal and vertical parallelism from \( G_M \). This bi-parallel grammar may be divided horizontal and vertical. The items \( G_1', G_2', G_3', G_4' \) bordered with squares are the grammars in witch the grammar \( G_M \) may be divided. This splitting will be done in two steps: horizontal splitting and then vertical splitting.

Horizontal splitting process is done in a top-down manner by a mechanism from compound grammars. We think that the top-down manner is preferred to bottom-up manner because in a derivation tree with not balanced branch, not all the terminal have the same hierarchic level. The superscript index \( i \) of grammar notation \( G_i' \) denotes the horizontal hierarchic level of that grammar.

From horizontal splitting step the grammars resulted are also matrix grammars.

Vertical splitting is done by dividing the matrix rules in vertical branches, branches that contains the same nonterminals. The resulted grammars are context free-grammars or matrix grammars.

2.1 Horizontal splitting

Horizontal splitting is done in a top-down manner like in section 1.2. In addition one we propose a new operator \( GMC \) (Grammar Matrix Constructor) defined below:

\[
GMC(N') = \{ m_k : m_k \in M \text{ and there exist } A \rightarrow w \in m_k \text{ with } A \in N' \}
\]

(25)

The operator \( GMC(N') \) takes from the initial matrix set \( M \), the matrices \( m_k \) that contains rewriting rules with right-hand nonterminals included in the set \( N' \). A generic matrix grammar \( G_M = (N,T,S,M) \) will be split in \( hN \) matrix grammars \( G_i' = (N_i', T_i', S_i', M_i) \), \( 1 \leq i \leq hN \) until \( N_{hN+1} \) is empty.

Formal construction of the horizontal grammars \( G_i' = (N_i', T_i', S_i', M_i) \) is proposed below:

\[
N_i' = \{ S \}
\]

(26)

\[
M_i' = GMC(N_i')
\]

\[
T_i' = \{ x : x \in (N \cup T) \text{ and there exist } S \rightarrow u_i x u_2 \in m_k \text{ with } m_k \in M_i' \} \cup \{ u_1, u_2 \in (N \cup T)^* \}
\]

\[
S_i' = \{ S \}
\]

and for \( 1 \leq i \leq hN \)

\[
N^{i+1} = \{ A : A \in N \text{ with } N^{i+1} \text{ contains rewriting rules with right-hand nonterminals included in the set } N' \text{ and there exist } A \rightarrow u_1 A u_2 \in m_k \text{ with } m_k \in M^{i+1}_i \text{ and } u_1, u_2 \in (N \cup T)^* \text{ such that } A_i \in N_i' \}
\]

\[
M^{i+1}_i = GMC(N^{i+1}_i)
\]

\[
T^{i+1}_i = \{ x : x \in (N \cup T) \text{ and there exist } A \rightarrow u_1 x u_2 \in m_k \text{ with } m_k \in M^{i+1}_i \text{ and } u_1, u_2 \in (N \cup T)^* \}
\]

\[
S^{i+1}_i = N^{i+1}_i
\]

Because always \( S^{i+1}_i = N_i' \), We propose a briefly notation of a such matrix grammar:

\[
G_i' = (N_i', T_i', M_i')
\]

(27)

Horizontal splitting for grammar of example 1 generates \( hN=2 \) matrix grammars:

\[
G_1' = (N_1', T_1', M_1')
\]

(28)

\[
N_1' = \{ S \}
\]

\[
M_1' = \{ S \rightarrow ABC \}
\]

\[
T_1' = \{ A, B, C \}
\]

and

\[
G_2' = (N_2', T_2', M_2')
\]

\[
N_2' = \{ A, B, C \}
\]

\[
M_2' = \{ m_2, m_3 \}
\]

\[
m_2 : [A \rightarrow a, B \rightarrow b, C \rightarrow c]
\]

\[
m_3 : [A \rightarrow a, B \rightarrow b, C \rightarrow c]
\]

\[
T_2' = \{ a, b, c \}
\]
The grammars $G^1$ and $G^2$ are horizontal parallel. According to section 1.2.1 one inherits the context from $G^2$ to $G^1$ by generating strings starting from $G^2$ and next from $G^1$.

Even the derivation process is done in parallel by the horizontal grammars the final string may be reconstructed in a top-down manner by two ways: trace pooling and substitution. The first method is presented in detail in [10].

### 2.1.1 Reconstruction by substitution
A faster way to reconstruct the final string is to assemble the strings generated by each horizontal grammar. For that every grammar $G^i$ with $i > 1$ must generate strings containing $ki$ sub-strings delimited (with ; for example), one sub-string from each non-terminal of $N$ (the initial non-terminal set of $G_{0i}$), with $ki$=card($N^i$) the number of non-terminals of $G^i$ as is shown below:

$$N^i = \{ A_1, A_2, ..., A_k \}$$

$$L(G^i) = \{ w_{ij}/w_{ij}, ..., w_{k1} : w_{ij} \in (T^*)^i \text{ and there exist } A_i \Rightarrow_{Gi} w_{ij}, A_i \in N^i \text{ for } 1 \leq j \leq ki \}$$

For a proper reconstruction the initial nonterminals set $N$ must be ordered. Also the nonterminals set of each horizontal grammars $N^i$ must preserve the same initial order. These restrictions make that the strings generated by derivation processes $A_i \Rightarrow_{Gi} w_{ij}$ (that is $w_{ij}$) will be placed in the same order for all the horizontal grammars.

The final assembly process consist of several substitution $s_i$, $i > 1$ defined below:

$$s_i : (T^{i-1})^i \rightarrow (L_{Gi} \cup T)^i$$

$$s_i(x) = s_i(y) \quad x, y \in (T^{i-1})^i$$

$$s_i(a) = a \quad \text{for } a \in T$$

$$s_i(a) = w_{ij} \quad \text{if } a = A_i \in N^i$$

The language generated by such horizontal hierarchy is:

$$L_H = s_{ln}(s_{ln-1}(...s_2(L(G_1)),...)) = L(G_M)$$

The first substitution $s_2$ is applied to the string generated by $G^1$, so result a new string. To that string is applied $s3$ and result a new string and so on until the last grammar $G^i$. Also an example of this method is presented in [10].

### 2.2 Vertical splitting
After the horizontal splitting process, one results two or many horizontal grammars. The horizontal grammars that have more than one rule in the matrix rules may be vertical splits.

In the above example, the grammar $G^i$ has no parallel rules and obvious only the grammar $G^2$ may be vertical divided. In figure 3 the grammar $G^2$ is divided in three grammars $G^2_1$, $G^2_2$, $G^2_3$. In this notation mode the superscript index is the horizontal level of the grammar and the subscript index is the vertical number of the grammar.

For a some $k$ horizontal level, with $G^i = (N^i, T^i, M^i)$ the horizontal grammar from $k$ level, $N^i = A_{i1}, A_{i2}, ..., A_{in}$ and $M^i = \{ m_1, m_2, ..., m_{in} \}$, the construction process of the vertical grammars are done in a left to right manner i.e. $G_{ij}^1, G_{ij}^2, ..., G_{ij}^{vn}$, with $vn$ the maximum number of vertical grammars for level $k$. For each $G_{ij}^j$, $1 \leq j \leq vn$ the construction begin with the nonterminals set $N^j_i$ then the terminal set $T^j_i$ and the matrix set $M^j_i$.

The first set $N^j_i$ shall contain:
- the first nonterminal $A_i$ of $N^i$
- those nonterminals of $N^i$ that occurs in the right side of rewriting rules with $A_i$ in the left side (the vertical “branches” communication). These rewriting rules occur in the matrix set $M^j_i$.

The second appending step of nonterminals shall be redone with all $A_i$ from $N^j_i$ and so on until in the resulting set does not appear new nonterminals.

We denoted with $GNC(N^i)$ (Grammar Nonterminals Constructor) the operator that make the second append step for a given some set $N^j_i$ and with $GNC^v(N^i_j)$ the repeated application of $GNC$ for resulted sets until there are not new resulting sets:

$$GNC(N^i) = \{ A_i : A_i \in N^i \text{ and there exist } A_i \in N^i \text{ and } m \in M \text{ and } A_i \Rightarrow_{Gi} m, A_i \in N^i \}$$

$$GNC^v(N^i) = GNC(GNC^v_{n-1}(N^i))$$

$$GNC^v_{n-1}(N^i) = GNC^v(N^i)$$

In this manner all the vertical “branches” that communicate are kept together.

The first terminals set $T^i_1$ contain all the terminals that occurs in the right hand of rewriting rules with the left hand $A_i \in N^i$. The matrix set $M^i_1$ contains only matrices that contain rewriting rules with left hand $A_i \in N^i$ and from these matrices only those rules.

The set $N^k_1$ shall contain the next nonterminal of $N^k$ that is not contained in $N^k_1$ and $GNCw(N^k_2)$. The sets $T^k_1$ and $M^k_1$ are constructed similar with $T^k_1$ and $M^k_1$.

The construction process of $G_{ij}^1$ continues until the set $N^j_i$ is empty.

The formal construction of the vertical grammars $G_{ij}^i = (N^i_{ij}, T^i_{ij}, M^i_{ij})$ is shown below:

$$N^i_{ij} = \{ A_i \} \cup GNC^v(N^i_{ij})$$

$$T^i_{ij} = \{ a_k : a_k \in T^k \text{ and there exist } A_i \in N^k \text{ and } m \in M \text{ and } A_i \Rightarrow_{Gi} m, u, u \in (N_{i1}^k \cup T^k_{ij}) \}$$
\[ M_i = \{ m_{ij} \mid m_{ij} = \{ A_i \rightarrow w_i : A_i \in N^k \land A_i \rightarrow w_i \in m_i \land m_i \in M_k \} \} \]

and for \( k \geq 2 \)

\[ N_k = \{ A_i \mid A_i \in N^k \land (N_1 \cup N_2 \cup \ldots \cup N_{k-1}) \cup G^k \} \]

\[ T_k = \{ a_k : a_k \in T^k \land \text{and there exist } A_i \in N^k \land m_i \in M_k \land A_i \rightarrow u_i \rightarrow \underbrace{m_u \in M_k}_{\text{and } u \in (N^k \cup T^k)^*} \} \]

The horizontal grammar \( G^k \) from (29) may be split in three vertical grammars \( G^i_1, G^i_2, G^i_3 \) by formulas (35) so:

\[ G^i_1 = (N^i_1, T^i_1, M^i_1) \text{ with:} \]

\[ N^i_1 = \{ A \} \]

\[ M^i_1 = \{ m_{21}, m_{31} \} \]

\[ m_{21} : [A \rightarrow aA] \]

\[ m_{31} : [A \rightarrow a] \]

\[ T^i_1 = \{ a \} \]

\[ G^i_2 = (N^i_2, T^i_2, M^i_2) \text{ with:} \]

\[ N^i_2 = \{ B \} \]

\[ M^i_2 = \{ m_{22}, m_{32} \} \]

\[ m_{22} : [B \rightarrow bB] \]

\[ m_{32} : [B \rightarrow b] \]

\[ T^i_2 = \{ b \} \]

\[ G^i_3 = (N^i_3, T^i_3, M^i_3) \text{ with:} \]

\[ N^i_3 = \{ C \} \]

\[ M^i_3 = \{ m_{23}, m_{33} \} \]

\[ m_{23} : [C \rightarrow cC] \]

\[ m_{33} : [C \rightarrow c] \]

\[ T^i_3 = \{ c \} \]

The grammars \( G^1_1, G^1_2, G^1_3 \) from this example are context-free, but in general case (vertical “branches” are communicating) these grammars are also matrix grammars.

### 2.2.1 Synchronization of vertical grammars

For a proper string generating process, the vertical grammars \( G^j_k \) must execute the derivation steps in a synchronous mode. The grammars must do all the rewriting rules from the same matrix \( m_i \in M_k \) and all \( G^j_k \) must follow the same order of matrices \( m_j \).

We propose a synchronization mechanism that permit to every vertical grammar \( G^j_k \) from a level \( k \) to done the derivation steps in parallel with the other grammars from the same level. For that mechanism, it is necessary for every level \( k \), a common trace-list (chained or linear list) of the matrices \( m_j \) used by the vertical grammars of a level \( k \) in derivation steps. We denoted this matrix trace-list with \( lk \). The elements of \( lk \) are of the form:

\[ \text{(step, matrix)} \]

where step is step number of derivation process and matrix is the matrix number used in step. For a matrix \( m_j \) used in some step of derivation process, matrix value is \( i \) (the first index), that is the initial matrix number of horizontal grammar \( G^j_k \) (even the initial matrix number of the main matrix grammar \( G_k \)).

The elements of list \( lk \) are appended always to the end of the list. The search of a some element \( (\text{step, matrix}) \) is performed from the head to tail of \( lk \).

We denote \( lk \).step the step from the last item appended to \( lk \).

In addition every grammar \( G^j_k \) has its own state item \( (\text{step}) \) that will show the last derivation step in \( G^j_k \). We denote with \( j \).step the last state of \( G^j_k \).

Initial \( j \).step is set to zero. After each derivation step \( j \).step is incremented.

For a state \( j \).step, an immediate derivation with matrix \( m_{ij} \in M^i_j \) holds in \( G^j_k \) in the state \( j \).step+1 if on on only if:

- \( lk \) is empty: \((j \).step+1, i) is appended to \( lk \) or \((37)\)
- \( lk \).step>j.step and there exist \((j \).step+1, i) \( \in \) \( lk \) or \( lk \).step=j.step : \((j \).step+1, i) appended to \( lk \)

If the string generated after \( j \).step in \( G^j_k \) contain nonterminals and \( lk \) is not empty (some vertical parallel grammar has done one or more derivation steps) and \( lk \).step>j.step and there exist \((j \).step+1, i) \( \in \) \( lk \), then \( j \).step is incremented and one try a new the derivation in step \( j \).step+1.

The language generated by vertical parallel grammars must be on the form \( L(G^j_k) \) from (32):

\[ L(G^j_k) = (L(G^1_k); L(G^2_k); \ldots ; L(G^k_k)) \]

For the grammars \( G^1_1, G^2_2, G^3_3 \) from (36) and for string \((a^* ; b^* ; c^*) \) the final matrix trace-list is:

\[ (1, 2, 3) \]

All the grammars must pass follow the states \( 0, 1 \) and \( 2 \).

### 3 Properties

Because the initial grammar is a matrix grammar, bi-parallel grammars \( G^j_k = (N^j_k, T^j_k, M^j_k) \) from section 2 and are also matrix grammars but without a start nonterminal \( S \). The language \( L_{BP} \) generated by such hierarchy of bi-parallel grammars is a matrix grammar language \( L_{BP} = L_M \) and in Chomsky hierarchy:

\[ L_2 \subseteq L_{BP} \subseteq L_1 \]

The bi-parallel grammars are smaller than the initial matrix grammars and therefore easier to manage.
The main property is that the bi-parallel grammars work in parallel and asynchronous. These grammars generate separate strings. The final reconstruction step generates the whole string, in a proper order.

4 Restrictions
Horizontal splitting and formula (26) for \( N^{i+1} = \{ A : A \in N( N^i \cup N^2 \cup \ldots \cup N^j) \) and there exist \( A_i \rightarrow u_1 A u_2 \in m_k \) with \( m_k \in P_j, u_1, u_2 \in (N \cup T)^* \) such that \( A_i \in N^{-1} \) does not work properly if there exist \( A \rightarrow u A_k u_2 \in m_k \) with \( A \in N \) and \( A_k \in N \) with \( j \leq i \) (the horizontal levels communicate). In that case \( A \) must be appended to \( N' \) and redone the construction of the grammars \( G_k \) for \( j \geq k \geq i \).

For a proper reconstruction of the strings generated by the horizontal hierarchy grammars, the initial nonterminals set must be ordered. Also the nonterminals set of horizontal grammars must preserve the same initial order.

5 Applicability
The horizontal parallel grammars may be used in hierarchic systems (management, economic processes) with horizontal and vertical parallelism.

The horizontal splitting can be used as a decomposition function for hierarchic system (decomposition of a decision/action). The final reconstruction can be used as an aggregate function (composition of a decisions/actions) in hierarchic systems presented in [6].

Production planning where parts of a product are manufactured in parallel is a good example of modeling by a bi-parallel grammar. An application is in implemented state these months.

The strings generating process may be done (faster and safe) in a MIMD (Multiple Instruction Multiple Data) parallel environment.

References: