Several Aspects of Context Freeness for Hyperedge Replacement Grammars

SILVIU DUMITRESCU Department of Informatics Transilvania University of Brasov Iuliu Maniu 50 ROMANIA <u>s.dumitrescu@info.unitbv.ro</u> http://cs.unitbv.ro

Abstract: - In this paper we survey several aspects related to normal forms of hyperedge replacement grammars. Considering context free hyperedge replacement grammars we introduce, inspired by string grammars, Chomsky Normal Form and Greibach Normal Form. The algorithm of conversion is quite the same with the algorithm for string grammars. The important difference is related to the fact that hyperedge grammars are two-dimensional and that's why parsing productions, in order to transform into string grammars, can be done only nondeterministic. A detailed example of conversion to both normal forms is introduced to clarify all the algorithm steps.

Key-Words: - Hyperedge Replacement Grammars, Normal Form, Chomsky, Greibach, Context Freeness, Nondeterministic

1 Introduction

In many fields of computer science, the information is represented by diagrams rather than strings. That's why a study in domain of graphs and formalizations of graphs could be very interesting. A hypergraph represents a generalized graph and consists by a number of hyperedges [2]. A hyperedge is an atomic item labeled with a label in a nonempty set, called alphabet, and a fixed number of tentacles. On each tentacle is attached a node. Nodes are involved in hyperedge replacement. With labeled hyperedges we can define productions. Productions consist of a label as left hand side and a replacing structure as right hand side. If a labeled hyperedge, with the left hand side of a production is replaced with the right hand side, then this is called direct derivation. So, we can define a language as a set of structures derivable from the start structure.

In this paper we consider the alphabet of labels divided into two disjoint sets: the alphabet of terminals, which labels only structures as right hand side of some productions, and the alphabet of nonterminals, which labels structures as both sides of productions, same as in string grammars is.

Some hyperedge grammars have only one set of labels [3]. In that case the set of nonterminals is empty and the terminal structures are not labeled. In this grammars derivations could be maximum parallel such as are in Lindenmayer systems. The languages generated by such grammars include visual structures like fractals [8], because the grown take place in all directions in the same time. With hyperedge replacement grammars we can generate digital images or we can recognize images [12].

In this paper all the grammars considered are context free. So, it does not matter how we choose the starting hyperedge in the replacement and it is not relevant how many times we repeat the replacement, but it's important to have, in each step of the derivation, a production where the label of the replaced hyperedge exists on its left side.

In the main section of this paper we'll consider a grammar without λ -productions and without rewritings. As it's shown in [4] this could be done. The algorithm is nondeterministic, that means it doesn't matter how we'll split the left side of the production because the choice doesn't influence the result. Parsing has different aspects as we can see in [1] or [11].

2 Problem Formulation

2.1 Definitions and notations

In this section, we recall the basic notions and results on hyperedge replacement.

It is well known that a graph is a pair G = (V, E), where V is a set of nodes and E is a set of 2-element subsets of V, called edges.

Definition 1: [5] Hypergraph - a tuple (V, E, att, lab, ext) where V is the finite set of nodes, E is the finite set of hyperedges, att: $E \rightarrow$

 V^* is the application of attaching, which assigns a sequence of pair wise distinct nodes to every hyperedge, lab: $E \rightarrow C$ is the application of labeling, which assigns a label to every hyperedge from arbitrary but fixed and not empty set C, and $ext \in V^*$ is a sequence of pairwise distinct external nodes.

In this paper we denote by \aleph_C the set of hypergraphs over C.

Definition 2: [5] Type of a hyperedge – type: $C \rightarrow N$, type(lab(e)) = |att(e)|, $e \in E$, $E \in H$, $H \in \aleph_C$.

We denote by type(H), the type of the hypergraph $H \in \aleph_C$, and understand the number of external nodes.





In Fig. 1 we represent a hypergraph with: $V = \{v_1, v_2, v_3, v_4\}, E = \{e_1, e_2, e_3, e_4, e_5, e_6\}, att(e_1) = v_1v_4, att(e_2) = v_3v_4, att(e_3) = v_2v_3, att(e_4) = v_1v_2, att(e_5) = v_3, att(e_6) = v_1v_2v_3v_4, lab(e_1) = A, lab(e_2) = F, lab(e_3) = D, lab(e_4) = B, lab(e_5) = E, lab(e_6) = C, type(A) = 2, type(B) = 2, type(C) = 4, type(D) = 2, type(E) = 1, type(F) = 2. We consider e_1 as a 2-edge, e_2 as a 2-edge, e_3 as a 2-edge, e_4 as a 2-edge, e_5 as a 1-edge, e_6 as a 4-edge. The previous hypergraph has type 2.$

Definition 3: [5] Hyperedge Replacement Grammar – a system HRG = (N, T, P, S), where N is the set of nonterminals, T is the set of terminals, N and T are disjoint, $N \cup T \subseteq C$, P is the set of productions, P = $\{(A,R) | A \in N, R \in \aleph_C \text{ with type}(A) = type(R)\},$ and $S \in N$ is the starting symbol.

We denote by H[e|R] the hypergraph obtained from H replacing hyperedge e, $e \in H$,

by hypergraph R. Then replacing process is made by cutting the hyperedge e from H and adding the hypergraph R so that the i-th external node of R is glued over the i-th attached node of e with i = 1,type(e). Moreover, the external nodes of H[e|R] are the same with the once of H.

Definition 4: [5] Direct derivation using productions of P, H \Rightarrow H', H $\in \aleph_C$, if and only $(lab_H(e), R) \in P$ and H' = H[e|R].

A sequence of direct derivations of the form $H_0 \Rightarrow H_1 \Rightarrow ... \Rightarrow H_k$ is called derivation of length k.

The language generated by an hyperedge replacement grammar, HRG, is denoted by L(HRG), and represents all hypergraphs labeled in T and obtained starting with the hypergraph labeled with S using productions of P.

2.2 Context Freeness

We study in this paper the properties of hyperedge context free grammars. Intuitively this means, during derivation, at a specific step, starting with a hypergraph in which the hyperedges are labeled with nonterminals, applying a production depends only on the existence of a hyperedge labeled with a nonterminal and modifies nothing else from the initial hypergraph. Context freeness says something more, doesn't which matter hyperedge is the first one in the derivation process and which one is next.

Inspired from string grammars we defined in [4] a λ – production by a production (A, R) \in P where A \in N and R is a set of external nodes.

For each context free hyperedge replacement grammar there is an equivalent one λ -free. That was proved in [4]. λ - freeness means that the set of productions have no λ - productions or if have then the only λ – production has the starting symbol S but, S doesn't appear in any production as right hand side.

We say that a production $(A, R) \in P$ is a rewriting if the hypergraph R has only one hyperedge and the number of external nodes equals the number of attachment nodes. We can build an equivalent grammar without rewritings as is proved in [4]. In conclusion we can build a normalform inspired by Chomsky Normal Form.

Theorem 1: Chomsky Normal Form – for a hyperedge replacement grammar, HRG = (N, T, P, S), without rewritings and λ -free, there is an equivalent grammar HRGNF = (N₁, T, P₁, S) in Chomsky Normal Form. That means all productions in P₁ are of the form (A, H), where A \in N and |lab(e)| = 1, lab(e) \in T, e \in H or |lab(e)| = 2, lab(e_i) \in N, e_i \in H, i = 1,2.

3 Greibach Normal Form

In Greibach Normal Form for string grammars [6] each production has the right hand side starting with a terminal perhaps followed by some nonterminals.

The algorithm which builds a hyperedge replacement grammar in Greibach Normal Form has as input a hyperedge replacement grammar in Chomsky Normal Form, HRG = (N, T, P, S).

Step 1: In order to introduce Greibach Normal Form we define the set of A – productions.

Definition 5: For each nonterminal symbol, $A \in N$, we define the set of A-productions using operator "|". Let (A, R_i) , $i = 1, n_A$, with R_i a hypergraph having exact 2 hyperedges labeled in N or exact one hyperedge labeled in T, be all productions in P with variable A on the left. We define the set of A – productions by $(A, R_1 | R_2 | ... | R_{nA})$.

After transformation we have in HRG n sets of A-productions (A, $R_1 | R_2 | ... | R_{nA}$), for all $A \in N$, where |N| = n. Obvious HRG contains the same productions but reordered.

Step 2: In this step we give to each nonterminal label a rank, S to be A_1 and so on.

A hypergraph as right-hand side of a production can be described as an ordered string of two nonterminals, A_iA_j , with $i \le j$, or as a string of one terminal. In the first case this could be done by parsing the hypergraph starting with the label of minimum index and continuing with the other one.

After that, in all sets of productions, starting with A_1 and proceeding to A_n we modify productions such as if $A_i \rightarrow A_j\gamma$ is a production, then j>i. Let say that we are in the

set of A_k – productions where we have $A_k \rightarrow A_j\gamma$ a production with j<k. We'll generate a new set of A_k - productions by substituting A_j with the right-hand side of each production from the set of A_j – productions. Let $A_j \rightarrow \beta_1 | \beta_2 | \dots | \beta_{nAj}$ be the set of A_j – productions. The new set of A_k - productions will be $A_k \rightarrow \beta_1\gamma | \beta_2\gamma | \dots | \beta_{nAj}\gamma$. By repeating the process k-1 times, at most, we obtain productions of the form $A_k \rightarrow A_{i}\gamma$ with $l \ge k$ or starting with a terminal. It's quite obvious that the new set of productions generate the same language.

Step 3: In this step we'll replace all the productions $A_k \rightarrow A_l\gamma$, with l = k. An arbitrary set of A – productions is divided into two subsets. Let $A \rightarrow A\alpha_1 | A\alpha_2 | ... | A\alpha_r$ be the subset of A - productions for which A is the leftmost symbol of the right-hand side and $A \rightarrow \beta_1 | \beta_2 | ... | \beta_s$ be the remaining subset of A – productions. We construct a new hyperedge replacement grammar, $HRG_1 = (N \cup \{B\}, T, P_1, S)$, by adding the symbol B to V and replacing all productions from the set of A – productions by: (1) $A \rightarrow \beta_i, A \rightarrow \beta_i B$, i = 1, s and (2) $B \rightarrow \alpha_i, B \rightarrow \alpha_i B$, i = 1, r.

Lemma 1: $L(HRG) = L(HRG_1)$.

Proof: " \subseteq " We consider in G the sequence of replacements: $A \Rightarrow A\alpha_{i1} \Rightarrow A\alpha_{i2}\alpha_{i1}$ $\Rightarrow \dots \Rightarrow A\alpha_{ip}\alpha_{ip-1\dots}\alpha_{i1} \Rightarrow \beta_j\alpha_{ip}\alpha_{ip-1\dots}\alpha_{i1}$. This sequence can be replaced in G₁ by: $A \Rightarrow \beta_j B \Rightarrow$ $\beta_j\alpha_{ip}B \Rightarrow \beta_j\alpha_{ip}\alpha_{ip-1}B \Rightarrow \dots \Rightarrow \beta_j\alpha_{ip}\alpha_{ip-1\dots}\alpha_{i2}B \Rightarrow$ $\beta_j\alpha_{ip}\alpha_{ip-1\dots}\alpha_{i1}$.

" \supseteq " In the same way we can proof the reverse transformation. §

We repeat the above process for each variable and finally we have only productions by the forms: (1) $A_i \rightarrow A_j\gamma$, with j > i, (2) $A_i \rightarrow a\gamma$, with $a \in T$ or (3) $B_i \rightarrow \gamma$, with $\gamma \in (N \cup \{B_1, B_2, ..., B_{i-1}\})^*$.

Step 4: In this step we transform all sets of A_i – productions, i = 1,n, such as the right side of each production starts with a terminal symbol. The process begins with the set of A_n – productions. Since A_n is the highest-numbered variable, the leftmost symbol on the right-hand side of any production for A_n is a terminal. We continue with all sets of A_i – productions, i = n-1,1. All these productions have the leftmost symbol, on the right-side, a terminal or a nonterminal of rank greater than i. We replace nonterminal symbol by the right-hand side of the productions corresponding to the set of A_j productions, j = i+1,n. The grammar resulting from this step generates the same language as the initial grammar. We proved that in step 2.

Step 5: In this step we exam only the productions for the variables $B_1, B_2, ..., B_n$. Because HRG is in Chomsky Normal Form and because of previous transformations, we have, in all B_i sets, productions by the forms: (1) $B_i \rightarrow A_i\gamma$, with γ not empty, or (2) $B_i \rightarrow a\gamma$, $a \in T$, i = 1, n. Now we have to apply again the step 4 for all productions having B_i as right-hand side.

Finally we have a new grammar GNF = (N_1, T, P_1, S) , where N_1 is the set of nonterminals having the nonterminals symbols from HRG and some new ones, T is the set of terminals having same symbols as HRG, S is the start symbol and P is the set of productions by the form $X \rightarrow a\alpha$, where X is a nonterminal symbol, a is a terminal symbol and α is a possibly empty string of nonterminal symbols.

Theorem 2: Greibach Normal Form – every context free language L without empty words can be generated by a grammar for which every production has the right-hand side formed by a terminal and a possibly empty string of nonterminals. As we proved above L(HRG) =L(GNF).

After the process of normalization the number of productions could be square than initial.

Example 1: This example will present the algorithms which transform a hyperedge replacement grammar into a Greibach Normal Form via Chomsky Normal Form.

We consider the grammar introduced in [4]. This grammar is special because it represents an example of generative power of hyperedge replacement grammars. As we have shown the generative power of context free hyperedge replacement grammars is grater than the generative power of context free string grammars.

Let HRG = $(\{S, A\}, \{a, b, c\}, P, S)$ be the grammar, with P = $\{(S, H_1|H_2), (A, H_3|H_4)\}$. In a graphic representation productions are:



Fig. 2

In Fig. 2 we have 4 hypergraphs defined

 $\begin{array}{ll} H_1 \colon & V_{H1} = \{v_1, v_2, v_3, v_4, v_5, v_6\}, \\ & E_{H1} = \{e_1, e_2, e_3, e_4\}, \\ & att_{H1}(e_1) = v_1 v_3, att_{H1}(e_2) = v_6 v_4, \\ att_{H1}(e_3) = v_6 v_5, att_{H1}(e_4) = v_3 v_4 v_5 v_2, \\ & lab_{H1}(e_1) = a, \ lab_{H1}(e_2) = b, \\ lab_{H1}(e_3) = c, \ lab_{H1}(e_4) = A, \\ & ext_{H1} = v_1 v_2. \end{array}$

 $\begin{array}{ll} H_2: & V_{H2} = \{v_1, v_2, v_3, v_4\}, \\ & E_{H2} = \{e_1, e_2, e_3\}, \\ & att_{H2}(e_1) = v_1v_2, att_{H2}(e_2) = v_2v_3, \\ att_{H2}(e_3) = v_3v_4, \\ & lab_{H2}(e_1) = a, \ lab_{H2}(e_2) = b, \\ lab_{H2}(e_3) = c, \\ & ext_{H2} = v_1v_4. \end{array}$

$$\begin{array}{ll} H_3: & V_{H3} = \{v_1,\,v_2,\,v_3,\,v_4,\,v_5,\,v_6,\,v_7\},\\ & E_{H3} = \{e_1,\,e_2,\,e_3,\,e_4\},\\ & att_{H3}(e_1) \,=\, v_1v_5,\,att_{H3}(e_2) \,=\, v_2v_6,\\ att_{H3}(e_3) = v_3v_7,\,att_{H3}(e_4) = v_5v_6v_7v_4,\\ & lab_{H3}(e_1) \,=\, a,\ lab_{H3}(e_2) \,=\, b,\\ lab_{H3}(e_3) = c,\,lab_{H3}(e_4) = A, \end{array}$$

by:

 $ext_{H3} = v_1v_2v_3v_4.$

$$\begin{array}{ll} H_4: & V_{H4} = \{v_1, v_2, v_3, v_4, v_5\}, \\ & E_{H4} = \{e_1, e_2, e_3\}, \\ & att_{H4}(e_1) = v_1 v_2, att_{H4}(e_2) = v_2 v_3, \\ & att_{H4}(e_3) = v_4 v_5, \\ & lab_{H4}(e_1) = a, \ lab_{H4}(e_2) = b, \\ & lab_{H4}(e_3) = c, \\ & ext_{H4} = v_1 v_3 v_4 v_5. \end{array}$$

A derivation in this grammar is shown in [4]. The language generated by this grammar is $L(HRG) = \{(a^nb^nc^n)^{\bullet}, n \ge 1\}$. We denote by $(a^nb^nc^n)^{\bullet}$ the hypergraph labeled in nonterminal set and parsed linearly. Note: the result graph is a linear graph that's why parsing is deterministic.

First, we transform this grammar into an equivalent Chomsky Normal Form. It is obvious that the HRG grammar is λ – free and without rewritings. The definitions for λ – free and rewritings are introduced in [4]. Also there you can find the algorithms to eliminate λ - productions and rewritings.

Step 1.1. We may begin by replacing terminals on the right with new nonterminals. With these new nonterminals we make new productions. The resulting set of productions is:





Step 1.2. In this step we replace the productions longer than 3 variables with corresponding number of productions where each member from right hand side has exactly 2 variables. The productions involved are in set of S – productions and A – productions. We introduce 6 more productions such as the final sets of productions are:















In Fig. 4 we have the graphic representation of 10 sets of productions which define a hyperedge replacement grammar in Chomsky Normal Form $CNF = (N_C, T, P_C, S)$ with $N = \{S, A, A_a, A_b, A_c, A_1, B_1, C_1, D_1, E_1\}$, $T = \{a, b, c\}$ and $P_C = \{(S, H_1|H_2), (A_1, H_3), (B_1, H_4), (A, H_5|H_6), (C_1, H_7), (E_1, H_8), (D_1, H_9), (A_a, H_{10}), (A_b, H_{11}), (A_c, H_{12})\}.$

We can make a nondeterministic linearization of P_C which leads to string productions. A result of this is shown bellow:

$$\begin{split} S & \rightarrow A_a A_1 \mid A_a B_1 \\ A_1 & \rightarrow A B_1 \\ B_1 & \rightarrow A_b A_c \\ A & \rightarrow A_a C_1 \mid D_1 A_c \\ C_1 & \rightarrow E_1 A_b \\ E_1 & \rightarrow A A_c \\ D_1 & \rightarrow A_a A_b \\ A_a & \rightarrow a \\ A_b & \rightarrow b \end{split}$$

 $A_c \mathop{\rightarrow} c$

In the next stage we transform this grammar into an equivalent one in Greibach Normal Form, accordingly to the algorithm of normalization.

Step 2.1. In G we already have ten sets of A – productions.

Step 2.2. We denote variables as follows: S with S_1 , A_a with S_2 , A_1 with S_3 , B_1 with S_4 , A with S_5 , A_b with S_6 , A_c with S_7 , C_1 with S_8 , D_1 with S_9 and E_1 with S_{10} . So, P_C has the following productions:

$$\begin{array}{l} S_1 \rightarrow S_2 S_3 \mid S_2 S_4 \\ S_2 \rightarrow a \\ S_3 \rightarrow S_5 S_4 \\ S_4 \rightarrow S_6 S_7 \\ S_5 \rightarrow S_2 S_8 \mid S_9 S_7 \\ S_6 \rightarrow b \\ S_7 \rightarrow c \\ S_8 \rightarrow S_{10} S_6 \\ S_9 \rightarrow S_2 S_6 \\ S_{10} \rightarrow S_5 S_7 \end{array}$$

Since the right-hand side of productions for S₁, S₂, S₃, S₄, S₆, S₇, S₈ start with terminal or higher-numbered variable, we focus of the productions S₅ \rightarrow S₂S₈, S₉ \rightarrow S₂S₆, S₁₀ \rightarrow S₅S₇ and substitute the left-most appearance of S₂ with the right-hand side of S₂ – productions and of S₅ with the right-hand side of S₅ – productions, respectively.

The resulting set of productions is:

$$\begin{split} S_1 &\rightarrow S_2 S_3 \mid S_2 S_4 \\ S_2 &\rightarrow a \\ S_3 &\rightarrow S_5 S_4 \\ S_4 &\rightarrow S_6 S_7 \\ S_5 &\rightarrow a S_8 \mid S_9 S_7 \\ S_6 &\rightarrow b \\ S_7 &\rightarrow c \\ S_8 &\rightarrow S_{10} S_6 \\ S_9 &\rightarrow a S_6 \\ S_{10} &\rightarrow a S_8 S_7 \mid S_9 S_7 S_7 \end{split}$$

The only production which doesn't respect the rules for this step is $S_{10} \rightarrow S_9S_7S_7$. So, we have to substitute the left-most appearance of S_9 with the right-hand side of S_9 – production. The last production became:

$$S_{10} \rightarrow aS_8S_7 \mid aS_6S_7S_7$$

Step 2.3. We don't have to take care about this step because we don't have productions where the variable from the left hand side has the same number as the leftmost variable from the right hand side.

Step 2.4. S_9 – productions and S_{10} – productions start with terminals. These are used in previous productions. The result is the following:

$$\begin{split} S_1 &\rightarrow S_2 S_3 \mid S_2 S_4 \\ S_2 &\rightarrow a \\ S_3 &\rightarrow S_5 S_4 \\ S_4 &\rightarrow S_6 S_7 \\ S_5 &\rightarrow a S_8 \mid a S_6 S_7 \\ S_6 &\rightarrow b \\ S_7 &\rightarrow c \\ S_8 &\rightarrow a S_8 S_7 S_6 \mid a S_6 S_7 S_7 S_6 \\ S_9 &\rightarrow a S_6 \\ S_{10} &\rightarrow a S_8 S_7 \mid a S_6 S_7 S_7 \end{split}$$

The set of S_6 - production is involved in S_4 – production. So, S_6 variable is replaced from the leftmost position of S_4 – production. The set of S_5 - production is involved in S_3 – production. So, S_5 variable is replaced from the leftmost position of S_3 – production. The set of S_2 - production is involved in S_1 – production. So, S_2 variable is replaced from the leftmost position of S_1 – production. The set of S_2 - production is involved in S_1 – production. So, S_2 variable is replaced from the leftmost position of S_1 – production. The new set of productions is now:

$$\begin{array}{l} S_1 \to aS_3 \mid aS_4 \\ S_2 \to a \\ S_3 \to aS_8S_4 \mid aS_6S_7S_4 \\ S_4 \to bS_7 \\ S_5 \to aS_8 \mid aS_6S_7 \\ S_6 \to b \\ S_7 \to c \\ S_8 \to aS_8S_7S_6 \mid aS_6S_7S_7S_6 \\ S_9 \to aS_6 \\ S_{10} \to aS_8S_7 \mid aS_6S_7S_7 \end{array}$$

Step 2.5. All the productions are starting in the right hand side with a terminal and

continuing with some nonterminals. Because we don't have new variable symbols introduced into the process of normalization the Step 5 is not required.

Thus, the equivalent grammar with G, in Greibach Normal Form, is $GNF = (N_1, T, P_1, T)$ S_1), where $N_1 = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}\}$ and P_1 is the previous set of string productions. In terms of hyperedge replacement grammars these can be written as follows: $\{(S_1,$ $H_1|H_2$), (S₂, H₃), (S₃, H₄|H₅), (S₄, H₆), (S₅, $H_7|H_8$), (S₆, H₉), (S₇, H₁₀), (S₈, H₁₁|H₁₂), (S₉, H_{13}), $(S_{10}, H_{14}|H_{15})$ and graphic represented bellow:















Fig. 5

In Fig. 5 we have 15 hypergraphs defined by:

- $\begin{array}{ll} H_1: & V_{H1} = \{v_1, v_2, v_3, v_4, v_5\}, \\ & E_{H1} = \{e_1, e_2\}, \\ & att_{H1}(e_1) = v_1 v_2, att_{H1}(e_2) = v_3 v_4 v_5 v_2, \\ & lab_{H1}(e_1) = a, \ lab_{H1}(e_2) = S_3, \\ & ext_{H1} = v_1 v_5. \end{array}$
- $\begin{array}{ll} H_2: & V_{H2} = \{v_1,\,v_2,\,v_3\}, \\ & E_{H2} = \{e_1,\,e_2\}, \\ & att_{H2}(e_1) = v_1v_2,\,att_{H2}(e_2) = v_2v_3, \\ & lab_{H2}(e_1) = a,\,lab_{H2}(e_2) = S_4, \\ & ext_{H2} = v_1v_3. \end{array}$

$$\begin{array}{ll} H_3: & V_{H3} = \{v_1, v_2\}, \\ & E_{H3} = \{e_1\}, \\ & att_{H3}(e_1) = v_1v_{2,} \\ & lab_{H3}(e_1) = a, \\ & ext_{H3} = v_1v_2. \end{array}$$

$$\begin{array}{ll} H_4: & V_{H4} = \{v_1, v_2, v_3, v_4, v_5\}, \\ & E_{H4} = \{e_1, e_2, e_3\}, \\ & att_{H4}(e_1) = v_1 v_2, \ att_{H4}(e_2) = \\ v_2 v_3 v_4 v_5, att_{H4}(e_3) = v_3 v_4, \\ & lab_{H4}(e_1) = a, \ lab_{H4}(e_2) = \\ S_8, \\ lab_{H4}(e_3) = S_4, \\ & ext_{H4} = v_1 v_3 v_4 v_5 \end{array}$$

 $\begin{array}{ll} H_5: & V_{H5} = \{v_1,\,v_2,\,v_3,\,v_4,\,v_5\}, \\ & E_{H5} = \{e_1,\,e_2,\,e_3,\,e_4\}, \\ & att_{H5}(e_1) \,=\, v_1v_2,\,att_{H5}(e_2) \,=\, v_2v_3, \\ att_{H5}(e_3) = v_3v_4,\,att_{H5}(e_4) = v_4v_5, \\ & lab_{H5}(e_1) \,=\, a,\ lab_{H5}(e_2) \,=\, S_6, \\ lab_{H5}(e_3) = S_4,\,lab_{H5}(e_4) = S_7, \\ & ext_{H5} = v_1v_3v_4v_5. \end{array}$

- $\begin{array}{ll} H_6: & V_{H6} = \{v_1,\,v_2,\,v_3\}, \\ & E_{H6} = \{e_1,\,e_2\}, \\ & att_{H6}(e_1) = v_1v_2,\,att_{H6}(e_2) = v_2v_3, \\ & lab_{H6}(e_1) = b, \ lab_{H6}(e_2) = S_7, \\ & ext_{H6} = v_1v_3. \end{array}$
- $H_7: \quad V_{H7} = \{v_1, v_2, v_3, v_4, v_5\}, \\ E_{H7} = \{e_1, e_2\}, \\ att_{H7}(e_1) = v_1 v_2, att_{H7}(e_2) = v_2 v_3 v_4 v_5, \\ lab_{H7}(e_1) = a, \ lab_{H7}(e_2) = S_8, \\ ext_{H7} = v_1 v_3 v_4 v_5.$

```
H<sub>8</sub>: V_{H8} = \{v_1, v_2, v_3, v_4\},\
```

 $E_{H8} = \{e_1, e_2, e_3\},\$ $att_{H8}(e_1) = v_1v_2$, $att_{H8}(e_2) = v_2v_3$, $att_{H8}(e_3) = v_3 v_4$, $lab_{H8}(e_1) = a, \ lab_{H8}(e_2) = S_6,$ $lab_{H8}(e_3) = S_7$, $ext_{H8} = v_1v_2v_3v_4$. H₉: $V_{H9} = \{v_1, v_2\},\$ $E_{H9} = \{e_1\},\$ $att_{H9}(e_1) = v_1 v_2$. $lab_{H9}(e_1) = b$, $ext_{H9} = v_1v_2$ H₁₀: $V_{H10} = \{v_1, v_2\},\$ $E_{H10} = \{e_1\},\$ $att_{H10}(e_1) = v_1 v_2$ $lab_{H10}(e_1) = c$, $ext_{H10} = v_1v_2$. H₁₁: $V_{H11} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\},\$ $E_{H11} = \{e_1, e_2, e_3, e_4\},\$ $att_{H11}(e_1) = v_1v_2$, $att_{H11}(e_2) = v_3v_4$, $att_{H11}(e_3) = v_6v_5$, $att_{H11}(e_4) = v_6v_4v_7v_2$, $lab_{H11}(e_1) = a, \ lab_{H11}(e_2) = S_6,$ $lab_{H11}(e_3) = S_7$, $lab_{H11}(e_4) = S_8$, $ext_{H11} = v_1v_3v_5v_7$ H_{12} : $V_{H12} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\},\$ $E_{H12} = \{e_1, e_2, e_3, e_4, e_5\},\$ $att_{H12}(e_1) = v_1v_2$, $att_{H12}(e_2) = v_2v_3$. $att_{H12}(e_3) = v_3v_4$, $att_{H12}(e_4) = v_5v_6$, $att_{H12}(e_5) = v_6 v_7$, $lab_{H12}(e_1) = a, \ lab_{H12}(e_2) = S_6,$ $lab_{H12}(e_3) = S_6$, $lab_{H12}(e_4) = S_7$, $lab_{H12}(e_5)$ $= S_{7}$ $ext_{H12} = v_1v_4v_5v_7$. H₁₃: $V_{H13} = \{v_1, v_2, v_3\},\$ $E_{H13} = \{e_1, e_2\},\$ $att_{H13}(e_1) = v_1v_2$, $att_{H13}(e_2) = v_2v_3$, $lab_{H13}(e_1) = a$, $lab_{H13}(e_2) = S_6$, $ext_{H13} = v_1v_3$, H_{14} : $V_{H14} = \{v_1, v_2, v_3, v_4, v_5, v_6\},\$ $E_{H14} = \{e_1, e_2, e_3\},\$ $att_{H14}(e_1) = v_1v_2$, $att_{H14}(e_2) = v_4v_5$, $att_{H14}(e_3) = v_3 v_2 v_5 v_6$ $lab_{H14}(e_1) = a, \ lab_{H14}(e_2) = S_8,$

 $lab_{H14}(e_3) = S_7,$

 $ext_{H14} = v_1v_3v_4v_6$.

$$\begin{array}{ll} H_{15} \colon & V_{H15} = \{v_1, v_2, v_3, v_4, v_5, v_6\}, \\ & E_{H15} = \{e_1, e_2, e_3, e_4\}, \\ & att_{H15}(e_1) = v_1v_2, att_{H15}(e_2) = v_2v_3, \\ & att_{H15}(e_3) = v_4v_5, att_{H15}(e_4) = v_5v_6, \\ & lab_{H15}(e_1) = a, \ lab_{H15}(e_2) = S_6, \\ & lab_{H15}(e_3) = S_7, \ lab_{H15}(e_4) = S_7, \\ & ext_{H15} = v_1v_3v_4v_6. \end{array}$$

A derivation in this grammar could be:











The word resulted by parsing the generated linear graph of this derivation is $a^4b^4c^4$.

During entirely derivation we kept the number of exterior nodes and those positions.

4 Conclusions

Context free hyperedge replacement grammars have a behavior very much like context-free Chomsky grammars. The important difference is related to transformation of a planar structure into a linear one. All the algorithms involved in transformations are nondeterministic. The reason why we introduced these normal forms for hyperedge replacement grammars is to make a step forward to create some recognizing machines like push down transducers for string grammars.

References:

[1] C. E. Aron, E. M. Popa, M. A. Musan, Recursive Programs Analysis using Context Free Languages, *WSEAS Transactions on Information Science & Applications*, Vol. 4, Issue 1, 2007, pg. 91 – 96.

[2] J. Dassow, A. Habel, St. Taubenger, *Chain-Code Pictures and Collages Generated by Hyperedge Replacement, Graph Grammars and Their Applications to Computer Science*, Springer-Verlag, 1996.

[3] F. Drewes, Tree Based Picture Generation, *Theoretical Computer Science*, No. 246, 2000, pg. 1-51

[4] S. Dumitrescu, Aspects of Context Freeness for Hyperedge Replacement Grammars, *Foundations of Computing and Decision Sciences*, Vol. 30, No. 2, 2005, pg. 91 – 101

[5] S. Dumitrescu, Visual Structures and Formal Languages, Transilvania University Press, 2006

[6] S. Dumitrescu, About Closer Properties of Hyperedge Replacement Grammars, Proceedings - Several Aspects of Biology, Chemistry, Computer Science, Mathematics and Physics, 2005, pg. 135 - 140

[7] S. Dumitrescu, About Normal Forms for Hyperedge Replacement Grammars, *Proceeding of the 12th WSEAS International Conference on Computers*, 2008, pg. 208 – 211.
[8] Deng Fang, Xi Li-Fneg, An Application of

L-system and IFS in 3D Fractal Simulation, WSEAS Transactions on Systems, Vol. 7, 2008, pg. 352 – 362.

[9] G. Rozenberg, Handbook of Graph Grammars and Computing by Graph Transformation, World Scientific, 1997

[10] A. Salomaa, *Formal Language*, Academic Press, 1973

[11] L. Sangeorzan, M. Popescu, E. Helerea, C. Aldea, S. Dumitrescu, W. Schwarz, M. Popescu, Aspects Regarding the Quality of Education on Process Based on Web 2.0 Technologies, *Proceeding of the 8th WSEAS International Conference on Applied Informatics and Communications*, 2008, pg. 499 – 505.

[12] P. Singh, S. Batra, HR Sharma, A Review of Digital Signatures and Status in India, *WSEAS Transactions on Computers*, Vol. 4, Issue 4, 2005, pg. 408 – 410.