Context-Free Graph Grammar

Kai Mittermüller[1]

[1] Universität des Saarlandes, 66123 Saarbrücken, Germany
E-mail: s9kamitt@stud.uni-saarland.de

Abstract. In this survey the concept of hyperedge replacement is presented as an elementary approach to graph and hypergraph generation. In particular, hyperedge replacement graph grammars are discussed as a (hyper)graph-grammatical counterpart to context-free string grammars. Basic properties of the context-free graph grammar are shown, like the pumping lemma. We also discuss hyperedge replacement grammars as generators for string languages and the power of those generators.

1 Hyperedge replacement grammars

Graph grammars have been developed as an extension of the concept of formal grammars on strings to grammars on graphs. Among string grammars context-free grammars have proved extremely useful in practical applications and powerful enough to generate a wide spectrum of interesting formal languages. It is therefore not surprising that analogous notions (context-freeness) have been developed also for graph grammars.
Hyperedge replacement is an elementary approach of graph and hypergraph rewriting. Hyperedge replacement is as powerful as hypernode replacement. In this survey we will only discuss hyperedge replacement.

Definition 1 (based on [1], p.131):

Let Γ be an alphabet of *edge labels*, and let Σ be an alphabet of *selectors*. A hypergraph over Γ and Σ is a tuple $H = (V, E, lab, nod, ext)$, where V is the finite set of *nodes*, E is the finite set of *edges* (or hyperedges), disjoint with V, $lab: E \to \Gamma$ is the *labeling function*, nod is the *incidence function* that associates with each $e \in E$ a partial function $nod(e) : \Sigma \to V$, and ext is the *external function*, a partial function $\Sigma \to V$. The *type* of H is $type(H) = dom(ext)$, the domain of the external function. For each $e \in E$, the type of e is $type(e) = dom(nod(e))$.
The components of a hypergraph H may be denoted by V_H, E_H, att_H, lab_H, ext_H, respectively. Furthermore, given a set $X \subseteq \Gamma$ of labels we denote by E_R^N the set $\{e \in E_H \mid lab_H(e) \in X\}$.
The number of nodes plus the number of hyperedges of H is called the size of H, denoted by $|H|$.

Definition 2 (from on [1],[2]):

Let H, K $\in$ $HGR_{\Gamma,\Sigma}$ be disjoint hypergraphs, and let e $\in$ E_H with type(e) = type(K). Let (H - e) + K be the result of removing e from H and adding K, i.e., (H – e) + K = (V, E, lab, nod, ext) where $V = V_H \cup V_K$, $E = (E_H - \{e\}) \cup E_K$, $lab = lab_H \cup lab_K$ restricted to E, $nod = nod_H \cup nod_K$ restricted to E, and $ext = ext_H$. The substitution of K for e in H is the hypergraph H[e/K] = ((H – e) + K)/R where $R = \{(nod_H(e,s), ext_K(s)) \mid s \in type(e)\}$. Another notation for H[e/K] is H $\otimes$ K.

Example:

Let us consider hypergraphs with multiple edges. We can replace an hyperedge X of a hypergraph G with another hypergraph H by removing X from G, adding H disjointly, and fusing the q-node of H with the source of X and the r-node of H with the target of X. The resulting graph is denoted by G[X/H]:

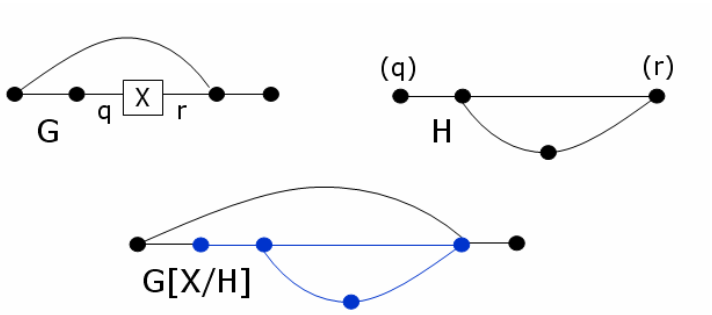

Figure 1: The replacement of an edge

Hyperedge replacement enjoys some nice properties well-known from other rule-based formalisms. First of all, we have a sequentialization and parallelization property. It does not matter whether we replace some hyperedges of a hypergraph one after another, or simultaneously. The second property is confluence. Hyperedges of a hypergraph can be replaced in any order, without affecting the result. (In fact this follows already from the sequentialization and parallelization property. If we replace hyperedges simultaneously there is no order among them at all.) The last and maybe most important property is associativity. If a hyperedge is replaced and afterwards a hyperedge of the new part is replaced with a third hypergraph, the same is obtained by first replacing the latter hyperedge and then replacing the first one with the result. These properties are stated formally below.

Sequentialization and parallelization: Let H be a hypergraph with pairwise distinct $e_1, \ldots, e_n \in E_H$ and let H_i be a hypergraph with $type(H_i) = type_H(e_i)$ for $i = 1, \ldots, n$. Then

$$H[e_1/H_1, \ldots, e_n/H_n] = H[e_1/H_1]\ldots[e_n/H_n].$$

Confluence: Let H be a hypergraph with distinct $e_1, e_2 \in E_H$ and let H_i be a hypergraph with $type(H_i) = type_H(e_i)$ for $i \in \{1,2\}$. Then

$$H[e_1/H_1]\,[e_2/H_2] = H[e_2/H_2]\,[e_1/H_1].$$

Associativity: Let H, H_1, H_2 be hypergraphs with $e_1 \in E_H$ and $e_2 \in E_H$, such that $type_H(e_1) = type(H_1)$ and $type_H(e_2) = type(H_2)$. Then

$$H[e_1/H_1][e_2/H_2] = H[e_1/H_1\,[e_2/H_2]].$$

We have now seen how hyperedge replacement on hypergraphs works. Now grammars, how base on this formalism, will be introduced. In this case we define a simple notation for terminal hyperedges with only two tentacles. These edges can be equivalent written as directed edges. The direction of the edges follows from the labelling of the tentacles.

Definition 3 (from [1], p.139):

A *hyperedge replacement grammar* (or HR grammar) is a tuple HGR = (N, T, Σ, P, S), where N is the Σ – typed alphabet of nonterminal edge labels. A Σ – typed alphabet is an alphabet N together with a mapping type: $N \rightarrow P(\Sigma)$, the set of subsets of Σ. T is the alphabet of terminal edge labels (disjoint with N), Σ is the alphabet of selectors, P is the finite set of productions, and $S \in N$ is the initial nonterminal. Every production is of the form $X \rightarrow D$ with $X \in N$ and $D \in HGR_{N \cup T, \Sigma}$, such that type(X) = type (D) and, for every $e \in E_D$ with $lab_D(e) \in N$, $type(e) = type(lab_D(e))$.
A hyperedge replacement grammar HGR = (N, T, Σ, P, S) is said to be of *order* r if for all $(A,R) \in P$, $type(R) \leq r$. A hyperedge replacement language L is of order r if there is a hyperedge replacement grammar HRG of order r with L(HGR) = L.

Example

Let us consider an HR grammar G_3 such that $L(G_3)$ is the set of all "triple stars", of the form shown in figure 2 (where edge labels are dropped).

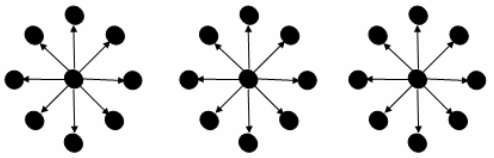

Figure 2: A triple star

Thus a triple star is the disjoint union of three copies of the same star. $G_3 = (N, T, \Sigma, P, S)$ with $N = \{S, X\}$, $T = \{*\}$, $\Sigma = \{p, q, r\}$, $type(S) = \emptyset$, $type(X) = \{p, q, r\}$, and P contains the three production shown in figure 3 (where the edge label * of the three ordinary edges is dropped).

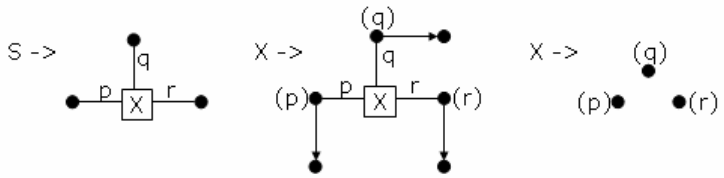

Figure 3: Productions of HR grammar G_3, generating triple starts

A triple star with 3n edges is generated by starting with the first production, applying the second production n times (each time adding one edge to each star), and finally removing the nonterminal edge by the third production. Note that from the point of view of context-free (string) grammar, $L(G_3)$ is rather noncontext-free: it is similar to the language $\{a^n b^n c^n \mid n \geq 0\}$

2 Context-freeness lemma

We have see, how hyperedge replacement grammars work. Now we want to show that hyperedge replacement grammars are context-free. Hypergraph context-freeness is not the same as string context-freeness. Hypergraph grammars can generate string languages which are more powerful then context-free string languages. But this will be discussed later in Chapter 4.

Theorem 4 - Context-free lemma (from [1], p.144):

Let G = (N, T, Σ, P, S) be an HR grammar. Let H and K be hypergraphs in $HGR_{N\cup T,\Sigma}$ and $HGR_{T,\Sigma}$, respectively, let $e_1, \ldots, e_n$ be all the nonterminal edges of H, and let e_i be labeled by X_i in H. Then $H \Rightarrow^* K$ if and only if there exist hypergraphs $K_1, \ldots, K_n$ in $HGR_{T,\Sigma}$ (disjoint with H) such that $K = H[e_1/K_1]\ldots[e_n/K_n]$ and $sn(X_i, e_i) \Rightarrow^* K_i$ for all $1 \le i \le n$. Moreover, the length of the derivation $H \Rightarrow^* K$ equals the sum of the lengths of the derivations $sn(X_i, e_i) \Rightarrow^* K_i$.

Proof:
We prove the equivalence by induction on the length of the derivation $H \Rightarrow^* K$, and on the sum of the lengths of the derivations $sn(X_i, e_i) \Rightarrow^* K_i$.
(Only if) Let $H \Rightarrow^* K$. If the derivation has length 0, and so K = H, then n = 0 and there is nothing to prove. Otherwise, because of confluence of substitution, we may assume w.l.o.g. that in the first step of the derivation a production copy p is applied to edge e_n, say $p : X_n \to D$. Thus, $H \Rightarrow H[e_n/D] \Rightarrow^* K$. Let $e_{n+1}, \ldots, e_{n+d}$ be all nonterminal edges of D, with labels $X_{n+1}, \ldots, X_{n+d}$. By induction, there exist hypergraphs K_i, for $1 \le i \le n+d$ and $i \ne n$, such that $sn(X_i, e_i) \Rightarrow^* K_i$ and

$$K = H[e_n/D]\,[e_1/K_1]\ldots[e_{n-1}/K_{n-1}]\,[e_{n+1}/K_{n+1}]\ldots[e_{n+d}/K_{n+d}].$$

Define $K_n = D[e_{n+1}/K_{n+1}]\ldots[e_{n+d}/K_{n+d}]$. By induction (in the other direction) $D \Rightarrow^* K_n$, and hence (using p) $sn(X_n, e_n) \Rightarrow D \Rightarrow^* K_n$. By confluence of substitution (n – 1 times),

$$K = H[e_1/K_1]\ldots[e_{n-1}/K_{n-1}]\,[e_n/D]\,[e_{n+1}/K_{n+1}]\ldots[e_{n+d}/K_{n+d}],$$

and by associativity of substitution (d times) this is equal to

$$H[e_1/K_1]\ldots[e_{n-1}/K_{n-1}]\,[e_n/D[e_{n+1}/K_{n+1}]\ldots[e_{n+d}/K_{n+d}]] = H[e_1/K_1]\ldots[e_{n-1}/K_{n-1}][e_n/K_n].$$

(If) This part of the proof is analogous to the one above. Consider $K = H[e_1/K_1]\ldots[e_n/K_n]$ with $sn(X_i, e_i) \Rightarrow^* K_i$. If all these derivations have length 0, then n = 0 and K = H. Otherwise, by confluence, we may assume that the last derivation has positive length. Let $sn(X_n, e_n) \Rightarrow D \Rightarrow^* K_n$ be the first step of the last derivation, with production copy $p : X_n \to D$. By induction (in the other direction), there exist K_i, for $n+1 \le i \le n+d$, such that $K_n = D[e_{n+1}/K_{n+1}]\ldots[e_{n+d}/K_{n+d}]$ and $sn(X_i, e_i) \Rightarrow^* K_i$. By the calculation above (in reverse),

$$K = H[e_n/D]\,[e_1/K_1]\ldots[e_{n-1}/K_{n-1}]\,[e_{n+1}/K_{n+1}]\ldots[e_{n+d}/K_{n+d}].$$

Hence, by induction, $H[e_n/D] \Rightarrow^* K$, and so (using p) $H \Rightarrow H[e_n/D] \Rightarrow^* K$.

3 Pumping lemma

After seeing what context-freeness for hypergraph grammars means, it would be good to have a possibility to check whether a hypergraph grammar is context-free or not. Luckily for us exists also a pumping lemma for hypergraph grammars. This pumping lemma is similar to the pumping lemma for regular string languages. (see figure 4).

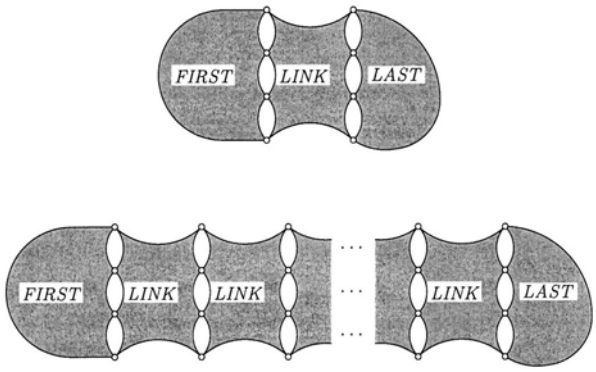

Figure 4: principle of pumping lemma for hypergraph grammars.

Before we can start with the pumping lemma for context-free hypergraph grammars, we must first introduce some definitions.

Definition 5 (from [2], p.19)

The set TREE(P) of derivation trees over P is recursively defined as follows:

- $N \subseteq$ TREE(P) with root(A) = A and result(A) = $A^\bullet$ for $A \in N$
- For every production $(A,R) \in P$ and every mapping branch $E_R^N \rightarrow$ TREE(P) such that we have type(e) = type (root(branch(e))) for all $e \in E_R^N$, the triple t(A,R,branch) is in TREE(P).
  Furthermore we let root(t) = A and result(t) = R[result(branch(e))] for all $e \in E_R^N$

One should notice that a derivation tree contains nonterminal labels $A \in N$ as subtrees if and only if its result is not terminal.

Definition 6 (from [2], p.22/23):

A hypergraph H is *substantial* if $V_H \neq ext_H$ or $|E_H| > 1$. A hyperedge replacement grammar each of whose productions has a substantial right-hand side is *growing*. A growing hyperedge replacement grammar generates only substantial hypergraphs.
H is said *X-handled* if there is a unique type(X)-edge $e \in E_H$ with label $X \in \Sigma_H$. In this case, e is called the X-handle of H.

Theorem 7 - Pumping lemma (based on [2], p.23):

Let L be some hyperedge replacement language of order r. Then there exists constants p and q such that the following is true: For every hypergraph H in L with $|H| > p$ there are an X-handled hypergraph FIRST, a substantial X-handled type(X)-hypergraph LINK, and a type(X)-hypergraph LAST for some $X \in \Sigma_H$ with $H = FIRST \otimes LINK \otimes LAST$, $|Link \otimes Last| \leq q$ and $type(LINK) \leq r$, such that $FIRST \otimes LINK^k \otimes LAST \in L$ for all $k \in \mathbb{N}$

Proof:
Let HRG= (N, T, Σ, P, S) ∈ HR grammar with L(HGR) = L and n the number of nonterminals. Since there are only finitely many one-substantial hypergraphs in L we may assume HGR is growing. Let max be the size of the largest right-hand side of HRG. Let $t \in TREE(P)$ with root(t) = S and H = result(t) a hypergraph. If $|H| > max^n$, then contains a path from the root to to a leaf longer than n such that one of the nonterminals, say X, occurs twice. In other words, t has a subtree t' with root X which has a proper subtree t'' with root X. Choose LAST = result(t''), LINK = result(t' - t'') and FIRST = result(t – t') where t' – t'' is obtained from t' by removing the subtree t'' and t – t' by removing t' from t. Then, FIRST and LINK are X-handled, LINK and LAST are type(X)-hypergraphs, and $H = FIRST \otimes LINK \otimes LAST$. Since HGR is growing, LINK must be substantial. As in the case of pumping lemma for regular string languages one can now show $FIRST \otimes LINK^k \otimes LAST \in L$ for all $k \in \mathbb{N}$

4 String-generating hyperedge replacement grammars

Now, how already hinted in Chapter 2, we will see how hypergraph grammars can generate string languages. These string languages are more powerful then context-free string languages. In fact, string-generating hyperedge replacement grammars are as powerful as Multiple Context-Free Grammars.

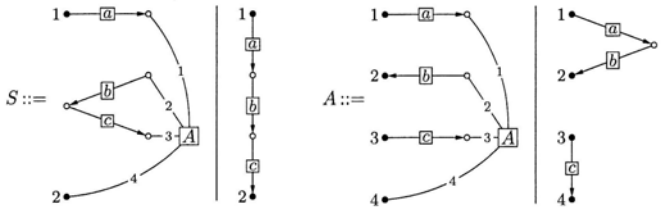

Figure 5: Productions of the hyperedge replacement grammar $A^nB^nC^n$

Example

Let us consider the hyperedge replacement grammar $A^nB^nC^n = (\{S,A\}, \{a, b, c\}, P, S)$ where P consists of the production depicted in figure 5. Beginning with S, the application of the second production yields the string graph (abc). By applying the first production, then applying the third production n-1 times, followed by an application of the fourth production, we obtain the derivation in Figure 4. Furthermore, the only hypergraphs in $L(A^nB^nC^n)$ are string graphs of the form $(a^nb^nc^n)$ for $n \geq 1$. Thus, $L(A^nB^nC^n) = \{(a^nb^nc^n) \mid n \geq 1\}$.

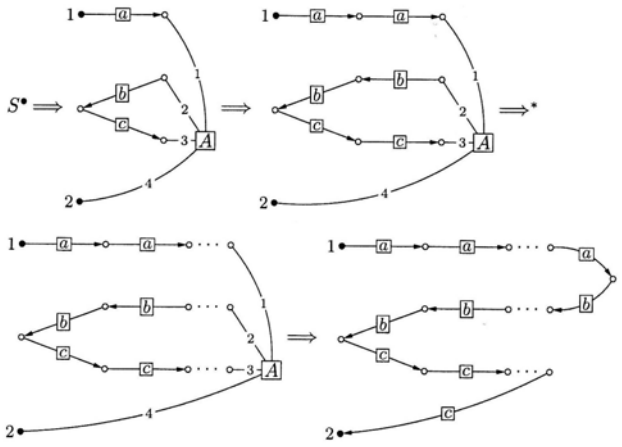

Figure 4: A derivation in $A^nB^nC^n$

Theorem 8:

Let L_{2k+2} be the languages $\{a_1^n a_2^n \ldots a_{2k+1}^n a_{2k+2}^n \mid n \geq 1\}$, where $a_1, \ldots, a_{2k+2}$ are different symbols. Then L_{2k+2} is in $STR(HR_{2k+2})$ but not in $STR(HR_{2k+1})$, for all $k \geq 1$. $STR(HR_i)$ are all languages generated by an hypergraph grammars at which each hyperedge has at most i tentacles.

Proof:
It is obvious that $H_i \subseteq H_{i+1}$, $i \geq 2$, because you can simply add an extra tentacle to each nonterminal hyperedge. In the last substitution the node of the extra tentacle can be consolidated with any other node connected by the same hyperedge. In this case we get the same derivation. This implies $STR(HR_i) \subseteq STR(HR_{i+1})$. We now want to show that $STR(HR_{2k+1}) \neq STR(HR_{2k+2})$.

The string-graph language $L_{2k+2} = \{a_1^n \ldots a_{2k+2}^n \mid n \in \mathbb{N}\}$ can be generated by the hyperedge replacement grammar of order 2k+2, for every $k \geq 1$. The construction is a straightforward generalization of the one presented in previous example. Thus, L_{2k+2} is a string-graph language of order 2k+2. Making use of the pumping lemma, it can be shown that L_{2k+2} is no string-graph language of order 2k+1. To see this, assume L_{2k+2} is a language of order 2k+1. Let FIRST, LINK, and LAST be as in the pumping lemma, for some sufficiently large member of L_{2k+2}. Then LINK has type less than 2k+2 and is substantial. The latter implies $E_{LINK} \neq 0$ because $|V_H| = |E_H| + 1$ for all $H \in L_{2k+2}$.
Since FIRST $\otimes$ LINK $\otimes$ LAST is a string graph the connected components of LINK must be paths whose first and last nodes are in $[ext_{LINK}] \cup [att_{LINK}(e)]$, where e is the X-handle of LINK. Hence, the number of connected components of LINK which contain at least one edge is at most k. Moreover, none of these connected components can contain edges with different labels, since otherwise pumping obviously yields graphs not in L_{2k+2}. But then LINK lacks at least one of the labels, so FIRST $\otimes$ LINK $\otimes$ LAST $\in$ L implies FIRST $\otimes$ LAST $\notin$ L because $E_{LINK} \neq \varnothing$. This proofs that $STR(HR_{2k+1}) \subset STR(HR_{2k+2})$.

Theorem 9:

$STR(HR_2)$ is the class of context-free languages.

Proof:
L is context-free $\Rightarrow L \in STR(HR_2)$.
Because L is context-free, there exists a grammar in Chomsky normal form that generates L. In Chomsky normal form there exists only production of the following types:

$$1)\ A \rightarrow BC \qquad \text{or} \qquad 2)\ A \rightarrow a \qquad \text{or} \qquad 3)\ S \rightarrow \varepsilon$$

These productions can equivalent be encoded in a hypergraph grammar (see figure 6).

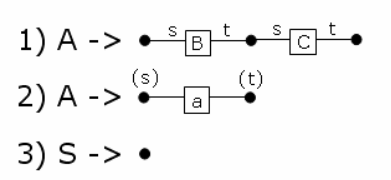

Figure 6: Production of hypergraph grammar in Chomsky normal form

The proof for the direction $L \in STR(HR_2) \Rightarrow L$ is context-free works analogous as described above. Because we know that each hyperedge in the grammar for L can only have two tentacles. In this case each production can only produce hyperedges in a series. These series can be easy encoded as context-free grammar.

Conclusion

In this survey we outlined the theory of hyperedge replacement as a grammatical device for the generation of hypergraph and string languages. Hypergraph replacement grammars are context-free and in this case we have seen a possibility to check if it is possible to model a language as hypergraph grammar (pumping lemma). At all, hypergraph replacement grammars are a nice theoretical model, but they are not use in practise, because in difference to context-free string grammars the membership problem turns out to be NP-complete. Only restricted subclasses lead to polynomial membership algorithms.

References

1. J. Engelfriet: Context-Free Graph Grammars, Handbook of Formal Languages, Springer-Verlag, 1997
2. F. Drewes, H.-J. Kreowski, A. Habel: Hyperedge replacement graph grammars, Handbook of Graph Grammars and Computing by Graph Transformation, World Scientific, 1997
3. A. Habel, H.-J. Kreowski: Some structural aspects of hypergraph languages generated by hyperedge replacement, in Proc. STACS'87, Lecture Nodes in Computer Science 247, Springer-Verlag, Berlin, 1987