Tree Grammars with multilinear Interpretation

Y. Guan, G. Hotz, A. Reichert *

Lehrstuhl Prof. Dr. Günter Hotz
Fachbereich 14 – Informatik
Universität des Saarlandes
W-6600 Saarbrücken, Germany

*Diploma Thesis

Introduction

In the following report, the relation between rewriting on trees and rewriting on strings is inspected.

The correspondence of *regular tree rewriting* with context-free string rewriting is a well-known fact [Br69]. Generalizing from this will lead us to a type of string grammar called *coupled-context-free grammar (CCFG)*.

A hierarchy of grammar (resp. language) classes is obtained with the context-free Chomsky-grammars (CFG) as its first step. Each language class in this hierarchy is contained in the class of context-sensitive Chomsky-languages.

Generation power increases only slightly stepping up in the hierarchy. This is measured by the ability for "counting up to k", i.e. for generating the language

$$COUNT_k := \{a_1^n \cdots a_k^n \mid n \in \mathbf{N}_0\} \subset \{a_1, \ldots, a_k\}^*$$

and the ability for "copying k times", i.e. for generating the language

$$COPY_k := \{ww^k \mid w \in \{a,b\}^*\} \subset \{a,b\}^*$$

where k is a non-negative integer.

Under this aspect CFGs are able to count up to 2 and are not able to copy, *Tree Adjoining Grammars (TAG)* [Jo85] are able to count up to 4 and to copy one time. The corresponding language classes CFL and TAL form the first two steps in our hierarchy. The k-th grammar class in the hierarchy is able to "count up to $2k+2$" and to "copy k times".

Recent investigations in Computer Linguistics have shown that many natural language phenomena can be described by grammars whose generation power is slightly stronger than that of CFGs [Jo86]. This also motivates the study of the grammar and language classes in our hierarchy.

Further work in the theory, especially the study of parsing algorithms, can be found in [Gu92].

Contents

Chapter 1

Algebraic Theory of Formal Languages

1.1 The theory of free $\times$–categories

We introduce in this section some basic definitions and facts from the theory of free $\times$–categories.

Definition 1 *monoid of words*
Let $[1..n] := \{i \in \mathbf{N} \mid 1 \leq i \leq n\}$ for $n \in \mathbf{N}_0$, i.e. $[1..0] = \emptyset$. For a finite set (*alphabet*) B,

$$B^* := \{w \mid w : [1..n] \to B \text{ is a mapping, } n \in \mathbf{N}_0\}$$

is called the set of *words over* B.

The (unique) mapping $\varepsilon_B : [1..0] \to B$ is the *empty word* over B, and $B^+ := B^* \setminus \{\varepsilon_B\}$ is the set of *nonempty words* over B.

Let "$\cdot$" denote the *concatenation* on B^*, i.e. for $u : [1..m] \to B,\ v : [1..n] \to B$

$$u \cdot v : [1..(m+n)] \to B$$

is the mapping defined by

$$(u \cdot v)(i) = \begin{cases} u(i) & , \quad 1 \leq i \leq m \\ v(i-m), & m+1 \leq i \leq m+n \end{cases}$$

The monoid $(B^*, \cdot, \varepsilon_B)$ is called *monoid of words over* B.

$|w| := n$ is called the *length* of $w : [1..n] \to B$. If B^n denotes the set of all words of length n in B^*, it holds

$$B^m \cdot B^n := \{u \cdot v \mid u \in B^m,\ v \in B^n\} = B^{m+n}.$$

$B^0 = \{\varepsilon_B\}$ will also be written as $\{\Box\}$.

The monoid of words over B is *freely generated* by the set B^1 which is identified with B.

Let us now give some category terminology which will be illustrated by a category of mappings.

For a set B, let M be the set of all mappings $f : B^m \to B^n,\ m, n \in \mathbf{N}_0$. Define mappings $Q, Z : M \to \mathbf{N}_0$ by $Q(f) := m,\ Z(f) := n$, for f as above.

The *(sequential) composition* $f \circ g$ is defined for $f, g \in M$ iff $Q(f) = Z(g)$ and is the mapping

$$f \circ g : B^{Q(g)} \to B^{Z(f)}, \qquad (f \circ g)(x) := f(\,g(x)\,) \quad \forall x \in B^{Q(g)}.$$

Thus, $Q(f \circ g) = Q(g)$ and $Z(f \circ g) = Z(f)$. Let $\mathbf{1}_n,\ n \in \mathbf{N}_0$, denote the identity on B^n. It holds:

$$K_1 = (\mathbf{N}_0, M, Q, Z, \circ) \text{ is a } \textit{category.}$$

Definition 2 *category*
$K = (O, M, Q, Z, \circ)$ is a *category* iff

(K1) O and M are sets, $Q, Z : M \to O$ are mappings,
$\circ : M \times M \rightsquigarrow M$ is a (partial) operation on M.

(K2) For $f, g \in M$, the composition $f \circ g \in M$ is defined iff $Q(f) = Z(g)$. In this case $Q(f \circ g) = Q(g)$ and $Z(f \circ g) = Z(f)$.

(K3) If $(f \circ g) \circ h$ is defined for $f, g, h \in M$, then also $f \circ (g \circ h)$. In this case holds $(f \circ g) \circ h = f \circ (g \circ h)$.

(K4) For each $u \in O$, there is an element $\mathbf{1}_u \in M$ with the following properties:

1. $Q(\mathbf{1}_u) = u = Z(\mathbf{1}_u)$,
2. $f \circ \mathbf{1}_u = f$ for all $f \in M$ with $Q(f) = u$,
3. $\mathbf{1}_u \circ g = g$ for all $g \in M$ with $Z(g) = u$.

$Obj\,K := O$ is the set of *objects*, and $Mor\,K := M$ is the set of *morphisms* of category K. Q and Z are called *source* resp. *target mapping* of K. The morphisms $\mathbf{1}_u$ are called *units* of K, the set of all units is denoted by $UNITS(K)$.

For $f \in Mor\,K$ with $Q(f) = u$ and $Z(f) = v$, we write $f : u \to v$ or $u \xrightarrow{f} v$. For subsets $U, V \subset Mor\,K$, let

$$K(U, V) := \{f \in Mor\,K \mid Q(f) \in U,\ Z(f) \in V\}$$

If $U = \{u\}$ is a singleton, we also write $K(u, V)$ (analogous for V).

Our next definition introduces a special type of category called $\times$–*category* which plays an important role in the description of formal languages, see [Ho66], [Ben70] or [Ben75]. In an $\times$–category, the sets of objects and morphisms each form a monoid. Source and target mappings are homomorphisms between these monoids. We extend the category K_1 from above to an $\times$–category by defining an additional operation on the set of morphisms.

Let $\otimes : M \times M \to M$ denote the *parallel composition* of mappings from M, i.e. for

$$f : B^m \to B^n, \qquad g : B^r \to B^s, \qquad m, n, r, s \in \mathbf{N}_0,$$

it holds:

$$f \otimes g : B^{m+r} \to B^{n+s}, \qquad f \otimes g\,(u \cdot v) := f(u) \cdot g(v) \quad \forall\, u \in B^m, v \in B^r.$$

Then the following properties hold:

1. $(\mathbf{N}_0, +, 0)$, the set of objects with addition, and $(M, \otimes, \mathbf{1}_0)$, the set of morphisms with parallel composition, are monoids.

2. The source and target mappings $Q, Z : M \to \mathbf{N}_0$ are homomorphisms between these monoids.

3. $(f_1 \otimes f_2) \circ (g_1 \otimes g_2) = (f_1 \circ g_1) \otimes (f_2 \circ g_2) \quad \forall f_i,\ g_i \in M,\ Q(f_i) = Z(g_i),\ i = 1, 2.$

The structure $MAP(B, \mathbf{N}_0) := (\, \mathbf{N}_0, M, Q, Z, \circ, (+, \otimes)\,)$ is an $\times$*–category.*

For simplicity, the monoid operations on the set of objects (here: $+$) resp. on the set of morphisms (here: $\otimes$) both are denoted by the same symbol (here: $\times$).

Definition 3 $\times$*–category*

$X = (O, M, Q, Z, \circ, \times)$ is called $\times$*–category* iff

(K) $(O, M, Q, Z, \circ)$ is a category.

(X1) $(O, \times)$ and $(M, \times)$ are monoids, and Q, Z are monoid-homomorphisms.

(X2) For all $f_i, g_i \in M$ with $Q(f_i) = Z(g_i),\ i = 1, 2$, holds:

$$(f_1 \times f_2) \circ (g_1 \times g_2) = (f_1 \circ g_1) \times (f_2 \circ g_2).$$

Definition 4 *subcategory,* $\times$*–subcategory*

Let U and K be categories ($\times$–categories) with $Obj\, U \subset Obj\, K,\ Mor\, U \subset Mor\, K$. If the operations of U and K are identical on U, U is called a *subcategory ($\times$–subcategory)* of K. This is denoted by $U \subset K$.

Definition 5 *generated subcategory,* $\times$*–subcategory*

Let K be a category ($\times$–category), $E \subset Obj\, K,\ A \subset Mor\, K$. The smallest subcategory ($\times$–subcategory) of K which includes E and A is called the *subcategory ($\times$–subcategory) generated by* (E, A) *in* K and is denoted by $\langle E, A \rangle_K$.

(E, A) is called *generating system* of $\langle E, A \rangle_K$. If $E = Obj\, K$, we simply call A generating system of $\langle E, A \rangle_K$.

Generated subcategories are characterized by the following

Lemma 1.1.1 (see [Ho74])

$$\langle E, A\rangle_K = \left(\widehat{E_K}, \widehat{A_K}, Q, Z, \circ, \times\right)$$

with

$$\begin{aligned}\widehat{E_K} &= \bigcap\{Obj\,U \mid U \subset K,\ E \subset Obj\,U,\ A \subset Mor\,U\},\\ \widehat{A_K} &= \bigcap\{Mor\,U \mid U \subset K,\ E \subset Obj\,U,\ A \subset Mor\,U\},\end{aligned}$$

and $Q, Z, \circ, \times$ are the restrictions of the corresponding mappings and operations of K.

The morphisms in subcategory $\langle E, A\rangle_K$ are exactly those morphisms from K which can be built from the elements of A and the units $\mathbf{1}_e$, $e \in E$, applying the operations of the category.

In the following, we will often make use of mappings between categories (resp. $\times$–categories), so called *functors*, which are compatible with the category operation(s). Functors are similar to homomorphisms (of algebras), but not in all respects. For example, the image of a category under a functor must not be a category.

Definition 6 *covariant functor*
Let K_1, K_2 be categories, $\phi_1 : Obj\,K_1 \to Obj\,K_2$ and $\phi_2 : Mor\,K_1 \to Mor\,K_2$ be mappings. We call $\phi = (\phi_1, \phi_2)$ a (*covariant*) *functor* from K_1 into K_2 if

(F1) $Q(\phi_2(f)) = \phi_1(Q(f))$ and $Z(\phi_2(f)) = \phi_1(Z(f)) \quad \forall f \in Mor\,K_1$,

(F2) $\phi_2(f \circ g) = \phi_2(f) \circ \phi_2(g) \quad \forall f, g \in Mor\,K_1$ with $Q(f) = Z(g)$,

(F3) $\phi_2(\mathbf{1}_u) = \mathbf{1}_{\phi_1(u)} \quad \forall u \in Obj\,K_1$.

A functor is a pair of mappings that is compatible with the operations of the category and with the source and target mappings Q and Z. A covariant functor preserves the "direction" of morphisms. There are also functors which invert the direction of morphisms:

Definition 7 *contravariant functor*
Let K_1, K_2, ϕ_1, ϕ_2 as above. $\phi = (\phi_1, \phi_2)$ is a *contravariant functor* from K_1 into K_2 if

(F1') $Q(\phi_2(f)) = \phi_1(Z(f))$ and $Z(\phi_2(f)) = \phi_1(Q(f)) \quad \forall f \in Mor\,K_1$,

(F2') $\phi_2(f \circ g) = \phi_2(g) \circ \phi_2(f) \quad \forall f, g \in Mor\,K_1$ with $Q(f) = Z(g)$,

(F3') $\phi_2(\mathbf{1}_u) = \mathbf{1}_{\phi_1(u)} \quad \forall u \in Obj\,K_1$.

Remark: Condition (F1) can be derived from (F2), (F3) and the uniqueness of units (analogously for (F1')).
For the special case of $\times$–categories we get the following definition:

Definition 8 $\times$–*functor*
Let K_1 and K_2 be $\times$–categories. A functor $\phi = (\phi_1, \phi_2)$ from K_1 into K_2 is called $\times$–*functor* if ϕ_1 and ϕ_2 are monoid homomorphisms with respect to $\times$.

An $\times$–category is *free (generated)* if its morphisms can be uniquely presented by the elements of a special generating system. Two presentations are considered equal if they can be transformed into each other using only axioms of the $\times$–category. A possible definition is given by

Definition 9 *free (generated) $\times$–category*
An $\times$–category X is *free* (*generated*) if there is a subset $A \subset MorX$ with the following property:

If K is an arbitrary $\times$–category, $\phi_1 : Obj\,X \to Obj\,K$ a monoid homomorphism and $\phi_2' : A \to Mor\,K$ a mapping compatible with source and target, i.e.

$$\forall f \in A \qquad Q(\phi_2'(f)) = \phi_1(Q(f)) \text{ and } Z(\phi_2'(f)) = \phi_1(Z(f)),$$

there is a (unique) extension $\phi_2 : Mor\,X \to Mor\,K$ of ϕ_2' such that $\phi = (\phi_1, \phi_2)$ is an $\times$–functor from X into K.

In this case, $A \subset MorX$ is called *free generating system* of X.

Given sets A, V and mappings $Q, Z : A \to V^*$, there is a free $\times$–category with free generating system A and monoid of objects V^*. We denote this free $\times$–category by $\mathcal{F}(A, V, Q, Z)$, $\mathcal{F}(A, Q, Z)$, $\mathcal{F}(A, V)$ or simply by $\mathcal{F}(A)$ if the missing parameters are given in the context.

Formal constructions of $\mathcal{F}(A, V, Q, Z)$ can be found in references [Ho65] or [Ben75]. We give some properties of $\mathcal{F}(A, V, Q, Z)$, for a proof see [HoCl72]:

1. $\mathcal{F}(A, V, Q, Z)$ has the unique free generating system A.
2. Each non-unit morphism $f \in \mathcal{F}(A, V, Q, Z)$ has a *sequential presentation*

$$f = (\mathbf{1}_{u_m} \times a_m \times \mathbf{1}_{v_m}) \circ \cdots \circ (\mathbf{1}_{u_1} \times a_1 \times \mathbf{1}_{v_1})$$

   where $u_i, v_i \in V^*$ and $a_i \in A,\ 1 \le i \le m,\ m \in \mathbf{N}$.

In general, the sequential presentation of a morphism is not uniquely determined. Nevertheless, the number of occurrences of each generator in different sequential presentations for a morphism is the same. The sum of all occurrences of generators in a sequential presentation is called the *length* of the presented morphism.

Morphisms of a free $\times$–category $\mathcal{F}(A, V, Q, Z)$ can be visualized geometrically by directed planar *nets*.

Each generator $a \in A$ with $Q(a) = u,\ Z(a) = v$, is considered as a "box" named a with $|u|$ inputs and $|v|$ outputs which hangs in a rectangle. Inputs and outputs are marked with elements from V. The i-th input (resp. output) from the left is marked with $u(i)$ (resp. $v(i)$). In the case $\#V = 1$, i.e. $V^* \cong \mathbf{N}_0$, the markings are omitted.

Each unit $\mathbf{1}_u,\ u \in V^*$, is considered as a "bundle of wires" marked with the word u.

Starting from these elementary nets, we can construct more complex ones by applying the $\times$–category-operations $\circ$ and $\times$.

$F \times G$ is defined as the juxtaposition of the nets F and G together with concatenation of input and output markings. (In the figure, the markings are omitted.)

$F \circ G$ is defined iff the output marking of G equals the input marking of F and corresponds to setting G on top of F. The input marking of the composed net $F \circ G$ is given by $Q(F \circ G) = Q(G)$ and the output marking is given by $Z(F \circ G) = Z(F)$.

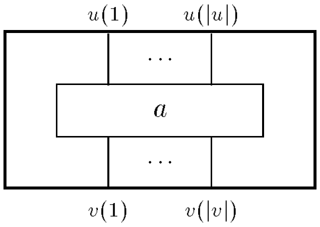

Figure 1.1: generator

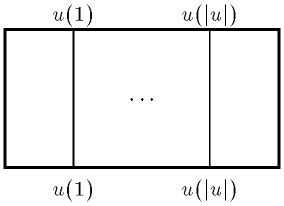

Figure 1.2: unit

The resulting nets are collected in classes where two nets are in the same class if they can be transformed into each other by "deformations" corresponding to the axioms (X2) resp. (K4) of the $\times$–category.

The operations $\circ$ and $\times$ are inherited to the classes, the morphisms of the free $\times$–category correspond exactly to these classes of nets. Nets in the same class are called *inessentially different* or *similar*.

More information on the geometric aspects of free $\times$–categories can be found in references [Ho65] and [Ho74].

An important special case of free $\times$–categories will now be defined. It is characterized by the fact that each generator has exactly one output.

Definition 10 *category of trees, context-free $\times$–category*

A free $\times$–category $\mathcal{F}(A, V, Q, Z)$ is called *category of trees* or *context-free $\times$–category* if

$$|Z(a)| = 1 \qquad \forall a \in A\ .$$

An analogous definition applies to the case $|Q(a)| = 1\ \forall a \in A$. If $\#V = 1$, we identify V^* with $\mathbf{N}_0$ and consider Q and Z as mappings from A to $\mathbf{N}_0$. For $\mathcal{F}(A, V, Q, Z)$ we write in this case also $\mathcal{B}(A, Q, Z)$.

Definition 11 *tree*

Let $\mathcal{F} := \mathcal{F}(A, V, Q, Z)$ be a category of trees. A morphism $f \in Mor\,\mathcal{F}$ is called a *tree* if

$$f = a \circ f', \quad a \in A,\ f' \in Mor\,\mathcal{F}.$$

We denote with $Trees(M)$ the subset of all trees in a set of morphisms $M \subset Mor\,\mathcal{F}$. Every morphism from a category of trees can be decomposed under $\times$ into units and trees:

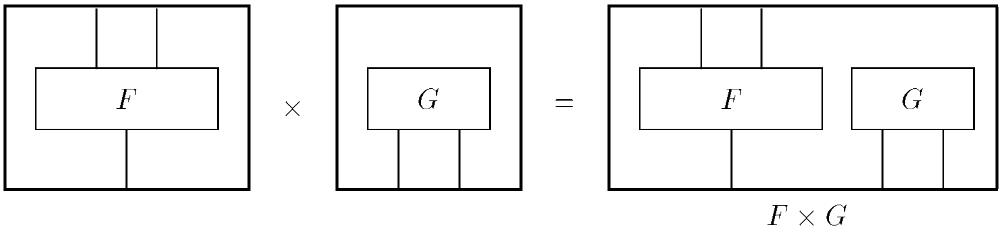

Figure 1.3: parallel composition

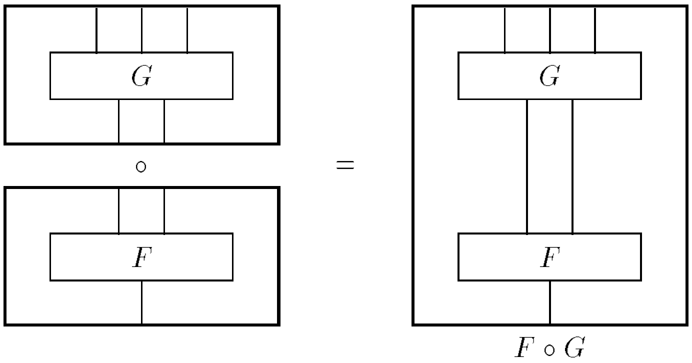

Figure 1.4: sequential composition

Lemma 1.1.2 (see [HoCl72]) *Let $\mathcal{F} := \mathcal{F}(A, V, Q, Z)$ be a category of trees.*
Each $f \in Mor\,\mathcal{F}$, $Z(f) = w$, $|w| = k \in \mathbf{N}_0$, has a unique decomposition

$$f = f_1 \times \cdots \times f_k$$

with $f_i \in Mor\,\mathcal{F}$, $Z(f_i) = w(i)$, $1 \leq i \leq k$.

Thus, each f_i in this decomposition is either the unit $\mathbf{1}_{w(i)}$ or a tree.

In the following section, we will describe formal languages using the theory introduced above. This treatment of formal language theory is sometimes called the *Algebraic Theory of Formal Languages* [Ben75].

1.2 Describing formal languages by ×–categories

The importance of the theory of ×–categories for the description of formal languages bases on the relation between *semi-Thue–systems* and free ×–categories, see [Ho66] or [Ben75].

A semi-Thue–system is a rewriting system on words and consists of a finite set of *rewriting rules*. Each rewriting rule is an ordered pair (u, v) of words and carries a unique name, e.g. p, which will be notated by $p : u \to v$.

A rule $p : u \to v$ is *applicable* to a word w iff u is a subword of w. *Application* of rule p to w is the substitution of subword u by v. The *semi-Thue–relation* describes which words can be transformed into each other by a finite number of rule applications.

If word w can be transformed into w' by the rules of the semi-Thue–system, w' is *derivable* from w. In this case, there is a sequence of words $w_0, \ldots, w_n$, $n \in \mathbf{N}_0$, with $w = w_0$, $w' = w_n$ and each w_{i+1} can be derived from w_i by application of one rule. Such a sequence is called a *derivation* in the semi-Thue–system.

Describing derivations by sequences of words in general gives no unique description of the generation process. But in many cases there is need for a complete description, e.g. in studying the ambiguity of semi-Thue–systems. An adequate formal description can be given using the theory of free ×–categories. We first give the usual definitions.

Definition 12 *semi-Thue–system*
$S = (P, V, Q, Z, \stackrel{*}{\Rightarrow}_S)$ or $S = (P, V, Q, Z)$ is called *semi-Thue–system* iff

1. P is a finite set (*productions, rewriting rules*).
2. V is a finite set (*alphabet*).
3. $Q, Z : P \to V^*$ are mappings (*source, target*).
4. $\stackrel{*}{\Rightarrow}_S$, the *semi-Thue–relation*, is the reflexive, transitive closure of the relation

$$\Rightarrow_S \quad := \quad \{\, (w_1 u w_2,\ w_1 v w_2) \mid p : u \to v \in P,\ w_1, w_2 \in V^* \,\}.$$

The common definition of a derivation is

Definition 13 *derivation (sequence)*
A sequence $(w_0, \ldots, w_n)$, $n \in \mathbf{N}_0$, with

1. $w_i \in V^*$, $0 \le i \le n$,
2. $w_i \Rightarrow_S w_{i+1}$, $0 \le i < n$,

is called *derivation (sequence) of length n from w_0 to w_n in S.*

It holds $w \stackrel{*}{\Rightarrow}_S w'$ iff there is a derivation from w to w' in S. The words w and w' are called *source* and *target* of the derivation.

To each semi-Thue–system $S = (P, V, Q, Z, \stackrel{*}{\Rightarrow}_S)$ corresponds a unique free ×–category

$$\mathcal{F}(S) := \mathcal{F}(P, V, Q, Z)$$

The objects of $\mathcal{F}(S)$ are the words over V, morphisms are the classes of "inessentially different" derivations in S. If we visualize morphisms geometrically (see preceding section), we can think of derivations as nets built up by productions from P with wires marked by symbols from V. The source resp. target of a derivation forms the input resp. output marking of the corresponding net. The relation between a semi-Thue–system S and the free ×–category $\mathcal{F}(S)$ is given by

Lemma 1.2.1 ([Ho66]) *For all $w, w' \in V^*$ holds:*

$$w \overset{*}{\Rightarrow}_S w' \iff \mathcal{F}(S)(w, w') \neq \emptyset.$$

This lemma expresses that there is a derivation from source w to target w' iff there is a morphism in the free $\times$–category with source w and target w'.

Each free $\times$–category $\mathcal{F} := \mathcal{F}(P, V, Q, Z)$ defines a semi-Thue–system $S := (P, V, Q, Z)$. S is uniquely determined by $\mathcal{F}$ because the free generating system of $\mathcal{F}$ is unique. Therefore we can write without ambiguity $S = S(\mathcal{F})$. It holds

Theorem 1 ([Ho66]) *There is a bijection between semi-Thue–systems $S = (P, V, Q, Z, \overset{*}{\Rightarrow}_S)$ and the free $\times$–categories $\mathcal{F}(P, V, Q, Z)$.*

Derivations $(w, \ldots, w')$ of positive length in the semi-Thue–system $S(\mathcal{F})$ correspond to the sequential presentations of the morphisms $f : w \to w' \in \mathcal{F}(S)$. The length zero derivations (w) in $S(\mathcal{F})$ correspond to the units $\mathbf{1}_w$ of $\mathcal{F}(S)$.

One of the most important formalisms for defining languages are the *Chomsky-grammars* which are based on semi-Thue–systems. We give the usual definition.

Definition 14 *(Chomsky-)grammar*
$G = (S, T, s)$ is a *(Chomsky-)grammar* iff

1. $S = (V, P, Q, Z)$ is a semi-Thue–system.
2. $T \subset V$, $T \neq \emptyset$. T is the set of *terminals* of G.
   $N := V \setminus T$, $N \neq \emptyset$, is the set of *nonterminals* of G.
3. $s \in N$ is the *axiom* of the grammar.
4. $Q(p) \in N^+$ for each production $p \in P$.

The *language generated* by grammar $G = (S, T, s)$ is the following set of terminal words:

$$L(G) := \left\{ w \in T^* \mid s \overset{*}{\Rightarrow}_S w \right\}.$$

An important special case of Chomsky-grammars are the *context-free* grammars defined as follows:

Definition 15 *context-free semi-Thue–system, Chomsky-grammar*
A semi-Thue–system $S = (P, V, Q, Z)$ resp. a Chomsky-grammar $G = (S, T, s)$ is *context-free* if for all $p \in P$, $|Q(p)| = 1$ holds.
Remark: If we exchange source and target mapping, $\mathcal{F}(S)$ becomes a category of trees according to our definition. Exchanging source and target is no essential modification because it can be described by a bijective $\times$–functor.

We now give an equivalent definition of Chomsky-grammars. Let $\mathcal{F}(S)$ be the free $\times$–category corresponding to semi-Thue–system S with Alphabet V, let T be a subset of V and s an element of $V \subset T$. Then we call

$$G = (\mathcal{F}(S), T, s)$$

a Chomsky-grammar and the language generated by G is

$$L(G) = \{w \in T^* \mid \mathcal{F}(S)(s, w) \neq \emptyset\}\,.$$

This definition emphasizes the derivation processes (morphisms) and is obviously equivalent to the previous one.

In the next section, we will give different possible definitions for the language generated by a context-free grammar. Our intention is to define the language by an *interpretation* of derivations. Derivations are formalized as morphisms (nets) from a free $\times$–category, their interpretation is determined by an interpretation of the free generating system (production set of the grammar). Each production is interpreted as a function whose arguments are strings, sequential (parallel) composition of derivations is interpreted as sequential (parallel) composition of functions. Thus, one gets an interpretation for each morphism in the $\times$–category.

1.3 Context-free Chomsky Languages

In this section, we will give different presentations for the language generated by a context-free grammar. The "syntax" defined by the grammar is given in terms of a free $\times$–category (of trees), and the generated string language is defined by an "interpretation" of a certain set of trees.

We will give a rather extensive treatment here because this presentation will be used in the next chapter for defining other language classes.

Our description is essentially the same as the one given in [Go77]. In that article, the string language generated by a context-free grammar is described as an example of *initial algebra semantics*.

The productions of a CFG are regarded as operators which generate a so called *initial many-sorted algebra* I. The term *initial* means that any algebra A of the same type is the homomorphic image of I under a unique homomorphism. The members of I can be regarded as expressions built up by the operators (productions of the CFG). These expressions represent the *derivation trees* of the grammar. The string language generated by the grammar is obtained by evaluating a certain set of expressions, i.e. by interpreting the expressions in an appropriate many-sorted algebra.

Every operator is interpreted as a function with arguments from T^*, i.e. terminal words. The arity of each function is the number of nonterminals on the right side of the corresponding grammar production. If the arguments are chosen to be words derived from these nonterminals, function application gives the terminal word derived from the left side of the production.

Assignment of these functions to the operators defines an homomorphism from the initial algebra into the algebra generated by the functions. That homomorphism is the interpretation of the derivations.

A similar but more general formulation is given in [Ho67]. In that paper, to every CFG two $\times$–categories are assigned:

- a *syntactic* $\times$–category (free $\times$–category)
- a *semantic* $\times$–category ($\times$–category of mappings)

The syntactic category corresponds to the initial algebra described above. It is more general than the initial algebra because not only derivation trees but all derivations are represented in that $\times$–category. The string language generated by the grammar is given by an $\times$–functor from the syntactic to the semantic category. This $\times$–functor is the analogue to the homomorphism above.

Based on this description, a class K of formal languages is defined, the *languages generated by coupled substitutions*. Special subclasses are investigated and closure properties are proved.

Languages from class K are defined by finitely generated $\times$–subcategories of the syntactic $\times$–category. Given a context-free production system P, each production $p \in P$ is split into a terminal-free production and a function f_p which describes application of p. The functions f_p generate an $\times$–category $\mathcal{C}$ with sequential and parallel composition of mappings as $\times$–category-operations. For E a finite subset of $\mathcal{C}$, $\mathcal{C}(E)$ denotes the $\times$–subcategory generated by E in $\mathcal{C}$. It is shown that there is an $\times$–subcategory $\mathcal{U}$ of the free $\times$–category $\mathcal{F}(P)$ whose members represent the possible "constructions" of the functions from $\mathcal{C}(E)$.

Only members of that $\times$–subcategory $\mathcal{U}$ are taken as valid derivations for the strings of the language to be defined. In each valid derivation, productions of P must be applied in a *coupled* fashion which is given by the functions from E.

Finally, the string language is defined by choosing a language S of start words and applying to them the functions from $\mathcal{C}(E)$.

The class K contains string languages whose syntax cannot be defined by Chomsky-grammars although every string language from K could be defined by a Chomsky-grammar (with a different syntax). The reason for this is that *coupling* of productions cannot be described by Chomsky-grammars.

This possibility could be interesting for linguistic considerations. For the description of natural language phenomena, formal systems are required whose *generative power* is slightly stronger than CFGs but which allow expressing more syntactic dependencies than CFGs [Jo86].

In the following, we will consider tree rewriting and corresponding string rewriting systems which are related by an interpretation function. We will obtain a rewriting mechanism similar to the one mentioned above. The difference is that we get $\times$–subcategories which don't have to be finitely generated.

We formalize the above notions. At first, we will describe the string language of a CFG as "trivial" interpretation of a set of derivations.

Let $G = (S, T, s)$ be a context-free grammar, i.e.

1. $S = (P, V, Q, Z)$ is a context-free semi-Thue–system,
2. $V = N \cup T,\ N \cap T = \emptyset,\ N, T \neq \emptyset,$
3. $Q(p) \in N$ for all $p \in P$,
4. $s \in N$.

Let M be the set of all mappings $\{u\} \xrightarrow{f} \{v\},\ u, v \in V^*$, let $\circ : M \times M \rightsquigarrow M$ be the composition and $\times : M \times M \rightarrow M$ the parallel composition of mappings from M, i.e.

$$\begin{array}{ccccc} \{v\} \xrightarrow{f} \{w\} & \circ & \{u\} \xrightarrow{g} \{v\} & := & \{u\} \xrightarrow{f \circ g} \{w\} \\ \{u\} \xrightarrow{f} \{v\} & \times & \{u'\} \xrightarrow{f'} \{v'\} & := & \{uu'\} \xrightarrow{f \times f'} \{vv'\} \end{array}$$

If we define $d(f) := u$ and $c(f) := v$ for $\{u\} \xrightarrow{f} \{v\}$ from M,

$$\mathcal{R}(V) := (V^*, M, d, c, \circ, \times)$$

is an $\times$–category.

Let $\tau_1 : V^* \rightarrow V^*$ be the identity and $\tau_2' : P \rightarrow M$ be defined by

$$\tau_2'(p) \ := \ \{Q(p)\} \xrightarrow{f_p} \{Z(p)\} \qquad \forall p \in P.$$

τ_2' can be uniquely extended to $\tau_2 : Mor\,\mathcal{F}(S) \rightarrow M$ such that $\tau = (\tau_1, \tau_2)$ is an $\times$–functor from $\mathcal{F}(S)$ into $\mathcal{R}(V)$.

This $\times$–functor τ assigns to each derivation with source u and target v the function $\{u\} \xrightarrow{f} \{v\}$ and "forgets" the construction of the derivation.

Thus, the string language generated by CFG G is

$$L(G) = \{\tau(\Psi)(s) \mid \Psi \in \mathcal{F}(S)(s, T^*)\}.$$

$L(G)$ is defined by applying the interpretation functor τ to the set of all derivations (trees) with source s and terminal target. This interpretation is "trivial" in the sense that for each morphism from $\mathcal{F}(S)$ it is completely determined by source and target.

We now define the string language of CFG G in terms of syntactic and semantic $\times$–category.

The syntactic $\times$–category contains the terminal-free derivations of the grammar. The grammar productions can be regarded as "boxes" from which derivations (nets, trees) are built up using the $\times$–category-operations $\circ$ and $\times$. Given such a derivation, the derived string is given by "evaluating" that derivation in the semantic $\times$–category.

The syntactic $\times$–category is defined as follows:

To each grammar production

$$p : x_0 \to u_1 \cdot x_1 \cdot u_2 \cdots u_k \cdot x_k \cdot u_{k+1}, \quad x_i \in N,\ u_j \in T^*,\ k \in \mathbf{N}_0,$$

we assign the terminal-free production

$$p : x_0 \to x_1 \ldots x_k$$

and the mapping $f_p : (T^*)^k \to T^*$ given by

$$f_p(w_1, \ldots, w_k) := u_1 \cdot w_1 \cdot u_2 \cdots u_k \cdot w_k \cdot u_{k+1} \quad \forall w_1, \ldots, w_k \in T^*$$

If we denote source and target of the terminal-free production p by $Q'(p)$ resp. $Z'(p)$, we get a new semi-Thue–system

$$S' = (P, N, Q', Z')$$

and a corresponding free $\times$–category $\mathcal{F}(S')$. We call this free $\times$–category the *syntactic* $\times$–category of G.

The following example shows a derivation and its terminal-free version.

Example 1:

Let $S = (P, N \cup T, Q, Z)$ with $P = \{p, q\},\ N = \{s\},\ T = \{a, b, c, d\}$. Let the source and target mappings be defined by

$$Q(p) = Q(q) = s, \qquad Z(p) = asbsc,\ Z(q) = d.$$

In the semi-Thue–system $S' = (P, N, Q', Z')$, p and q have source and target

$$Q'(p) = Q'(q) = s, \qquad Z'(p) = ss,\ Z'(q) = \varepsilon.$$

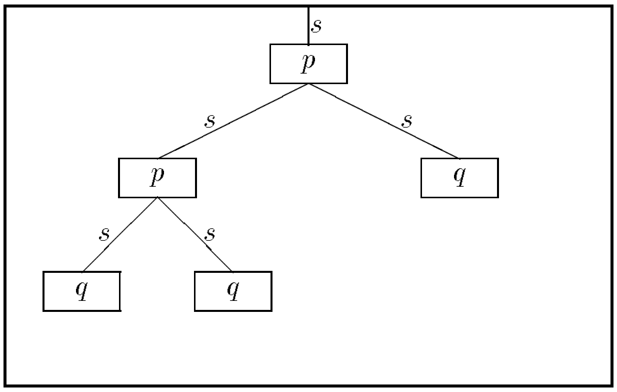

Figure 1.5: A terminal-free derivation tree

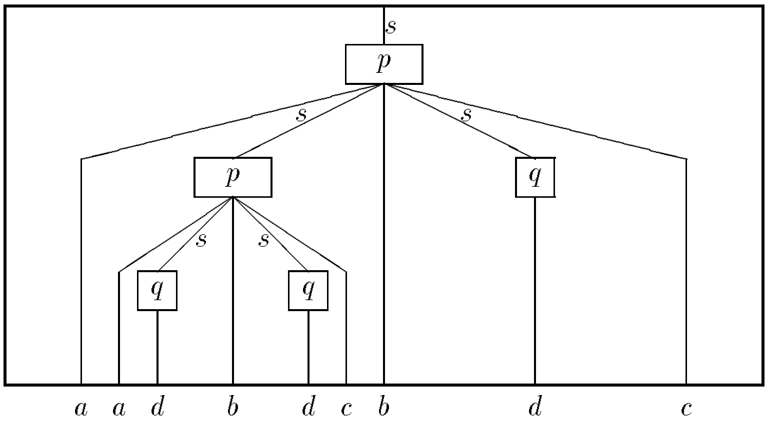

Figure 1.6: Original derivation tree

We now define the *semantic* $\times$–category of grammar G:
Let $F_P := \{f_p \mid p \in P\}$. F_P generates an $\times$–subcategory

$$\mathcal{C}(F_P, T^*) \subset MAP(T^*, \mathbf{N}_0)$$

$\mathcal{C}(F_P, T^*)$ contains exactly those mappings that can be generated by sequential and parallel composition from the mappings f_p and the identities on $(T^*)^n, n \in \mathbf{N}_0$.

We call this $\times$–category $\mathcal{C}(F_P, T^*)$ the *semantic* $\times$–category of G.

Let us now define the interpretation functor for the derivations in $\mathcal{F}(S')$. Define mappings

$$\begin{array}{lll} \eta_1 : N^* \to \mathbf{N}_0, & \eta_1(w) := |w|, & w \in N^*, \\ \eta_2' : P \ \ \to F_P, & \eta_2'(p) := f_p \ , & p \in P. \end{array}$$

By extension of η_2' onto $Mor\,\mathcal{F}(S')$ we get a unique (contravariant) $\times$–functor $\eta = (\eta_1, \eta_2)$ from $\mathcal{F}(S')$ to $\mathcal{C}(F_P, T^*)$.

η maps each derivation in the semi-Thue–system S' (each net, tree in $\mathcal{F}(S')$) to a function from $\mathcal{C}(F_P, T^*)$ and so it connects syntactic and semantic $\times$–category.

To each derivation $\Psi : s \to \varepsilon$ from $\mathcal{F}(S')$, a mapping $\eta(\Psi) : (T^*)^0 \to T^*$ is assigned which is identified with the string $\eta(\Psi)(\Box) \in T^*$.

For the productions p and q from our example above, we get functions f_p and f_q with

$$\begin{array}{rcl} f_p(w_1, w_2) & = & a \cdot w_1 \cdot b \cdot w_2 \cdot c \quad \forall w_1, w_2 \in T^* \\ f_q(\Box) & = & d \end{array}$$

The interpretation of the (terminal-free) derivation tree shown in the figure is obtained by evaluating an equivalent categorical expression, e.g.

$$((q \times q) \circ p \times q) \circ p$$

which gives

$$f_p\left(f_p\left(f_q(\Box), f_q(\Box)\right), f_q(\Box)\right) \quad = \quad aadbdcbdc$$

The string language $L(G) = L(S, T, s)$ now can be defined by the syntactic $\times$–category $\mathcal{F}(S')$, the semantic $\times$–category $\mathcal{C}(F_P, T^*)$ and the $\times$–functor η.

Let $B(S, x) := \mathcal{F}(S)(x, T^*)$ and $B(S', x) := \mathcal{F}(S')(x, \varepsilon)$ for $x \in N$. That is, $B(S, x)$ is the set of all derivation trees of a terminal word from nonterminal x in the original grammar, and $B(S', x)$ is the set of all derivation trees of the empty word from x in the $\times$–category generated by the terminal-free productions. Then the following theorem holds:

Theorem 2

$$L(G) = \{\eta(\Psi)(\Box) \mid \Psi \in B(S', s)\}$$

Proof:

The proof bases on the connection between the free $\times$–categories $\mathcal{F}(S)$ and $\mathcal{F}(S')$. The result is that removing the terminal symbols from the productions preserves the syntactic structure, i.e. terminals are unessential. We formalize this fact.

Let $\lambda_1 : V^* \to N^*$ be the homomorphism

$$\lambda_1(x) := \begin{cases} x, & x \in N \\ \varepsilon, & x \in T \end{cases}$$

which deletes the terminal symbols from a string, and let

$$\lambda_2' : P \to \mathcal{F}(S'), \qquad \lambda_2'(p) := p$$

be the mapping which assigns to each production its terminal-free version (which carries the same name).

Mappings λ_1 and λ_2' define a unique $\times$–functor $\lambda = (\lambda_1, \lambda_2)$ from $\mathcal{F}(S)$ into $\mathcal{F}(S')$. This functor describes removal of terminals from the productions resp. from the derivations in S. The set of trees which defines the language generated by G is only unessentially changed by that mapping. This is expressed by the following facts:

Fact 1.3.1 *Every tree $\Psi \in B(S,x)$, $x \in N$, has a unique presentation*

$$\Psi = (\mathbf{1}_{u_1} \times \Psi_1 \times \mathbf{1}_{u_2} \times \cdots \times \mathbf{1}_{u_k} \times \Psi_k \times \mathbf{1}_{u_{k+1}}) \circ p$$

with $p \in P$, $Q(p) = x$, $Z(p) = u_1 \cdot x_1 \cdot u_2 \cdots u_k \cdot x_k \cdot u_{k+1}$, $u_j \in T^*$, $x_i \in N$, $\Psi_i \in B(S, x_i)$, $k \in \mathbf{N}_0$, $1 \le i \le k$, $1 \le j \le k+1$.

Fact 1.3.2 *Every tree $\Psi' \in B(S',x)$, $x \in N$, has a unique presentation*

$$\Psi' = (\Psi_1' \times \cdots \times \Psi_k') \circ p$$

with $p \in P$, $Q'(p) = x$, $Z'(p) = x_1 \cdots x_k$, $x_i \in N$, $\Psi_i' \in B(S', x_i)$, $k \in \mathbf{N}_0$, $1 \le i \le k$.

Remark: Follows directly from the definition of a tree and from lemma 1.1.2.

Fact 1.3.3 *If $|.|$ denotes the length of the morphisms in $\mathcal{F}(S)$ resp. in $\mathcal{F}(S')$ and is λ the $\times$–functor defined above, then for all $\Psi \in \mathcal{F}(S)$ holds:*

$$|\lambda(\Psi)| = |\Psi| \ .$$

Follows from the definition of λ.

Lemma 1.3.1 *λ is a bijection between $B(S,x)$ and $B(S',x)$.*

Proof:
We show by induction on the length of the trees that every $\Psi' \in B(S',x)$ has exactly one origin in $B(S,x)$ under λ.

Basis: $|\Psi'| = 1$

Then $\Psi' = p \in P$, $Z'(p) = \varepsilon$, $Z(p) \in T^*$ and therefore $p \in \lambda^{-1}(\Psi') \cap B(S,x)$. By fact 1.3.3, every g in $\lambda^{-1}(\Psi')$ has length 1, so it has the form $g = \mathbf{1}_{w_1} \times q \times \mathbf{1}_{w_2}$ with $w_1, w_2 \in T^*$ and $q \in P$. If $g \in B(S,x)$, then $w_1 = w_2 = \varepsilon$ and therefore $g = q$. Since λ is the identity on P, it follows $g = p$.

Step: $|\Psi'| > 1$

Let $\Psi' = (\Psi_1' \times \cdots \times \Psi_k') \circ p$ be the unique presentation of Ψ' from fact 1.3.2.

By induction hypothesis, every Ψ_i' in $B(S,x)$ has a unique origin Ψ_i. The unique origin of p in $B(S,x)$ is p itself. By construction of S',

$$Z(p) = u_1 \cdot x_1 \cdot u_2 \cdots u_k \cdot x_k \cdot u_{k+1}, \quad x_i \in N,\ u_j \in T^*,$$

and therefore

$$\Psi := (\mathbf{1}_{u_1} \times \Psi_1 \times \mathbf{1}_{u_2} \times \cdots \times \mathbf{1}_{u_k} \times \Psi_k \times \mathbf{1}_{u_{k+1}}) \circ p \quad \in \lambda^{-1}(\Psi') \cap B(S,x).$$

Let g be an arbitrary element of $\lambda^{-1}(\Psi') \cap B(S,x)$ and

$$g = (\mathbf{1}_{w_1} \times g_1 \times \mathbf{1}_{w_2} \times \cdots \times \mathbf{1}_{w_l} \times g_l \times \mathbf{1}_{w_{l+1}}) \circ q, \qquad l \in \mathbf{N}_0,$$

be the unique representation of g from fact 1.3.1. Then

$$\begin{aligned} \lambda(g) &= (\lambda(g_1) \times \cdots \times \lambda(g_l)) \circ \lambda(q) \\ &= \Psi' \\ &= (\Psi'_1 \times \cdots \times \Psi'_k) \circ p \end{aligned}$$

From $\lambda(q) = q$ and the uniqueness of the presentation it follows

$$q = p, \quad l = k, \quad Q'(\lambda(g_i) = x_i, \quad \lambda(g_i) = \Psi'_i \in B(S', x_i)$$

So we get

$$g = (\mathbf{1}_{u_1} \times g_1 \times \mathbf{1}_{u_2} \times \cdots \times \mathbf{1}_{u_k} \times g_k \times \mathbf{1}_{u_{k+1}}), \quad g_i \in B(S, x_i),\ 1 \leq i \leq k.$$

By induction hypothesis, the origin of Ψ'_i in $B(S, x_i)$ is uniquely determined and from $\lambda(g_i) = \Psi'_i = \lambda(\Psi_i)$ it follows $g_i = \Psi_i$. So we have $g = \Psi$ and the lemma is shown. $\square$

Lemma 1.3.2

$$\forall\ \Psi \in B(S,x),\ x \in N\ :\ \eta(\lambda(\Psi))(\square) = Z(\Psi)$$

Proof: Induction over the length of $\Psi \in B(S,x)$.
Basis: $|\Psi| = 1$
Then $\Psi = p : x \to w \in P,\ w \in T^*$ and $f_p : \{\square\} \to T^*,\ f_p(\square) = w$. From $\lambda(p) = p$, it follows $\eta(\lambda(\Psi))(\square) = \eta(p)(\square) = f_p(\square) = w = Z(\Psi)$.

Induction step: $|\Psi| > 1$
Let

$$\Psi = (\mathbf{1}_{w_1} \times \Psi_1 \times \mathbf{1}_{w_2} \times \cdots \times \mathbf{1}_{w_k} \times \Psi_k \times \mathbf{1}_{w_{k+1}}) \circ p$$

be the unique presentation of Ψ from fact 1.3.1. Then

$$\begin{aligned} Z(\Psi) &= w_1 \cdot Z(\Psi_1) \cdot w_2 \ldots w_k \cdot Z(\Psi_k) \cdot w_{k+1} \\ &= w_1 \cdot \eta(\lambda(\Psi_1))(\square) \cdot w_2 \ldots w_k \cdot \eta(\lambda(\Psi_k))(\square) \cdot w_{k+1} \\ &= f_p(\ \eta(\lambda(\Psi_1))(\square), \ldots \eta(\lambda(\Psi_k)\ )(\square) \\ &= \eta(\lambda(\Psi))(\square) \end{aligned}$$

and the lemma is shown. $\square$

From both lemmata, we get

$$\begin{aligned} L(G) &= \{Z(\Psi) \mid \Psi \in B(S,x)\} \\ &= \{\eta(\lambda(\Psi))(\square) \mid \Psi \in B(S,x)\} \\ &= \{\eta(\Psi')(\square) \mid \Psi' \in B(S',x)\} \end{aligned}$$

and the theorem is proved. $\square$

Remark: From fact 1.3.2, one immediately gets a recursive definition for the sets $B(S', x)$ which can be formulated as a (tree) grammar on $\mathcal{F}(S')$. This tree grammar is similar to the *regular grammar on a many-sorted algebra* defined in [Mai74]. The productions of the (context-free) semi-Thue–system S' become the terminal symbols and the sets $B(S', x)$, shortly written as $[x]$, $x \in N$, become the nonterminals of this tree grammar. The production set $\tilde{P}$ of the tree grammar is given as follows:

For every $p : x \to x_1 \cdots x_k \in P(S')$, there is a production

$$\tilde{p} : [x] \to ([x_1] \times \cdots \times [x_k]) \circ p \quad \in \tilde{P}$$

in the tree grammar.

For consistency, we define for the nonterminals $[x]$ source and target mappings by

$$Q'([x]) := x, \quad Z'([x]) := \varepsilon$$

Then for every production $\tilde{p}$ of the tree grammar the following *source/target-condition* holds:

$$\tilde{p} : [x] \to \Psi \in \tilde{P} \quad \Longrightarrow \quad Q'([x]) = Q'(\Psi),\ Z'([x]) = Z'(\Psi)$$

This condition will also be assumed for the tree grammars defined in the next chapter.

We generalize from this type of grammar in a natural way. Not only do we allow “leaves” to be rewritten, but also boxes occurring anywhere in the tree. Our rewriting mechanism does not allow duplication or deletion of subtrees as do the *context-free tree grammars* defined e.g. in [Mai74].

1.4 Coupled-context-free Grammars

Coupled-context-free Grammars (CCFG) are a generalization of context-free grammars. The CCFG allows several context-free rewritings to be executed in parallel. These parallel rewritings are not executed independently but they are *coupled* in the following way. Nonterminal symbols are grouped together to *brackets* (in a generalized sense). Rewriting rules and rewriting relation are defined such that each rule application respects the bracketing structure of the sentence. In each rewriting step, only corresponding brackets can be substituted. The result of the substitution is correctly bracketed iff the original word was. Terminal symbols are "normal" alphabet symbols which can be regarded as "trivial" brackets (which enclose no word). We will consider not only pairs of opening and closing brackets (in our sense: bracket components), but also brackets consisting of an arbitrary finite number of components.

We will formalize the above notions starting from the usual definition of the *semi-Dyck-languages*.

A semi-Dyck-language can be considered as the set of all correctly bracketed words over a finite set of *bracket pairs*. This can be formalized by giving a set of equations of the form $\{a_1\bar{a}_1 = \varepsilon, \ldots, a_k\bar{a}_k = \varepsilon\}$, where a_i is considered as opening bracket corresponding to closing bracket $\bar{a}_i$. These equations induce a congruence relation on the free monoid over the alphabet consisting of all bracket pairs. Two words are congruent iff they can be transformed into each other by a finite number of equation applications, i.e. insertions or deletions of factors $a_i\bar{a}_i$.

The set of all *correctly bracketed* words, the *semi-Dyck-language*, is the congruence class of the empty word. For a formal definition see e.g. [Ho90] or [Ber79].

We want to generalize from these notions by allowing *bracket tuples* consisting of a finite number of components.

We consider a finite alphabet (brackets) and define for each bracket the number of words that can be enclosed, called *degree* of the bracket.

Definition 16 *bracket alphabet*
Let K be a finite set (*brackets*) and $g : K \to \mathbf{N}_0$ a mapping.

$$[K, g] := \{(k, i) \in K \times \mathbf{N} \mid 1 \leq i \leq g(k) + 1\}$$

is called *bracket alphabet* and $g(k)$ is the *degree* of bracket $k \in K$.

For (k, i) we write k_i. Let K_n be the set of all brackets in K of degree $n \in \mathbf{N}_0$. Sometimes we will identify brackets of degree 0 with their (single) component.

Definition 17 *bracket congruence*
The congruence $\equiv_K$ on $[K, g]^*$, generated by the equations

$$\left\{ \left(\prod_{i=1}^{g(k)+1} k_i \right) = \varepsilon, \quad k \in K \right\}$$

is called *bracket congruence* for the bracket alphabet $[K, g]$.

Definition 18 *bracket language*
The *bracket language* $D(K, g)$ over the bracket alphabet $[K, g]$ is

$$D(K, g) := \{w \in [K, g]^* \mid w \equiv_K \varepsilon\},$$

i.e. the set of words that can be reduced to the empty word by application of the defining equations. We simply write $D(K)$ if g is implicitly given.

The bracket language can be characterized constructively:
$D(K)$ is the smallest subset of $[K, g]^*$ satisfying conditions 1, 2 and 3:

1. $\varepsilon \in D(K)$,
2. $D(K) \cdot D(K) \subseteq D(K)$,
3. $k \in K_n,\ w_1, \ldots, w_n \in D(K) \quad \Longrightarrow \quad k_1 \cdot w_1 \cdot k_2 \cdots k_n \cdot w_n \cdot k_{n+1} \ \in D(K)$.

Let us give some examples:

1. Let $K = \{x, y\}$ and $g(x) = g(y) = 1$, so $[K, g] = \{x_1, x_2, y_1, y_2\}$. Then the following are elements of $D(K, g)$:

$$\varepsilon, \quad x_1 x_2, \quad x_1 y_1 y_2 x_1 x_2 x_2, \quad x_1 x_1 x_1 x_2 x_2 x_2$$

   If we write "{" and "}" instead of x_1 and x_2, "(" and ")" instead of y_1 and y_2, then the words given above become

$$\varepsilon, \quad \{\ \}, \quad \{\ (\ )\ \{\ \}\ \}, \quad \{\ \{\ \{\ \}\ \}\ \}$$

   In this example, $D(K, g)$ is the semi-Dyck-language with corresponding pairs of brackets $x_1 \leftrightarrow x_2$ and $y_1 \leftrightarrow y_2$. The bracket languages with each bracket of degree 1 are exactly the semi-Dyck-languages.

2. For a bracket alphabet $[K, g]$ with $K = K_0$, we have $D(K, g) \cong K^*$. In this case, $[K, g]$ and K are identified.

In our considerations there will be often bracket alphabets $[K, g]$ with $K = N \dot{\cup} T$, $T = T_0$. We denote the corresponding bracket language by $D(N, T)$ and consider it as a subset of $([N, g] \cup T)^*$.

We give some properties of the bracket languages $D(K)$:

1. For each subset $M \subset K_0$ it holds $M^* \subset D(K)$.
2. $D(K)$ is a free submonoid of $[K, g]^*$. The free generating system of $D(K)$ is

$$E(K) := \bigcup_{k \in K} \{k_1 \cdot d_1 \cdot k_2 \cdots k_n \cdot d_n \cdot k_{n+1} \mid k \in K_n,\ d_1, \ldots, d_n \in D(K)\}.$$

3. If $d \in D(K)$, then for $u, v \in [K, g]^*$

$$udv \in D(K) \quad \Longleftrightarrow \quad uv \in D(K).$$

4. For $u, v \in [K, g]^*,\ u \cdot v \in D(K)$, it holds:

$$u \in D(K) \quad \Longleftrightarrow \quad v \in D(K).$$

We define now the *coupled-context-free grammars.*

Definition 19 *coupled-context-free grammar (CCFG)*
Let $[V,g]$ be a bracket alphabet with $V = N \dot{\cup} T,\ N,T \neq \emptyset$ and $T = T_0$. Let $D(N,T)$ be the corresponding bracket language.

$$G_{ccf} = \left([V,g], T, P_{ccf}, s, \overset{*}{\Rightarrow}_{ccf}\right)$$

is called *coupled-context-free grammar* if

1. P_{ccf} is a finite set of *coupled-context-free productions*

$$p : (x_1, \ldots, x_{k+1}) \rightarrow (y_1, \ldots, y_{k+1})$$

   with $x \in N_k,\ y_1, \ldots, y_{k+1} \in [V,g]^*,\ y_1 \cdots y_{k+1} \in D(N,T)$.

   We call $grad(p) := k$ the *degree* of production p. Define the degree of the grammar as the maximum degree of a production.

2. The *rewriting relation* $\overset{*}{\Rightarrow}_{ccf}$ is the reflexive, transitive closure of the relation

   $\Rightarrow_{ccf}\ :=$
   $\{\ \ (\ u_1 \cdot x_1 \cdot v_1 \cdot x_2 \cdots x_k \cdot v_k \cdot x_{k+1} \cdot u_2,\ \ u_1 \cdot y_1 \cdot v_1 \cdot y_2 \cdots y_k \cdot v_k \cdot y_{k+1} \cdot u_2\ )\ |$
   $u_1, u_2 \in [V,g]^*,\ v_1, \ldots, v_k \in D(N,T),$
   $p : (x_1, \ldots, x_{k+1}) \rightarrow (y_1, \ldots, y_{k+1}) \in P_{ccf}\quad \}.$

3. $s \in N_0$ is the *axiom* of the grammar.

The *language generated* by G_{ccf} is

$$L(G_{ccf}) := \left\{ w \mid w \in T^*,\ s \overset{*}{\Rightarrow}_{ccf} w \right\}.$$

The coupled-context-free grammars generalize ordinary context-free grammars. Consider the case where $N = N_0$, i.e. each nonterminal is a single symbol. Then we can identify each $x_1 \in [N,g]$ with $x \in N$, thus $[N,g]$ with N, and V with $[V,g]$.

The productions of G_{ccf} have the form $p : (x_1) \rightarrow (y_1),\ x_1 \in [N,g] = N,\ y_1 \in [V,g]^* = V^*$, i.e. they are context-free productions.

The relation $\Rightarrow_{ccf}$ is in this case

$$\Rightarrow_{ccf}\ = \{(u_1 \cdot x_1 \cdot u_2,\ u_1 \cdot y_1 \cdot u_2) \mid u_1, u_2 \in V^*,\ p : (x_1) \rightarrow (y_1) \in P_{ccf}\},$$

i.e. $\overset{*}{\Rightarrow}_{ccf}$ is the semi-Thue-relation.

Example 1:
Let $G_{ccf} := (\hat{V}, T, P, s, \overset{*}{\Rightarrow}_{ccf}),\qquad \hat{V} := \{s,\ x_1, x_2,\ a, b, c, e\},\qquad T := \{a, b, c, e\},$
$P := \{\ p : s \rightarrow x_1 e x_2,\quad q : (x_1, x_2) \rightarrow (a x_1 b, c x_2),\quad r : (x_1, x_2) \rightarrow (ab, c)\ \}.$
A derivation in G_{ccf} is

$$s \Rightarrow_{ccf} x_1 e x_2 \overset{n}{\Rightarrow}_{ccf} a^n x_1 b^n e c^n x_2 \Rightarrow_{ccf} a^{n+1} b^{n+1} e c^{n+1},\ n \in \mathbf{N}_0$$

It is easy to see that $L(G_{ccf}) = \{a^n b^n e c^n \mid n \in \mathbf{N}\}$. This is a context-sensitive language which is not context-free. By the way, $L(G_{ccf}) \in TAL$, i.e. $L(G_{ccf})$ is generated by a *Tree Adjoining Grammar (TAG)* [Jo85].

Example 2:
Let $G_{ccf} := (\hat{V}, T, P, s, \overset{*}{\Rightarrow}_{ccf})$, $\hat{V} := \{s,\ x_1, x_2,\ a, b\}$, $T := \{a, b\}$, $P :=$

$\{p:\ s \to x_1 x_2,\ q_a:\ (x_1, x_2) \to (a x_1, a x_2),\ q_b:\ (x_1, x_2) \to (b x_1, b x_2),\ q_\varepsilon:\ (x_1, x_2) \to (\varepsilon, \varepsilon)\}$

The derivations of G_{ccf} with source s are of the form

$$\begin{array}{lll} s & \overset{p}{\Rightarrow}_{ccf} & x_1 \cdot x_2 \\ & \overset{q_{a_1}}{\Rightarrow}_{ccf} & a_1 x_1 \cdot a_1 x_1 \\ & \vdots & \\ & \overset{q_{a_n}}{\Rightarrow}_{ccf} & a_1 \ldots a_n x_1 \cdot a_1 \ldots a_n x_2 \\ & \overset{q_\varepsilon}{\Rightarrow}_{ccf} & a_1 \ldots a_n \cdot a_1 \ldots a_n \end{array}$$

where $a_1, \ldots, a_n \in \{a, b\},\ n \in \mathbf{N}_0$.

Thus, $L(G_{ccf}) = \{w \cdot w \mid w \in \{a,b\}^*\} = COPY_1$. This is also a non-context-free, context-sensitive language which is contained in the class TAL. These examples show that by introducing nonterminals of degree 1 one gets rewriting systems with more generation power than context-free grammars.

The following lemma shows that CCFG-rewriting preserves bracketing, especially that correctly bracketed words are invariant under CCFG-rewriting.

Lemma 1.4.1 *For $w, w' \in [V, g]^*$ holds:* $w \overset{*}{\Rightarrow}_{ccf} w' \quad \Longrightarrow \quad w \equiv_V w'$

Proof: Induction on the derivation length.

Basis: $w \overset{0}{\Rightarrow}_{ccf} w'$. Then $w = w'$ and thus $w \equiv_V w'$.

Induction step: $w \overset{n+1}{\Rightarrow}_{ccf} w',\ n \in \mathbf{N}_0$.

There is a $w'' \in [V, g]^*$, such that $w \overset{n}{\Rightarrow}_{ccf} w'' \Rightarrow_{ccf} w'$. By definition of $\Rightarrow_{ccf}$, there are a production $p : (x_1, \ldots, x_{k+1}) \to (y_1, \ldots, y_{k+1})$ and words $w_1'', w_2'' \in [V, g]^*,\ d_1, \ldots, d_k \in D(N, T)$ with

$$\begin{array}{rcl} w'' & = & w_1'' \cdot x_1 \cdot d_1 \cdot x_2 \cdots x_k \cdot d_k \cdot x_{k+1} \cdot w_2'' \\ w' & = & w_1'' \cdot y_1 \cdot d_1 \cdot y_2 \cdots y_k \cdot d_k \cdot y_{k+1} \cdot w_2'' \end{array}$$

From $d_i \equiv_V \varepsilon,\ 1 \le i \le k$ and $x_1 \cdots x_{k+1} \equiv_V \varepsilon$ follows

$$\begin{array}{rcl} w'' & = & w_1'' \cdot x_1 \cdot d_1 \cdot x_2 \cdots x_k \cdot d_k \cdot x_{k+1} \cdot w_2'' \\ & \equiv_V & w_1'' \cdot x_1 \cdots x_{k+1} \cdot w_2'' \\ & \equiv_V & w_1'' \cdot w_2'' \end{array}$$

By definition of P_{ccf}, we have $y_1 \cdots y_{k+1} \equiv_V \varepsilon$ and thus

$$\begin{array}{rcl} w' & = & w_1'' \cdot y_1 \cdot d_1 \cdot y_2 \cdots y_k \cdot d_k \cdot y_{k+1} \cdot w_2'' \\ & \equiv_V & w_1'' \cdot y_1 \cdots y_{k+1} \cdot w_2'' \\ & \equiv_V & w_1'' \cdot w_2'' \end{array}$$

By induction hypothesis, we have $w \equiv_V w''$ thus $w \equiv_V w'$. □

Conclusion: For each $w \in [V,g]^*$ with $s \overset{*}{\Rightarrow}_{ccf} w$ holds: $w \in D(N,T)$.
That is, "sentential forms" of a CCFG are members of the bracket language $D(N,T)$.

Let us now investigate the connection between derivations of a CCFG and the derivations of the context-free grammar which is obtained by "uncoupling" of the productions. We get this CFG by using the components of the CCFG-productions as single context-free productions and rewriting by the semi-Thue–relation.

Definition 20 *context-free base grammar*
For a CCFG $G_{ccf} = ([V,g],T,P_{ccf},s,\overset{*}{\Rightarrow}_{ccf})$, we call

$$G_{cf} := ([V,g],T,P_{cf},s,\overset{*}{\Rightarrow})$$

with

$$P_{cf} := \{p_i : x_i \to y_i \mid \exists p : (x_1,\ldots,x_{k+1}) \to (y_1,\ldots,y_{k+1}) \in P_{ccf},\ 1 \leq i \leq k+1\}$$

the *context-free base grammar* for CCFG G_{ccf}. ($\overset{*}{\Rightarrow}$ is the semi-Thue–relation.)

Clearly, G_{cf} is a context-free grammar. Derivations of CFG G_{cf} are the morphisms from the free $\times$–category $\mathcal{F}(G_{cf}) := \mathcal{F}(P_{cf},[V,g])$ generated by P_{cf}. We want to characterize the derivations of CCFG G_{ccf} using $\mathcal{F}(G_{cf})$. The following theorem holds:

Theorem 3 *If $G_{ccf} = ([V,g],T,P_{ccf},s,\overset{*}{\Rightarrow}_{ccf})$ is a CCFG and $u,v \in [V,g]^*$, then*

$$u \overset{*}{\Rightarrow}_{ccf} v \quad \Longleftrightarrow \quad \mathcal{U}(P_{ccf})(u,v) \neq \emptyset\ .$$

$\mathcal{U}(P_{ccf})$ is the $\times$–subcategory of $\mathcal{F}(G_{cf})$ with free generating system

$$\begin{aligned} E(P_{ccf}) := \{ \quad & p_1 \times \mathbf{1}_{d_1} \times p_2 \times \cdots \times p_k \times \mathbf{1}_{d_k} \times p_{k+1} \mid \\ & p \in P_{ccf},\ grad(p) = k,\ d_i \in D(N,T),\ 1 \leq i \leq k \quad \}. \end{aligned}$$

Proof:

Let $f \in \mathcal{U}(P_{ccf})(u,v),\ u,v \in [V,g]^*$. We have one of the following cases:

1. f is a unit, $f = \mathbf{1}_u,\ u = v$. Then $u \overset{*}{\Rightarrow}_{ccf} v$ holds.

2. $f \in E(P_{ccf})$ is a generator, $f = p_1 \times \mathbf{1}_{d_1} \times p_2 \times \cdots \times p_k \times \mathbf{1}_{d_k} \times p_{k+1}$,
   $p : (x_1,\ldots,x_{k+1}) \to (y_1,\ldots,y_{k+1}) \in P_{ccf},\ d_1,\ldots,d_k \in D(N,T)$,
   $Q(f) = x_1 \cdot d_1 \cdot x_2 \cdots x_k \cdot d_k \cdot x_{k+1}, \quad Z(f) = y_1 \cdot d_1 \cdot y_2 \cdots y_k \cdot d_k \cdot y_{k+1}$.

   Then $Q(f) \Rightarrow_{ccf} Z(f)$ by definition of $\Rightarrow_{ccf}$.

3. $f = f' \times f'',\ f',f'' \in \mathcal{U}(P_{ccf})$. If $Q(f') \overset{*}{\Rightarrow}_{ccf} Z(f')$ and $Q(f'') \overset{*}{\Rightarrow}_{ccf} Z(f'')$, then $Q(f) \overset{*}{\Rightarrow}_{ccf} Z(f)$ by definition.

4. $f = f'' \circ f',\ f',f'' \in \mathcal{U}(P_{ccf})$. If $Q(f') \overset{*}{\Rightarrow}_{ccf} Z(f') = Q(f'')$ and $Q(f'') \overset{*}{\Rightarrow}_{ccf} Z(f'')$, then $Q(f') \overset{*}{\Rightarrow}_{ccf} Z(f'')$ by definition.

Each morphism from $\mathcal{U}(P_{ccf})$ can be uniquely decomposed into elements from $E(P_{ccf})$ and units, from which follows one direction of the claim. We will now show the opposite direction.

Let $u, v \in [V, g]^*$ and let $u \overset{n}{\Rightarrow}_{ccf} v$ be a derivation of G_{ccf} from u to v, $n \in \mathbf{N}_0$.
We use induction on the length of the derivation.
Basis: $u \overset{0}{\Rightarrow}_{ccf} v$. Then $u = v$ and from $\mathbf{1}_u \in \mathcal{U}(P_{ccf})(u, v)$ follows the claim.
Step: $u \overset{n+1}{\Rightarrow}_{ccf} v$. Then there is a $v' \in [V, g]^*$ with $u \overset{n}{\Rightarrow}_{ccf} v' \Rightarrow_{ccf} v$ and

$$\begin{aligned} v' &= w_1 \cdot x_1 \cdot d_1 \cdot x_2 \cdots x_k \cdot d_k \cdot x_{k+1} \cdot w_2 \ , \\ v &= w_1 \cdot y_1 \cdot d_1 \cdot y_2 \cdots y_k \cdot d_k \cdot y_{k+1} \cdot w_2 \ , \end{aligned}$$

with $w_1, w_2 \in [V, g]^*$, $d_1, \ldots, d_k \in D(N, T)$, and there is a production $p : (x_1, \ldots, x_{k+1}) \to (y_1, \ldots, y_{k+1}) \in P_{ccf}$.

By induction hypothesis, there is a morphism $f' \in \mathcal{U}(P_{ccf})(u, v')$ and by definition, $E(P_{ccf})$ contains the element $p(d_1, \ldots, d_k) := (p_1 \times \mathbf{1}_{d_1} \times p_2 \times \cdots \times p_k \times \mathbf{1}_{d_k} \times p_{k+1})$.

Then $f := (\mathbf{1}_{w_1} \times p(d_1, \ldots, d_k) \times \mathbf{1}_{w_2}) \circ f' \in \mathcal{U}(P_{ccf})(u, v)$ which proves the theorem. □

The characterization of the CCFG-derivations as elements from an $\times$–subcategory of a context-free $\times$–category could be helpful in solving the word problem "$w \in L(G_{ccf})$?". A possible algorithm could first solve the word problem for the context-free base grammar and then (efficiently ?) test the computed derivations for membership in the $\times$–subcategory.

There is hope to get better recognition/parsing algorithms for interesting language classes by that strategy (e.g. for TALs, which can be generated by CCFGs as will be shown).

It is also possible to describe the derivations of a CCFG as members of a *finitely* generated $\times$–subcategory. To achieve this, we describe the generated string language by an interpretation of a certain set of morphisms (as in the previous section). These morphisms are built up by the terminal-free productions of the CCFG.

Terminal-free productions have the following form:

$$p : (x_1, \ldots, x_{k+1}) \to (y_1, \ldots, y_{k+1})$$

with $y_1 \cdots y_{k+1} \in D(N)$, $y_i \in [N, g]^*$, $1 \le i \le k + 1$.

We split each component of a CCFG-production into a (terminal-free) production and an interpretation function. The new context-free production inherits the name of the component, its source and target are obtained by erasing all terminal symbols. The new production system generates a free $\times$–category $\mathcal{F}'(G_{ccf})$.

The language generated by the CCFG is defined by morphisms $f : s \to \varepsilon$ from $\mathcal{F}'(G_{ccf})$, exactly by those which present a coupled derivation. The coupled derivations are exactly the members of the $\times$–subcategory $\mathcal{U}'(P_{ccf})$ which is generated by

$$E'(P_{ccf}) := \{p_1 \times \mathbf{1}_{d_1} \times p_2 \times \cdots \times p_k \times \mathbf{1}_{d_k} \times p_{k+1} \mid p \in P_{ccf},\ grad(p) = k\}$$

with $d_i \in D(N)$, $1 \le i \le k$.

A word $w \in T^*$ is a member of $L(G_{ccf})$ iff there is a derivation $f : s \to \varepsilon \in \mathcal{U}'(P_{ccf})$ with $\eta(f)(\square) = w$.

A derivation of this form can be obtained by rewriting in each step only a simple nonterminal or a bracket nonterminal enclosing only empty words. That is, in each step a word $x_1 \cdots x_{k+1}$, $x \in N_k$, is rewritten. Since finally the empty word is derived, all bracket nonterminals can be substituted *after* substituting enclosed words.

Such a derivation can be presented as the sequential composition of generators of the form $p_1 \times \cdots \times p_{k+1}$ where $p \in P_{ccf}$ is a production of degree k from the CCFG. Thus, every word in the language of the CCFG has a derivation which is generated by the finite system

$$E''(P_{ccf}) := \{p_1 \times \cdots \times p_{k+1} \mid p \in P_{ccf},\ k = grad(p)\}$$

Without eliminating terminals from the productions, a finite generating system is in general not sufficient because arbitrary long terminal subwords could separate bracket nonterminals. Thus, there would be need for generators of the form

$$p_1 \times \mathbf{1}_{w_1} \times p_2 \times \cdots \times p_k \times \mathbf{1}_{w_k} \times p_{k+1}, \quad w_1, \ldots, w_k \in T^*$$

In the following sections, we will investigate the connection between CCFGs and rewriting on trees. We first define grammars on trees and prove some formal properties of these systems.

Chapter 2

Tree Grammars with Multilinear Interpretation

2.1 Tree Grammars

We will define grammars on categories of trees. These grammars are similar to context-free grammars on strings in the sense that single nonterminals are rewritten by composed structures (nets, trees) independently of the "context". An occurrence of a nonterminal symbol in a string can always be rewritten if there is a rule with that nonterminal as its left side. In order to get this property also for nets (or trees), we restrict the right side of each grammar production to be a net with the same number of inputs and outputs as the nonterminal symbol on its left side. This property will be called *source/target-condition* for the rewriting rules.

Let $\mathcal{B}_V := \mathcal{B}(V, Q, Z)$ be a category of trees ($Z(v) = 1 \quad \forall v \in V$), $V = N \dot\cup T$, $N, T \neq \emptyset$, and let $\mathcal{B}_T$ be the free $\times$–subcategory of $\mathcal{B}_V$ generated by T (corresponds to T^* in the case of string grammars). Members of $\mathcal{B}_V$ are nets (as presented in the first chapter) which can be decomposed into trees and units under the $\times$-operation. All wires are marked with the same symbol which is therefore omitted (Thus the source of a net can be identified with the number of its inputs, the same goes for the target).

Definition 21 *tree grammar*
Let $\mathcal{B}_V, \mathcal{B}_T$ be given as above. $G = (V, T, P, s, \stackrel{*}{\Rightarrow}_P)$ is called *tree grammar* over $\mathcal{B}_T$ if

1. P is a finite set of *productions* $p : x \to y$ with
   (a) $x \in N, \ y \in \mathcal{B}_V$.
   (b) $Q(x) = Q(y), \ Z(x) = Z(y) = 1.$ \quad (*source/target-condition*)
2. The *rewriting-relation* $\stackrel{*}{\Rightarrow}_P$ is the reflexive, transitive closure of the relation $\Rightarrow_P$ on $\mathcal{B}_V$ defined as follows: $\forall f, \ f' \in \mathcal{B}_V$

$$f \Rightarrow_P f' :\Longleftrightarrow \begin{cases} f = g \circ (\mathbf{1}_r \times x \times \mathbf{1}_s) \circ h, \ f' = g \circ (\mathbf{1}_r \times y \times \mathbf{1}_s) \circ h \quad r, s \in \mathbf{N}_0, \\ p : x \to y \in P, \ g \in \mathcal{B}_V(r + Z(x) + s, \mathbf{N}_0), \ h \in \mathcal{B}_V(\mathbf{N}_0, r + Q(x) + s) \end{cases}$$

3. The *axiom* of G is an element $s \in N$ with $Q(s) = 0$.

Remark: The source/target-condition guarantees that each occurrence of a nonterminal x in a net can always be rewritten by application of a production $p : x \to y$, and that the result of the substitution is a well defined net. If rewriting always takes place in trees without inputs, there is always a presentation of the tree with $r = s = 0$. Then the definition of the rewriting relation $\Rightarrow_P$ above can be simplified to $g \circ x \circ h \Rightarrow_P g \circ y \circ h$.

We illustrate the definitions by the following pictures.

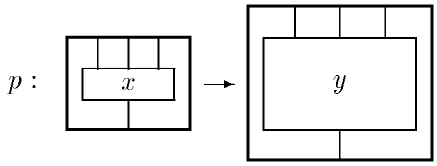

Figure 2.1: rewriting rule

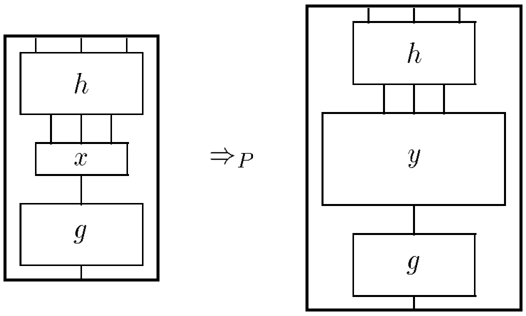

Figure 2.2: rule application

The *tree language* generated by tree grammar G is

$$T(G) := \left\{ f \mid s \overset{*}{\Rightarrow}_P f,\ f \in \mathcal{B}_T \right\}.$$

We draw some simple conclusions from these definitions:

1. $f, g \in \mathcal{B}_V,\ f \overset{*}{\Rightarrow}_P g \quad \Longrightarrow \quad Q(f) = Q(g),\ Z(f) = Z(g).$

2. $T(G) \subseteq \mathcal{B}_T(0,1)$, i.e. a tree grammar generates trees without inputs.

3. If $T(G_1)$ and $T(G_2)$ are tree languages, then $T(G_1) \cup T(G_2)$ is a tree language.

As we will see, the generation power of a tree grammar depends on the number of inputs of its nonterminals. This motivates the following

Definition 22 *degree of a tree grammar*

$$grad\, G := \max \{Q(x) \mid p : x \to y \in P\}.$$

We denote the class of tree languages in $\mathcal{B}_T$ which are generated by a tree grammar of degree k by $CF(\mathcal{B}_T, k),\ k \in \mathbf{N}_0$.

We will always assume that the grammars are reduced (the definition of reduced grammar is analogous to context-free string grammars and is omitted).

A *derivation* from f_0 to f_n in G is a sequence

$$f_0 \Rightarrow_P f_1 \Rightarrow_P \ldots \Rightarrow_P f_n$$

with $f_i \in \mathcal{B}_V,\ 0 \leq i \leq n,\ n \in \mathbf{N}_0$. The number n is called the *length* of the derivation.

The following lemma shows that the derivations of our tree grammars can be "factorized" like the derivations of context-free string grammars [Ha78].

Lemma 2.1.1 *Let $G = (V, T, P, s)$ be a tree grammar and let $\alpha, \beta \in \mathcal{B}_V$.*
If $\alpha \stackrel{r}{\Rightarrow}_G \beta,\ r \in \mathbf{N}_0$, and $\alpha = \alpha_1 \circ \alpha_2$, then there are $r_1, r_2 \in \mathbf{N}_0,\ \beta_1, \beta_2 \in \mathcal{B}_V$ with

$$r = r_1 + r_2, \quad \beta = \beta_1 \circ \beta_2, \quad \alpha_1 \stackrel{r_1}{\Rightarrow}_G \beta_1,\ \alpha_2 \stackrel{r_2}{\Rightarrow}_G \beta_2$$

A similar result holds for $\alpha = \alpha_1 \times \alpha_2$.

Proof: Induction on r.

Basis: $\alpha = \alpha_1 \circ \alpha_2 \stackrel{0}{\Rightarrow}_G \beta$. Then $\alpha = \beta$ and for $\beta_i := \alpha_i,\ r_i := 0,\ i = 1, 2$, follows the claim.

Induction step:
Let $\alpha = \alpha_1 \circ \alpha_2 \stackrel{r+1}{\Rightarrow}_G \beta$. This derivation can be decomposed into $\alpha \Rightarrow_G \gamma \stackrel{r}{\Rightarrow}_G \beta$ for suitable $\gamma \in \mathcal{B}_V$, and there is a rule $p : x \to y \in P$ applied in the first step. The substituted nonterminal x is (wlg.) assumed to occur in α_1. Then we can write α_1 as

$$\alpha_1 = \alpha_1' \circ (\mathbf{1}_u \times x \times \mathbf{1}_v) \circ \alpha_1''$$

where $\alpha_1', \alpha_1'' \in \mathcal{B}_V,\ u, v \in \mathbf{N}_0$ and

$$\gamma = \underbrace{\alpha_1' \circ (\mathbf{1}_u \times y \times \mathbf{1}_v) \circ \alpha_1''}_{\gamma_1} \circ \underbrace{\alpha_2}_{\gamma_2}$$

It holds $\alpha_1 \stackrel{1}{\Rightarrow}_G \gamma_1$ and $\alpha_2 \stackrel{0}{\Rightarrow}_G \gamma_2$, and thus

$$\alpha = \alpha_1 \circ \alpha_2 \stackrel{1}{\Rightarrow}_G \gamma_1 \circ \gamma_2 \stackrel{r}{\Rightarrow}_G \beta$$

By induction hypothesis, there are $s_1, s_2 \in \mathbf{N}_0,\ \beta_1, \beta_2 \in \mathcal{B}_V$ with

$$\beta = \beta_1 \circ \beta_2, \quad r = s_1 + s_2, \quad \gamma_1 \stackrel{s_1}{\Rightarrow}_G \beta_1, \quad \gamma_2 \stackrel{s_2}{\Rightarrow}_G \beta_2$$

Thus

$$\alpha_1 \stackrel{1}{\Rightarrow}_G \gamma_1 \stackrel{s_1}{\Rightarrow}_G \beta_1,$$
$$\alpha_2 \stackrel{0}{\Rightarrow}_G \gamma_2 \stackrel{s_2}{\Rightarrow}_G \beta_2.$$

Let $r_1 := s_1 + 1,\ r_2 := s_2$, then $(r+1) = r_1 + r_2,\ \alpha_1 \overset{r_1}{\Rightarrow}_G \beta_1,\ \alpha_2 \overset{r_2}{\Rightarrow}_G \beta_2$. □

The proof for the product $\alpha = \alpha_1 \times \alpha_2$ is similar. This lemma shows that our tree grammars behave like context-free string grammars, but work with other structures.

To each tree grammar there is an equivalent one in a normal form.

Lemma 2.1.2 *For each tree grammar G there is an equivalent tree grammar G' with productions of the form 1. or 2.:*

1. $p : x \to y, \qquad y \in \mathcal{B}_N,$
2. $p : x \to y, \qquad y \in T.$

Proof: We assume wlg that there are no productions $p : x \to \mathbf{1}_1$. If $p : x \to y$ is a production which is not of the form 1. or 2., then for each terminal t in y we introduce a nonterminal x_t with $Q(x_t) := Q(t),\ Z(x_t) := Z(t)$ and a production $p_t : x_t \to t$. In y, all occurrences of t are replaced by nonterminal x_t.

Applying this construction to all productions not satisfying condition 1. or 2. leads to a grammar of the desired form which is obviously equivalent to the original one. □

This normal form can be modified such that each nonterminal tree on the right side of a production has "height" 2.

Lemma 2.1.3 *For each tree grammar G there is an equivalent tree grammar G' whose productions have the form 1. or 2.:*

1. $p : x \to y_0 \circ (y_1 \times \cdots \times y_k) \qquad y_0 \in N, y_i \in N \cup \{\mathbf{1}_1\},\ 1 \leq i \leq k,\ Q(y_0) = k,$
2. $p : x \to t, \qquad t \in T.$

Proof: Let G be a grammar already in the normal form of the previous lemma. Then we only have to consider productions

$$p : x \to y_0 \circ (z_1 \times \cdots \times z_k), \qquad y_0 \in N,\ z_i \in \mathcal{B}_N,\ 1 \leq i \leq k,\ Q(y_0) = k$$

where one of the z_i is neither a nonterminal nor a unit.

For each such z_i, a new nonterminal $x_{<p,i>}$ with $Q(x_{<p,i>}) := Q(z_i),\ Z(x_{<p,i>}) := Z(z_i) = 1$ is introduced. This occurrence of z_i in production p is substituted by the new nonterminal $x_{<p,i>}$. We add a new production $q_{<p,i>} : x_{<p,i>} \to z_i$. By substituting all such z_i, production p gets the desired form. The process is repeated for the new production system. It terminates because the right side of each new production $q_{<p,i>}$ has length less than the right side of p. It is obvious that the generated tree language remains unchanged, thus the resulting grammar is equivalent to the original one. □

Let us illustrate the definitions with two examples. These examples also give an impression of the role played by the degree of a grammar for generation power.

Example 1:
$V = \{s, f, g\}, \qquad T = \{f, g\}, \qquad Q(s) = Q(g) = 0,\ Q(f) = 2, \qquad Z(x) = 1 \quad \forall x \in V.$
$P = \{p : s \to g,\ q : s \to f \circ (\mathbf{1}_1 \times g) \circ s\}.$

$G = (V, T, P, s)$ generates the tree language $T(G) = \{[f \circ (\mathbf{1}_1 \times g)]^n \circ g \mid n \in \mathbf{N}_0\}.$

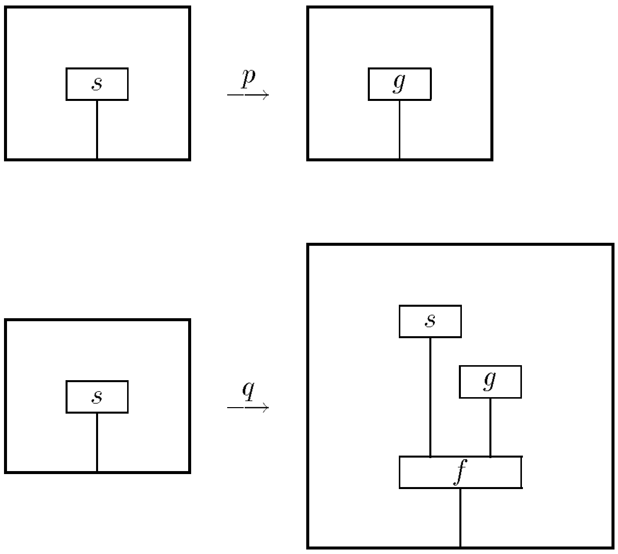

Figure 2.3: tree grammar productions, degree = 0

G has degree 0, $T(G) \in CF(\mathcal{B}_T, 0)$. Grammars of degree 0 correspond to the *regular tree grammars* known from the literature [GeSt84].

Example 2:
$V = \{s, x, f, g, h\}$, $\quad T = \{f, g, h\}$,
$Q(s) = Q(g) = 0,\ Q(x) = Q(h) = 2,\ Q(f) = 1,\quad Z(v) = 1 \quad \forall v \in V$.
$P = \{p_1 : s \to x \circ (g \times g),\ p_2 : x \to h,\ p_3 : x \to f \circ x \circ (f \times f)\}$.

$G = (V, T, P, s)$ generates the tree language

$$T(G) = \{f^n \circ h \circ (f^n \times f^n) \circ (g \times g) \mid n \in \mathbf{N}_0\}.$$

The degree of G is 2, $T(G) \in CF(\mathcal{B}_T, 2)$.

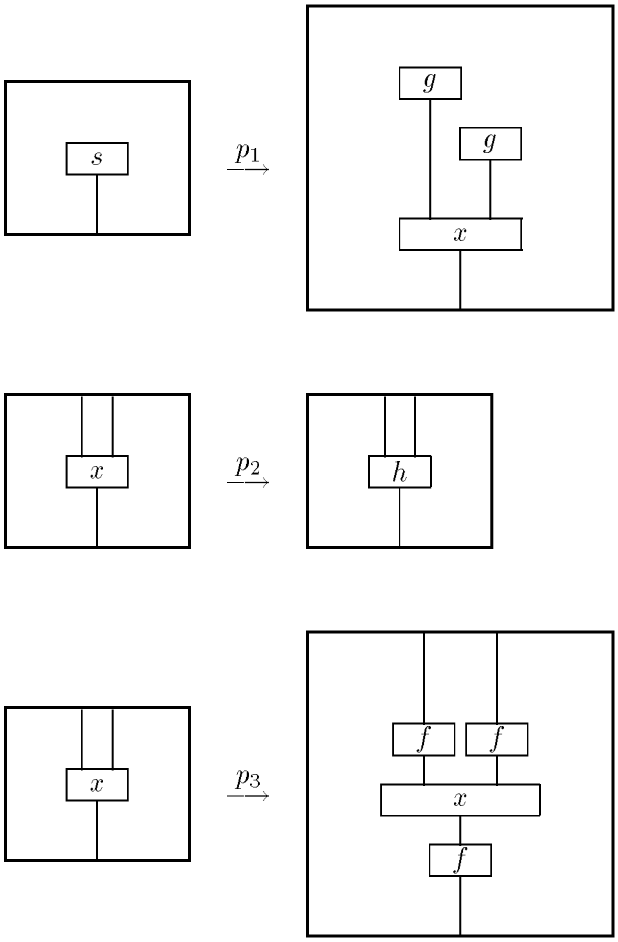

Figure 2.4: tree grammar productions, degree = 2

2.2 Multilinear Interpretation of Tree Languages

In this section, we define a "translation" of tree languages generated by tree grammars into string languages.

To every generator of a category of trees a *multilinear* mapping is assigned giving a *multilinear interpretation* for each net.

This interpretation maps each tree language into a string language. We always start from an interpretation of the terminals of a tree grammar and extend it "canonically" to the nonterminals of the grammar. Thus, for each derived tree (not only for terminal trees), an interpretation is defined.

The productions of the tree grammar are translated into coupled-context-free productions and the rewriting relation is translated into coupled-context-free rewriting.

Let us first give the formal definition of a *multilinear* mapping. We use the $\times$−category $MAP(., \mathbf{N}_0)$ which was defined in the first chapter.

Definition 23 *multilinear mapping*

A mapping $f : (\Delta^*)^m \to \Delta^*,\ m \in \mathbf{N}_0$, is called *multilinear* if there are words $w_1, \dots, w_{m+1} \in \Delta^*$ such that

$$f(x_1, \dots, x_m) = w_1 \cdot x_1 \cdot w_2 \cdots w_m \cdot x_m \cdot w_{m+1} \qquad \forall x_i \in \Delta^*,\ 1 \le i \le m.$$

A mapping $f : (\Delta^*)^m \to (\Delta^*)^n,\ m, n \in \mathbf{N}_0$, is called *multilinear* if f has a decomposition $f = f_1 \times \dots \times f_n$ into multilinear mappings $f_i : (\Delta^*)^{m_i} \to \Delta^*,\ m_i \in \mathbf{N}_0,\ 1 \le i \le n$. (For $n = 0$, f is multilinear iff $f = \mathbf{1}_0$.)

Lemma 2.2.1 *If F is a set of multilinear mappings $f \in MAP(\Delta^*, \mathbf{N}_0)$, the $\times$–subcategory $\mathcal{C}(F)$, generated by F in $MAP(\Delta^*, \mathbf{N}_0)$, contains only multilinear mappings.*

Proof:
It suffices to show that sequential and parallel composition of mappings preserve multilinearity.
For parallel composition, this follows immediately from the definition. Consider the sequential composition.

Let $f \in \mathcal{C}(F)(m, n),\ g \in \mathcal{C}(F)(l, m),\ l, m, n \in \mathbf{N}_0$, be multilinear. By definition, there are multilinear mappings $f_1, \dots, f_n$ and $g_1, \dots, g_m$ in $MAP(\Delta^*, \mathbf{N}_0)$ with $f = f_1 \times \dots \times f_n,\ g = g_1 \times \dots \times g_m,\ m_i := Q(f_i),\ Z(f_i) = 1,\ l_j := Q(g_j),\ Z(g_j) = 1$.

Remark: The mappings f_i and g_j need not be elements from $\mathcal{C}(F)$. With the notations above we have $m = \sum_{i=1}^{n} m_i$ and $l = \sum_{j=1}^{m} l_j$.

For the composition $f \circ g$ we get:

$$\begin{aligned}(f \circ g)(x_1, \dots, x_l) &= f(\ g(x_1, \dots, x_l)\ ) \\ &= f(\ y_1, \dots, y_m\ )\end{aligned}$$

with $(\ y_1, \dots, y_m\ ) := (g_1 \times \dots \times g_m)(\ x_1, \dots, x_l\ )$.

$$\begin{aligned}(\ y_1, \dots, y_m\ ) &= (\ g_1(x_1, \dots, x_{l_1}),\ \dots,\ g_m(x_{l-l_m+1}, \dots, x_l)\ ) \\ &= (\ u_{1,1}x_1u_{1,2} \dots u_{1,l_1}x_{l_1}u_{1,l_1+1},\ \dots,\ u_{m,1}x_{l-l_m+1}u_{m,2} \dots u_{m,l_m}x_lu_{m,l_m+1}\ )\end{aligned}$$

$$\begin{aligned}f(\ y_1, \dots, y_m\ ) &= (\ f_1(y_1, \dots, y_{m_1}),\ \dots,\ f_n(y_{m-m_n+1}, \dots, y_m)\ ) \\ &= (\ v_{1,1}y_1v_{1,2} \dots v_{1,m_1}y_{m_1}v_{1,m_1+1},\ \dots,\ v_{n,1}y_{m-m_n+1}v_{n,2} \dots v_{n,m}y_mv_{n,m+1}\ )\end{aligned}$$

The $u_{j,*},\ v_{i,*}$ in the equations above are elements from Δ^*.

If we let

$$\begin{aligned}h_1 &:= f_1 \circ (g_1 \times \dots \times g_{m_1}) \\ &\vdots \\ h_n &:= f_n \circ (g_{m-m_n+1} \times \dots \times g_m)\ ,\end{aligned}$$

every h_i is a multilinear mapping in $MAP(\Delta^*, \mathbf{N}_0)$, and from

$$\begin{aligned}(f \circ g) &= (f_1 \times \cdots \times f_n) \circ (g_1 \times \cdots \times g_m) \\ &= f_1 \circ (g_1 \times \cdots \times g_{m_1}) \times \cdots \times f_n \circ (g_{m-m_n+1} \times \cdots \times g_m) \\ &= h_1 \times \cdots \times h_n\end{aligned}$$

follows the claim. □

We will now define a multilinear interpretation for languages generated by tree grammars.

Let $G = (V, T, P, s)$ be a tree grammar, $N := V \setminus T$ the nonterminal alphabet of G, and let $\mathcal{B}_V := \mathcal{B}(V, Q, Z)$, $\mathcal{B}_T := \mathcal{B}(T, Q|_T, Z|_T)$ be the categories of trees generated by V resp. T. Let Δ_T be an alphabet and Δ_T^* the monoid of words over Δ_T.

To every terminal symbol $t \in T,\ Q(t) = k \in \mathbf{N}_0$, we assign a multilinear mapping

$$f_t : (\Delta_T^*)^k \to \Delta_T^* \quad \in MAP(\Delta_T^*, \mathbf{N}_0),$$

i.e.

$$f_t(w_1, \ldots, w_k) = u_1 \cdot w_1 \cdot u_2 \cdots u_k \cdot w_k \cdot u_{k+1}$$

where $u_1, \ldots, u_{k+1} \in \Delta_T^*$.

Let $F_T := \{f_t \mid t \in T\}$, and let $\mathcal{C}(F_T)$ be the $\times$–subcategory of $MAP(\Delta_T^*, \mathbf{N}_0)$ generated by F_T. As shown in lemma 2.2.1, the $\times$–category $\mathcal{C}(F_T)$ contains only multilinear mappings.

Let η_1 be the identity on $\mathbf{N}_0$ and $\eta_2' : T \to \mathcal{C}(F_T)$ defined by $\eta_2'(t) := f_t,\ t \in T$. The mappings η_1 and η_2' induce a uniquely determined $\times$–functor $\eta = (\eta_1, \eta_2)$ from the (free) $\times$–category $\mathcal{B}_T$ into the $\times$–category of mappings $\mathcal{C}(F_T)$. We call the $\times$–functor η a *multilinear interpretation* for the category of trees $\mathcal{B}_T$.

To each net from $\mathcal{B}_T$ a multilinear mapping is assigned by η. The tree language $T(G)$ is mapped by η into a string language over Δ_T.

Let us now extend the interpretation η canonically onto $\mathcal{B}_V$, thus giving an interpretation for every derived tree.

Let $\Delta_N := [N, Q|_N]$ be the bracket alphabet (see chapter 1) corresponding to N, i.e.

$$\Delta_N = \{x_i \mid x \in N,\ 1 \le i \le Q(x) + 1\}.$$

We set $\Delta := \Delta_N \cup \Delta_T$ and extend the mappings $f_t,\ t \in T$, onto Δ^*.

For each nonterminal symbol $x \in N,\ Q(x) = k$, we define a mapping

$$f_x : (\Delta^*)^k \to \Delta^*$$

by

$$f_x(w_1, \ldots, w_k) := x_1 \cdot w_1 \cdot x_2 \cdots x_k \cdot w_k \cdot x_{k+1} \qquad \forall w_1, \ldots, w_k \in \Delta^*.$$

Let $F_V := \{f_v \mid v \in V\}$ and $\mathcal{C}(F_V)$ be the $\times$–subcategory of $MAP(\Delta^*, \mathbf{N}_0)$ generated by F_V. We extend $\eta_2' : T \to \mathcal{C}(F_T)$ onto V by setting $\eta_2'(x) := f_x,\ x \in N$. η_1 and η_2' define uniquely an $\times$–functor $\eta = (\eta_1, \eta_2)$ from $\mathcal{B}_V$ to $\mathcal{C}(F_V)$.

The $\times$–functor η is called *multilinear interpretation for the tree grammar G*.

We extended a multilinear interpretation for the terminal category to the nonterminals by interpreting each nonterminal as a mapping which encloses his arguments in brackets (named by the label of the production). The bracket components mark positions in the string where substitutions are possible. In the interpretation of a terminal tree no more brackets occur.

Definition 24 *tree grammar with multilinear interpretation*
If $G = (V, T, P, s)$ is a tree grammar, $\eta_T : \mathcal{B}_T \to \mathcal{C}(F_T)$ is a multilinear interpretation for the terminal category of trees $\mathcal{B}_T$ and if $\eta : \mathcal{B}_V \to \mathcal{C}(F_V)$ is the canonical extension onto $\mathcal{B}_V$ then (G, η) is called *tree grammar with multilinear interpretation (BGMI).*

The *language generated* by the BGMI (G, η) is

$$L(G, \eta) = \{\eta(\Psi)(\Box) \mid \Psi \in T(G)\}.$$

The *degree* of a BGMI is defined as the degree of the underlying tree grammar. The class of string languages defined by BGMIs of degree k is denoted by BGMI(k).

In the next section, we will give some formal properties of the classes BGMI(k), and we will investigate the connection with coupled-context-free grammars.

2.3 Some formal properties of the classes BGMI(k)

Lemma 2.3.1 *$BGMI(k)$ is closed under union.*

Proof: Let $(G, \eta) = (V, T, P, s, \eta)$ and $(G', \eta') = (V', T', P', s', \eta')$ be BGMI of degree $k \in \mathbf{N}_0$. Let Δ_T and Δ'_T be the alphabets for the interpretation of the terminals from G and G'. Wlg we assume all alphabets (except $\Delta_T, \Delta_{T'}$) to be disjoint. We define the BGMI

$$G'' := (V \cup V' \cup \{\sigma\},\ T \cup T',\ P \cup P' \cup \{p_1 : \sigma \to s,\ p_2 : \sigma \to s'\},\ \sigma)$$

with source and target mappings

$$Q'' := Q \cup Q' \cup \{(\sigma, 0)\},\ Z'' := Z \cup Z' \cup \{(\sigma, 1)\}.$$

Define the interpretation $\eta_{T\cup T'} : \mathcal{B}_{T\cup T'} \to \mathcal{C}(F_{T\cup T'})$ by

$$\eta_{T\cup T'}(t) := \begin{cases} \eta_T(t), & t \in T \\ \eta'_T(t), & t \in T'. \end{cases}$$

η'' is defined by canonical extension of $\eta_{T\cup T'}$ onto $\mathcal{B}(V \cup V' \cup \{\sigma\})$. Then

1. $T(G'') = T(G) \cup T(G')$.
2. $grad(G'') = \max\{grad(G), grad(G')\} = k$.
3. $\eta''(f) = \begin{cases} \eta(f), & f \in \mathcal{B}_V \\ \eta'(f), & f \in \mathcal{B}_{V'} \end{cases}$
4. $L(G'', \eta'') = L(G, \eta) \cup L(G', \eta')$. □

Lemma 2.3.2 *$BGMI(k)$ is closed under concatenation.*

Proof: Let (G, η) and (G', η') be BGMI given as above. Let

$$G'' := (V \cup V' \cup \{\sigma\},\ T \cup T' \cup \{[conc_2]\},\ P \cup P' \cup \{p : \sigma \to [conc_2] \circ (s \times s')\},\ \sigma)$$

Q'', Z'' are defined as above and for $[conc_2]$ we let $Q''([conc_2]) := 2,\ Z([conc_2]) := 1$. Let $\eta''([conc_2])$ be the concatenation on $(V \cup V' \cup \{\sigma\})^*$ which is multilinear. Then

1. $grad(G'') = k$.
2. $T(G'') = \{[conc_2] \circ (f \times f') \mid f \in T(G),\ f' \in T(G')\}$.
3. $L(G'', \eta'') = L(G, \eta) \cdot L(G', \eta')$. □

Lemma 2.3.3 *$BGMI(k)$ is closed under Kleene-star.*

Proof: Let (G, η) be as above and

$$G'' := (V \cup \{\sigma\},\ T \cup \{[conc_2], [eps]\},\ P \cup \{p_1 : \sigma \to [eps],\ p_2 : \sigma \to [conc_2] \circ (s \times \sigma),\ \sigma)$$

Let source and target of σ, $[conc_2]$ be as above, and let $Q''([eps]) := 0,\ Z''([eps]) := 1,\ \eta''([eps])(\Box) := \varepsilon$.

Then

1. $T(G'') = \{([conc_2] \circ (s \times \mathbf{1}_1))^n \circ [eps] \mid n \in \mathbf{N}_0\}$.
2. $L(G'', \eta'') = \{w_1 \cdots w_n \cdot \varepsilon \mid w_i \in L(G, \eta),\ n \in \mathbf{N}_0\} = L(G, \eta)^*$. □

Lemma 2.3.4 *$BGMI(k)$ is closed under homomorphism.*

Proof: Let (G, η) be as above and $h : \Delta_T^* \to \Sigma_T^*$ a homomorphism. Let $\Sigma = \Delta_N \cup \Sigma_T$, and extend $h : \Delta^* \to \Sigma^*$ by letting h the identity on Δ_N.

If t is a terminal of the grammar with interpretation $\eta(t) : (\Delta^*)^k \to \Delta^*,\ k = Q(t) \in \mathbf{N}_0$,

$$\eta(t)\,(w_1, \ldots, w_k) = u_1 \cdot w_1 \cdot u_2 \cdots u_k \cdot w_k \cdot u_{k+1}, \quad \forall w_1, \ldots, w_k \in \Delta^*$$

where $u_j \in \Delta_T^*,\ 1 \le j \le k+1$,
then we define a new interpretation $\theta(t) : (\Sigma^*)^k \to \Sigma^*$ by

$$\theta(t)\,(w_1, \ldots, w_k) := h(u_1) \cdot w_1 \cdot h(u_2) \cdots h(u_k) \cdot w_k \cdot h(u_{k+1}) \quad \forall w_1, \ldots, w_k \in \Sigma^*.$$

The interpretation of the nonterminals remains unchanged. Since for each terminal t with k inputs

$$\begin{aligned} h(\ \eta(t)(w_1, \ldots, w_k)\ ) &= h(\ u_1 \cdot w_1 \cdot u_2 \cdots u_k \cdot w_k \cdot u_{k+1}\ ) \\ &= h(u_1) \cdot h(w_1) \cdot h(u_2) \cdots h(u_k) \cdot h(w_k) \cdot h(u_{k+1}) \\ &= \theta(t)(\ h(w_1), \ldots, h(w_k)\ ) \end{aligned}$$

it follows easily by induction

$$L(G, \theta) = h(\ L(G, \eta)\ ).$$

□

Lemma 2.3.5 $COPY_k \in BGMI(k) \qquad \forall k \in \mathbf{N}_0$.

Proof:
Let $G_k := (V_k, T_k, P_k, s)$ with

$$V_k := \{s, x, [\varepsilon, a], [\varepsilon, b], [a, \varepsilon], [b, \varepsilon], [\varepsilon], [conc_k]\}, \quad T_k := V_k \setminus \{s, x\}$$

Let $Q(s) = Q([\varepsilon]) = 0,\ Q(x) = k,\ Q(v) = 1$ for the other nonterminals, and let the target of all symbols be 1. Define production set P_k by

$$\begin{aligned} P_k := \{ \quad & p_1\!: s \to x \circ ([\varepsilon] \times \cdots \times [\varepsilon]), \\ & p_2\!: x \to [a, \varepsilon] \circ x \circ ([\varepsilon, a] \times \cdots \times [\varepsilon, a]), \\ & p_3\!: x \to [b, \varepsilon] \circ x \circ ([\varepsilon, b] \times \cdots \times [\varepsilon, b]), \\ & p_4\!: x \to [conc_k] \quad \} \end{aligned}$$

The interpretation of the terminals is given by

$$\begin{aligned}
\eta([\varepsilon, a])(w) &:= \varepsilon \cdot w \cdot a,\\
\eta([\varepsilon, b])(w) &:= \varepsilon \cdot w \cdot b,\\
\eta([a, \varepsilon])(w) &:= a \cdot w \cdot \varepsilon,\\
\eta([b, \varepsilon])(w) &:= b \cdot w \cdot \varepsilon,\\
\eta([\varepsilon])(\square) &:= \varepsilon,\\
\eta([conc_k])(w_1, \dots, w_k) &:= w_1 \cdots w_k.
\end{aligned}$$

It is not difficult to see that the generated tree language is

$$\begin{aligned}
T(G_k) = \{ \quad & [a_1, \varepsilon] \circ \cdots \circ [a_n, \varepsilon] \circ [conc_k] \circ \\
& ([\varepsilon, a_n] \times \cdots \times [\varepsilon, a_n]) \circ \cdots \circ ([\varepsilon, a_1] \times \cdots \times [\varepsilon, a_1]) \circ ([\varepsilon] \times \cdots \times [\varepsilon]) \mid \\
& n \in \mathbf{N}_0,\ a_i \in \{a, b\},\ 1 \le i \le n \quad \}
\end{aligned}$$

Thus

$$L(G_k, \eta) = \left\{ (a_1 \cdots a_n) \cdot (a_1 \cdots a_n)^k \mid a_i \in \{a, b\},\ n \in \mathbf{N}_0 \right\} = COPY_k.$$

$\square$

Lemma 2.3.6 $COUNT_{2k+2} \in BGMI(k) \qquad \forall k \in \mathbf{N}_0.$

Proof:
Let $G_k := (V_k, T_k, P_k, s)$ with
$V_k := \{s, x, [a_1, a_{2k+2}], [a_2, a_3], \dots, [a_{2k}, a_{2k+1}], [\varepsilon], [conc_k]\}, \qquad T_k := V_k \setminus \{s, x\}.$

Let $Q(s) = 0,\ Q(x) = k,\ Q([\varepsilon]) = 0,\ Q([conc_k] = k,\ Q(v) = 1$ for all other symbols and $Z(v) = 1 \quad \forall v \in V_k$. The production set is

$$\begin{aligned}
P_k := \{ \quad & p_1\colon s \to x \circ ([\varepsilon] \times \cdots \times [\varepsilon]),\\
& p_2\colon x \to [a_1, a_{2k+2}] \circ x \circ ([a_2, a_3] \times \cdots \times [a_{2k}, a_{2k+1}]),\\
& p_3\colon x \to [conc_k] \quad \}.
\end{aligned}$$

The interpretation of the terminal symbols is

$$\begin{aligned}
\eta([a_i, a_j])(w) &:= a_i \cdot w \cdot a_j,\\
\eta([\varepsilon])(\square) &:= \varepsilon,\\
\eta([conc_k])(w_1, \dots, w_k) &:= w_1 \cdots w_k.
\end{aligned}$$

Then

$$T(G_k) = \{[a_1, a_{2k+2}]^n \circ [conc] \circ ([a_2, a_3] \times \cdots \times [a_{2k}, a_{2k+1}])^n \mid n \in \mathbf{N}_0\}$$

and

$$L(G_k, \eta) = \{a_1^n \cdots a_{2k+2}^n \mid n \in \mathbf{N}_0\} = COUNT_{2k+2}.$$

$\square$

2.4 Tree Grammars with Multilinear Interpretation and Coupled-context-free Grammars

In this section, we investigate the connection between tree grammars with multilinear interpretation and coupled-context-free grammars. At first we will show that for every BGMI there is an equivalent CCFG.

Let (G, η) be a BGMI, i.e. $G = (V, T, P, s)$ is a tree grammar over the category of trees $\mathcal{B}_T$, and η is a multilinear interpretation. Let $N := V \setminus T$ be the nonterminal alphabet of G, and let Q, Z be the source resp. target mapping of $\mathcal{B}_V$.

Each terminal from G is interpreted under η as a multilinear function over an alphabet Δ_T. Each nonterminal has the "canonical" interpretation as a function which encloses its arguments in the corresponding bracket from $\Delta_N := [N, Q|_N]$. To nonterminal $x \in N$ corresponds a bracket of degree $Q(x)$ with components $x_1, \dots, x_{Q(x)+1}$. Let the total alphabet be denoted by $\Delta := \Delta_N \cup \Delta_T$.

We assign to a BGMI (G, η) the following CCFG G_{ccf}:

$$G_{ccf} := (\Delta, \Delta_T, P_{ccf}, \sigma)$$

with production system

$$\begin{aligned} P_{ccf} := \{ \quad & p : (x_1, \dots, x_{k+1}) \to (y_1, \dots, y_{k+1}) \mid \\ & p : x \to y \in P, \quad Q(x) = Q(y) = k, \\ & \eta(y)(w_1, \dots, w_k) = y_1 \cdot w_1 \cdot y_2 \cdots y_k \cdot w_k \cdot y_{k+1} \quad \} \end{aligned}$$

and axiom $\sigma := \eta(s)(\Box) \in \Delta_N$. ($\sigma$ is identified with s.)

To every tree grammar production for a nonterminal with k inputs, a coupled-context-free production with $k+1$ components is assigned. The right sides of the production components are determined by the interpretation of the right side of the tree grammar production. We have to show first that G_{ccf}, as defined above, is in fact a CCFG.

The alphabets Δ_N and Δ_T obviously fulfill the definition of a CCFG and the axiom σ is a bracket of zero degree. Consider the production system P_{ccf}. The claim follows immediately from the following

Lemma 2.4.1 *If $f \in \mathcal{B}_V(k, 1)$, $k \in \mathbf{N}_0$, and $w_1, \dots, w_k \in D(N, \Delta_T)$, then*

$$\eta(f)(w_1, \dots, w_k) \in D(N, \Delta_T).$$

Proof: Induction on the size $|f|$.
Basis: $|f| = 0$, i.e. $f = \mathbf{1}_1$. Then $\eta(f)(w) = w \in D(N, \Delta_T)$ by assumption.
Induction step: Let f be an element from $\mathcal{B}_V(k, 1)$ of positive length. Then

$$f = v \circ (g_1 \times \cdots \times g_l)$$

with $v \in V$, $l = Q(v)$, $g_i \in \mathcal{B}_V(k_i, 1)$, $1 \leq i \leq l$, $\sum_{i=1}^{l} k_i = k$.

By induction, for $g_1, \dots, g_l$ holds

$$\begin{aligned} w_1, \dots, w_{k_1} \in D(N, \Delta_T) \quad & \Longrightarrow \quad \eta(g_1)(w_1, \dots, w_{k_1}) \in D(N, \Delta_T), \\ & \vdots \\ w_{k-k_l+1}, \dots, w_k \in D(N, \Delta_T) \quad & \Longrightarrow \quad \eta(g_l)(w_{k-k_l+1}, \dots, w_k) \in D(N, \Delta_T). \end{aligned}$$

For v we have to consider the two cases

1. $v \in T: \quad \eta(v)(u_1, \ldots, u_l) = v_1 \cdot u_1 \cdot v_2 \cdots v_l \cdot u_l \cdot v_{l+1}, \quad v_i \in \Delta_T^*$.
   Then $u_1, \ldots, u_l \in D(N, \Delta_T) \Longrightarrow \eta(v)(u_1, \ldots, u_l) \in D(N, \Delta_T)$ by definition.

2. $v \in N: \quad \eta(v)(u_1, \ldots, u_l) = v_1 \cdot u_1 \cdot v_2 \cdots v_l \cdot u_l \cdot v_{l+1}, \quad v \in N_k$.
   Then $u_1, \ldots, u_l \in D(N, \Delta_T) \Longrightarrow \eta(v)(u_1, \ldots, u_l) \in D(N, \Delta_T)$ by definition.

Thus we have: $w_1, \ldots, w_k \in D(N, \Delta_T) \quad \Longrightarrow$

$$\eta(f)(w_1, \ldots, w_k) = \eta(v)\left(\eta(g_1)(w_1, \ldots, w_{k_1}), \ldots, \eta(g_l)(w_{k-k_l+1}, \ldots, w_k)\right) \in D(N, \Delta_T)$$

and the lemma is shown. □

From this lemma follows that for every production $p : x \to y \in P$

$$\eta(y)(\varepsilon, \ldots, \varepsilon) = y_1 \cdots y_{k+1} \in D(N, \Delta_T),$$

so every $p : (x_1, \ldots, x_{k+1}) \to (y_1, \ldots, y_{k+1}) \in P_{ccf}$, as defined above, is in fact a CCFG-production.

Having shown that G_{ccf} is a well-defined CCFG, we now show that the language of the BGMI (G, η) is contained in $L(G_{ccf})$.

Lemma 2.4.2 *Let (G, η) and G_{ccf} as above and let $\mathcal{B}_V$ be the category of trees generated by V. Then for all $f \in \mathcal{B}_V$ holds:*

$$s \overset{*}{\Rightarrow}_G f \quad \Longrightarrow \quad \sigma \overset{*}{\Rightarrow}_{ccf} \eta(f)(\Box)$$

Proof: Induction on the length of the derivation.
Basis: $s \overset{0}{\Rightarrow}_G f$. Then $f = s$ and $\eta(f)(\Box) = \sigma$, from which follows the claim.
Induction step: Let $s \overset{*}{\Rightarrow}_G f$ be a derivation of positive length. Then there are $g, h \in \mathcal{B}_V$ and $p : x \to y \in P$ such that

$$s \overset{*}{\Rightarrow}_G g \circ x \circ h \Rightarrow_G g \circ y \circ h = f$$

where $Q(g) = Z(g) = 1,\ Q(h) = 0,\ Z(h) = Q(x) = Q(y) = k \in \mathbf{N}_0$.

By lemma 2.2.1, the interpretations of g and h

$$\begin{array}{rl} \eta(g): & \Delta^* \to \Delta^* \\ \eta(h): & \{\Box\} \to (\Delta^*)^k \end{array}$$

are multilinear mappings, i.e. there are $u_1, u_2,\ w_1, \ldots, w_k \in \Delta^*$ such that

$$\begin{array}{rcll} \eta(g)(w) & = & u_1 \cdot w \cdot u_2 & \forall w \in \Delta^*, \\ \eta(h)(\Box) & = & (w_1, \ldots, w_k) & \end{array}$$

and h has a decomposition $h = (h_1 \times \cdots \times h_k)$ with $h_i \in \mathcal{B}_V(0,1),\ \eta(h_i)(\Box) = w_i,\ 1 \le i \le k$.

Thus

$$\eta(g \circ x \circ h)(\Box) = u_1 \cdot x_1 \cdot w_1 \cdot x_2 \cdots x_k \cdot w_k \cdot x_{k+1} \cdot u_2$$

and

$$\eta(g \circ y \circ h)(\Box) = u_1 \cdot y_1 \cdot w_1 \cdot y_2 \cdots y_k \cdot w_k \cdot y_{k+1} \cdot u_2$$

By lemma 2.4.1, $w_1, \ldots, w_k \in D(N, \Delta_T)$ and by definition of G_{ccf}, production system P_{ccf} contains the rule $p : (x_1, \ldots, x_{k+1}) \to (y_1, \ldots, y_{k+1})$ where $y_1, \ldots, y_{k+1}$ are given by $\eta(y)$.

From the definition of $\Rightarrow_{ccf}$ and by induction follows

$$\sigma \overset{*}{\Rightarrow}_{ccf} \eta(g \circ x \circ h)(\Box) \Rightarrow_{ccf} \eta(g \circ y \circ h)(\Box) = \eta(f)(\Box)$$

which proves the lemma. $\Box$

From the previous lemma follows $L(G, \eta) \subseteq L(G_{ccf})$. We now show the opposite inclusion.

Lemma 2.4.3 *Let (G, η) and G_{ccf} as above. Then for all $w \in \Delta^*$ holds:*

$$\sigma \overset{*}{\Rightarrow}_{ccf} w \quad \Longrightarrow \quad \exists f \in \mathcal{B}_V(0,1),\ s \overset{*}{\Rightarrow}_G f,\ w = \eta(f)(\Box)$$

Proof: Induction on the length of the derivation $\sigma \overset{n}{\Rightarrow}_{ccf} w,\ n \in \mathbf{N}_0$.
Basis: $\sigma \overset{0}{\Rightarrow}_{ccf} w$. Then $w = \sigma$ and for $f := s$ we get the claim.
Induction step: $\sigma \overset{n+1}{\Rightarrow}_{ccf} w,\ n \in \mathbf{N}_0$. Then there is a decomposition

$$\sigma \overset{n}{\Rightarrow}_{ccf} \underbrace{u_1 \cdot x_1 \cdot d_1 \cdot x_2 \cdots x_k \cdot d_k \cdot x_{k+1} \cdot u_2}_{w'} \Rightarrow_{ccf} \underbrace{u_1 \cdot y_1 \cdot d_1 \cdot y_2 \cdots y_k \cdot d_k \cdot y_{k+1} \cdot u_2}_{w}$$

with $u_1, u_2 \in \Delta^*,\ d_1, \ldots, d_k \in D(N, \Delta_T)$, and

$$p : (x_1, \ldots, x_{k+1}) \to (y_1, \ldots, y_{k+1})$$

is a production in P_{ccf}.

By induction assumption, there is a tree $f' \in \mathcal{B}_V(0,1)$ with $s \overset{*}{\Rightarrow}_G f'$ and $\eta(f')(\Box) = w'$.

We have to show that there exists a tree $f \in \mathcal{B}_V(0,1)$ such that

1. $f' \Rightarrow_G f$
2. $\eta(f)(\Box) = w$

Let n_x be the number of occurrences of x in the tree f'. Each occurrence gives a unique decomposition of f'. For $j = 1, \ldots, n_x$, let this decomposition be denoted by

$$f' = g^{(j)} \circ x \circ h^{(j)}$$

where $g^{(j)} \in \mathcal{B}_V(1,1),\ h^{(j)} \in \mathcal{B}_V(0,k),\ k = Q(x)$.

The interpretation w' of f' has corresponding decompositions

$$w' = u_1^{(j)} \cdot x_1 \cdot d_1^{(j)} \cdot x_2 \cdots x_k \cdot d_k^{(j)} \cdot x_{k+1} \cdot u_2^{(j)}$$

defined by

$$\begin{aligned} \eta(g^{(j)})(v) &= u_1^{(j)} \cdot v \cdot u_2^{(j)} \qquad \forall v \in \Delta^* \\ \eta(h^{(j)})(\Box) &= (d_1^{(j)}, \ldots, d_k^{(j)}) \end{aligned}$$

Clearly, $d_1^{(j)}, \ldots, d_k^{(j)} \in D(N, \Delta_T)$, i.e. $x_1, \ldots, x_{k+1}$ form corresponding bracket components.

Since there are exactly n_x occurrences of nonterminal x in the tree f', there are exactly n_x occurrences of the "opening" bracket x_1 in w'.

From

$$w' = u_1 \cdot x_1 \cdot d_1 \cdot x_2 \cdots x_k \cdot d_k \cdot x_{k+1} \cdot u_2$$

follows that there must be an index $j \in \{1, \ldots, n_x\}$ such that $u_1^{(j)} = u_1$. Since the bracket components $x_2, \ldots, x_{k+1}$ corresponding to component x_1 are uniquely determined, it follows $d_1 = d_1^{(j)}, \ldots, d_k = d_k^{(j)}$ and thus $u_2 = u_2^{(j)}$. The decomposition

$$f' = g^{(j)} \circ x \circ h^{(j)}$$

now fulfills

$$\begin{aligned} \eta(g^{(j)})(v) &= u_1 \cdot v \cdot u_2 \qquad \forall v \in \Delta^* \\ \eta(h^{(j)})(\Box) &= (d_1, \ldots, d_k) \end{aligned}$$

By construction of G_{ccf}, P must contain production $p : x \to y$. For the tree $f \in \mathcal{B}_V(0,1)$, given by the derivation

$$s \overset{*}{\Rightarrow}_G \underbrace{g^{(j)} \circ x \circ h^{(j)}}_{f'} \Rightarrow_G \underbrace{g^{(j)} \circ y \circ h^{(j)}}_{=:f}$$

in the tree grammar G, it holds

$$\begin{aligned} \eta(f)(\Box) &= \eta(g^{(j)}) \left(\eta(y)(\eta(h^{(j)})) \right) \\ &= u_1 \cdot y_1 \cdot d_1 \cdot y_2 \cdots y_k \cdot d_k \cdot y_{k+1} \cdot u_2 \\ &= w \end{aligned}$$

which shows the lemma. □

From lemma 2.4.2, lemma 2.4.3 and the definition of the generated string language, it follows

Theorem 4 *For each BGMI (G, η) there is an equivalent CCFG G_{ccf}.*

We will now show the opposite direction by defining for each CCFG an equivalent BGMI.

The first lemma shows that each word from the bracket language $D(N,T)$ defined by the alphabet of a CCFG can be obtained by multilinear interpretation of a tree from a suitably defined category of trees.

We construct to a given CCFG an equivalent BGMI. Let G_{ccf} be a CCFG, i.e.

$$G_{ccf} = ([V, g], T, P_{ccf}, s)$$

where $g : V \to \mathbf{N}_0$ is the degree mapping for the brackets from V. Let $N := V \setminus T$ denote the nonterminal alphabet of G_{ccf}.

We define the following category of trees:

$$\mathcal{B}_{\tilde{V}} := \mathcal{B}(N \cup \tilde{T}, Q, Z)$$

with $\tilde{V} := N \cup \tilde{T}$, $\tilde{T} := \{\tilde{t} \mid t \in T\} \cup \{[eps], [conc]\}$, and source and target given by

$$\begin{aligned} Q(x) &= g(x) && x \in N, \\ Q(\tilde{t}) &= 0 && t \in T, \\ Q([eps]) &= 0 && \\ Q([conc]) &= 2 && \\ Z(v) &= 1 && v \in N \cup \tilde{T}. \end{aligned}$$

For each terminal in $\tilde{T}$, an interpretation is defined by

$$\begin{aligned} \eta(\tilde{t})(\square) &= t \qquad \forall t \in T, \\ \eta([eps])(\square) &= \varepsilon \\ \eta([conc])(w_1, w_2) &= w_1 \cdot w_2 \qquad \forall w_1, w_2 \in [V,g]^*. \end{aligned}$$

Every string w from the bracket language $D(N,T)$ has a unique decomposition into terminals $t \in T$ and strings from $E(N,T) \setminus T$, i.e. strings of the form $x_1 \cdot d_1 \cdot x_2 \cdots x_k \cdot d_k \cdot x_{k+1}$ with $x \in N_k$, $d_1, \ldots, d_k \in D(N,T)$.

For a given word $w \in D(N,T)$, we define recursively a tree $\Psi_w \in \mathcal{B}_{\tilde{V}}(0,1)$ with interpretation $\eta(\Psi_w) = w$:

1. $\Psi_\varepsilon := [eps]$,
2. $\Psi_t := \tilde{t} \qquad t \in T$,
3. $\Psi_w := x \circ (\Psi_{d_1} \times \cdots \times \Psi_{d_k})$
   $w = x_1 \cdot d_1 \cdot x_2 \cdots x_k \cdot d_k \cdot x_{k+1},\ x \in N_k,\ d_1, \ldots, d_k \in D(N,T)$,
4. $\Psi_{u \cdot v} := [conc] \circ (\Psi_u \times \Psi_v) \qquad u \in E(N,T), v \in D(N,T) \setminus \{\varepsilon\}$.

It is easily verified that this definition gives a unique tree $\Psi_w \in \mathcal{B}_{\tilde{V}}(0,1)$ for each string $w \in D(N,T)$ such that $\eta(\Psi_w)(\square) = w$.

The next lemma shows that given an arbitrary decomposition of a string $y \in D(N,T)$, there is a tree in $\mathcal{B}_{\tilde{V}}$ whose interpretation is defined by that decomposition.

Lemma 2.4.4 *Let $[V,g] = [N \cup T, g]$, $D(N,T)$, $\tilde{V}$, $\tilde{T}$ and η be given as above.*
If $y \in D(N,T)$ is an arbitrary string from the bracket language and $(y_1, \ldots, y_{k+1})$ is a decomposition of y, i.e.

$$y = y_1 \cdots y_{k+1} \qquad k \in \mathbf{N}_0,\ y_i \in [V,g]^*,$$

then there is a tree

$$\Psi_{(y_1, \ldots, y_{k+1})} \in \mathcal{B}_{\tilde{V}}(k,1)$$

with interpretation

$$\eta(\Psi_{(y_1, \ldots, y_{k+1})})(v_1, \ldots, v_k) = y_1 \cdot v_1 \cdot y_2 \cdots y_k \cdot v_k \cdot y_{k+1} \qquad \forall v_1, \ldots, v_k \in [V,g]^*.$$

Proof: We introduce a new symbol $[input]$ of degree $g([input]) = 0$. The interpretation of this symbol is canonical, i.e. $\eta([input])(\square) = [input]_1$ (which is identified with the symbol itself). If we let

$$y[input] := y_1 \cdot [input] \cdot y_2 \cdots y_k \cdot [input] \cdot y_{k+1} \in D(N \cup \{[input]\}, T),$$

we can apply our construction and obtain a tree

$$\Psi_{y[input]} \in \mathcal{B}(\tilde{V} \cup \{[input]\}, Q, Z)(0,1)$$

with interpretation $y[input]$.

By construction, there are exactly k occurrences of $[input]$ in the tree $\Psi_{y[input]}$, thus there is a decomposition

$$\Psi_{y[input]} = \Psi \circ \underbrace{([input] \times \cdots \times [input])}_{k\ times}$$

where $\Psi \in \mathcal{B}(\tilde{V}, Q, Z)(k, 1)$ is a tree with k inputs which does not contain the symbol $[input]$.

It is easy to see that

$$\eta(\Psi)(v_1, \ldots, v_k) = y_1 \cdot v_1 \cdot y_2 \cdots y_k \cdot v_k \cdot y_{k+1} \qquad \forall v_1, \ldots, v_k \in [V, g]^*$$

We denote this tree with $\Psi_{(y_1, \ldots, y_{k+1})}$. □

Lemma 2.4.5 *Let*

$$G_{ccf} = ([N \cup T, g], T, P_{ccf}, s)$$

be a CCFG and let the category of trees

$$\mathcal{B}_{\tilde{V}} := \mathcal{B}(N \cup \tilde{T}, Q, Z)$$

be constructed as above. Let further η be the canonical extension onto $\tilde{V}$ of the interpretation given for the terminals from $\tilde{T}$.

If (G, η) is defined by

$$G := (N \cup \tilde{T}, \tilde{T}, P, s)$$

with production system

$$P := \{p : x \to \Psi_{(y_1, \ldots, y_{k+1})} \mid p : (x_1, \ldots, x_{k+1}) \to (y_1, \ldots, y_{k+1}) \in P_{ccf}\}$$

where $\Psi_{(y_1, \ldots, y_{k+1})}$ is given by lemma 2.4.4, then it holds

1. *(G, η) is a BGMI,*
2. *$grad(G) = grad(G_{ccf})$,*
3. *$L(G, \eta) = L(G_{ccf})$.*

Proof: By construction, (G, η) fulfills the definition of a BGMI and the degree of G equals the degree of G_{ccf}. It remains to show that for each $w \in [N \cup T, g]^*$

$$s \stackrel{*}{\Rightarrow}_{ccf} w \iff \exists \Psi \in \mathcal{B}(N \cup \tilde{T}, Q, Z),\ s \stackrel{*}{\Rightarrow}_G \Psi,\ \eta(\Psi)(\Box) = w$$

The proof is analogous to lemmata 2.4.2 and 2.4.3 and is omitted. □

Thus, we get

Theorem 5 *For each coupled-context-free grammar there is an equivalent tree grammar with multilinear interpretation of the same degree.*

For the special case of zero degree, we get

Corollary 1 *The class BGMI(0) is the class of context-free string languages.*

This follows from the fact that a coupled-context-free grammar of degree 0 is a simple context-free grammar with the semi-Thue–relation as rewriting relation. □

Chapter 3

Tree Adjoining Grammars and BGMI

3.1 Basic definitions

A *Tree Adjoining Grammar (TAG)* [Jo85] is a rewriting system on the derivation trees of a context-free grammar. It is mainly intended as a formalism for the definition of tree languages rather than a string generating system. TAGs have recently found great interest in Computer Linguistics because they seem to be an adequate formalism for describing many natural language phenomena. In this area, one is interested in language classes and rewriting systems slightly more powerful than context-free grammars [Jo86].

We will not consider the linguistic relevance of TAGs but the generated language class, the *Tree Adjoining Languages (TAL)*. This language class properly contains the class of context-free languages and is a proper subclass of the *Indexed Languages*. We will show that the classes BGMI(1) and TAL coincide.

We will first describe the TAG-rewriting formalism and then show the equivalence. The following definition of TAG-rewriting is taken from [Jo85].

Definition 25 *Tree Adjoining Grammar (TAG)*
A *Tree Adjoining Grammar* $G = (I, A)$ consists of

- a finite set I of *initial trees*,
- a finite set A of *auxiliary trees*.

These *elementary* trees are derivation trees of a given CFG $g = (N \cup T, T, P, s)$.

The initial trees are derivation trees with source s and target $w \in T^*$, the auxiliary trees β are derivation trees with source x and target $u \cdot x \cdot v,\ x \in N,\ u, v \in T^*$. The unique leaf marked with x in an auxiliary tree β is called *foot-node* of β.

The set of all auxiliary trees with source x will be denoted by A_x, thus $A = \bigcup_{x \in N} A_x$. The general form of an initial and an auxiliary tree is shown in the picture.

Starting with initial trees, new trees can be produced by *adjunction* of auxiliary trees at suitable tree nodes. Adjunction of an auxiliary tree β at a node k in a tree γ is defined iff k is marked with $x \in N$ and $\beta \in A_x$. Adjunction consists of the following steps:

- Split tree γ at node k into upper tree γ_1 and subtree γ_2.

Figure 3.1: Initial and auxiliary tree of a TAG

- Paste γ_1 and γ_2 together after inserting β giving the new tree γ'.

We omit introducing the usual terminology for these definitions because we will give a treatment in the theory of $\times$–categories. Formal definitions can be found in [Vi88].

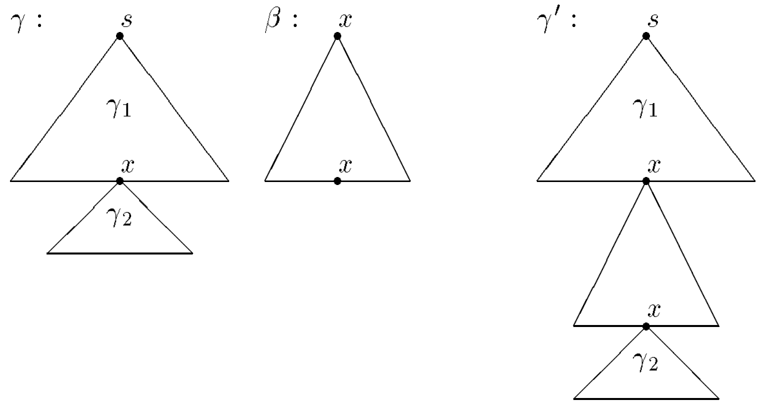

Figure 3.2: Adjoining operation

If γ' results from γ by adjoining, we write $\gamma \Rightarrow_{adj} \gamma'$. The *tree language generated* by TAG G is

$$T(G) := \{\Psi \mid \exists\, \sigma \in I,\ \sigma \stackrel{*}{\Rightarrow}_{adj} \Psi\}.$$

Clearly, $T(G)$ is a set of derivation trees of the CFG g with source s and terminal target.
The *string language* generated by TAG G is

$$L(G) := \{yield(\Psi) \mid \Psi \in T(G)\}.$$

The function *yield* assigns to each derivation tree the string obtained by "reading the leaf markings from left to right". Obviously, $L(G)$ is a subset of the context-free language $L(g)$.

A TAG is a rewriting system on the derivation trees of a context-free grammar. From the theory of *tree automata* it is known that the set of derivation trees of a CFG is a *recognizable tree language* [Th67]. The tree language of a TAG need not be recognizable. Further, there are TALs which are not context-free, but all TALs are *indexed languages*. Often, the following generalization of TAGs is considered.

Definition 26 *TAG with constraints*

A *TAG with constraints* (TAGC) $G = (I, A, C)$ is a TAG with the following extension:

To each inner node k (marked with a nonterminal $x \in N$) in an elementary tree, one of the following *constraints* $C(k)$ is assigned:

1. *Obligatory–Adjoining Constraint*: $C(k) = (OA, \Gamma),\ \Gamma \subset A_x$
   Meaning: At node k a tree *must* be adjoined, otherwise the tree in which k occurs is not an element of the generated tree language. Only trees from Γ may be adjoined to node k.

2. *Selective–Adjoining Constraint*: $C(k) = (SA, \Gamma),\ \Gamma \subset A_x$
   Meaning: At node k any tree from Γ may be adjoined.

3. *Null–Adjoining Constraint*: $C(k) = NA = (SA, \emptyset)$
   Meaning: No tree may be adjoined at node k.

The adjunction operation for TAGCs is defined straightforward. If k is an x-marked node in a tree γ, $x \in N$, and $\beta \in A_x$ is an auxiliary tree, β may be adjoined at node k iff $C(k) = (SA, \Gamma)$ or $C(k) = (OA, \Gamma)$ and $\beta \in \Gamma$. By adjunction, node k loses its old constraint and inherits the constraint of the root node of β. The constraints of the remaining nodes are not changed.

The tree language of a TAGC $G = (I, A, C)$ is

$$T(G) := \{\Psi \mid \exists\, \sigma \in I,\ \sigma \stackrel{*}{\Rightarrow}_{adj} \Psi,\ C(k) \neq (OA, \Gamma) \text{ for all inner nodes } k \text{ of } \Psi\}.$$

The generated string language $L(G)$ is defined as it is for TAGs.

Clearly, every TAG is a TAGC: assign to each inner node k marked with $x \in N$ the constraint $C(k) = (SA, A_x)$.

In the following, we may assume that each elementary tree contains only NA- or OA-constraints [Jo85].

We want to compare TAG-rewriting with our BGMI-formalism. Therefore, we first describe TAG-rewriting in the theory of $\times$–categories.

3.2 Description of TAG-rewriting by $\times$–categories

Let $G = (I, A)$ be a TAG, $g = (N \cup T, T, P, s)$ be the underlying context-free grammar and $\mathcal{F}(g) := \mathcal{F}(P, N \cup T, Q, Z)$ the $\times$–category defined by g (see chapter 1).

Each initial tree $\alpha \in I$ is a derivation tree of g with source s and target from T^*. In the free $\times$–category $\mathcal{F}(g)$, for each $\alpha \in I$ there is a unique corresponding morphism

$$N_\alpha : s \to w, \qquad w \in T^*.$$

Thus, we can describe I as a finite subset of $\mathcal{F}(g)(s, T^*)$. Clearly, all nets in this set are trees according to our definition (see chapter 1).

Auxiliary trees $\beta \in A$ are derivation trees of g with source $x \in N$ and targets from T^*xT^*. For each $\beta \in A_x$, there is a unique (non-unit) morphism

$$N_\beta : x \to u \cdot x \cdot v, \qquad u, v \in T^*,\ x \in N.$$

As in section 1.3, we consider terminal-free derivations (nets) and the *syntactic* and *semantic categories* $\mathcal{F}'(g)$ resp. $\mathcal{C}(g)$. Source and target mappings of $\mathcal{F}'(g)$ are denoted with Q' resp. Z'.

We get the following description of a TAG $G = (I, A)$:

1. I is a finite subset of $\mathcal{F}'(g)(s, \varepsilon)$.
2. A_x is a finite subset of $Trees(\mathcal{F}'(g)(x, x))$, $x \in N$.
3. $A = \bigcup_{x \in N} A_x$.

We use the representation of morphisms from $\mathcal{F}'(g)$ as *nets* to visualize the adjunction operation. Each initial tree $\alpha \in I$ corresponds to a net $N_\alpha \in \mathcal{F}'(g)(s, \varepsilon)$ of the following form:

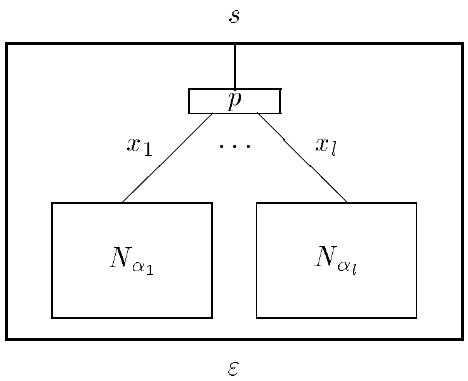

Figure 3.3: Net corresponding to initial tree of a TAG

The *initial net* N_α has a unique decomposition

$$N_\alpha = (N_{\alpha_1} \times \cdots \times N_{\alpha_l}) \circ p$$

where $p \in P,\ Z'(p) = x_1 \cdots x_l,\ x_i \in N,\ N_{\alpha_i} \in \mathcal{F}'(g)(x_i, \varepsilon), 1 \leq i \leq l$.

Production $p : s \to x_1 \cdots x_l$ is the terminal-free version of production

$$p : s \to u_1 \cdot x_1 \cdot u_2 \cdots u_l \cdot x_l \cdot u_{l+1}, \qquad u_1, \ldots, u_{l+1} \in T^*$$

from the CFG g. In the initial tree α, production p forms the root and its children. If $r_1, \ldots, r_l$ are the children of the root of α marked with a nonterminal, the net N_{α_i} corresponds to the subtree α_i of α with root r_i, $1 \leq i \leq l$.

If $f_q : (T^*)^k \to T^*$, $k = |Z'(q)|$, denotes the interpretation function for production $q \in P$, and η denotes the resulting multilinear interpretation for $\mathcal{F}'(g)$, then

$$\begin{aligned} \eta(N_{\alpha_i})(\Box) &= yield(\alpha_i), \\ \eta(N_\alpha)(\Box) &= f_p(\ \eta(N_{\alpha_1}), \ldots, \eta(N_{\alpha_l})\ ) \\ &= yield(\alpha). \end{aligned}$$

In the simplest case, $p : s \to w$, $w \in T^*$, is a terminal production of grammar g and then $l = 0$, $N_\alpha = p$ and $\eta(N_\alpha)(\Box) = f_p(\Box) = w$.

Consider now the auxiliary trees $\beta \in A$ of the TAG G. Let $\beta \in A_x$, $x \in N$, be an auxiliary tree. Then the corresponding net $N_\beta \in \mathcal{F}'(g)$ has the following form:

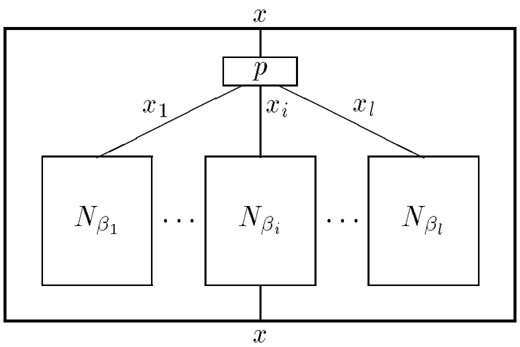

Figure 3.4: Net corresponding to auxiliary tree of a TAG

N_β has a unique decomposition

$$N_\beta = (N_{\beta_1} \times \cdots \times N_{\beta_i} \times \cdots \times N_{\beta_l}) \circ p$$

where $p \in P$, $Q'(p) = x$, $Z'(p) = x_1 \cdots x_l$, $l \in \mathbf{N}$, $i \in \{1, \ldots, l\}$.

Source and target of nets N_{β_j}, $1 \leq j \leq l$, are given by

$$\begin{aligned} Q'(N_{\beta_j}) &= x_j, \\ Z'(N_{\beta_j}) &= \begin{cases} x, & j = i \\ \varepsilon, & j \neq i. \end{cases} \end{aligned}$$

Production $p : x \to x_1 \cdots x_l$ is the terminal-free version of

$$p : x \to u_1 \cdot x_1 \cdot u_2 \cdots u_l \cdot x_l \cdot u_{l+1}, \qquad u_1, \ldots, u_{l+1} \in T^*$$

from CFG g, which forms the root and its children in the auxiliary tree β.

In N_β, there is a unique "root wire" marked with x (corresponding to the root node of β) and a unique "foot wire" marked with x (corresponding to the foot node of β).

In the simplest case, β is a single production $p \in P$ of the form $p : x \to u_1 x u_2$, $u_1, u_2 \in T^*$. Then, $l = 1$ and $N_{\beta_i} = \mathbf{1}_x$ holds.

Describing derivation trees of g as nets from $\mathcal{F}'(g)$ is in a sense dual to the usual description. Each inner node of a derivation tree corresponds to a wire in the net, marked with the corresponding nonterminal. The productions of g (which are implicitly presented in the usual derivation trees), are explicitly presented in the nets.

Consider now the adjunction operation as an operation on nets. Let γ be a derivation tree of g containing an inner node k marked with $x \in N$. γ can be decomposed into subtree γ_2 with root k and the upper tree γ_1 (k is contributed to both trees if k is not the root of γ). If k is the root of γ, γ_1 is empty. Let $w_1 \cdot w_2 \cdot w_3$ be the yield of γ, where w_2 is the yield of γ_2.

In the net N_γ, there is a wire s_k marked with $x \in N$ which corresponds to node k. This wire defines a decomposition $N_\gamma = N_{\gamma_2} \circ N_{\gamma_1}$. The adjunction of auxiliary tree $\beta \in A_x$ at node k in γ corresponds to replacing the "wire" s_x by the net N_β.

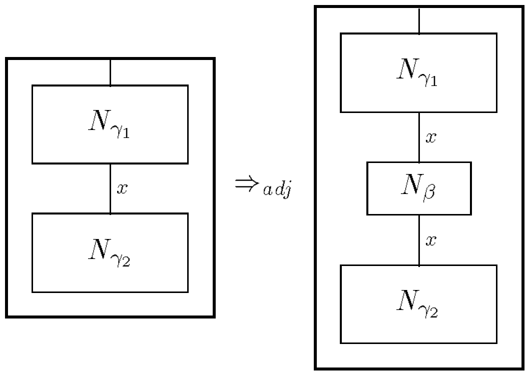

Figure 3.5: Adjunction operation on nets

In terms of categorical expressions, we get:

The adjunction $\gamma \Rightarrow_{adj} \gamma'$, where auxiliary tree β is adjoined at node k in γ, corresponds to the substitution of a unit $\mathbf{1}_x$ by the net N_β.

$$N_\gamma = N_{\gamma_2} \circ N_{\gamma_1} = N_{\gamma_2} \circ \mathbf{1}_x \circ N_{\gamma_1} \Rightarrow_{adj} N_{\gamma_2} \circ N_\beta \circ N_{\gamma_1} = N_{\gamma'}$$

If we consider the interpretation, we get

$$\begin{aligned} \eta(N_{\gamma_1})(w) &= w_1 \cdot w \cdot w_3 \qquad \forall w \in T^*, \\ \eta(N_\beta)(w) &= u \cdot w \cdot v \qquad \forall w \in T^*, \\ \eta(N_{\gamma_2})(\Box) &= w_2. \end{aligned}$$

Thus

$$\begin{aligned} \eta(N_{\gamma'})(\Box) &= \eta(N_{\gamma_2} \circ N_\beta \circ N_{\gamma_1})(\Box) \\ &= w_1 \cdot u \cdot w_2 \cdot v \cdot w_3 \\ &= yield(\gamma'). \end{aligned}$$

By adjoining, the yield of a tree is expanded at two positions which are dependent by the structure of the tree. This is similar to our coupled-context-free rewriting and BGMI-formalism. For this reason, we will now investigate the connection between the different formalisms.

3.3 The equivalence of TAGC and BGMI of degree 1

We show in this section that TAGC and BGMI of degree 1 generate the same class of string languages, i.e. that the language classes TAL and BGMI(1) coincide. At first, we consider the adjoining operation for TAGCs in their description by nets. To avoid confusion, we call morphisms always *nets* even when they are trees according to our definition (see chapter 1). The languages generated by our tree grammars are here called *net languages* for the same reason.

In a TAGC, every nonterminal node k of an elementary (initial or auxiliary) tree γ carries a constraint $C(k)$. Since nodes in derivation trees correspond to "wires" in nets, we assign constraint $C(k)$ to the wire s_k corresponding to k.

If wire s_k carries constraint $C(k) = (OA, B)$, all substitutions $s_k \to N_\beta$, $\beta \in B$, are allowed. Since $C(k)$ is an obligatory constraint, tree γ is not a member of the tree language generated by the TAG. Analogously, net $N_\gamma \in \mathcal{F}'(g)$ is not a member of the generated net language.

If wire s_k carries a null-adjoining-constraint, then s_k cannot be substituted anymore, but it has no influence on membership to the generated net language.

Wires with OA-constraint play the role of nonterminal symbols in tree grammars. The constraint set B contains all possible alternatives that can be substituted for the wire. Analogously to the fact that trees containing nonterminals are not members of the generated tree language, a net containing an OA-constraint is not a member of the generated net language.

We will simulate TAGC-rewriting by tree grammars with multilinear interpretation. According to the remarks above, the idea is obvious:

In each elementary net, we substitute all wires s_k marked with $x \in N$ and constraint (OA, B), by a nonterminal $[x, B]$. We define source and target of the nonterminal $[x, B]$ by $Q'([x, B]) = Z'([x, B]) := x$ to get well-defined nets.

To each nonterminal $[x, B]$, the following set of productions is assigned:

$$\{[x, B] \to N_\beta \mid \beta \in B\}$$

where N_β is the (terminal-free) net corresponding to the auxiliary tree β.

The wires with NA-constraints remain unchanged.

For every initial tree $\alpha \in I$, we introduce a production $\sigma \to N_\alpha$ where σ is a new nonterminal. We define $Q'(\sigma) := s$ and $Z'(\sigma) := \varepsilon$ to be consistent. The result is a grammar on the category of trees $\mathcal{F}'(g)$. (It is obvious that adjunctions in the original TAGC correspond uniquely to rewritings in this tree grammar).

We defined tree grammars only on categories of trees with set of objects $\mathbf{N}_0$. Now we have nets whose wires are marked with symbols from an arbitrary alphabet. This is no problem because we can also simulate TAGC-rewriting using the restricted formalism.

Consider the tree grammar productions $[x, B] \to N_\beta$. It holds:

$$Q'([x, B]) = Z'([x, B]) = x = Q'(N_\beta) = Z'(N_\beta).$$

The source/target-condition is fulfilled for every production. This guarantees that each production is always applicable to an occurrence of its left-side-nonterminal. So it is impossible that an unapplicable production becomes applicable by omitting the wire markings.

The resulting nets are built up by the new nonterminal symbols and by the productions of the underlying CFG g. These productions define the interpretation of the net. According to our definition, the interpretation of a production does not depend on source and target markings but only on source and target length. Thus, we can omit source and target markings of the productions without changing the interpretation.

If we perform the construction given above and then omit the markings of the wires, we get a BGMI of degree 1. The discussion shows that the generated string language equals the string language of the original TAGC.

Theorem 6 *For each TAGC, there is a BGMI which generates the same string language. The degree of the BGMI is at most 1.*

We now show that for each BGMI of degree 1 one can construct a TAGC which generates the same string language. Our construction gives a TAGC described by terminal-free derivation trees together with interpretation functions for the context-free productions. It is easy to get from this a TAGC in the usual form.

Let (G, η) be a BGMI of degree 1, where

$$G = (N \cup T, T, P, s)$$

is the underlying tree grammar with source and target mappings $Q, Z : N \cup T \to \mathbf{N}_0$.

We assume that each nonterminal of G (except the axiom s) has source 1. This is possible because for each BGMI of degree 1, we can give an equivalent one satisfying the condition, as follows:

1. Introduce new terminals $[eps]$ and $[conc]$ with $Q([eps]) := 0$, $Q([conc]) := 2$ and the obvious interpretation as the function giving the empty word resp. the binary concatenation function.
2. For each nonterminal $z \neq s$ with source $Q(z) = 0$, replace z by a new nonterminal z' with source $Q(z') = 1$. Replace each occurrence of z in the right side of a production by $z' \circ [eps]$.
3. Each production $q : z \to y$ is replaced by $q' : z' \to [conc] \circ (\mathbf{1}_1 \times y)$.

This construction preserves the source/target-condition for the productions. It is obvious that the new BGMI generates the same *string* language and satisfies the conditions above.

Remark: This construction is easily extended to grammars of arbitrary degree. Thus, if one is only interested in the string language of a BGMI of degree k, one can always assume that all nonterminals (except the axiom) have degree exactly k.

We now construct a TAGC which is equivalent to a given BGMI (G, η) with

$$G = (N \cup T, T, P, s)$$

Assume that the tree grammar G fulfills the conditions above.

At first, modify the source and target mappings Q and Z of G to mappings

$$Q, Z : N \cup T \to \{\#\}^*$$

by identifying $\mathbf{N}_0$ and $\{\#\}^*$. ($\#$ is a new symbol.) That is, we mark each wire of a net from $\mathcal{B}(N \cup T, Q, Z)$ with the symbol "#".

The constructed nets are members of the free $\times$–category

$$\mathcal{F} := \mathcal{F}(T \cup T', N \dot\cup \{\#\}, Q', Z').$$

T' will be a set of new terminals (only a technical tool). Objects of this $\times$–category are words over the alphabet $N \dot{\cup} \{\#\}$, i.e. these nets are built up by terminals of G and new terminals, and its wires are marked with nonterminals of G or with symbol "#".

Source and target mappings $Q', Z' : T \cup T' \to (N \dot{\cup} \#)^*$ are defined by extending the source and target mappings Q and Z of G, the additional values are given in the construction below.

The terminal set T' is defined as follows: For each $x \in N$, introduce new terminal symbols $[\# \to x]$ and $[x \to \#]$. Define source and target by $Q'([\# \to x]) := \#$, $Z'([\# \to x]) := x$ and inverse for symbol $[x \to \#]$. Let the interpretation of these terminals be the identity.

Let $p : x \to y$ be a production of the BGMI G. Define the auxiliary tree β_p corresponding to p by the following two steps:

- Replace each occurrence of a nonterminal z in y by the net $[z \to \#] \circ [\# \to z]$. Thus, z is replaced by a wire marked with z, which is "delimited" by the terminals $[\# \to z]$ and $[z \to \#]$. This wire is given the constraint (OA, A_z), where A_z is the set of all auxiliary trees for z obtained from the construction. The resulting net β'_p is a member of $\mathcal{F}(\#, \#)$.
- Now we must transform β'_p into an auxiliary tree in A_x because it was constructed from production $p : x \to y$. Let $\beta_p := [\# \to x] \circ \beta'_p \circ [x \to \#] \in \mathcal{F}(x, x)$. This gives a net whose (single) input and output wire are marked with x. Both wires are given a null-adjoining-constraint because their only function is to identify the net as a member of A_x. All wires marked with "#" are also given an NA-constraint.

By this construction, we get for every production $p : x \to y$ exactly one auxiliary net $\beta_p \in A_x$.

We may assume that the axiom s of the BGMI does not occur in the right side of a production and that the only production for s has the form $p : s \to x \circ [eps]$, where x is a nonterminal and $[eps]$ is the terminal described above. Then we let $\alpha := [\# \to x] \circ [eps]$ be the only initial net and give the wire marked x the constraint (OA, A_x). (This wire represents nonterminal x of the tree grammar.)

We obtain by this construction a TAGC (in our presentation) from a given BGMI G of degree 1. The productions of G correspond uniquely to the auxiliary trees of the TAGC.

By induction on the length of derivation resp. adjunction sequences follows the equivalence of these systems. Thus, we have

Theorem 7 *To every BGMI of degree 1, one can construct a TAGC generating the same string language.*

Bibliography

[Ben70] Benson D.B. (1970), "Syntax and semantics: a categorical view", *Information and Control* **17**, 145–160.

[Ben75] Benson D.B. (1975), "The basic algebraic structures in categories of derivations", *Information and Control* **28**, 1–29.

[Ber79] Berstel, J. (1979), "Transductions and Context-Free Languages", Teubner, Stuttgart.

[Br69] Brainerd, W.S. (1969), "Tree generating Regular Systems", *Information and Control* **14**, 217–231.

[GeSt84] Gecseg F., Steinby M. (1984), "Tree Automata", Akadémiai Kiadó, Budapest.

[Go77] Goguen J.A., Thatcher J.W., Wagner E.G., Wright J.B. (1977), "Initial Algebra Semantics and Continuous Algebras", *Journal of the ACM* **24**, 68–95.

[Gu92] Guan, Y. (1992), "Klammergrammatiken, Netzgrammatiken und Interpretationen von Netzen", Dissertation, (May '92).

[Ha78] Harrison, M. (1978), "Introduction to Formal Language Theory", Addison-Wesley.

[Hop79] Hopcroft J.E., Ullman J.D. (1979), "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley.

[Ho65] Hotz G. (1965), "Eine Algebraisierung des Syntheseproblems von Schaltkreisen", *EIK* **1**, 185–205, 209–231.

[Ho66] Hotz G. (1966), "Eindeutigkeit und Mehrdeutigkeit formaler Sprachen" , *EIK* **2**, 235–246.

[Ho67] Hotz G. (1967), "Erzeugung formaler Sprachen durch gekoppelte Ersetzungen", 4. Colloquium über Automatentheorie, ed. by F.L. Bauer, K. Samelson, Math. Institut der TH München, 62–73.

[HoCl72] Hotz G., Claus V. (1972), "Automatentheorie und Formale Sprachen III" , B.I.-Wissenschaftsverlag, Mannheim.

[Ho74] Hotz G. (1974), "Schaltkreistheorie", de Gruyter, Berlin.

[Ho90] Hotz G. (1990), "Einführung in die Informatik", Teubner, Stuttgart.

[Jo85] Joshi A. (1985), "An introduction to Tree Adjoining Grammars", Technical report, University of Pennsylvania.

[Jo86] Joshi A. (1986), "The convergence of mildly context-sensitive grammar formalisms", Workshop on "Processing of Linguistic Structure", Stanford, Jan. 87, (prelim. version).

[Mai74] Maibaum T.S.E. (1974), "A generalized approach to formal languages", *Journal of CSS* **8**, 409–439.

[Th67] Thatcher J.W. (1967), "Characterizing derivation trees of context-free grammars through a generalization of Finite Automata Theory", *Journal of CSS* **1**, 317–322.

[Vi88] Vijay–Shanker K. (1988), "A study of Tree Adjoining Grammars", Dissertation, University of Pennsylvania.