Generalized Context-Free Grammars

Nikè van Vugt

May 24, 1996

Abstract

We consider several language generating formalisms from the literature, such as string-valued attribute grammars with only s-attributes, parallel multiple context-free grammars, relational grammars and top-down tree-to-string transducers, of which we have chosen the OnlyS string-valued attribute grammars to be our vantage point. We prove that OnlyS string-valued attribute grammars, parallel multiple context-free grammars and relational grammars generate the same class of languages, and we prove that every language accepted by an OnlyS string-valued attribute grammar is the image of a top-down tree-to-string transducer.

The main result of this thesis is the proof of equivalence of the special string-valued attribute grammars, the multiple context-free grammar, the special relational grammar and the finite copying top-down tree-to-string transducer.

In order to prove these equivalences, definitions of some of these formalisms have been slightly modified, and normal forms have been (re)defined and proven.

Acknowledgements

Working on this thesis has been a great pleasure to me. Of course, I could not do it all on my own. I would like to thank everyone who was interested in my thesis and in the progress I made. In particular Joost Engelfriet, for coming up with this subject for a thesis and for all his suggestions, Hendrik Jan Hoogeboom, who was my supervisor, for all his help and encouraging comments, and Jurriaan Hage, for helping me with correcting, structuring and struggling with proofs that wouldn't work. My very special thanks go out to Jurriaan and Hendrik Jan for always being there when I needed their support.

Contents

Chapter 1

Introduction

Among computational linguists there has been considerable interest in grammatical formalisms, with generative power in between the context-free languages and the context-sensitive languages, to describe the syntax of natural languages. Some of those formalisms are the combinatory categorial grammars ([A],[S85],[S86]), head grammars ([P]), linear indexed grammars ([Ga]), tree adjoining grammars ([JLT], [J], [V]) and generalized context-free grammars (gcfg's, introduced in [P]).

In this thesis we have compared a subclass of gcfg's, called the parallel multiple context-free grammars (pmcfg's), with a number of formalisms that have been developed in the area of theoretical computer science, in particular the OnlyS string-valued attribute grammar (OnlyS-sag, see for instance [E86], [K]), an adapted version of the relational grammar (rg) of [GR], and the top-down tree-to-string transducer with recognizable input languages (yT(REC) transducer) from [ERS]. We have shown that

$$\text{PMCFL} = \text{OnlyS-SAL} = \text{RL} \subseteq \text{yTL(REC)}$$

where XXXL denotes the class of languages generated by xxx grammars. In order to do this, we have described and proven some normal forms for these types of grammars.

Our main result is the equivalence of the 'special' versions of pmcfg, OnlyS-sag, rg and yT(REC) transducers, in which duplication of information is restricted.

This thesis is structured as follows : in Chapter 2 we describe the context-free grammar and the attribute grammar, that we have chosen to be our vantage point. In Chapter 3 we consider the (parallel) multiple context-free grammars from [SMFK]. Chapter 4 describes the variant of the relational grammar from [GR] that we have considered. In Chapter 5 we discuss the (finite copying) top-down tree-to-string transducers from [ERS]. Finally, Chapter 6 summarizes our results and indicates possible directions for further research.

Chapter 2

Context-free grammars and attribute grammars

2.1 Preliminaries

We write $\mathbf{N}$ for the set of nonnegative integers, and $\mathbf{Q}$ for the set of rational numbers. We write S^n for $S \times S \times \ldots \times S$, the n-fold cartesian product of set S. We denote the empty set by $\emptyset$.

An *alphabet* is a finite set. An alphabet S is *ranked* if it has an associated total function $d : S \rightarrow \mathbf{N}$, such that, for each $s \in S$, $d(s)$ is the *rank* or *dimension* of s. Note that, in our definition, an element of a ranked alphabet has exactly one rank.

We denote the empty string by λ. The length of a string w is denoted as $|w|$, while $|w|_a$ is the number of symbols a that occurs in the string w, and w^R is the reverse of w.

For a total function $f : D \rightarrow R$ and a nonempty subset D' of D, we denote the *restriction of* f *to* D' by $f|_{D'}$ (so $f|_{D'} : D' \rightarrow R$ and, for all $d \in D'$, $f|_{D'}(d) = f(d)$). By R^D we denote the set of all functions with domain D and range R.

The set of *derived functions of the free monoid* Σ^* (generated by the elements of an alphabet Σ) is the set containing all functions f_v, where $v \in (\Sigma \cup \{y_1, \ldots, y_m\})^*$ (with $y_1, \ldots, y_m \notin \Sigma$ and $m \geq 0$), such that $f_v(w_1, \ldots, w_m)$ is the result of substituting w_i for y_i throughout v, with $1 \leq i \leq m$ and $w_i \in \Sigma^*$ (definition from [E86]).

A finite *directed graph* is a pair (Q, E), where Q is a finite set of nodes and $E \subseteq Q \times Q$ is a set of edges. An edge $e_2 = (p_2, q_2)$ is called a *successor* of an edge $e_1 = (p_1, q_1)$ if $q_1 = p_2$. A *path* in a directed graph is a sequence $e_1 e_2 \ldots e_n$ ($n \geq 0$ and $e_j \in E$ for $1 \leq j \leq n$) such that, if $n \geq 2$, then e_{i+1} is a successor of e_i for $1 \leq i \leq n-1$; n is the *length* of $e_1 \ldots e_n$. A path $(p_1, q_1)(p_2, q_2) \ldots (p_n, q_n)$ with $n \geq 1$ in a directed graph is a (directed) *cycle* if $p_1 = q_n$. A directed graph is *cyclic* if it contains a cycle; otherwise it is *acyclic*.

A directed graph t is a *directed tree* if there is a node r, called the *root* of t, such that, for each node x of t, there is *exactly* one path from r to x. In the sequel we will simply write *tree* instead of directed tree. For a tree t, $\text{node}(t)$ is the set of nodes of t and $\text{root}(t) \in \text{node}(t)$ is the root of t. Each node x of a tree t is the root of a *subtree* t' of t. The nodes of t' are x and all its descendants, while the edges of t' are the edges that exist in t between the nodes of t'.

We will discuss grammars, languages and classes of languages using the following conventions : if xxx is a class of grammars, then xxxg abbreviates 'xxx grammar', xxxl denotes the language generated by an xxxg and XXXL is the class of xxxl's. E.g., we use cfg and cfl for 'context-free grammar' and 'context-free language', respectively; CFL denotes the class of context-free languages. Two grammars G and G' are called *equivalent* if the language generated by G is equal to the language generated by G'.

2.2 Context-free grammars

The classical context-free grammar forms the basis of many generative devices considered in this thesis.

Definition 2.1
A *context-free grammar* (cfg) is a 4-tuple $G = (N, T, P, S_0)$ where

i. N is a finite set of *nonterminals*;

ii. T is a finite set of *terminals*, and $N \cap T = \emptyset$;

iii. $P \subseteq N \times (N \cup T)^*$ is a finite set of *productions*;

iv. $S_0 \in N$ is the *initial* symbol.

□

A production (X, θ) is written as $X \to \theta$. It is called a *terminating* production if $\theta \in T^*$. If θ contains at least one nonterminal it is called a *nonterminating* production. By convention, we will use capitals for nonterminals and lower case letters for terminals.

We now formalize how words are generated by a cfg. Let $G = (N, T, P, S_0)$ be a cfg and let $\Sigma = N \cup T$.

Let $x, y \in \Sigma^*$ and let $p : X \to \theta \in P$. We say that *x directly derives y in G (using p)* if $x = x_1 X x_2$ and $y = x_1 \theta x_2$ for certain $x_1, x_2 \in \Sigma^*$. This derivation step is denoted as $x \Rightarrow_G y$ (or $x \Rightarrow_G^p y$).

Let $\Rightarrow_G^*$ be the reflexive and transitive closure of the binary relation $\Rightarrow_G \subseteq \Sigma^* \times \Sigma^*$. We say that *$x$ derives y in G* if $x \Rightarrow_G^* y$. We will omit the subscript 'G' if the cfg under consideration is obvious from the context.

The *language $\mathcal{L}(G)$ generated by G* is defined as $\mathcal{L}(G) = \{x \in T^* \mid S_0 \Rightarrow_G^* x\}$. The elements of $\mathcal{L}(G)$ are called *words* of G.

A *derivation tree* of G is a node labelled ordered tree in which every internal node has <u>either</u> label X_0 and exactly one child, that has label λ, where $X_0 \to \lambda$ is in P, <u>or</u> label X_0 and $k \geq 1$ children, with labels $\theta_1, \ldots, \theta_k$ in $N \cup T$, where $X_0 \to \theta_1 \ldots \theta_k$ is in P.

For $x \in \text{node}(t)$, $\text{label}_t(x)$ is the label of x in t. A node labelled with a (non)terminal is a (non)terminal node. A derivation tree of G is called *complete* if the root has label S_0 and the leaves are labelled with terminals or with λ.

Example 2.1
We give a cfg G that generates the language $L = \{w \in \{a, b\}^* \mid |w|_a = 2 \times |w|_b\}$.

$G = (\{X, A\}, \{a, b\}, \{X \to bAA, X \to AbA, X \to AAb, X \to \lambda, A \to aX, A \to Xa\}, X)$.
Some sample derivations in G are
$X \Rightarrow AbA \Rightarrow aXbA \Rightarrow aXbXa \Rightarrow aXbbAAa \Rightarrow aAAbbbAAa \Rightarrow^*$
$aaXXabbbXaXaa \Rightarrow^* aaabbbaaa$, and
$X \Rightarrow AbA \Rightarrow XabA \Rightarrow abA \Rightarrow abaX \Rightarrow abaAbA \Rightarrow abaXabA \Rightarrow abaabA \Rightarrow abaabaX \Rightarrow$
$abaabaAAb \Rightarrow^* abaabaaab$. □

In the sequel we will not always explicitly specify all the components of a cfg. It suffices to give the productions and the initial symbol; all the elements of N and T can be found implicitly in the productions.

2.3 Attribute grammars

Most of the definitions in subsections 2.3.1 and 2.3.2 are suggested by J. Engelfriet ([E94], see also [DJL], [Gi]).

2.3.1 Syntax

In this subsection we define the notion of attribute grammar, which can be viewed as a cfg in which every nonterminal can have finitely many attributes. To be able to assign values to those attributes, we need the concept of semantic domain.

Definition 2.2
A 2-tuple (Ω, Φ) is a *semantic domain* if Ω is a finite collection of sets and Φ is a set of total functions $f : W_1 \times \ldots \times W_m \to W_{m+1}$ where $m \geq 0$ and $W_i \in \Omega$. In particular, when $m = 0$, Φ can contain elements of $W_1 \in \Omega$, i.e., constants. □

Hence, in a semantic domain (Ω, Φ) the functions in Φ define the typed operations that may be applied to the values of a specific direct product of sets in Ω, and the result of applying such an operation is a member of a specific set in Ω.

Example 2.2
(1) A sample semantic domain is $D_1 = (\Omega_1, \Phi_1)$ with $\Omega_1 = \{\mathbf{N}\}$ and $\Phi_1 = \{C_n \mid n \in \mathbf{N}\} \cup \{\text{succ}\}$, where $C_n() = n$ for all $n \in \mathbf{N}$, and $\text{succ}(x) = x + 1$ for all $x \in \mathbf{N}$.

(2) Another example of a semantic domain is $D_2 = (\{\Sigma^*\}, \Phi_2)$, where Σ is an alphabet, and Φ_2 consists of all derived functions of the free monoid Σ^*. The values that we can use are strings over Σ, and the basic operation is concatenation (from [E86]). Note that every string $x \in \Sigma^*$ is a constant in Φ, it is represented by the unction $f_x()$.
(3) The semantic domain $D_3 = (\Omega_3, \Phi_3)$ is a combination of D_1 and D_2 : $\Omega_3 = \{\mathbf{N}, \Sigma^*\}$, $\Phi_3 = \Phi_1 \cup \Phi_2 \cup \{ \text{ len } \}$, where len : $\Sigma^* \to \mathbf{N}$ assigns the length to a string. □

We are now ready to give a definition of attribute grammars. As stated before, an attribute grammar is in fact a cfg in which a finite number of attributes is added to each nonterminal. In a (complete) derivation tree, the values of these attributes can be passed from one nonterminal to another, and functions can be applied to those values.

The possible attribute values and the functions that can be applied to them are given by a semantic domain.

There are two kinds of attributes, that are called synthesized and inherited attributes. The value of a synthesized attribute of a nonterminal X is usually composed of (or synthesized from) the values of attributes of the nonterminal children of X in a derivation tree, while the value of an inherited attribute of X is usually passed through by (or inherited from) the nonterminal parent of X in a derivation tree (but it is also possible that the value of an i-attribute of X is constructed with the value of an s-attribute of a sibling of X).

An ag translates complete derivation trees into attribute values. The language generated by an ag consists of the values that a designated attribute of the initial symbol can have in complete derivation trees of the grammar.

Definition 2.3
An *attribute grammar* (ag) is a 4-tuple $G = (G_0, D, B, R)$, where

i. $G_0 = (N, T, P, S_0)$ is a cfg such that S_0 does not occur in the right-hand side of any production. N is a ranked alphabet with rank function $d : N \rightarrow \mathbf{N}$;

ii. $D = (\Omega, \Phi)$ is a semantic domain;

iii. $B = (S, I, \alpha_0, W)$ is an *attribute description*, i.e.,

- For each $X \in N \cup T$, $S(X)$ and $I(X)$ are finite disjoint sets of *synthesized* (s-) and *inherited* (i-) *attributes* of X, respectively. There are three restrictions : (1) $I(S_0) = \emptyset$, (2) for each $X \in N$, $|S(X) \cup I(X)| = d(X)$, and (3) for all $X \in T : S(X) = I(X) = \emptyset$;
- $\alpha_0 \in S(S_0)$ is the *designated attribute*;
- For each $a \in \bigcup\{S(X) \cup I(X) \mid X \in N\}$, $W(a) \in \Omega$ denotes the set of attribute values of a.

iv. For a production $p : X_0 \rightarrow w_1 X_1 w_2 \ldots w_k X_k w_{k+1} \in P$ (with $k \geq 0$, $w_i \in T^*$ for $1 \leq i \leq k+1$, and $X_j \in N$ for $0 \leq j \leq k$), a pair $\langle a, j\rangle$ such that $a \in S(X_j) \cup I(X_j)$ is called an *attribute* of p (with $0 \leq j \leq k$). The set of all attributes of p is denoted $A(p)$. For each $p \in P$, $R(p)$ is a finite set of *semantic rules* of p, defined as follows. Each semantic rule of p is specified by a function $f \in \Phi$, say of type $W_1 \times \ldots \times W_m \rightarrow W_0$ ($m \geq 0$), and a sequence of $m+1$ (not necessarily distinct) attributes of p : $(\langle a_0, j_0\rangle, \langle a_1, j_1\rangle, \ldots, \langle a_m, j_m\rangle)$ with $W(a_\ell) = W_\ell$ for $0 \leq \ell \leq m$. Such a semantic rule is denoted as the string $\langle a_0, j_0\rangle = f(\langle a_1, j_1\rangle, \ldots, \langle a_m, j_m\rangle)$ and we say that it defines $\langle a_0, j_0\rangle$ using $\langle a_i, j_i\rangle$ for $1 \leq i \leq m$. $R(p)$ contains one semantic rule defining each attribute $\langle a, 0\rangle$ with $a \in S(X_0)$, and one semantic rule defining each attribute $\langle a, j\rangle$ with $1 \leq j \leq k$ and $a \in I(X_j)$, and no other semantic rules.

□

Notes :

i. If our cfg $G_0 = (N, T, P, S_0)$ has productions in the right-hand side of which the initial symbol occurs, we can easily construct an equivalent cfg G_0' that does not have this kind of productions : $G_0' = (N \cup \{Z\}, T, P \cup \{Z \to S_0\}, Z)$, where Z is a new nonterminal.

ii. Although a nonterminal with rank 0 cannot contribute to the values of the attributes of the ag, it is technically more convenient to allow such nonterminals, in particular it facilitates the proof of Lemma 2.3.

iii. Sometimes, an ag is a 5-tuple $G = (G_0, D, B, R, C)$, where C is a set of *semantic conditions*. These semantic conditions impose an extra restriction (in addition to the semantic rules) on the number of possible correct decorations of derivation trees (see below). Since we will only consider ag's with $C = \emptyset$ for every $p \in P$, we have left C out of our definition.

iv. Note that terminals do not have attributes; for technical reasons it is sometimes convenient to have the notation $S(X)$ and $I(X)$ available for them.

The cfg G_0 of an ag G is called the *underlying cfg* of G. It is possible for an ag to have only s-attributes. Such an ag will be called an *OnlyS-ag*.

We will introduce some notation in order to make it easier to discuss attributes of an ag G :
S-Att $= \{\alpha \mid \alpha \in S(X)$ for some $X \in N\}$ is the set of s-attributes of G,
I-Att $= \{\alpha \mid \alpha \in I(X)$ for some $X \in N\}$ is the set of i-attributes of G,
Att $=$ *S-Att* $\cup$ *I-Att* is the set of attributes of G, and,
for every $X \in N$, $A(X) = S(X) \cup I(X)$ is the set of attributes of X in G.

When we discuss an ag G', for instance, we write *S-Att′*, *I-Att′*, and *Att′* for the sets of s-attributes, i-attributes and attributes of G', respectively. $A'(X)$ denotes the set of attributes of X, where X is a nonterminal of G'.

For a production $p : X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$, we define $\text{inner}(p) = \{\langle \alpha, j\rangle \mid (j = 0 \text{ and } \alpha \in S(X_0)) \text{ or } (1 \leq j \leq k \text{ and } \alpha \in I(X_j))\}$, the set of *inner attributes* of p, and $\text{outer}(p) = \{\langle \alpha, j\rangle \mid (j = 0 \text{ and } \alpha \in I(X_0)) \text{ or } (1 \leq j \leq k \text{ and } \alpha \in S(X_j))\}$, the set of *outer attributes* of p.

We say that an ag is in *normal form* if, for every production, the semantic rules define the inner attributes of the production in terms of the outer attributes of the production. Formally, an ag is in normal form if, for every semantic rule $\langle a_0, j_0\rangle = f(\langle a_1, j_1\rangle, \ldots, \langle a_m, j_m\rangle)$ of every production $p : X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$, $\langle a_i, j_i\rangle \in \text{outer}(p)$ (for $1 \leq i \leq k$).

Restriction 2.1
From now on, we will assume that our ag's are in normal form.

We will not prove that this is indeed a normal form, but it can be shown that, for every (non-circular, see below) ag, (repetitive) replacing of definiendum by definiens results in an equivalent ag in normal form ([E94], [B]).

For an OnlyS-ag in normal form, the outer attributes of a production $p : X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$ are the (s-)attributes of $X_1, \ldots, X_k$, while the inner attributes of p are the (s-)attributes of X_0. Consequently, the semantic rules of p define the attributes of X_0 in terms of the attributes of $X_1, \ldots, X_k$.

2.3.2 Semantics

Let $G = (G_0, D, B, R)$ be an ag, with $G_0 = (N, T, P, S_0)$, $D = (\Omega, \Phi)$ and $B = (S, I, \alpha_0, W)$.

Every derivation tree of G_0 is also a derivation tree of G, and there are no other derivation trees of G. An *occurrence of a production* $p : X_0 \rightarrow \theta$ in a derivation tree t of G consists of a node x_0 of t, labelled X_0, and its children, such that the labels of the children, from left to right, form the string θ. We say that p *is applied at* x_0.

For a derivation tree t and a node x of t that is not labelled with λ, we define the *attributes of* x as $A(x) = \{\langle \alpha, x\rangle \mid \alpha \in A(\text{label}_t(x))\}$. $I(x)$ and $S(x)$ are defined analogously. For t we define the *attributes of* t as $A(t) = \bigcup\{A(x) \mid x \in \text{node}(t)\}$, i.e., $A(t) = \{\langle \alpha, x\rangle \mid x \in \text{node}(t) \text{ and } \alpha \in A(\text{label}_t(x))\}$; $\langle \alpha, x\rangle$ is also called an *attribute instance.*

We now have to determine the values of every attribute of every node, i.e., the derivation tree has to be 'decorated' with attribute values. A *decoration* of t is a total function val : $A(t) \rightarrow \bigcup \Omega$, such that, for each $\langle \alpha, x\rangle \in A(t)$, $\text{val}(\langle \alpha, x\rangle) \in W(\alpha)$. The pair $\langle t$,val$\rangle$ is called a *decorated tree.*

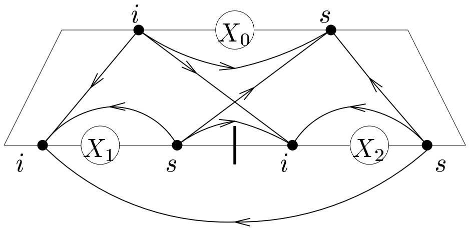
We also have to define when a decoration is correct, i.e., we have to define when the attribute values satisfy the semantic rules. For an occurrence of a production $p : X_0 \rightarrow w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$ with nonterminal nodes $x_0, x_1, \ldots, x_k$ in a derivation tree t, if $R(p)$ contains the semantic rule $\langle a_0, j_0\rangle = f(\langle a_1, j_1\rangle, \ldots, \langle a_m, j_m\rangle)$, then the string $\langle a_0, x_{j_0}\rangle$=$f$($\langle a_1, x_{j_1}\rangle$,..,$\langle a_m, x_{j_m}\rangle$) is called a *semantic instruction* of t; it is a string over $A(t) \cup \Phi \cup \{(,\ ),\ ,,\ =\}$. The set of all semantic instructions is called $R(t)$, i.e., $R(t)$ is a set of equations (where the unknown elements are the elements of $A(t)$) that has to be solved. A decoration val of t is defined to be a *correct decoration* if for every semantic instruction $\langle a_0, x_{j_0}\rangle$=$f$($\langle a_1, x_{j_1}\rangle$,..,$\langle a_m, x_{j_m}\rangle$) the following holds : $\text{val}(\langle a_0, x_{j_0}\rangle) = f(\text{val}(\langle a_1, x_{j_1}\rangle), \ldots, \text{val}(\langle a_m, x_{j_m}\rangle))$.

Another concept that we need to be able to work with ag's is the dependency graph. If $R(p)$ contains a semantic rule $\langle a_0, j_0\rangle = f(\langle a_1, j_1\rangle, \ldots, \langle a_m, j_m\rangle)$, we say that $\langle a_i, j_i\rangle$ *is necessary for* $\langle a_0, j_0\rangle$, with $1 \leq i \leq m$. If $R(t)$ contains a semantic instruction $\langle a_0, x_{j_0}\rangle$=$f$($\langle a_1, x_{j_1}\rangle$,..,$\langle a_m, x_{j_m}\rangle$) we say that $\langle a_i, x_{j_i}\rangle$ is necessary for $\langle a_0, x_{j_0}\rangle$, with $1 \leq i \leq m$.

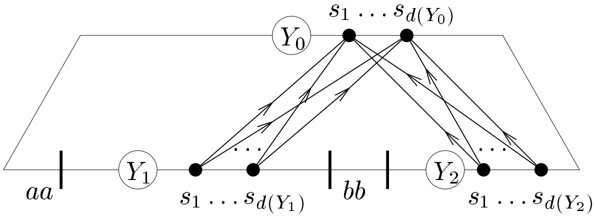
So 'is necessary for' is a binary relation on $A(p)$ and on $A(t)$. Since a finite directed graph is a binary relation on a finite set, we now have two finite directed graphs : $D(p) = (A(p)$, is necessary for), and $D(t) = (A(t)$, is necessary for). $D(p)$ and $D(t)$ are called the *dependency graphs* of p and t, respectively.

We will depict the dependency graph of a production $p : X_0 \rightarrow w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$ as follows. Open circles denote the nonterminal nodes of the derivation tree, while smaller black dots denote the attributes. We draw the s-attributes of a nonterminal to the right of the circle corresponding to (and labelled with) that nonterminal, and the i-attributes to the left. Dependencies between attributes are represented by arrows between the corresponding black dots. An arrow from $\langle a_1, j_1\rangle$ to $\langle a_0, j_0\rangle$ denotes that $\langle a_1, j_1\rangle$ is necessary for $\langle a_0, j_0\rangle$. Every open circle with label X is connected to the attributes of X by a horizontal line, and with those horizontal lines a trapezium is built, to delimit the occurrence of p in the derivation tree. The terminal strings are written below the horizontal line connecting the circles for X_1 through X_k, and terminals and nonterminals are separated by thicker vertical lines. Often we will omit a terminal string that is equal to λ, and the corresponding separator. The dependency graph of a derivation tree t is depicted by 'putting together' the dependency graphs of the productions that occur in t (see Example 2.4 (2)).

As an example of the possible dependencies in an ag in normal form, we draw the dependency graph of $p : X_0 \to X_1X_2$, where p is a production of an ag in normal form. For simplicity, we assume that each of the nonterminals X_0, X_1 and X_2 has one s-attribute s and one i-attribute i. To show every possible dependency, we assume moreover that every outer attribute of p is necessary for every inner attribute of p.



For an OnlyS-ag in normal form, there are even less dependencies possible, as we have only s-attributes. We give the dependency graph for $q : Y_0 \to aaY_1bbY_2$, where Y_0, Y_1 and Y_2 are nonterminals and a and b are terminals. Again, to show all possibilities, we assume that every outer attribute of q (i.e., every s-attribute of Y_1 and Y_2) is necessary for every inner attribute of q (i.e., every s-attribute of Y_0). Let Y_i have s-attributes s_1 through $s_{d(Y_i)}$, for $i = 0, 1, 2$, where $d(Y_i)$ denotes the rank of Y_i.



An ag G is called *circular* if there is a derivation tree t of G such that $D(t)$ is cyclic; otherwise the ag is called *non-circular*.

Restriction 2.2
We will only consider non-circular ag's.

We impose this restriction on our ag's because we want to be sure that there is exactly one correct decoration for every complete derivation tree. It is known that this is the case for non-circular ag's without semantic conditions (see [E86]).

An ag G defines the *string-value translation* $\tau(G)$, that is defined as $\tau(G) = \{\langle \text{yield}(t), \text{val}(\langle \alpha_0, \text{root}(t)\rangle)\rangle \mid t$ is a complete derivation tree of G, val is a correct decoration of $t\}$. This translation adds a unique meaning to each complete derivation tree. There are other translations possible, like the '(derivation) tree-(decorated) tree', the 'string-(decorated) tree' and the '(derivation) tree-value' translation, but the string-value translation is the appropriate notion in the context of this thesis.

Usually, the language generated by G, $\mathcal{L}(G)$, is defined to be the language that consists of all strings that satisfy both the context-free syntax (of the underlying cfg) and the context-sensitive syntax (of the semantic conditions). Since we consider only ag's without semantic conditions, in our case $\mathcal{L}(G)$ would have been equal to $\mathcal{L}(G_0)$, and consequently this definition of $\mathcal{L}(G)$ would not be very useful to us : it is context-free.

Since we are mainly interested in the meanings of the complete derivation trees of an ag G, we will define the *language generated by* G as $\mathcal{L}(G) = \{\text{val}(\langle \alpha_0, \text{root}(t)\rangle) \mid t$ is a complete derivation tree of G, and val is a correct decoration of $t\}$. In [E86] this language is called OUT(G), the output set of G. Observe that this is not always a *string* language.

2.3.3 Examples and notation

We first give a formal description of a sample ag.

Example 2.3
We construct an ag G that counts the number of occurrences of a's, b's and c's, respectively, in words of the form $awbw^Rc$, where $w \in \{a,b,c\}^*$. Therefore, we make a cfg G_0 that generates words of the required form ($awbw^Rc$), and to each production of G_0 we add semantic rules that will count the occurrences of a's, b's and c's, as follows.

$G = (G_0, D, B, R)$ where
$G_0 = (\{X,Y\}, \{a,b,c\}, \{X \to aYc, Y \to aYa, Y \to bYb, Y \to cYc, Y \to b\}, X)$,
$D = (\Omega, \Phi)$ with $\Omega = \{\mathbf{N}, \mathbf{N}^3\}$ and $\Phi = \{f, g, id, C_0, C_1\}$,
where $f(x,y,z) = (x+1, y, z+1)$, $g(x) = x+2$, $id(x) = x$, $C_0() = 0$ and $C_1() = 1$,
$B = (S, I, \alpha_0, W)$ with $S(X) = \{\delta\}$, $S(Y) = \{\alpha, \beta, \gamma\}$, $I(X) = \emptyset$, $I(Y) = \emptyset$, $\alpha_0 = \delta$,
$W(\delta) = \mathbf{N}^3$, $W(\alpha) = W(\beta) = W(\gamma) = \mathbf{N}$, and
$R(X \to aYc) = \{\ \langle\delta,0\rangle = f(\langle\alpha,1\rangle,\ \langle\beta,1\rangle,\ \langle\gamma,1\rangle)\ \}$,
$R(Y \to aYa) = \{\ \langle\alpha,0\rangle = g(\langle\alpha,1\rangle),\ \langle\beta,0\rangle = id(\langle\beta,1\rangle),\ \langle\gamma,0\rangle = id(\langle\gamma,1\rangle)\ \}$,
$R(Y \to bYb) = \{\ \langle\alpha,0\rangle = id(\langle\alpha,1\rangle),\ \langle\beta,0\rangle = g(\langle\beta,1\rangle),\ \langle\gamma,0\rangle = id(\langle\gamma,1\rangle)\ \}$,
$R(Y \to cYc) = \{\ \langle\alpha,0\rangle = id(\langle\alpha,1\rangle),\ \langle\beta,0\rangle = id(\langle\beta,1\rangle),\ \langle\gamma,0\rangle = g(\langle\gamma,1\rangle)\ \}$,
$R(Y \to b) = \{\ \langle\alpha,0\rangle = C_0(),\ \langle\beta,0\rangle = C_1(),\ \langle\gamma,0\rangle = C_0()\ \}$.

G is an OnlyS-ag and translates words u that are generated by its underlying cfg G_0 into 3-tuples (x,y,z) in $\mathbf{N}^3$ such that $x = |u|_a$, $y = |u|_b$ and $z = |u|_c$.
$\mathcal{L}(G) = \{(x,y,z) \mid x,y,z \in \mathbf{N}$ and x, y and z are odd $\}$. □

Since the notation used in the previous example is not easy to read, we will now introduce some conventions and abbreviations to describe concrete ag's.

The formal description of a production is $p : X_0 \to w_1X_1w_2 \ldots w_kX_kw_{k+1}$, as indicated in Definition 2.3. In a concrete production we will *implicitly* add the same numbering to the occurrences of the nonterminals as we did in this formal description.

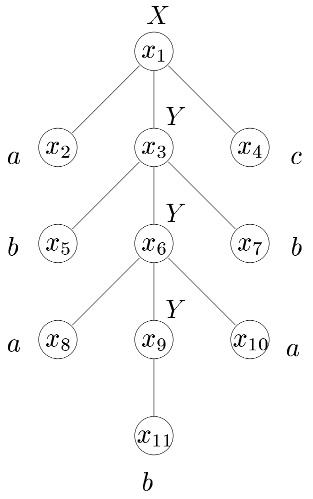
E.g., in the production $A \to aAbB$ there are three occurrences of nonterminals numbered 0 (the left-hand A), 1 (the right-hand A) and 2 (the B). Furthermore, we will write a_j instead of $\langle a, j\rangle$ when we are discussing attributes.

We will often describe an ag as follows : we give all the productions of P and to the right of each production we write all the semantic rules that define its inner attributes. Moreover, we will use infix notation for the functions in the semantic rules. We will not explicitly give the attributes of each nonterminal, the semantic domain and so forth; all the additional information will be implicit from the productions and semantic rules.

Example 2.4
(1) The OnlyS-ag from Example 2.3 is described as follows, when we use the conventions mentioned above.

$X \to aYc$	$\delta_0 = (\alpha_1 + 1, \beta_1, \gamma_1 + 1)$		
$Y \to aYa$	$\alpha_0 = \alpha_1 + 2$	$\beta_0 = \beta_1$	$\gamma_0 = \gamma_1$
$Y \to bYb$	$\alpha_0 = \alpha_1$	$\beta_0 = \beta_1 + 2$	$\gamma_0 = \gamma_1$
$Y \to cYc$	$\alpha_0 = \alpha_1$	$\beta_0 = \beta_1$	$\gamma_0 = \gamma_1 + 2$
$Y \to b$	$\alpha_0 = 0$	$\beta_0 = 1$	$\gamma_0 = 0$

An example of a correctly decorated derivation tree $\langle t, \text{val}\ \rangle$ is the following, where t is



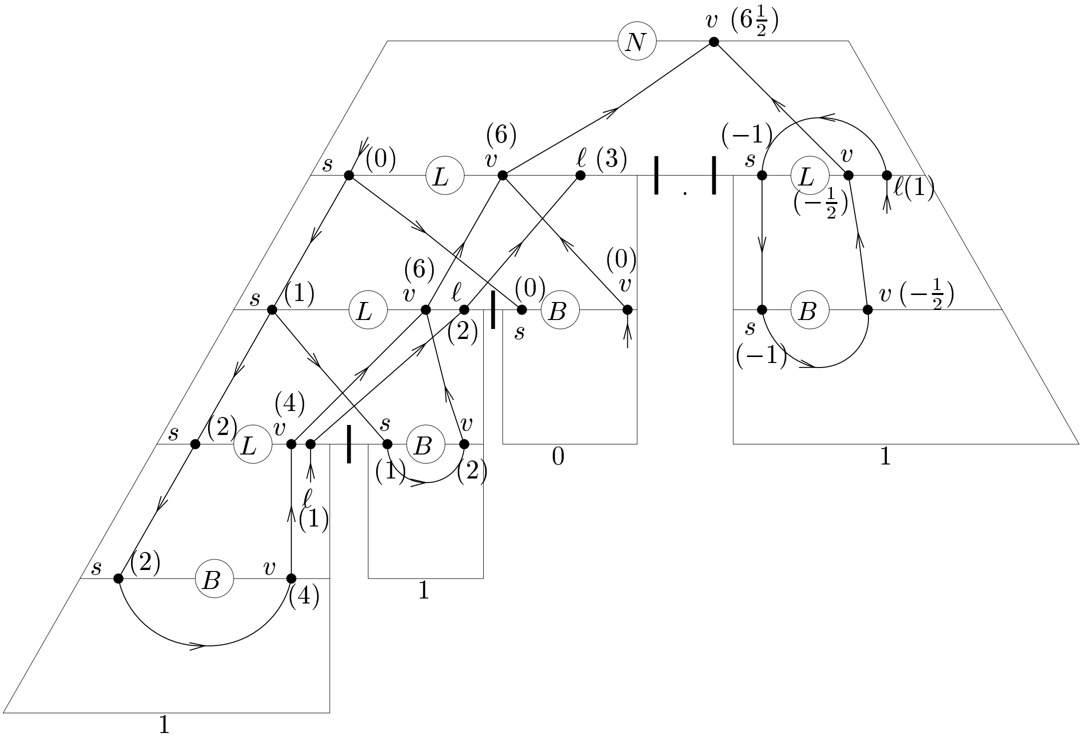
and $\text{val}(\langle\alpha, x_9\rangle) = 0$, $\text{val}(\langle\beta, x_9\rangle) = 1$, $\text{val}(\langle\gamma, x_9\rangle) = 0$,
$\text{val}(\langle\alpha, x_6\rangle) = 2$, $\text{val}(\langle\beta, x_6\rangle) = 1$, $\text{val}(\langle\gamma, x_6\rangle) = 0$,
$\text{val}(\langle\alpha, x_3\rangle) = 2$, $\text{val}(\langle\beta, x_3\rangle) = 3$, $\text{val}(\langle\gamma, x_3\rangle) = 0$,
$\text{val}(\langle\delta, x_1\rangle) = (3, 3, 1)$.

(2) We give an ag G', with synthesized and inherited attributes, that defines the translation $\tau(G') = \{\langle x, y\rangle \mid x$ is a string of the form w or $w.v$, with $w, v \in \{0,1\}^+$, that represents a binary number (with or without fraction) and y is the decimal value of $x\}$ (from [K]).

$B \to 0$	$v_0 = 0$			
$B \to 1$	$v_0 = 2^{s_0}$			
$L \to B$	$v_0 = v_1$	$s_1 = s_0$	$\ell_0 = 1$	
$L \to LB$	$v_0 = v_1 + v_2$	$s_2 = s_0$	$s_1 = s_0 + 1$	$\ell_0 = \ell_1 + 1$
$N \to L$	$v_0 = v_1$	$s_1 = 0$		
$N \to L.L$	$v_0 = v_1 + v_2$	$s_1 = 0$	$s_2 = -\ell_2$	

where v is the value of B, L, or N, ℓ is the length of L, s is the scale of B or L, $W(v) = \mathbf{Q}$, $W(\ell) = \mathbf{N}$, $W(s) = \mathbf{N}$, $S\text{-}Att = \{v, \ell\}$, and $I\text{-}Att = \{s\}$.

The dependency graph $D(t)$ for a derivation tree t of 110.1 can be depicted as follows. The values of the attributes are written between parentheses.



□

2.3.4 Some simple normal forms for ag's

It is known that, for every cfg, we can construct an equivalent cfg in which every symbol (terminal or nonterminal) is productive, which means that it can generate a terminal word, and in which every symbol can be reached from the initial nonterminal. We will give a definition of this normal form (without proof that it is indeed a normal form) and we carry it over to ag's.

Definition 2.4
Let $G_0 = (N, T, P, S_0)$ be a cfg and let $X \in N \cup T$. X is called *useful* (*in* G_0) if X occurs in a complete derivation tree of G_0. If every $X \in N \cup T$ is useful in G_0, or if $G_0 = (\{S_0\}, \emptyset, \emptyset, S_0)$, G_0 is called a *reduced* cfg.

Let $G = (G_0, D, B, R)$ be an ag. X is called *useful in* G if X is useful in G_0. G is called a *reduced* ag if G_0 is a reduced cfg. □

It is possible that the initial symbol of a cfg is not productive. In that case, no symbol can be useful, so the language generated by the cfg is empty; an equivalent cfg is then $(\{S_0\}, \emptyset, \emptyset, S_0)$, as stated in the definition.

In the following lemma we explain that reduced ag's are a normal form of the ag's.

Lemma 2.1
For every ag we can construct an equivalent reduced ag.
Proof
We know that the reduced cfg is a normal form for cfg's. Reducing a cfg means deleting productions that contain symbols that are not useful. Reducing an ag then means deleting those productions and their semantic rules, but not *changing* any productions or semantic rules. Since the complete derivation trees of the cfg do not change, the translation defined by the ag does neither, and thus the language generated by the ag stays the same. Consequently, the reduced ag is equivalent to the original one. □

Nonterminals of rank 0 do not contribute to the value computed by the ag. Hence they do not contribute to the language of the ag we considere here. We can easily remove these nonterminals without changing the language.

Lemma 2.2
Let G be an ag with set of nonterminals N, rank function d and with $\mathcal{L}(G) \neq \emptyset$. Then we can construct an equivalent (reduced) ag G' with set of nonterminals N' and rank function d', such that $d'(X') \geq 1$ for all $X' \in N'$.
Proof
According to Lemma 2.1, we may assume that G is reduced.

We assume that every nonterminal of G is useful. Let $G = ((N, T, P, S_0), D, B, R)$. We remove every production $p : X_0 \to w_1 X_1 w_2 \dots w_k X_k w_{k+1}$ with $d(X_0) = 0$ from P, since X_0 cannot pass any (attribute) values from itself or its children to its parent in a derivation tree, nor from itself or its parent to its children. We remove p's semantic rules from R.

Furthermore, we replace every occurrence of a nonterminal X with rank 0 in the right-hand side of a production by λ. This does not have any effect on the semantic rules of those productions, since X did not have any attributes that could occur in these semantic rules.

The resulting ag $G' = ((N', T, P', S_0), D, B, R')$ where P' and R' are described above, $N' = \{X \mid X \in N \text{ and } d(X) > 0\}$, and G' has rank function $d' = d|_{N'}$.

Note that G' is reduced. □

In this proof, the assumption that G is reduced is necessary, because the construction described in the previous lemma can remove useless symbols with rank 0 from a production, which may cause the production to occur in a complete derivation tree of G' while it could not occur in a complete derivation tree of G (see the example in subsubsection 2.3.5.3). Obviously, this would change the language generated by the ag. If G is reduced, this cannot happen.

2.3.5 String-valued attribute grammars

In general, the attribute values of an ag can be of any type (integers, strings, vectors etc.). Since our main concern is string languages, we will now define an ag with a restricted semantic domain: the string-valued ag (see [E86]). Its attribute values are strings over an alphabet Σ, and the basic operation on these strings is concatenation.

2.3.5.1 Definitions and examples

Definition 2.5
A *string-valued attribute grammar* (sag) is an ag with semantic domain $(\{\Sigma^*\}, \Phi)$ for some alphabet Σ, where Φ consists of all derived functions of the free monoid Σ^* generated by the elements of Σ. □

Informally, this means that we specify our semantic rules by $\langle a_0, j_0 \rangle = u_1 \langle a_1, j_1 \rangle u_2 \ldots u_m \langle a_m, j_m \rangle u_{m+1}$, with $u_i \in \Sigma^*$ and $1 \leq i \leq m+1$. Thus the right-hand side of a semantic rule of a production p is given as an element of $(\Sigma \cup A(p))^*$.

Example 2.5
Consider the cfg G_0 from Example 2.3, that generates the language $K = \{\ awbw^Rc \mid w \in \{a,b,c\}^* \}$. Here is an example of a sag G that 'sorts' the letters a, b and c that occur in the words that are generated by G_0. Formally, G generates the language $M = \{\ a^n b^m c^k \mid |v|_a = n, |v|_b = m, |v|_c = k$ for some $v \in K\} = \{a^m b^n c^k \mid m, n, k$ odd $\}$. The translation defined by G is $\tau(G) = \{\langle v, a^n b^m c^k \rangle \mid v \in K,\ |v|_a = n,\ |v|_b = m,\ |v|_c = k\}$.

$X \to aYc$	$\delta_0 = a\alpha_1\beta_1 c\gamma_1$		
$Y \to aYa$	$\alpha_0 = aa\alpha_1$	$\beta_0 = \beta_1$	$\gamma_0 = \gamma_1$
$Y \to bYb$	$\alpha_0 = \alpha_1$	$\beta_0 = bb\beta_1$	$\gamma_0 = \gamma_1$
$Y \to cYc$	$\alpha_0 = \alpha_1$	$\beta_0 = \beta_1$	$\gamma_0 = cc\gamma_1$
$Y \to b$	$\alpha_0 = \lambda$	$\beta_0 = b$	$\gamma_0 = \lambda$

This sag is an OnlyS-sag. □

In the OnlyS-ag's that we have defined, it is possible that not every outer attribute of a production is used to define the inner attributes of this production. Consequently there can be loss of information. On the other hand, we can also have duplication : some of the outer attributes of a production may be used more than once to define the inner attributes of that production.

The following definition, that was suggested by J. Engelfriet (see also [Gi]), describes an OnlyS-ag that can have neither duplication nor loss of information.

Definition 2.6
A *special* attribute grammar is an OnlyS-ag such that, for every production p, every outer attribute of p is used exactly once in a definition of an inner attribute of p.
□

This means that every attribute in a dependency graph has exactly one outgoing edge (except for the designated attribute, of course).

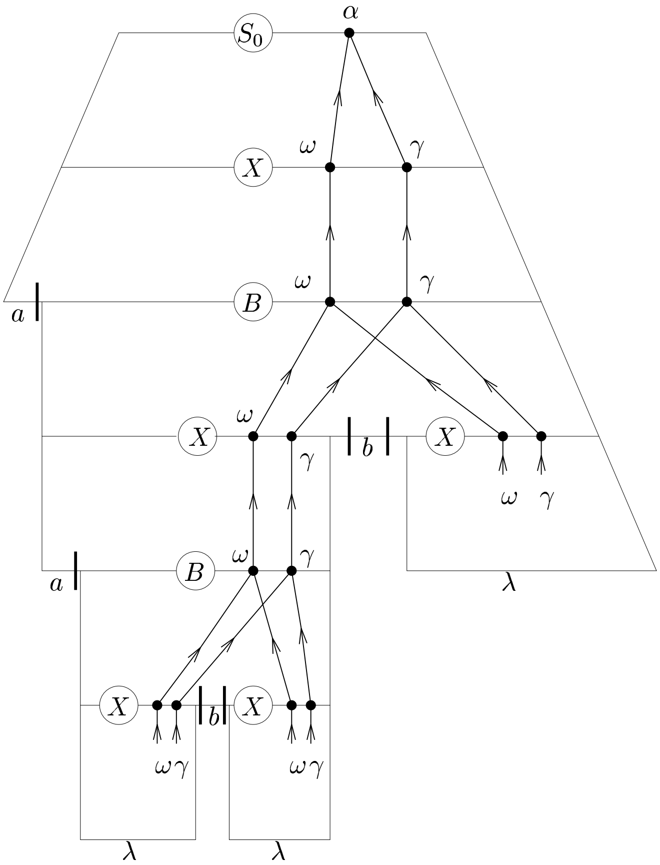
Since we will use the concept of special string-valued ag rather frequently, we introduce an abbreviation for it : ssag.

Example 2.6
(1) The sag of Example 2.5 is also an ssag, since, for every production p, every attribute in outer(p) is used exactly once to define an inner attribute of p.

(2) We give an ssag G for the (non-context-free) language $L = \{wc^n \mid w \in \{a,b\}^*, |w|_a = |w|_b = n\}$. Note that the attribute ω of a node x in a derivation tree has a value that is equal to the yield of the subtree rooted at x.

$S_0 \rightarrow X$	$\alpha_0 = \omega_1 \gamma_1$	
$X \rightarrow bA$	$\omega_0 = b\ \omega_1$	$\gamma_0 = c\ \gamma_1$
$X \rightarrow aB$	$\omega_0 = a\ \omega_1$	$\gamma_0 = c\ \gamma_1$
$X \rightarrow \lambda$	$\omega_0 = \lambda$	$\gamma_0 = \lambda$
$A \rightarrow XaX$	$\omega_0 = \omega_1 a\ \omega_2$	$\gamma_0 = \gamma_1 \gamma_2$
$B \rightarrow XbX$	$\omega_0 = \omega_1 b\ \omega_2$	$\gamma_0 = \gamma_1 \gamma_2$

The dependency graph of a derivation tree for $a^2b^2c^2$ is the following.



□

2.3.5.2 A normal form for OnlyS-sag's

We define a *structural normal form* for OnlyS-sag's. It is *structural* in the sense that every production and every semantic rule should have a special form.

In Lemma 2.2 from [SMFK] this normal form is formulated for (parallel) multiple context-free grammars. We have filled in some of the details of the proof of that lemma, and we prove this normal form for OnlyS-sag's instead of (parallel) multiple context-free grammars, because we thought that technically more convenient.

For proving that this is indeed a normal form for OnlyS-sag's, it would be convenient to know that the initial symbol has rank 1. If necessary, we can easily construct an equivalent ag that satisfies this requirement, by adding a new initial symbol Z, that has a single (synthesized) attribute α_0, and adding the production $Z \to S_0$ with semantic rule $\langle \alpha_0, 0 \rangle = \langle \alpha_0, 1 \rangle$. Here S_0 and α_0 are the initial symbol and the designated attribute of the original ag, respectively.

In the proof of the following lemma we will use nonterminals of the form X^U, where U is a subset of the set of s-attributes of the nonterminal X. The variant X^U of X represents a nonterminal whose attributes generate exactly the same values as the attributes of X that are contained in U (we could say that X^U is the restriction of X to U). We use these variants X^U of X on two occasions in the proof : first U contains only those attributes of X that are indeed used to compute attribute values of the parent of X in a derivation tree, and later U contains only those attributes of X that cannot have the value λ.

An OnlyS-sag is in structural normal form if it satisfies five properties mentioned in Lemma 2.3 below. Our proof will construct an equivalent ag in normal form in five consecutive steps. In each step an additional requirement will be satisfied (without violating the previous ones).

In this proof, the language generated by the underlying cfg will change, and so will the translation defined by the OnlyS-sag. But, of course, the language generated by the OnlyS-sag does not change, since the resulting OnlyS-sag should be equivalent to the original one.

Lemma 2.3
For every OnlyS-sag G, it is possible to construct an equivalent OnlyS-sag G', with initial symbol S_0', that has the following properties :

i. For every production p, every outer attribute of p is used at least once in a definition of an inner attribute of p. This is called the information-lossless condition.

ii. Every nonterminal that occurs in the left-hand side of a terminating production has rank 1.

iii. For every nonterminating production q, the right-hand side of every semantic rule of q can be written as an element of $(\text{outer}(q))^*$.

iv. For every nonterminal $X \neq S_0'$ of G', no attribute of X can have the value λ.

v. For every terminating production r with left-hand side $X \neq S_0'$, the length of the right-hand side of the semantic rule of r is 1. If the left-hand side of r is S_0', then the length of the right-hand side of the semantic rule of r is 0 or 1.

Proof

Let $G = (G_0, D, B, R)$ with $G_0 = (N, T, P, S_0)$, $D = (\{\Sigma^*\}, \Phi)$, $B = (S, I, \alpha_0, W)$, rank function $d : N \to \mathbf{N}$, and $d(S_0) = 1$.

i. We construct an OnlyS-sag G_1, that satisfies the information-lossless condition and is equivalent to G, from G as follows. Let $G_1 = ((N_1, T, P_1, S_0), D, (S_1, I_1, \alpha_0, W), R_1)$ with rank function d_1, and let $N_1 = P_1 = \emptyset$.

For every production $p : S_0 \to w_1 X_1 w_2 \dots w_k X_k w_{k+1}$ in P, we add to P_1 the production $p' : S_0 \to w_1 X_1^{V_1} w_2 \dots w_k X_k^{V_k} w_{k+1}$ with $R_1(p') = R(p)$, $V_i = \{s \in S(X_i) \mid s$ occurs in the right-hand side of the semantic rule in $R(p)\}$, $S_1(X_i^{V_i}) = V_i$ and $I_1(X_i^{V_i}) = \emptyset$ (for $1 \leq i \leq k$). We add the $X_i^{V_i}$ and S_0 to N_1, and we let $d_1(X_i^{V_i}) = |V_i|$ and $d_1(S_0) = d(S_0) = 1$. Note that the newly introduced productions and semantic rules have the required property.

We also have to make productions and semantic rules for the newly introduced nonterminals and for the nonterminals that we are going to introduce.
For every X^V in N_1 for which there are no productions in P_1 yet, consider all productions q for X_i in P, where
$q : X_i \to v_1 Y_1 v_2 \dots v_m Y_m v_{m+1}$ has semantic rules
$\langle s_\ell, 0\rangle = u_{\ell,1}\langle s_{\ell,1}, j_{\ell,1}\rangle u_{\ell,2} \dots u_{\ell,n_\ell}\langle s_{\ell,n_\ell}, j_{\ell,n_\ell}\rangle u_{\ell,n_\ell+1}$ for $1 \leq \ell \leq d(X_i)$.
For each of these productions q we add to P_1 the production
$q' : X_i^{V_i} \to v_1 Y_1^{U_1} v_2 \dots v_m Y_m^{U_m} v_{m+1}$ with $R_1(q') = \{\langle s, 0\rangle = \beta \mid \langle s, 0\rangle = \beta \in R(q)$ and $s \in V_i\}$ and, for $1 \leq j \leq m$, $U_j = \{s \in S(Y_j) \mid \langle s, j\rangle$ occurs in the right-hand side of a semantic rule in $R_1(q')\}$, $S_1(Y_j^{U_j}) = U_j$ and $I_1(Y_j^{U_j}) = \emptyset$. We add the $Y_j^{U_j}$ to N_1, and we let $d_1(Y_j^{U_j}) = |U_j|$. Again, the newly introduced productions and semantic rules have the desired property.

When there is no $X_i^{V_i}$ left for which we have to add productions, we have reached our goal. This construction will end eventually, since there are only finitely many nonterminals and there are at most $2^{|S(X)|}$ variants of a nonterminal X, for each of which we have to make finitely many productions.

ii. We construct G_2, that satisfies i and ii and is equivalent to G_1, from G_1. From Lemma 2.2, we may assume that every $X \in N_1$ has rank greater than 0 (the construction in Lemma 2.2 preserves i).

For every terminating production $p : X \to w$ in P_1, with $d_1(X) > 1$, $S_1(X) = \{s_1, \dots, s_{d_1(X)}\}$ and semantic rules $\langle s_i, 0\rangle = v_i$ (with $v_i \in \Sigma^*$ for $1 \leq i \leq d_1(X)$), we add $d_1(X)$ *new* nonterminals X_i to N_1, with $d_2(X_i) = 1$, and to P_1 we add $d_1(X)$ new terminating productions $X_i \to \lambda$, with $R_2(X_i \to \lambda) = \{\langle s, 0\rangle = v_i\}$. Let $S_2(X_i) = \{s\}$, $I_2(X_i) = \emptyset$ and $W_2(s) = \Sigma^*$.

Furthermore, we remove production p from P_1 and we add a new nonterminating production $q : X \to X_1 \dots X_{d_1(X)}$ to P_1, with $d_1(X)$ semantic rules : $R_2(q) = \{\langle s_1, 0\rangle = \langle s, 1\rangle, \dots, \langle s_{d_1(X)}, 0\rangle = \langle s, d_1(X)\rangle\}$.

In this way we obtain $G_2 = ((N_2, T_2, P_2, S_0), (\{\Sigma^*\}, \Phi), (S_2, I_2, \alpha_0, W_2), R_2)$, and rank function $d_2 : N_2 \to \mathbf{N}$, that satisfies both i and ii.

iii. We adjust G_2 in order to make it satisfy iii as well.

Consider a nonterminating production $p : X_0 \to w_1 X_1 w_2 \dots w_k X_k w_{k+1}$ in P_2, with semantic rules $\langle s_i, 0\rangle = u_{i,1}\langle a_{i,1}, j_{i,1}\rangle u_{i,2} \dots u_{i,m_i}\langle a_{i,m_i}, j_{i,m_i}\rangle u_{i,m_i+1}$. For every $u_{i,q}$, we add to N_2 a *new* nonterminal $X_{i,q}$, with $d_3(X_{i,q}) = 1$, and to P_2 we add $X_{i,q} \to \lambda$, with semantic rule $\langle s, 0\rangle = u_{i,q}$, $S_3(X_{i,q}) = \{s\}$, $I_3(X_{i,q}) = \emptyset$, and $W_3(s) = \Sigma^*$.

Furthermore, we replace p by
$$p' : X_0 \to w_1 X_1 w_2 \dots w_k X_k w_{k+1} X_{1,1} \dots X_{1,m_1+1} X_{2,1} \dots X_{d_2(X_0), m_{d_2(X_0)}+1}.$$
The semantic rules of p' are the same as those of p, but we replace $u_{i,q}$ by $\langle s, z_{i,q}\rangle$, where $z_{i,q} = k + \Sigma_{j<i}(m_j + 1) + q$ is the position of $X_{i,q}$ in the right-hand side of p'.

After executing this procedure for every nonterminating production we have $G_3 = ((N_3, T_3, P_3, S_0), (\{\Sigma^*\}, \Phi), (S_3, I_3, \alpha_0, W_3), R_3)$ and rank function $d_3 : N_3 \to \mathbf{N}$, that satisfies i through iii, and is equivalent to G_2.

iv. We construct G_4, that satisfies iv as well, from G_3.

First we adjust the terminating productions that have a semantic rule with right-hand side λ. We replace every terminating production $p : X \to w$, where $X \neq S_0$, with semantic rule $\langle s, 0\rangle = \lambda$ (where $S_3(X) = \{s\}$ and $d_3(X) = 1$, since G_3 satisfies ii), by the production $p' : X^{\emptyset} \to w$ with $d_4(X^{\emptyset}) = 0$, $S_4(X^{\emptyset}) = I_4(X^{\emptyset}) = \emptyset$ and $R_4(p') = \emptyset$. We add $X^{\emptyset}$ to N_3.

Next, we change the terminating productions that do not have a semantic rule with right-hand side λ. This is necessary because of the construction applied to nonterminating productions described below. We replace every terminating production $r : Y \to v$, with semantic rule $\langle s, 0\rangle = u$ (where $u \in \Sigma^+$, $S_3(Y) = \{s\}$ and $d_3(Y) = 1$, since G_3 satisfies ii), by the production $r' : Y^{\{s\}} \to v$ with $d_4(Y^{\{s\}}) = 1$, $S_4(Y^{\{s\}}) = \{s\}$, $I_4(Y^{\{s\}}) = \emptyset$ and $R_4(r') = \{\langle s, 0\rangle = u\}$. We add $Y^{\{s\}}$ to N_3.

Furthermore, we replace every nonterminating production
$q : X_0 \to w_1 X_1 w_2 \dots w_k X_k w_{k+1}$, with semantic rules
$$\langle s_i, 0\rangle = \langle s_{i,1}, j_{i,1}\rangle \dots \langle s_{i,n_i}, j_{i,n_i}\rangle$$
with $n_i \geq 0$ for $1 \leq i \leq d_3(X_0)$, by the productions $q' : X_0^{U_0} \to w_1 X_1^{U_1} w_2 \dots w_k X_k^{U_k} w_{k+1}$ with $U_j \subseteq S_3(X_j)$ for $1 \leq j \leq k$ (so we construct $2^{|S_3(X_1)|} \times \dots \times 2^{|S_3(X_k)|}$ new productions), the $X_\ell^{U_\ell}$ $(0 \leq \ell \leq k)$ are new nonterminals that we add to N_3, $S_4(X_\ell^{U_\ell}) = U_\ell$, $d_4(X_\ell^{U_\ell}) = |U_\ell|$ and $I_4(X_\ell^{U_\ell}) = \emptyset$.

If $X_0 \neq S_0$, or $X_0 = S_0$ and $\lambda \notin \mathcal{L}(G)$, then we set $R_4(q') = \{\langle s, 0\rangle = \beta' \mid \langle s, 0\rangle = \beta \in R_3(q)$, β' is constructed from β by replacing every $\langle s, j\rangle \notin U_j$ for $1 \leq j \leq k$

that occurs in β by λ, and $\beta' \neq \lambda\}$, and $U_0 = \{s \in S_3(X_0) \mid$ there is a semantic rule for $\langle s, 0\rangle$ in $R_4(q')\}$.

If $X_0 = S_0$ and $\lambda \in \mathcal{L}(G)$, then $U_0 = S_3(S_0)$, and $R_4(q') = \{\langle s,0\rangle = \beta' \mid \langle s,0\rangle = \beta \in R_3(q)$ and β' is constructed from β by replacing every $\langle s,j\rangle \notin U_j$ for $1 \leq j \leq k$ that occurs in β by $\lambda\}$. To simplify notation, we write S_0 instead of $S_0^{U_0}$.

To make sure that the resulting grammar satisfies ii, we have to reduce it and remove nonterminals with rank 0 (see Lemma 2.2).

The resulting grammar is $G_4 = ((N_4, T_4, P_4, S_0), (\{\Sigma^*\}, \Phi), (S_4, I_4, \alpha_0, W_4), R_4)$ with rank function $d_4 : N_4 \rightarrow \mathbf{N}$, which satisfies i through iv and is equivalent to G_3.

v. We construct the required OnlyS-sag G' from G_4.

Consider the terminating production $p : X \rightarrow w$ in P_4 with $S_4(X) = \{s\}$, one semantic rule $\langle s,0\rangle = v$ and $v = v_1 \ldots v_m$ (with $v_i \in \Sigma$, $1 \leq i \leq m$ and $m > 1$). We add m *new* nonterminals X_j (for $1 \leq j \leq m$) to N_4, with $d'(X_j) = 1$, and to P_4 we add m new terminating productions $X_j \rightarrow \lambda$. We let $R'(X_j \rightarrow \lambda) = \{\langle s,0\rangle = v_j\}$, $S'(X_j) = \{s\}$, $I'(X_j) = \emptyset$ and $W'(s) = \Sigma^*$.

Furthermore, we remove production p and we add a new nonterminating production $q : X \rightarrow wX_1 \ldots X_m$ to P_4 with one semantic rule : $R'(q) = \{\langle s,0\rangle = \langle s,1\rangle \ldots \langle s,m\rangle\}$. We apply this procedure to every terminating production in P_4 of which the semantic rule's right-hand side has length greater than 1.

Now we have $G' = ((N', T', P', S_0), (\{\Sigma^*\}, \Phi), (S', I', \alpha_0, W'), R')$ and rank function $d' : N' \rightarrow \mathbf{N}$, that satisfies i through v and is equivalent to G.

□

Note that, for an OnlyS-sag that satisfies iii and iv, the following holds : for every nonterminating production q with left-hand side $\neq S_0$, the right-hand side of every semantic rule of q can be written as an element of $(\text{outer}(q))^+$.

2.3.5.3 Example

Here is an example of applying Lemma 2.3 to a given OnlyS-sag G.
Let G be

$S \rightarrow AB$	$s_0 = \alpha_1\gamma_2$	
$A \rightarrow AB$	$\alpha_0 = \beta_2\alpha_1\gamma_2$	
$A \rightarrow a$	$\alpha_0 = \lambda$	
$B \rightarrow B$	$\beta_0 = a\beta_1 b$	$\gamma_0 = cd$
$B \rightarrow a$	$\beta_0 = ab$	$\gamma_0 = cd$

G generates the language $\mathcal{L}(G) = \{a^{n_1}b^{n_1} \ldots a^{n_k}b^{n_k}(cd)^{k+1} \mid n_i \geq 1$ for $1 \leq i \leq k$ and $k \geq 0\}$.
We bring G, step by step, in structural normal form. We use the following notational convention here : a set of (s-)attributes is written as a string that is the concatenation

of all the elements of the set, in an arbitrary order. Thus for instance $S(B) = \{\beta, \gamma\}$ will be written as $\beta\gamma$.
(i) G does not satisfy the information-lossless condition, because in the semantic rule $s_0 = \alpha_1\gamma_2$ of $S \to AB$, β_2 is not used, and in the semantic rules $\beta_0 = a\beta_1 b$ and $\gamma_0 = cd$ of $B \to B$, γ_1 does not occur. We start by making an equivalent OnlyS-sag G_1 that satisfies the information-lossless condition.

$$\begin{array}{lllll}
S & \to A^{\alpha}B^{\gamma} & s_0 = \alpha_1\gamma_2 & & \\
A^{\alpha} & \to A^{\alpha}B^{\beta\gamma} & \alpha_0 = \beta_2\alpha_1\gamma_2 & & \\
A^{\alpha} & \to a & \alpha_0 = \lambda & & \\
B^{\gamma} & \to B^{\emptyset} & \gamma_0 = cd & & \\
B^{\gamma} & \to a & \gamma_0 = cd & & \\
B^{\beta\gamma} & \to B^{\beta} & \beta_0 = a\beta_1 b & \gamma_0 & = cd \\
B^{\beta\gamma} & \to a & \beta_0 = ab & \gamma_0 & = cd \\
B^{\emptyset} & \to B^{\emptyset} & & & \\
B^{\emptyset} & \to a & & & \\
B^{\beta} & \to B^{\beta} & \beta_0 = a\beta_1 b & & \\
B^{\beta} & \to a & \beta_0 = ab & &
\end{array}$$

(ii) Next, we are going to make sure that the nonterminals that occur in the left-hand side of a terminating production have rank 1 (this is only necessary for $B^{\beta\gamma}$ and $B^{\emptyset}$, since A^{α}, B^{γ} and B^{β} already have rank 1). First, we have to reduce G_1. Since every symbol in G_1 is useful, G_1 is already reduced. Then we have to remove the nonterminals with rank 0 (i.e., $B^{\emptyset}$). This gives us G'_1 :

$$\begin{array}{lllll}
S & \to A^{\alpha}B^{\gamma} & s_0 = \alpha_1\gamma_2 & & \\
A^{\alpha} & \to A^{\alpha}B^{\beta\gamma} & \alpha_0 = \beta_2\alpha_1\gamma_2 & & \\
A^{\alpha} & \to a & \alpha_0 = \lambda & & \\
B^{\gamma} & \to \lambda & \gamma_0 = cd & & \\
B^{\gamma} & \to a & \gamma_0 = cd & & \\
B^{\beta\gamma} & \to B^{\beta} & \beta_0 = a\beta_1 b & \gamma_0 & = cd \\
B^{\beta\gamma} & \to a & \beta_0 = ab & \gamma_0 & = cd \\
B^{\beta} & \to B^{\beta} & \beta_0 = a\beta_1 b & & \\
B^{\beta} & \to a & \beta_0 = ab & &
\end{array}$$

Now we can apply the construction of Lemma 2.3 ii to G'_1, which leads to G_2 :

$$\begin{array}{lllll}
S & \to A^{\alpha}B^{\gamma} & s_0 = \alpha_1\gamma_2 & & \\
A^{\alpha} & \to A^{\alpha}B^{\beta\gamma} & \alpha_0 = \beta_2\alpha_1\gamma_2 & & \\
A^{\alpha} & \to a & \alpha_0 = \lambda & & \\
B^{\gamma} & \to \lambda & \gamma_0 = cd & & \\
B^{\gamma} & \to a & \gamma_0 = cd & & \\
B^{\beta\gamma} & \to B^{\beta} & \beta_0 = a\beta_1 b & \gamma_0 & = cd \\
X & \to \lambda & \xi_0 = ab & & \\
Y & \to \lambda & \xi_0 = cd & & \\
B^{\beta\gamma} & \to XY & \beta_0 = \xi_1 & \gamma_0 & = \xi_2 \\
B^{\beta} & \to B^{\beta} & \beta_0 = a\beta_1 b & & \\
B^{\beta} & \to a & \beta_0 = ab & &
\end{array}$$

(iii) Now we have to replace the constant strings that occur in the right-hand sides of the semantic rules of the nonterminating productions $B^{\beta\gamma} \to B^{\beta}$ and $B^{\beta} \to B^{\beta}$.

We do not follow the construction given in Lemma 2.3 iii exactly, however. We will reuse nonterminals, if possible, instead of adding new nonterminals (see for instance $B^{\beta\gamma} \to B^{\beta}UVY$, in which we have used the already present nonterminal Y rather than introducing a new one). We will do the same with the other nonterminals that we should introduce during this example.

This gives G_3 :

$S \to A^{\alpha}B^{\gamma}$	$s_0 = \alpha_1\gamma_2$	
$A^{\alpha} \to A^{\alpha}B^{\beta\gamma}$	$\alpha_0 = \beta_2\alpha_1\gamma_2$	
$A^{\alpha} \to a$	$\alpha_0 = \lambda$	
$B^{\gamma} \to \lambda$	$\gamma_0 = cd$	
$B^{\gamma} \to a$	$\gamma_0 = cd$	
$B^{\beta\gamma} \to B^{\beta}UVY$	$\beta_0 = \xi_2\beta_1\xi_3$	$\gamma_0 = \xi_4$
$U \to \lambda$	$\xi_0 = a$	
$V \to \lambda$	$\xi_0 = b$	
$X \to \lambda$	$\xi_0 = ab$	
$Y \to \lambda$	$\xi_0 = cd$	
$B^{\beta\gamma} \to XY$	$\beta_0 = \xi_1$	$\gamma_0 = \xi_2$
$B^{\beta} \to B^{\beta}UV$	$\beta_0 = \xi_2\beta_1\xi_3$	
$B^{\beta} \to a$	$\beta_0 = ab$	

Before we remove 'λ-semantic rules', we will rename the nonterminals of G_3, yielding G_3'. Note that this is not really necessary: it is only done for reasons of clarity.

$S \to AC$	$s_0 = \alpha_1\gamma_2$	
$A \to AD$	$\alpha_0 = \beta_2\alpha_1\gamma_2$	
$A \to a$	$\alpha_0 = \lambda$	
$C \to \lambda$	$\gamma_0 = cd$	
$C \to a$	$\gamma_0 = cd$	
$D \to BUVY$	$\beta_0 = \xi_2\beta_1\xi_3$	$\gamma_0 = \xi_4$
$U \to \lambda$	$\xi_0 = a$	
$V \to \lambda$	$\xi_0 = b$	
$X \to \lambda$	$\xi_0 = ab$	
$Y \to \lambda$	$\xi_0 = cd$	
$D \to XY$	$\beta_0 = \xi_1$	$\gamma_0 = \xi_2$
$B \to BUV$	$\beta_0 = \xi_2\beta_1\xi_3$	
$B \to a$	$\beta_0 = ab$	

(iv) Applying the construction of Lemma 2.3 iv now gives G_4' :

$S^{s} \to A^{\alpha}C^{\gamma}$	$s_0 = \alpha_1\gamma_2$
$S^{s} \to A^{\emptyset}C^{\gamma}$	$s_0 = \gamma_2$
$S^{s} \to A^{\alpha}C^{\emptyset}$	$s_0 = \alpha_1$
$S^{\emptyset} \to A^{\emptyset}C^{\emptyset}$	
$A^{\alpha} \to A^{\alpha}D^{\beta\gamma}$	$\alpha_0 = \beta_2\alpha_1\gamma_2$
$A^{\alpha} \to A^{\alpha}D^{\beta}$	$\alpha_0 = \beta_2\alpha_1$

A^{α}	$\to A^{\alpha}D^{\gamma}$	$\alpha_0 = \alpha_1\gamma_2$	
A^{α}	$\to A^{\alpha}D^{\emptyset}$	$\alpha_0 = \alpha_1$	
A^{α}	$\to A^{\emptyset}D^{\beta\gamma}$	$\alpha_0 = \beta_2\gamma_2$	
A^{α}	$\to A^{\emptyset}D^{\beta}$	$\alpha_0 = \beta_2$	
A^{α}	$\to A^{\emptyset}D^{\gamma}$	$\alpha_0 = \gamma_2$	
$A^{\emptyset}$	$\to A^{\emptyset}D^{\emptyset}$		
$A^{\emptyset}$	$\to a$		
C^{γ}	$\to \lambda$	$\gamma_0 = cd$	
C^{γ}	$\to a$	$\gamma_0 = cd$	
$D^{\beta\gamma}$	$\to B^{\beta}U^{\xi}V^{\xi}Y^{\xi}$	$\beta_0 = \xi_2\beta_1\xi_3$	$\gamma_0 = \xi_4$
D^{β}	$\to B^{\beta}U^{\xi}V^{\xi}Y^{\emptyset}$	$\beta_0 = \xi_2\beta_1\xi_3$	
$D^{\beta\gamma}$	$\to B^{\beta}U^{\xi}V^{\emptyset}Y^{\xi}$	$\beta_0 = \xi_2\beta_1$	$\gamma_0 = \xi_4$
$D^{\beta\gamma}$	$\to B^{\beta}U^{\emptyset}V^{\xi}Y^{\xi}$	$\beta_0 = \beta_1\xi_3$	$\gamma_0 = \xi_4$
$D^{\beta\gamma}$	$\to B^{\beta}U^{\emptyset}V^{\emptyset}Y^{\xi}$	$\beta_0 = \beta_1$	$\gamma_0 = \xi_4$
D^{β}	$\to B^{\beta}U^{\emptyset}V^{\xi}Y^{\emptyset}$	$\beta_0 = \beta_1\xi_3$	
D^{β}	$\to B^{\beta}U^{\xi}V^{\emptyset}Y^{\emptyset}$	$\beta_0 = \xi_2\beta_1$	
D^{β}	$\to B^{\beta}U^{\emptyset}V^{\emptyset}Y^{\emptyset}$	$\beta_0 = \beta_1$	
$D^{\beta\gamma}$	$\to B^{\emptyset}U^{\xi}V^{\xi}Y^{\xi}$	$\beta_0 = \xi_2\xi_3$	$\gamma_0 = \xi_4$
D^{β}	$\to B^{\emptyset}U^{\xi}V^{\xi}Y^{\emptyset}$	$\beta_0 = \xi_2\xi_3$	
$D^{\beta\gamma}$	$\to B^{\emptyset}U^{\xi}V^{\emptyset}Y^{\xi}$	$\beta_0 = \xi_2$	$\gamma_0 = \xi_4$
$D^{\beta\gamma}$	$\to B^{\emptyset}U^{\emptyset}V^{\xi}Y^{\xi}$	$\beta_0 = \xi_3$	$\gamma_0 = \xi_4$
D^{γ}	$\to B^{\emptyset}U^{\emptyset}V^{\emptyset}Y^{\xi}$		$\gamma_0 = \xi_4$
D^{β}	$\to B^{\emptyset}U^{\emptyset}V^{\xi}Y^{\emptyset}$	$\beta_0 = \xi_3$	
D^{β}	$\to B^{\emptyset}U^{\xi}V^{\emptyset}Y^{\emptyset}$	$\beta_0 = \xi_2$	
$D^{\emptyset}$	$\to B^{\emptyset}U^{\emptyset}V^{\emptyset}Y^{\emptyset}$		
U^{ξ}	$\to \lambda$	$\xi_0 = a$	
V^{ξ}	$\to \lambda$	$\xi_0 = b$	
X^{ξ}	$\to \lambda$	$\xi_0 = ab$	
Y^{ξ}	$\to \lambda$	$\xi_0 = cd$	
$D^{\beta\gamma}$	$\to X^{\xi}Y^{\xi}$	$\beta_0 = \xi_1$	$\gamma_0 = \xi_2$
D^{γ}	$\to X^{\emptyset}Y^{\xi}$		$\gamma_0 = \xi_2$
D^{β}	$\to X^{\xi}Y^{\emptyset}$	$\beta_0 = \xi_1$	
$D^{\emptyset}$	$\to X^{\emptyset}Y^{\emptyset}$		
B^{β}	$\to B^{\beta}U^{\xi}V^{\xi}$	$\beta_0 = \xi_2\beta_1\xi_3$	
B^{β}	$\to B^{\beta}U^{\xi}V^{\emptyset}$	$\beta_0 = \xi_2\beta_1$	
B^{β}	$\to B^{\beta}U^{\emptyset}V^{\xi}$	$\beta_0 = \beta_1\xi_3$	
B^{β}	$\to B^{\beta}U^{\emptyset}V^{\emptyset}$	$\beta_0 = \beta_1$	
B^{β}	$\to B^{\emptyset}U^{\xi}V^{\xi}$	$\beta_0 = \xi_2\xi_3$	
B^{β}	$\to B^{\emptyset}U^{\xi}V^{\emptyset}$	$\beta_0 = \xi_2$	
B^{β}	$\to B^{\emptyset}U^{\emptyset}V^{\xi}$	$\beta_0 = \xi_3$	
$B^{\emptyset}$	$\to B^{\emptyset}U^{\emptyset}V^{\emptyset}$		
B^{β}	$\to a$	$\beta_0 = ab$	

This grammar, however, does no longer satisfy the condition of Lemma 2.3 ii, since $A^{\emptyset}$ occurs in the left-hand side of a terminating rule ($A^{\emptyset} \to a$) and has rank 0. So we reduce

G'_4 (in which $S^\emptyset$, $C^\emptyset$, D^β, D^γ, $D^\emptyset$, $U^\emptyset$, $V^\emptyset$, $X^\emptyset$, $Y^\emptyset$ and $B^\emptyset$ are not useful), which leads to G''_4 :

$$\begin{array}{lllll}
S^s & \to A^\alpha C^\gamma & s_0 = \alpha_1\gamma_2 & & \\
S^s & \to A^\emptyset C^\gamma & s_0 = \gamma_2 & & \\
A^\alpha & \to A^\alpha D^{\beta\gamma} & \alpha_0 = \beta_2\alpha_1\gamma_2 & & \\
A^\alpha & \to A^\emptyset D^{\beta\gamma} & \alpha_0 = \beta_2\gamma_2 & & \\
A^\emptyset & \to a & & & \\
C^\gamma & \to \lambda & \gamma_0 = cd & & \\
C^\gamma & \to a & \gamma_0 = cd & & \\
D^{\beta\gamma} & \to B^\beta U^\xi V^\xi Y^\xi & \beta_0 = \xi_2\beta_1\xi_3 & \gamma_0 = \xi_4 & \\
U^\xi & \to \lambda & \xi_0 = a & & \\
V^\xi & \to \lambda & \xi_0 = b & & \\
X^\xi & \to \lambda & \xi_0 = ab & & \\
Y^\xi & \to \lambda & \xi_0 = cd & & \\
D^{\beta\gamma} & \to X^\xi Y^\xi & \beta_0 = \xi_1 & \gamma_0 = \xi_2 & \\
B^\beta & \to B^\beta U^\xi V^\xi & \beta_0 = \xi_2\beta_1\xi_3 & & \\
B^\beta & \to a & \beta_0 = ab & &
\end{array}$$

Now we remove from G''_4 the nonterminals with rank 0, of which $A^\emptyset$ is the only one.

Observe that it is important that a grammar is reduced before removing nonterminals with rank 0. If we had not reduced G'_4, we would have introduced, for instance, the production $D^\beta \to B^\beta U^\xi V^\xi$ by removing $Y^\emptyset$. But in G'_4, $Y^\emptyset$ is not useful, which causes $D^\beta \to B^\beta U^\xi V^\xi Y^\emptyset$ never to be used in a complete derivation tree. So $D^\beta \to B^\beta U^\xi V^\xi$ should not occur in our grammar, because it might now be used in a complete derivation tree, and thus it would change the language generated by the ag.

Using the construction of Lemma 2.2, the result is G_4 :

$$\begin{array}{llll}
S^s & \to A^\alpha C^\gamma & s_0 = \alpha_1\gamma_2 & \\
S^s & \to C^\gamma & s_0 = \gamma_2 & \\
A^\alpha & \to A^\alpha D^{\beta\gamma} & \alpha_0 = \beta_2\alpha_1\gamma_2 & \\
A^\alpha & \to D^{\beta\gamma} & \alpha_0 = \beta_2\gamma_2 & \\
C^\gamma & \to \lambda & \gamma_0 = cd & \\
C^\gamma & \to a & \gamma_0 = cd & \\
D^{\beta\gamma} & \to B^\beta U^\xi V^\xi Y^\xi & \beta_0 = \xi_2\beta_1\xi_3 & \gamma_0 = \xi_4 \\
U^\xi & \to \lambda & \xi_0 = a & \\
V^\xi & \to \lambda & \xi_0 = b & \\
X^\xi & \to \lambda & \xi_0 = ab & \\
Y^\xi & \to \lambda & \xi_0 = cd & \\
D^{\beta\gamma} & \to X^\xi Y^\xi & \beta_0 = \xi_1 & \gamma_0 = \xi_2 \\
B^\beta & \to B^\beta U^\xi V^\xi & \beta_0 = \xi_2\beta_1\xi_3 & \\
B^\beta & \to a & \beta_0 = ab &
\end{array}$$

(v) The last step of the construction involves breaking up right-hand sides of semantic rules that are too long. The result of this step is G', with $\mathcal{L}(G') = \mathcal{L}(G)$, which is in structural normal form.

$$\begin{array}{lll}
S^s & \to A^\alpha C^\gamma & s_0 = \alpha_1\gamma_2
\end{array}$$

$$
\begin{array}{llllll}
S^{s} & \to C^{\gamma} & s_0 & = \gamma_2 & & \\
A^{\alpha} & \to A^{\alpha} D^{\beta\gamma} & \alpha_0 & = \beta_2 \alpha_1 \gamma_2 & & \\
A^{\alpha} & \to D^{\beta\gamma} & \alpha_0 & = \beta_2 \gamma_2 & & \\
C^{\gamma} & \to WZ & \gamma_0 & = \xi_1 \xi_2 & & \\
W & \to \lambda & \xi_0 & = c & & \\
Z & \to \lambda & \xi_0 & = d & & \\
C^{\gamma} & \to aWZ & \gamma_0 & = \xi_1 \xi_2 & & \\
D^{\beta\gamma} & \to B^{\beta} U^{\xi} V^{\xi} Y^{\xi} & \beta_0 & = \xi_2 \beta_1 \xi_3 & \gamma_0 & = \xi_4 \\
U^{\xi} & \to \lambda & \xi_0 & = a & & \\
V^{\xi} & \to \lambda & \xi_0 & = b & & \\
X^{\xi} & \to U^{\xi} V^{\xi} & \xi_0 & = \xi_1 \xi_2 & & \\
Y^{\xi} & \to WZ & \xi_0 & = \xi_1 \xi_2 & & \\
D^{\beta\gamma} & \to X^{\xi} Y^{\xi} & \beta_0 & = \xi_1 & \gamma_0 & = \xi_2 \\
B^{\beta} & \to B^{\beta} U^{\xi} V^{\xi} & \beta_0 & = \xi_2 \beta_1 \xi_3 & & \\
B^{\beta} & \to a U^{\xi} V^{\xi} & \beta_0 & = \xi_1 \xi_2 & &
\end{array}
$$

Chapter 3

Generalized context-free grammars

3.1 Introduction

In this chapter we discuss generalized context-free grammars, which can be viewed as cfg's in which the nonterminals generate tuples of strings instead of strings.

We now give our definition of a generalized context-free grammar. We have slightly modified the definition given in [SMFK] in order to show the similarities between generalized context-free grammars and the other grammars that we describe in this thesis (in particular the attribute grammar), and in order to make the definition more precise.

We need some terminology concerning functions that have tuples of strings over some alphabet T as input and output values. For such a function f, let $a(f)$ denote the number of arguments of f, $d_i(f)$ (for $1 \leq i \leq a(f)$) the dimension of the i^{th} argument of f, and $r(f)$ the dimension of the result of applying f. Hence f is a function from $(T^*)^{d_1(f)} \times (T^*)^{d_2(f)} \times \ldots \times (T^*)^{d_{a(f)}(f)}$ to $(T^*)^{r(f)}$. The description of the i^{th} component of the result of applying f is denoted f^i, where $1 \leq i \leq r(f)$.

We also define, for each f, $\overline{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d_i(f)})$, the i^{th} argument of f, and $V(f) = \{x_{i,j} \mid 1 \leq i \leq a(f) \text{ and } 1 \leq j \leq d_i(f)\}$, a set of variables that denote the components of the arguments of f.

Definition 3.1
Let m be a positive integer. An *m-generalized context-free grammar* (m-gcfg or gcfg) is a 5-tuple $G = (N, T, D, P, S_0)$ where

i. N is a ranked alphabet of *nonterminal* symbols. The rank or dimension of each nonterminal is given by the total function $d : N \to \mathbf{N}$;

ii. T is a finite set of *terminal* symbols which is disjoint with N;

iii. D is a semantic domain $(\{O_1, O_2, \ldots, O_m\}, F)$, where O_i is the set of all tuples of dimension i of strings over T, i.e., $O_i = (T^*)^i$ for $1 \leq i \leq m$. There are no restrictions on F, thus each $f \in F$ is an arbitrary function from $O_{d_1(f)} \times O_{d_2(f)} \times \ldots \times O_{d_{a(f)}(f)}$ to $O_{r(f)}$. Obviously, $r(f)$ and $d_i(f)$ are not greater than m;

iv. P is a finite set of *(rewriting) rules* each of which is of the form $(f, X_0, X_1, \ldots, X_{a(f)})$ where $f \in F$ and $X_i \in N$ for $0 \leq i \leq a(f)$. If a rule $(f, X_0, X_1, \ldots, X_{a(f)})$ is in P, then $r(f) = d(X_0)$ and $d_i(f) = d(X_i)$ for $1 \leq i \leq a(f)$;

v. $S_0 \in N$ is the *initial* symbol, and $d(S_0) = 1$, i.e., S_0 generates strings (more precisely tuples of dimension 1).

□

Notes :

i. We will omit the parentheses of tuples of dimension 1.

ii. We have the range of the rank function defined to be $\mathbf{N}$, although, as in the case of ag's, nonterminals with rank 0 are 'not needed'.

We abbreviate $\{O_1, \ldots, O_m\}$ as O. An element $(f, X_0, X_1, \ldots, X_q)$ of P is written as $X_0 \rightarrow f[X_1, \ldots, X_q]$. If $q = 0$, i.e. if f is a constant element of $O_{r(f)}$, the rule is called a *terminating* rule; otherwise it is called a *nonterminating* rule.

The *language generated by* $X \in N$ in G, $\mathcal{L}_G(X) \subseteq (T^*)^{d(X)}$, is defined as follows. For $X \in N$, $\mathcal{L}_G(X)$ is the smallest set satisfying the following two conditions :

i. If a terminating rule $X \rightarrow \theta$ is in P, then $\theta \in \mathcal{L}_G(X)$;

ii. If, for $1 \leq i \leq q$, $\theta_i \in \mathcal{L}_G(X_i)$, $X \rightarrow f[X_1, \ldots, X_q]$ is in P and $f(\theta_1, \ldots, \theta_q)$ is defined, then $f(\theta_1, \ldots, \theta_q) \in \mathcal{L}_G(X)$.

We denote the generalized context-free language (gcfl) generated by G by $\mathcal{L}(G)$, and we define $\mathcal{L}(G) = \mathcal{L}_G(S_0) \subseteq T^*$.

We define a *derivation tree* in a gcfg G as follows :

i. For a terminating rule $X \rightarrow \theta$, the tree whose root (labelled with X) has only one child (labelled with θ) is a derivation tree of θ;

ii. If t_i is a derivation tree of θ_i whose root is labelled with X_i (for $1 \leq i \leq q$), $X \rightarrow f[X_1, \ldots, X_q]$ is in P and $f[\theta_1, \ldots, \theta_q]$ is defined, then a derivation tree of $f[\theta_1, \ldots, \theta_q]$ is a tree such that

   (a) the root is labelled with X,

   (b) the root has q children, which are connected to it by edges labelled with f, and

   (c) the subtree rooted at the i^{th} child is isomorphic to t_i (for $1 \leq i \leq q$).

iii. Every derivation tree is built using conditions i. and ii. a finite number of times.

So derivation trees of gcfg's and those of cfg's are very much alike, except that the leaves of the former are tuples of strings while those of the latter are strings, and the former have edge labels indicating functions that have to be applied to tuples generated by nonterminals.

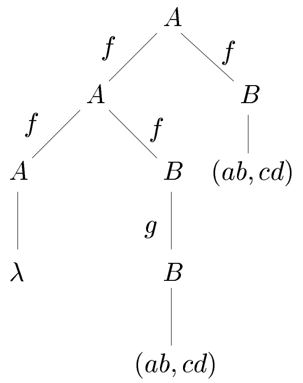
3.2 Parallel multiple context-free grammars

We will now define a class of gcfg's with a restricted semantic domain : the only operation that may be applied to the strings is concatenation. That makes the grammars in this class, the parallel multiple context-free grammars, very similar to the string-valued attribute grammars.

Definition 3.2
Let m be a positive integer. An *m-parallel multiple context-free grammar* (m-pmcfg or pmcfg) is an m-gcfg $G = (N, T, D, P, S_0)$, with $D = (O, F)$, which satisfies the following requirement : for each function $f \in F$, $f(\overline{x}_1, \ldots, \overline{x}_{a(f)})$ can be written as an element of $((V(f) \cup T)^*)^{r(f)}$. □

Example 3.1
The 2-pmcfg $G = (N, T, D, P, S_0)$, where $N = \{A, B\}$, $d(A) = 1$, $d(B) = 2$, $T = \{a, b, c, d\}$, $D = (\{T^*, T^* \times T^*\}, \{f, g\})$ with $f(x, (y, z)) = yxz$ and $g((y, z)) = (ayb, czd)$, $P = \{A \to f[A, B], A \to \lambda, B \to g[B], B \to (ab, cd)\}$ and $S_0 = A$, generates the language $L = \{a^{n_1} b^{n_1} \ldots a^{n_k} b^{n_k} c^{n_k} d^{n_k} \ldots c^{n_1} d^{n_1} \mid n_i \geq 1 \text{ for } 1 \leq i \leq k \text{ and } k \geq 0\}$.

The following tree is a derivation tree of $aba^2b^2c^2d^2cd \in L$. Instead of drawing nodes and putting the labels next to the nodes, we have replaced the nodes by their labels.



According to this tree, $\lambda \in \mathcal{L}_G(A)$ and $(ab, cd) \in \mathcal{L}_G(B)$. From the application of $B \to g[B]$, $(a^2b^2, c^2d^2) \in \mathcal{L}_G(B)$, and then, from the 'leftmost' application of $A \to f[A, B]$, $a^2b^2c^2d^2 \in \mathcal{L}_G(A)$. Finally, the 'topmost' application of $A \to f[A, B]$ gives $aba^2b^2c^2d^2cd \in \mathcal{L}_G(A)$. Consequently, $aba^2b^2c^2d^2cd \in \mathcal{L}(G)$ (and λ and $a^2b^2c^2d^2$ are in $\mathcal{L}(G)$ as well). □

When discussing ag's, we introduced a simpler, less formal, notation for describing them. For pmcfg's, we will use a similar notation : we will just give the rewriting rules, the descriptions of the functions that we use, and the initial symbol. The dimension of the nonterminals can be derived from the rules and the functions. This is illustrated in the following example.

Example 3.2
(1) The 2-pmcfg G of Example 3.1 can be described as follows :

$$\begin{array}{ll}
A \rightarrow f[A, B] & f(x, (y, z)) = yxz \\
A \rightarrow \lambda & \\
B \rightarrow g[B] & g((y, z)) = (ayb, czd) \\
B \rightarrow (ab, cd) &
\end{array}$$

(2) The language $\{ww \mid w \in \{a, b\}^*\}$ can be generated by the following 2-pmcfg :

$$\begin{array}{ll}
X \rightarrow f[Y] & f((x, y)) = xy \\
Y \rightarrow \alpha[Y] & \alpha((x, y)) = (ax, ay) \\
Y \rightarrow \beta[Y] & \beta((x, y)) = (bx, by) \\
Y \rightarrow (\lambda, \lambda) &
\end{array}$$

(3) A 4-pmcfg (with initial symbol X) that generates the language $\{ww^R ww^R \mid w \in \{a, b\}^*\}$ is the following :

$$\begin{array}{ll}
X \rightarrow f[Y] & f((u, x, y, z)) = uxyz \\
Y \rightarrow \alpha[Y] & \alpha((u, x, y, z)) = (au, xa, ay, za) \\
Y \rightarrow \beta[Y] & \beta((u, x, y, z)) = (bu, xb, by, zb) \\
Y \rightarrow (\lambda, \lambda, \lambda, \lambda) &
\end{array}$$

(4) Consider the following 5-pmcfg G_4 (with initial symbol X) :

$$\begin{array}{ll}
X \rightarrow f[Y] & f((u, v, w, x, y)) = uvwxy \\
Y \rightarrow g[Y] & g((u, v, w, x, y)) = (au, bv, cw, dx, ey) \\
Y \rightarrow (\lambda, \lambda, \lambda, \lambda, \lambda) &
\end{array}$$

G_4 generates the language $\{a^n b^n c^n d^n e^n \mid n \geq 0\}$.

(5) This sample 2-pmcfg (with initial symbol X) is from [SMFK]. It generates $\{a^{(n^2)} \mid n > 0\}$, based on the identity $(n + 1)^2 = n^2 + 2n + 1$.

$$\begin{array}{ll}
X \rightarrow f[Y] & f((x, y)) = axxy \\
Y \rightarrow g[Y] & g((x, y)) = (ax, axxy) \\
Y \rightarrow (\lambda, \lambda) &
\end{array}$$

(6) The language $\{a^{(2^n)} \mid n \geq 0\}$ can be generated by the following 1-pmcfg (also from [SMFK]) :

$$\begin{array}{ll}
X \rightarrow f[X] & f(x) = xx \\
X \rightarrow a &
\end{array}$$

$\square$

3.3 Results on pmcfg's

A pmcfg is very similar to an OnlyS-sag : they both generate strings and they both walk bottom-up through a derivation tree. Moreover, a nonterminal of rank n in a pmcfg is almost the same as a nonterminal of rank n in an OnlyS-sag, since the former generates tuples of dimension n, while the latter has n s-attributes. So the components of tuples in pmcfg's correspond to s-attributes in OnlyS-sag's, where the only difference is that tuples are ordered and sets of attributes are not (but we can add an ordering to a set). In the following theorem we will formalize this idea.

Since in an ag the initial symbol is not supposed to occur in the right-hand side of a production, there might be a problem when we want to translate a pmcfg G in which the initial symbol does occur in the right-hand side of a production to a sag. This problem, however, is easy to solve. For a pmcfg $G = (N, T, (O, F), P, S_0)$, with rank function $d : N \to \mathbf{N}$, an equivalent pmcfg that has the required property is $G' = (N \cup \{Z\}, T, (O, F \cup \{\mathrm{id}\}), P \cup \{Z \to \mathrm{id}[S_0]\}, Z)$, where Z is a new nonterminal with $d(Z) = 1$, and id is a new function with $\mathrm{id}(x) = x$ for all x.

Theorem 3.1
PMCFL = OnlyS-SAL
Proof
First we prove that for each m-pmcfg $G = (N, T, D, P, S_0)$ we can construct an OnlyS-sag $G' = ((N', T', P', S'_0), (\Omega, \Phi), B', R')$ such that $\mathcal{L}(G) = \mathcal{L}(G')$. We may assume that S_0 does not occur in the right-hand side of any rule in P.

Let $N' = N$, $S'_0 = S_0$, $\Omega = \{T^*\}$ and Φ consists of all derived functions of the free monoid T^*. Let N' have the same rank function d as N, so every $X \in N'$ will have $d(X)$ s-attributes : s_1 through $s_{d(X)}$. Then the attribute description B' of G' consists of $S'(X) = \{s_j \mid 1 \leq j \leq d(X)\}$ for each $X \in N'$, $I'(X) = \emptyset$ for each $X \in N'$, $\alpha'_0 = s_1$ and $W'(\alpha) = T^*$ for every $\alpha \in Att'$. To determine T' , P' and R', we have to consider each rule in P. Let $T' = P' = \emptyset$.

For each nonterminating rule $r : X_0 \to f[X_1, \ldots, X_k]$ in P, we add the production $p : X_0 \to f X_1 \ldots X_k$ to P', and the new terminal 'f' to T'. The semantic rules in $R'(p)$ are $\langle s_i, 0\rangle = f^i((\langle s_1, 1\rangle, \ldots, \langle s_{d(X_1)}, 1\rangle), \ldots, (\langle s_1, k\rangle, \ldots, \langle s_{d(X_k)}, k\rangle))$ for $1 \leq i \leq r(f)$. The terminal that is added to the production may look somewhat superfluous, but it is necessary to prevent us from adding the same production more than once, with different semantic rules.

For each terminating rule $r' : X_0 \to (v_1, \ldots, v_{d(X_0)})$ in P (where $v_i \in T^*$ for $1 \leq i \leq d(X_0)$) we add a *new* terminal 'r'' to T', and to P' we add the production $p' : X_0 \to r'$. The semantic rules for p' will be $\langle s_1, 0\rangle = v_1, \ldots, \langle s_{d(X_0)}, 0\rangle = v_{d(X_0)}$. Now $\mathcal{L}(G) = \mathcal{L}(G')$ and consequently PMCFL $\subseteq$ OnlyS-SAL.

Our next task is to prove that OnlyS-SAL $\subseteq$ PMCFL. Let $G = (G_0, D, B, R)$ be an OnlyS-sag, with $G_0 = (N, T, P, S_0)$, $D = (\{\Sigma^*\}, \Phi)$ for some alphabet Σ and G has rank function d. According to the remark just before Lemma 2.3 we may assume that $d(S_0) = 1$, and by Lemma 2.2 we assume that $d(X) > 0$ for every $X \in N$. For each terminating production $p : X_0 \to w$ in P, $R(p) = \{\langle s_1, 0\rangle = v_1, \ldots, \langle s_{d(X_0)}, 0\rangle = v_{d(X_0)}\}$, where $v_i \in \Sigma^*$ for $1 \leq i \leq d(X_0)$. For each nonterminating production q : $X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$, let

$\langle s_i, 0\rangle = f_i(\langle s_1, 1\rangle, \ldots, \langle s_{d(X_1)}, 1\rangle, \ldots, \langle s_1, k\rangle, \ldots, \langle s_{d(X_k)}, k\rangle) \in R(q)$
be the semantic rule for s_i (with $1 \leq i \leq d(X_0)$).
(It is possible to describe every function $f \in \Phi$ that occurs in $R(q)$ as a function that takes the attributes $\langle s_1, 1\rangle, \ldots, \langle s_{d(X_1)}, 1\rangle, \ldots, \langle s_1, k\rangle, \ldots, \langle s_{d(X_k)}, k\rangle$ of q in this order as its arguments, and of which the description of the result can be given as an element of $(A(q) \cup \Sigma)^*$. Note that not every argument has to be used in this description!)

Let $m = \max\{d(X) \mid X \in N\}$. We construct an m-pmcfg $G' = (N', T', D', P', S'_0)$, with $D' = (O', F')$, such that $\mathcal{L}(G') = \mathcal{L}(G)$. Let $N' = N$, where N and N' have the same rank function d, $T' = \Sigma$, $O' = \{(T')^*, ((T')^*)^2, \ldots, ((T')^*)^m\}$ and $S'_0 = S_0$. For a function $f \in \Phi$, that is described as above, we define $\overline{f}((x_{1,1}, \ldots, x_{1,j_1}), \ldots, (x_{n,1}, \ldots, x_{n,j_n}))$ to be $f(x_{1,1}, \ldots, x_{1,j_1}, \ldots, x_{n,1}, \ldots, x_{n,j_n})$ where $n \geq 0$ and $j_i \geq 1$ (for $1 \leq i \leq n$). For every nonterminating production $q : X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$ in P we add the following rule to $P' : X_0 \to \overline{f}[X_1, \ldots, X_k]$, where $\overline{f}^i = f_i$ for $1 \leq i \leq d(X_0)$, and we add $\overline{f}$ to F'. For every terminating production $p : X_0 \to w$ in P, we add the following rule to $P' : X_0 \to (v_1, \ldots, v_{d(X_0)})$. Now $\mathcal{L}(G') = \mathcal{L}(G)$ and thus OnlyS-SAL $\subseteq$ PMCFL.

Consequently PMCFL = OnlyS-SAL. □

Since we now know that the pmcfg is equivalent to the OnlyS-sag, we can bring every pmcfg in a structural normal form that corresponds to the one in Lemma 2.3 (Lemma 2.2 from [SMFK]). Note that it is important that the initial symbol of the pmcfg does not occur in the right-hand side of any rule!

Corollary 3.2
We can bring every pmcfg $G = (N, T, D, P, S_0)$ with $D = (O, F)$ and rank function d in structural normal form, i.e., we can adjust G in order to let it have the following properties :

i. For every $f \in F$, if $f(\overline{x}_1, \ldots, \overline{x}_{a(f)}) = (y_1, \ldots, y_{r(f)})$, then every $x \in V(f)$ appears at least once in $y_1 \ldots y_{r(f)}$ (the information-lossless condition).

ii. Every nonterminal that occurs in the left-hand side of a terminating rule has rank 1.

iii. For every nonterminating rule $X_0 \to f[X_1, \ldots, X_k]$ in P, the description of $f(\overline{x}_1, \ldots, \overline{x}_k)$ can be given as an element of $((V(f))^*)^{r(f)}$.

iv. For every nonterminal $X \neq S_0$, if $(\theta_1, \ldots, \theta_{d(X)}) \in \mathcal{L}_G(X)$, then $\theta_i \neq \lambda$ for $1 \leq i \leq d(X)$.

v. For every terminating rule $X \to \theta$, with $X \neq S_0$, $|\theta| = 1$. If $X = S_0$, then $|\theta|$ is 0 or 1.

□

3.4 Multiple context-free grammars

In pmcfg's, it is possible to use the elements of the generated tuples more than once to compute the elements of another tuple ('parallellism'). We will now define a class of pmcfg's in which this duplication cannot occur : the multiple context-free grammars.

Definition 3.3
Let m be a positive integer. An *m-multiple context-free grammar* (m-mcfg or mcfg) is an m-pmcfg $G = (N, T, D, P, S_0)$, with $D = (O, F)$, where all functions in $f \in F$ satisfy this condition : for each variable x in $V(f)$, the total number of occurrences of x in the description of f is at most one, i.e., each x is used at most once to describe the result of f. □

Example 3.3
(1) Here is an example of a 3-mcfg for the language $M = \{a^m b^n c^k \mid m, n, k \text{ odd }\}$ from Example 2.5. The grammar directly corresponds to the OnlyS-sag from that example.

$$\begin{array}{ll}
X \to f[Y] & f((x, y, z)) = axycz \\
Y \to \alpha[Y] & \alpha((x, y, z)) = (aax, y, z) \\
Y \to \beta[Y] & \beta((x, y, z)) = (x, bby, z) \\
Y \to \gamma[Y] & \gamma((x, y, z)) = (x, y, ccz) \\
Y \to (\lambda, b, \lambda) &
\end{array}$$

(2) The pmcfg's in Example 3.2 (1), (2), (3) and (4) are also mcfg's, but those in (5) and (6) are not, since in the description of the functions f and g in (5) and f in (6) the variable x is used more than once.

□

It is easy to see that (p)mcfg's are generalizations of cfg's. In a cfg $G = (N, T, P, S_0)$, a nonterminal X_0 that has $X_1, \ldots, X_k$ as its children will generate a string that is the concatenation of the strings generated by $X_1, \ldots, X_k$ (in this order!) and some terminal strings that can occur before, between and after the strings generated by $X_1, \ldots, X_k$: $X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$ (with $w_i \in T^*$ and $X_j \in N$). In a pmcfg, however, a nonterminal X_0 that has k children $X_1, \ldots, X_k$ can generate a tuple of strings, that are also concatenations of the strings in the tuples generated by $X_1, \ldots, X_k$ and terminals, but the order of the strings generated by the children may be disturbed, and these strings may be used any number of times. Since in an mcfg the strings generated by the children cannot be used more than once, mcfg's and cfg's are very much alike. In the next theorem we will prove that every cfg can be simulated by a 1-mcfg and vice versa.

Theorem 3.3
CFL = 1-MCFL
Proof
The easiest part of the proof is CFL $\subseteq$ 1-MCFL. Let $G = (N, T, P, S_0)$ be a cfg. We construct a 1-mcfg $G' = (N, T, D', P', S_0)$, with $D' = (O', F')$, and rank function d, from G as follows. Let $O' = \{T^*\}$ and, for every $X \in N'$, $d(X) = 1$. For each $p : X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$ in P we construct the following rule in P' : $X_0 \to f_p[X_1, \ldots, X_k]$ where $f_p(\overline{x}_1, \ldots, \overline{x}_k) = w_1 x_{1,1} w_2 \ldots w_k x_{k,1} w_{k+1}$. Let $F' = \{f_p \mid p \in P\}$. Now $\mathcal{L}(G) = \mathcal{L}(G')$ and thus CFL $\subseteq$ 1-MCFL.

Now we have to prove that 1-MCFL $\subseteq$ CFL. Let $G = (N, T, D, P, S_0)$, with $D = (O, F)$, be a 1-mcfg. We know that $O = \{T^*\}$ and that the rules in P are of the form $r : X_0 \to f[X_1, \ldots, X_k]$ with $k \geq 0$, where $f(x_{1,1}, \ldots, x_{k,1})$ can be written as an element of $(\{x_{1,1}, \ldots, x_{k,1}\} \cup T)^*$, i.e., $f(x_{1,1}, \ldots, x_{k,1}) = w_1 x_{i_1,1} w_2 \ldots w_t x_{i_t,1} w_{t+1}$ with $w_\ell \in T^*$

for $1 \leq \ell \leq t+1$, $i_j \in \{1, \ldots, k\}$ and $i_j = i_{j'} \Rightarrow j = j'$ (for $1 \leq j \leq t$ and $t \leq k$). We construct a cfg $G' = (N', T', P', S_0')$ with $\mathcal{L}(G') = \mathcal{L}(G)$ as follows : let $T' = T$ and $S_0' = S_0$. For each $r : X_0 \to f[X_1, \ldots, X_k]$ in P we construct the following production in $P' : X_0 \to w_1 X_{i_1} w_2 \ldots w_t X_{i_t} w_{t+1}$ and we add $X_{i_1}, \ldots, X_{i_t}$ and X_0 to N'. This proves 1-MCFL $\subseteq$ CFL.

Consequently CFL = 1-MCFL. □

We have already proven that the pmcfg is equivalent to the OnlyS-sag. Both formalisms have a restricted version (the mcfg and the ssag, respectively), in which duplication is prohibited. The only difference left between those two restricted versions is the fact that in mcfg's loss of information is possible, while in ssag's that is not allowed. In the next theorem we prove that that can be overcome with the help of the information-lossless condition of Corollary 3.2

Theorem 3.4
MCFL = SSAL
Proof
We know that each mcfg G is equivalent to an mcfg G' that satisfies the information-lossless condition of Corollary 3.2. This means that for G' the following is true : for each $f \in F'$ where $f(\overline{x}_1, \ldots, \overline{x}_{a(f)}) = (y_1, \ldots, y_{r(f)})$, every $x \in V(f)$ appears *exactly* once in $y_1 \cdots y_{r(f)}$.

According to Theorem 3.1, we can construct an OnlyS-sag G'' such that $\mathcal{L}(G') = \mathcal{L}(G'')$. It is clear from the proof of that theorem that for each production $p : X_0 \to f X_1 \ldots X_k$ in P'' the dependency graph $D(p)$ has the following property : for each s-attribute s of X_1 through X_k there is exactly one edge from s to some s-attribute of X_0. The designated attribute of an OnlyS-sag cannot have any outgoing edges. Hence the dependency graph of each derivation tree is an inverted tree.

On the other hand, it is clear from the proof of Theorem 3.1 that when an OnlyS-sag is special, the equivalent pmcfg that can be constructed is also an mcfg. □

Example 3.4
Using the construction in Theorem 3.1 we construct the following 2-mcfg G' for the language L given in Example 2.6(2).

$S \to f[X]$	$f((x,y)) = xy$
$X \to \beta[A]$	$\beta((x,y)) = (bx, cy)$
$X \to \alpha[B]$	$\alpha((x,y)) = (ax, cy)$
$X \to (\lambda, \lambda)$	
$A \to g[X, X]$	$g((x,y),(u,v)) = (xau, yv)$
$B \to h[X, X]$	$h((x,y),(u,v)) = (xbu, yv)$

□

Chapter 4

Relational grammars

4.1 Definitions and examples

We will give an adapted version (suggested by J. Engelfriet) of the definition of relational grammars from [GR]. Our relational grammars generate strings, while in [GR] trees are generated.

Let V be an infinite set of variables. We will use $x_1, x_2, x_3, \ldots, y_1, y_2, y_3, \ldots, x, y, z, \ldots, \overline{x}, \overline{y}, \overline{z}, \ldots$ as variable names.

Definition 4.1
Let T be an alphabet of terminal symbols, and let N be a ranked alphabet of nonterminal symbols with rank function $d : N \to \{1, 2, \ldots\}$. A *derivation tuple over N and T* is a tuple $((v_1, \ldots, v_k), R)$, with $k \geq 1$, that satisfies the following requirements :

i. $v_1, \ldots, v_k \in (T \cup V)^*$;

ii. R is a finite subset of NV^+ such that each element of R is of the form $Xx_1 \ldots x_n$ with $n = d(X)$ and the $x_1, \ldots, x_n$ are all distinct. Moreover, if $Xx_1 \ldots x_n$ and $Yy_1 \ldots y_m$ are in R, then $x_i \neq y_j$ (for $1 \leq i \leq n$, $1 \leq j \leq m$ and $m = d(Y)$);

iii. A variable occurs in $v_1 \ldots v_k$ if and only if it occurs in some element of R.

□

In [GR] derivation tuples are called 'parameterized relations'. We will however not use that name, since we think that 'derivation tuple' gives a better description of its use.

A derivation tuple is a 2-tuple $((v_1, \ldots, v_k), R)$. As we will see later in this chapter, the first component of this tuple, $(v_1, \ldots, v_k)$, indicates what has been derived so far. The second component is a set of elements of the form $Xx_1 \ldots x_{d(X)}$. Such an element indicates that the variables $x_1, \ldots, x_{d(X)}$ still have to be rewritten, with the help of a production with left-hand side X.

Moreover, a derivation of a relational grammar (defined below) starts with a derivation tuple, and in each derivation step a new derivation tuple is generated.

Note that, according to Definition 4.1, the variables that occur in R occur *at least* once in $v_1 \ldots v_k$, and that any variable occurring in $v_1 \ldots v_k$ appears *exactly* once in R.

The *rank* of a derivation tuple $((v_1, \ldots, v_k), R)$ is k. Two derivation tuples that differ only by the names of the variables are *isomorphic*. Two derivation tuples are *disjoint* if no variable occurs in both.

Definition 4.2
A *relational grammar* (rg) is a 4-tuple $G = (N, T, P, S_0)$, with

i. N is a ranked alphabet of *nonterminals*. The rank of each nonterminal is given by a total function $d : N \to \{1, 2, \ldots\}$;

ii. T is a finite set of *terminals*, and $N \cap T = \emptyset$;

iii. P is a finite set of *productions* of the form $X \to \rho$ where ρ is a derivation tuple over N and T with the same rank as $X \in N$;

iv. $S_0 \in N$ is the *initial* nonterminal.

□

A *derivation step* $\rho_1 \Rightarrow \rho_2$ between derivation tuples is defined as follows. Let $\rho_1 = ((v_1, \ldots, v_k), R)$ and let $Xx_1 \ldots x_n \in R$. Let $X \to \rho$ be a production in P and let $((w_1, \ldots, w_n), R')$ be an isomorphic copy of ρ that is disjoint with ρ_1. Then $\rho_1 \Rightarrow \rho_2$ where $\rho_2 = ((v'_1, \ldots, v'_k), (R \backslash \{Xx_1 \ldots x_n\}) \cup R')$ and $v'_i = v_i[x_1/w_1, \ldots, x_n/w_n]$, the result of substituting w_j for x_j in v_i (where $1 \leq i \leq k$ and $1 \leq j \leq m$).

A relational grammar G generates a set of m-tuples of terminal strings, where $m = d(S_0)$. To be precise, the *language generated by* G is defined to be $\mathcal{L}(G) = \{(v_1, \ldots, v_m) \mid ((x_1, \ldots, x_m), \{S_0 x_1 \ldots x_m\}) \Rightarrow^* ((v_1, \ldots, v_m), \emptyset)\}$.

This means that every derivation is started by the derivation tuple $((x_1, \ldots, x_m), \{S_0 x_1 \ldots x_m\})$. Then an appropriate production is applied to this derivation tuple, which gives us a new derivation tuple. The process of applying productions is executed until a derivation tuple $((v_1, \ldots, v_m), \emptyset)$ is encountered. Note that every derivation tuple in such a derivation has rank m.

Restriction 4.1
Since we are mainly interested in string languages, we will only consider rg's of which the initial nonterminal has rank 1.

We give an example of an rg. As we did with pmcfg's, we will omit the parentheses of tuples of dimension 1.

Example 4.1
Consider the relational grammar $G = (\{S, X\}, \{a\}, P, S_0)$, with $P = \{S_0 \to (axxy, \{Xxy\}), X \to ((ax, axxy), \{Xxy\}), X \to ((\lambda, \lambda), \emptyset)\}$. G generates the language $\{a^{(n^2)} \mid n > 0\}$ (see Example 3.2 (5)). A sample derivation of G is

$(s, \{S_0 s\}) \Rightarrow$
$(axxy, \{Xxy\}) \Rightarrow$
$(aa\overline{x}a\overline{x}a\overline{x}\,\overline{x}\,\overline{y}, \{X\overline{x}\,\overline{y}\}) \Rightarrow$
$(aaaxaaxaaxaxaxxy, \{Xxy\}) \Rightarrow$
$(aaaaaaaaa, \emptyset)$ □

From this example, it is clear that variables that occur in the second component of a derivation tuple may be used more than once in the first component. We have seen a similar property with ag's and pmcfg's. Now we can impose a restriction on relational grammars as we did on pmcfg's and ag's : we prohibit duplication of variables.

Definition 4.3
A *special* relational grammar (srg) is a relational grammar of which every derivation tuple $((v_1, \ldots, v_k), R)$ in the right-hand side of a production satisfies the following additional requirement : each variable in V occurs at most once in $v_1 \ldots v_k$. □

This means that every variable that occurs in an element of R occurs *exactly* once in $v_1 \ldots v_k$.

Example 4.2
Consider the special relational grammar $G = (N, T, P, S_0)$ where
$N = \{A, B\}$,
$T = \{a, b, c, d\}$,
$P = \{A \rightarrow (\lambda, \emptyset)$,
$\quad A \rightarrow (xyz, \{Bxz, Ay\})$,
$\quad B \rightarrow ((ab, cd), \emptyset)$,
$\quad B \rightarrow ((axb, czd), \{Bxz\})\}$, and
$S_0 = A$.
G generates the language $\{a^{n_1}b^{n_1} \ldots a^{n_k}b^{n_k}c^{n_k}d^{n_k} \ldots c^{n_1}d^{n_1} \mid n_i \geq 0$ for $1 \leq i \leq k$ and $k \geq 0\}$.
A sample derivation of G is

$$
\begin{aligned}
&(y, \{Ay\}) \Rightarrow \\
&(x\overline{y}z, \{Bxz, A\overline{y}\}) \Rightarrow \\
&(ab\overline{y}cd, \{A\overline{y}\}) \Rightarrow \\
&(abxyzcd, \{Ay, Bxz\}) \Rightarrow \\
&(aba\overline{x}byc\overline{z}dcd, \{Ay, B\overline{x}\,\overline{z}\}) \Rightarrow \\
&(aba^2b^2yc^2d^2cd, \{Ay\}) \Rightarrow \\
&(aba^2b^2x\overline{y}zc^2d^2cd, \{A\overline{y}, Bxz\}) \Rightarrow \\
&(aba^2b^2ab\overline{y}cdc^2d^2cd, \{A\overline{y}\}) \Rightarrow \\
&(aba^2b^2abcdc^2d^2cd, \emptyset),
\end{aligned}
$$

where x, y, z, $\overline{x}$, $\overline{y}$, and $\overline{z} \in V$. □

Both our rg and srg satisfy an information-lossless condition, because of the third requirement of our definition of derivation tuples. We could however define a derivation tuple in which loss of information is possible, by changing the condition in Definition 4.1 iii to 'if a variable occurs in $v_1 \ldots v_k$, then it occurs in some element of R'. And then we could prove that, for every rg (or srg), we can construct an equivalent rg (or srg) that satisfies an information-lossless condition similar to the ones mentioned in Lemma 2.3 and Corollary 3.2.

4.2 Results

4.2.1 Equivalence of rg and OnlyS-sag

We can look at an rg as if it was an OnlyS-sag, as follows. In a derivation tuple $((v_1, \ldots, v_k), R)$, the elements of R denote nonterminals with their s-attributes. A production $X \to ((w_1, \ldots, w_\ell), R')$ states that X has to be rewritten as the sequence of nonterminals that occur in R' (in an arbitrary order, because R' is a set), and that the ℓ s-attributes of X are defined by $w_1, \ldots, w_\ell$, respectively. Since the variables that occur in R' and also in $w_1 \ldots w_\ell$ represent the s-attributes of the nonterminals in R', we can say that the s-attributes of X are defined in terms of the s-attributes of its children. The translation of an OnlyS-sag into an rg is analogous.

Theorem 4.1
RL = OnlyS-SAL.
Proof
First, we prove that RL $\subseteq$ OnlyS-SAL.
Let $G = (N, T, P, S_0)$ be an rg with rank function $d : N \to \{1, 2, \ldots\}$. We will construct an equivalent OnlyS-sag $G' = ((N', T', P', S_0'), (\{T^*\}, \Phi), B', R')$ with rank function $d' : N \to \mathbf{N}$.

Let $N' = N \cup \{S_0'\}$, S_0' is a new nonterminal, Φ consists of all derived functions of the free monoid T^*, $B' = (S, I, \alpha_0, W)$ with $S(X) = \{s_1, \ldots, s_{d'(X)}\}$ for all $X \in N$, $I(X) = \emptyset$ for all $X \in N$, $S(S_0') = \{\alpha_0\}$, $I(S_0') = \emptyset$, $W(s) = T^*$ for all $s \in S\text{-}Att'$, $d'|_N = d$ and $d'(S_0') = 1$. Let $P' = \emptyset$ and $T' = \emptyset$.

Consider a production $p : X \to ((v_1, \ldots, v_{d(X)}), R)$ in P, and $R = \{X_1 x_{1,1} \ldots x_{1,n_1}, X_2 x_{2,1} \ldots x_{2,n_2}, \ldots, X_\ell x_{\ell,1} \ldots x_{\ell,n_\ell}\}$ (with $\ell \geq 0$ and $n_i = d(X_i)$ for $1 \leq i \leq \ell$). We construct a production $q : X \to p X_1 X_2 \ldots X_\ell$ in P', with $R'(q) = \{\langle s_1, 0\rangle = v_1[x_{i,j}/\langle s_j, i\rangle], \ldots, \langle s_{d(X)}, 0\rangle = v_{d(X)}[x_{i,j}/\langle s_j, i\rangle]\}$ for $1 \leq i \leq \ell$ and $1 \leq j \leq n_i$, and we add p to T'. We execute this procedure for every $p \in P$.

Furthermore, to make sure that S_0' does not occur in the right-hand side of any production in P', we add to P' the production $S_0' \to S_0$ with semantic rule $\langle \alpha_0, 0\rangle = \langle s_1, 1\rangle$. Now $\mathcal{L}(G') = \mathcal{L}(G)$.

Consequently, RL $\subseteq$ OnlyS-SAL.

We still have to prove that OnlyS-SAL $\subseteq$ RL.
Let $G = ((N, T, P, S_0), (\{\Sigma^*\}, \Phi), (S, I, \alpha_0, W), R)$ be an OnlyS-sag, for some alphabet Σ, and with rank function d. According to Lemma 2.2 we may assume that $d(X) > 0$ for all $X \in N$, and according to the remark just before Lemma 2.3 we may assume that $d(S_0) = 1$. We construct an equivalent rg $G' = (N, \Sigma, P', S_0)$, with the same rank function, as follows.

For every production $p : X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$ in P with semantic rules $\langle s_i, 0\rangle = u_{i,1}\langle a_{i,1}, j_{i,1}\rangle u_{i,2} \ldots u_{i,n_i}\langle a_{i,n_i}, j_{i,n_i}\rangle u_{i,n_i+1}$ for $1 \leq i \leq d(X_0)$, and with $S(X_m) = \{s_1, \ldots, s_{d(X_m)}\}$ for $0 \leq m \leq k$, we add to P' a production

$$\begin{aligned} q : X_0 \to ((\ & u_{1,1}\langle a_{1,1}, j_{1,1}\rangle u_{1,2} \ldots u_{1,n_1}\langle a_{1,n_1}, j_{1,n_1}\rangle u_{1,n_1+1}, \\ & \ldots, \\ & u_{\ell,1}\langle a_{\ell,1}, j_{\ell,1}\rangle u_{\ell,2} \ldots u_{\ell,n_\ell}\langle a_{\ell,n_\ell}, j_{\ell,n_\ell}\rangle u_{\ell,n_\ell+1}), \\ & \{X_1\langle s_1, 1\rangle\langle s_2, 1\rangle \ldots \langle s_{d(X_1)}, 1\rangle, \ldots, X_k\langle s_1, k\rangle\langle s_2, k\rangle \ldots \langle s_{d(X_k)}, k\rangle\}\ ) \end{aligned}$$

for $\ell = d(X_0)$. We let $\langle s, j\rangle \in V$ for $1 \leq j \leq k$ and $s \in S(X_j)$. Now $\mathcal{L}(G') = \mathcal{L}(G)$, and thus OnlyS-SAL $\subseteq$ RL.

Consequently RL = OnlyS-SAL. □

Corollary 4.2
RL = PMCFL. □

If the original rg is special, then the corresponding OnlyS-sag will be special as well, since from the restrictions on the derivation tuples of srg's (all the variables that occur in R' are distinct, and they all occur exactly once in $w_1 \ldots w_\ell$) we see that every s-attribute of a child is used exactly once to define the s-attributes of X. Furthermore, the s-attribute (recall that $d(S_0) = 1$) of the occurrence of S_0 at the root of a derivation tree can never be used to define another attribute, since the derivation tuple $(z, \{S_0 z\})$ starts the derivation. Thus the dependency graph of each derivation tree of such an OnlyS-sag is an inverted tree, and consequently the OnlyS-sag is special.

Theorem 4.3
SRL = SSAL
Proof
Assume that we use the construction described in Theorem 4.1 to translate a special rg into an OnlyS-sag. Because of the restrictions on the derivation tuples of the srg (for a derivation tuple $((w_1, \ldots, w_k), C)$, all the variables that occur in C are distinct, and they all occur exactly once in $w_1 \ldots w_k$) it is obvious that the resulting OnlyS-sag is special.

The translation from OnlyS-sag into rg gives an srg if the OnlyS-sag is special, since then the tuples in the right-hand sides of the constructed productions obviously satisfy the requirements for derivation tuples given in Definition 4.1 and the additional condition of Definition 4.3. □

Corollary 4.4
SRL = MCFL. □

Example 4.3
(1) An ssag corresponding to the srg of Example 4.2 is

$S_0 \to A$	$\alpha_0 = \alpha_1$	
$A \to p_1$	$\alpha_0 = \lambda$	
$A \to p_2 AB$	$\alpha_0 = \beta_2 \alpha_1 \gamma_2$	
$B \to p_3$	$\beta_0 = ab$	$\gamma_0 = cd$
$B \to p_4 B$	$\beta_0 = a\beta_1 b$	$\gamma_0 = c\gamma_1 d$

with designated attribute α.

(2) An srg that is equivalent to the ssag of Example 2.6 (2) is the following :

$S_0 \to (\omega\gamma, \{X\omega\gamma\})$
$X \to ((b\omega, c\gamma), \{A\omega\gamma\})$
$X \to ((a\omega, c\gamma), \{B\omega\gamma\})$

$X \to ((\lambda, \lambda), \emptyset)$
$A \to ((\omega_1 a \omega_2, \gamma_1 \gamma_2), \{X \omega_1 \gamma_1, X \omega_2 \gamma_2\})$
$B \to ((\omega_1 b \omega_2, \gamma_1 \gamma_2), \{X \omega_1 \gamma_1, X \omega_2 \gamma_2\})$

A sample derivation is

$(s, \{S_0 s\}) \Rightarrow$
$(\omega\, \gamma, \{X \omega \gamma\}) \Rightarrow$
$(a\, \overline{\omega}\, c\, \overline{\gamma}, \{B \overline{\omega}\, \overline{\gamma}\}) \Rightarrow$
$(a\, \omega_1\, b\, \omega_2\, c\, \gamma_1\, \gamma_2, \{X \omega_1 \gamma_1, X \omega_2 \gamma_2\}) \Rightarrow$
$(aa\, \omega\, b\, \omega_2\, cc\, \gamma\, \gamma_2, \{B \omega \gamma, X \omega_2 \gamma_2\}) \Rightarrow$
$(aa\, \omega_1\, b\, \overline{\omega}\, b\, \omega_2\, cc\, \gamma_1\, \overline{\gamma}\, \gamma_2, \{X \omega_2 \gamma_2, X \omega_1 \gamma_1, X \overline{\omega}\, \overline{\gamma}\}) \Rightarrow^*$
$(aabbcc, \emptyset)$.

(3) The following srg, with initial symbol X, generates the same language as the 3-mcfg from Example 3.3 (1).

$X \to (axycz, \{Yxyz\})$
$Y \to ((aax, y, z), \{Yxyz\})$
$Y \to ((x, bby, z), \{Yxyz\})$
$Y \to ((x, y, ccz), \{Yxyz\})$
$Y \to ((\lambda, b, \lambda), \emptyset)$

□

4.2.2 Correctness of the construction

Throughout this thesis, we give several constructions to transform one grammar formalism into another. We have however never given a formal proof of the correctness of those constructions. In this subsection we will give such a proof for the construction of translating an rg into an OnlyS-sag. This is the only correctness proof that we will give in this thesis, since the other ones are analogous.

To prove the correctness of the construction described in Theorem 4.1, we need to define the language generated by a nonterminal in an rg and an OnlyS-sag, respectively. For rg's, this definition is simple.

Definition 4.4
Let $G = (N, T, P, S_0)$ be an rg with rank function $d : N \to \{1, 2, \ldots\}$.

For every $X \in N$, the *language generated by* X *in* G, $\mathcal{L}_G(X) \subseteq (T^*)^{d(X)}$, is defined to be $\mathcal{L}_G(X) = \{(v_1, \ldots, v_{d(X)}) \mid ((x_1, \ldots, x_{d(X)}), \{X x_1 \ldots x_{d(X)}\}) \Rightarrow^* ((v_1, \ldots, v_{d(X)}), \emptyset)\}$.
□

Consequently, $\mathcal{L}_G(S_0) = \mathcal{L}(G)$.

For OnlyS-sag's, the definition is somewhat more complicated, since the nonterminals have sets of attributes (i.e., the attributes are not ordered). Therefore we use functions to associate values with the attributes, as follows.

Definition 4.5
Let $G = (G_0, D, B, R)$ be an OnlyS-sag, with $G_0 = (N, T, P, S_0)$, $D = (\{\Sigma^*\}, \Phi)$, $B = (S, I, \alpha_0, W)$ and rank function $d : N \to \mathbf{N}$.

For every $X \in N$, the *language generated by X in G*, $\mathcal{L}_G(X) \subseteq (\Sigma^*)^{S(X)}$, is defined to be $\mathcal{L}_G(X) = \{\sigma \mid \sigma : S(X) \to \Sigma^*$ is a total function, and there is a derivation tree t of G of which the root r is labelled with X, such that $\text{val}(\langle \alpha, r\rangle) = \sigma(\alpha)$ for every $\alpha \in S(X)$, and val is a correct decoration of $t\}$.

Furthermore, for every subset $U \neq \emptyset$ of $S(X)$, we define $\mathcal{L}_G(X)|_U$ to be the restriction of $\mathcal{L}_G(X)$ to U, i.e., $\mathcal{L}_G(X)|_U = \{\sigma|_U \mid \sigma \in \mathcal{L}_G(X)\}$. □

Now $\mathcal{L}_G(S_0)|_{\{\alpha_0\}} = \mathcal{L}(G)$.

Let $G = (N, T, P, S_0)$ be an rg with rank function $d : N \to \{1, 2, \ldots\}$. Let $G' = (G'_0, D', B', R')$ be an OnlyS-sag that is created using the construction described in Theorem 4.1, with $G'_0 = (N', T', P', S'_0)$, $D' = (\{T^*\}, \Phi)$, $B' = (S, I, \alpha_0, W)$ and rank function d', where $d'|_N = d$ and $d'(S'_0) = 1$. Note that $N' = N \cup \{S'_0\}$. Let $S(X) = \{s_1, \ldots, s_{d(X)}\}$ for all $X \in N$.

In the proof of the following lemma we will use the fact that, for a derivation $\alpha : ((x_1, \ldots, x_{d(X)}), \{Xx_1 \ldots x_{d(X)}\}) \Rightarrow ((w_1, \ldots, w_{d(X)}), \{X_1 x_{1,1} \ldots x_{1,d(X_1)}, \ldots, X_\ell x_{\ell,1} \ldots x_{\ell,d(X_\ell)}\}) \Rightarrow^* ((v_1, \ldots, v_{d(X)}), \emptyset)$ of G, there are unique derivations $\alpha_j : ((x_{j,1}, \ldots, x_{j,d(X_j)}), \{X_j x_{j,1} \ldots x_{j,d(X_j)}\}) \Rightarrow^* ((u_{j,1}, \ldots, u_{j,d(X_j)}), \emptyset)$ that are 'part' of α in the sense that some reordering of them is precisely α, and such that $v_i = w_i[x_{j,z}/u_{j,z}]$ for $1 \leq i \leq d(X)$, $1 \leq j \leq \ell$, $\ell \geq 0$ and $1 \leq z \leq d(X_j)$.

Lemma 4.5
For all $X \in N$, $(v_1, \ldots, v_{d(X)}) \in \mathcal{L}_G(X)$ iff $\sigma \in \mathcal{L}_{G'}(X)$ with $\sigma(s_i) = v_i$ for $1 \leq i \leq d(X)$. In particular, $v \in \mathcal{L}_G(S_0)$ iff $\sigma \in \mathcal{L}_{G'}(S'_0)$ with $\sigma(\alpha_0) = v$, and consequently $\mathcal{L}(G) = \mathcal{L}(G')$.

Proof
"only if" (with induction to the length of the derivation in G)

- (basis)
  Consider a derivation of length 1 of $(v_1, \ldots, v_{d(X)}) \in \mathcal{L}_G(X)$, $((x_1, \ldots, x_{d(X)}), \{Xx_1 \ldots x_{d(X)}\}) \Rightarrow ((v_1, \ldots, v_{d(X)}), \emptyset)$. This means that the production $p : X \to ((v_1, \ldots, v_{d(X)}), \emptyset)$ is in P. Then, following the construction given in Theorem 4.1, the production $q : X \to p$ is in P', with $R'(q) = \{\langle s_1, 0\rangle = v_1, \ldots, \langle s_{d(X)}, 0\rangle = v_{d(X)}\}$. And thus $\sigma \in \mathcal{L}_{G'}(X)$, where $\sigma(s_i) = v_i$ for $1 \leq i \leq d(X)$.

- (induction hypothesis)
  For all $X \in N$, if $(v_1, \ldots, v_{d(X)}) \in \mathcal{L}_G(X)$ and there is a derivation $((x_1, \ldots, x_{d(X)}), \{Xx_1 \ldots x_{d(X)}\}) \Rightarrow^n ((v_1, \ldots, v_{d(X)}), \emptyset)$ with $n \geq 1$, then $\sigma \in \mathcal{L}_{G'}(X)$ with $\sigma(s_i) = v_i$ for $1 \leq i \leq d(X)$.

- (induction step)
  Consider the derivation $\alpha : ((x_1, \ldots, x_{d(X)}), \{Xx_1 \ldots x_{d(X)}\}) \Rightarrow^{n+1} ((v_1, \ldots, v_{d(X)}), \emptyset)$ of $(v_1, \ldots, v_{d(X)}) \in \mathcal{L}_G(X)$. Let the first step of α be $((x_1, \ldots, x_{d(X)}), \{Xx_1 \ldots x_{d(X)}\}) \Rightarrow ((w_1, \ldots, w_{d(X)}), \{X_1 x_{1,1} \ldots x_{1,d(X_1)}, \ldots, X_\ell x_{\ell,1} \ldots x_{\ell,d(X_\ell)}\})$ using a production $p : X \to ((w_1, \ldots, w_{d(X)}), \{X_1 x_{1,1} \ldots x_{1,d(X_1)}, \ldots, X_\ell x_{\ell,1} \ldots x_{\ell,d(X_\ell)}\})$ from P. Since the derivations for $X_1, \ldots, X_\ell$ that are 'part' of α have length $\leq n$, we know by induction that, for $X_1, \ldots, X_\ell$, if $(u_{j,1}, \ldots, u_{j,d(X_j)}) \in \mathcal{L}_G(X_j)$, then $\sigma_j \in \mathcal{L}_{G'}(X_j)$, where $\sigma_j(s_z) = u_{j,z}$ for $1 \leq j \leq \ell$ and $1 \leq z \leq$

$d(X_j)$. According to the construction given in Theorem 4.1, p is translated to $q : X \to pX_1 \ldots X_\ell$ in P', with $R'(q) = \{\langle s_1, 0\rangle = w_1[x_{j,z}/\langle s_z, j\rangle], \ldots, \langle s_{d(X)}, 0\rangle = w_{d(X)}[x_{j,z}/\langle s_z, j\rangle]\}$. Now we can say that $\sigma \in \mathcal{L}_{G'}(X)$, where

$$\begin{aligned}\sigma(s_i) &= w_i[x_{j,z}/\sigma(\langle s_z, j\rangle)]\\ &= w_i[x_{j,z}/u_{j,z}]\\ &= v_i\end{aligned}$$

for $1 \le i \le d(X)$.

Moreover, the production $S'_0 \to S_0$ with semantic rule $\langle \alpha_0, 0\rangle = \langle s_1, 1\rangle$ is in P'. If $v \in \mathcal{L}_G(S_0)$, then $\sigma \in \mathcal{L}_{G'}(S_0)$ with $\sigma(s_1) = v$, as we just proved.

Let t be a derivation tree of $S_0 \Rightarrow^* w$ (for some $w \in (T')^*$) with $\text{val}(\langle s_1, \text{root}(t)\rangle) = v$, for a correct decoration val of t. Then we can construct a derivation tree t' of $S'_0 \Rightarrow S_0 \Rightarrow^* w$ with $\text{val}(\langle \alpha_0, \text{root}(t')\rangle) = v$, and thus $\sigma \in \mathcal{L}_{G'}(S'_0)$ with $\sigma(\alpha_0) = v$.

"if" (with induction to the depth of the correctly decorated derivation tree of G')

- (basis)
  Consider $\sigma \in \mathcal{L}_{G'}(X)$, with $\sigma(s_i) = v_i$ for $1 \le i \le d(X)$. If there is a derivation tree t of depth 1 of G' of which the root r is labelled with X, and with $\text{val}(\langle s_1, r\rangle) = v_i$ and val is a correct decoration of t, then apparently a production $p : X \to w$ with semantic rules $\langle s_1, 0\rangle = v_1, \ldots, \langle s_{d(X)}, 0\rangle = v_{d(X)}$ is in P'. According to the construction of Theorem 4.1, this production is a translation from a production $q : X \to ((v_1, \ldots, v_{d(X)}), \emptyset)$ in P, and thus $(v_1, \ldots, v_{d(X)}) \in \mathcal{L}_G(X)$.

- (induction hypothesis)
  For all $X \in N$, if $\sigma \in \mathcal{L}_{G'}(X)$ with $\sigma(s_i) = v_i$ for $1 \le i \le d(X)$ and X is the label of the root r of a derivation tree t of depth $n \ge 1$ of G', with $\text{val}(\langle s_i, r\rangle) = v_i$ and val is a correct decoration of t, then $(v_1, \ldots, v_{d(X)}) \in \mathcal{L}_G(X)$.

- (induction step)
  Consider the occurrence of $q : X \to pX_1 \ldots X_k$ from P' (for $k \ge 0$), with $R'(q) = \{\langle s_i, 0\rangle = u_{i,1}\langle s_{i,1}, j_{i,1}\rangle u_{i,2} \ldots u_{i,n_i}\langle s_{i,n_i}, j_{i,n_i}\rangle u_{i,n_i+1} \mid 1 \le i \le d(X)\}$, at the root r, labelled with X, of a derivation tree t of G' with depth $n+1$, with $\text{val}(\langle s_i, r\rangle) = v_i$ for $1 \le i \le d(X)$, and val is a correct decoration of t. Then $\sigma \in \mathcal{L}_{G'}(X)$ with $\sigma(s_i) = v_i$.

  By induction, since the subtrees of t of which the nonterminal children of r are the roots (that are labelled with $X_1, \ldots, X_k$, respectively) have depth $\le n$, the following holds : if $\sigma_j \in \mathcal{L}_{G'}(X_j)$ with $\sigma_j(s_\ell) = v_{j,\ell}$ for $1 \le j \le k$ and $1 \le \ell \le d(X_j)$, then $(v_{j,1}, \ldots, v_{j,d(X_j)}) \in \mathcal{L}_G(X_j)$.

  Since q is in P', apparently a production

$$\begin{aligned}p : X \to ((\ &(u_{1,1}\langle s_{1,1}, j_{1,1}\rangle u_{1,2} \ldots u_{1,n_1}\langle s_{1,n_1}, j_{1,n_1}\rangle u_{1,n_1+1})[\langle s_\ell, j\rangle/x_{j,\ell}],\\ &\ldots,\\ &(u_{m,1}\langle s_{m,1}, j_{m,1}\rangle u_{m,2} \ldots u_{m,n_m}\langle s_{m,n_m}, j_{m,n_m}\rangle u_{m,n_m+1})[\langle s_\ell, j\rangle/x_{j,\ell}]\ ),\\ &\{X_1x_{1,1} \ldots x_{1,d(X_1)}, \ldots, X_kx_{k,1} \ldots x_{k,d(X_k)}\}\ )\end{aligned}$$

where $m = d(X)$, is in P. And since $(v_{j,1}, \ldots, v_{j,d(X_j)}) \in \mathcal{L}_G(X_j)$, now

$$\begin{array}{l}(\ (u_{1,1}\langle s_{1,1}, j_{1,1}\rangle u_{1,2} \ldots u_{1,n_1}\langle s_{1,n_1}, j_{1,n_1}\rangle u_{1,n_1+1})[\langle s_\ell, j\rangle / v_{j,\ell}], \\ \quad \ldots, \\ \quad (u_{m,1}\langle s_{m,1}, j_{m,1}\rangle u_{m,2} \ldots u_{m,n_m}\langle s_{m,n_m}, j_{m,n_m}\rangle u_{m,n_m+1})[\langle s_\ell, j\rangle / v_{j,\ell}] \ ) = \\ \quad (v_1, \ldots, v_{d(X)}) \in \mathcal{L}_G(X)\end{array}$$

Moreover, if $\sigma \in \mathcal{L}_{G'}(S_0')$ with $\sigma(\alpha_0) = v$, then also $\sigma \in \mathcal{L}_{G'}(S_0)$, since $p : S_0' \to S_0$ is the only production with left-hand side S_0' in P', and since $R'(p) = \{\langle \alpha_0, 0\rangle = \langle s_1, 1\rangle\}$. And then, as we just proved, $v \in \mathcal{L}_G(S_0)$. □

Since the construction for the translation of an rg into an OnlyS-sag introduces a new nonterminal S_0', we should also prove that $v \in \mathcal{L}(G)$ iff $\sigma \in \mathcal{L}_{G'}(S_0')$, with $\sigma(\alpha_0) = v$. This is obvious from the construction.

Chapter 5

Top-down tree-to-string transducers

5.1 Preliminaries

We will use an alphabet Σ with rank function $d : \Sigma \to \mathbf{N}$, and by Σ_n we denote the set $\{\sigma \mid \sigma \in \Sigma \text{ and } d(\sigma) = n\}$. A *tree over* Σ is either a symbol of rank 0 or a string of the form $\sigma(t_1 \dots t_n)$, where σ has rank n, with $n \geq 1$, and t_i is a tree over Σ (for $1 \leq i \leq n$). The set of all trees over Σ is denoted T_Σ, thus $T_\Sigma \subseteq (\Sigma \cup \{(,)\})^*$. A *tree language* over Σ is a subset of T_Σ.

A tree language is *recognizable* if it can be accepted by a finite tree automaton (see [T73]). The class of recognizable tree languages will be denoted by REC. (For properties of REC see for instance [T73], [T67], [TW] and [E75].)

We use $V = \{x_1, x_2, x_3, \dots\}$ as a denumerably infinite set of variables, $V_0 = \emptyset$ and, for $n \geq 1$, $V_n = \{x_1, \dots, x_n\}$. In examples we will use $x, y, z, \dots$ rather than $x_1, x_2, x_3, \dots$.

For an alphabet Ψ and strings $w_0 \in (\Psi \cup V_n)^*$ and $w_1, \dots, w_n \in \Psi^*$ (for $n \geq 0$), $w_0[w_1, \dots, w_n]$ denotes the result of substituting w_i for x_i in w_0 (where $1 \leq i \leq n$).

5.2 Definitions

We will now give the definitions concerning top-down tree-to-string transducers, that we have found in [ERS].

A top-down tree-to-string transducer translates an input tree over some ranked alphabet into a string, by walking top-down through the input tree and meanwhile making translations of the subtrees of the input tree. During this top-down walk through the input tree, several not necessarily distinct translations can be made of one subtree, and there can be subtrees of which no translation is made (so those subtrees are discarded).

Definition 5.1
A *top-down tree-to-string transducer* (yT transducer) is a construct $M = (Q, \Sigma, \Delta, q_0, R)$, where

i. Q is a finite set of *states*;

ii. Σ is the ranked *input alphabet*;

iii. Δ is the *output alphabet*;

iv. q_0 is the *initial state*;

v. R is a finite set of *(transducer) rules* of the form
$q(\sigma(x_1 \ldots x_k)) \to w_1 q_1(x_{i_1}) w_2 \ldots w_n q_n(x_{i_n}) w_{n+1}$
with $n, k \geq 0$; $\sigma \in \Sigma_k$; $q, q_1, \ldots, q_n$ are (not necessarily distinct) elements of Q; $w_1, \ldots, w_{n+1} \in \Delta^*$, and $1 \leq i_m \leq k$ for $1 \leq m \leq n$.

M is *deterministic* if different rules in R have different left-hand sides.

□

Let $M = (Q, \Sigma, \Delta, q_0, R)$ be a yT transducer.

Let $Q(S) = \{q(s) \mid q \in Q, s \in S\}$ for some set S. A *sentential form* of M is an element of $(Q(T_\Sigma) \cup \Delta)^*$, i.e., a string of the form $u_1 p_1(t_1) u_2 p_2(t_2) \ldots u_m p_m(t_m) u_{m+1}$ with $m \geq 0$, $p_i \in Q$, $t_i \in T_\Sigma$, $1 \leq i \leq m$ and, for $1 \leq j \leq m+1$, $u_j \in \Delta^*$.

For sentential forms s_1 and s_2 we write $s_1 \Rightarrow s_2$ if s_2 is obtained from s_1 by replacing a substring $q(\sigma(t_1 \ldots t_k))$ of s_1, for certain $t_1, \ldots, t_k \in T_\Sigma$, by $w_1 q_1(t_{i_1}) w_2 q_2(t_{i_2}) \ldots w_n q_n(t_{i_n}) w_{n+1}$, using the rule in Definition 5.1 v. As usual, $\Rightarrow^*$ is used to denote *derivations*, i.e., the reflexive and transitive closure of $\Rightarrow$.

The (tree-to-string) *translation defined by* M, also denoted by M, is defined by $M = \{\langle t, w\rangle \in T_\Sigma \times \Delta^* \mid q_0(t) \Rightarrow^* w\}$. We define the *language generated by* M to be $\mathcal{L}(M) = \{w \in \Delta^* \mid q_0(t) \Rightarrow^* w \text{ for some } t \in T_\Sigma\}$.

The class of (deterministic) top-down tree-to-string transducers will be denoted by yT (yDT). We will denote the class of languages they generate by yTL (yDTL).

Example 5.1
(1) We have found this example in [ERS].
Consider the yT transducer $M_1 = (\{q_0, q_1, q_2\}, \{\sigma, \tau, \delta\}, \{a, b, c, d\}, q_0, R_1)$ such that σ, τ and δ have ranks 2, 1 and 0, respectively, and R_1 consists of the rules
$q_0(\sigma(xy)) \to q_1(x) q_0(y) q_2(x)$,
$q_0(\delta) \to \lambda$,
$q_1(\tau(x)) \to a q_1(x) b$,
$q_1(\delta) \to \lambda$,
$q_2(\tau(x)) \to c q_2(x) d$,
$q_2(\delta) \to \lambda$
M_1 translates a tree of the form $\sigma(\tau^{n_1}(\delta)\sigma(\tau^{n_2}(\delta) \cdots \sigma(\tau^{n_k}(\delta)\delta) \cdots))$ into the string $a^{n_1} b^{n_1} a^{n_2} b^{n_2} \ldots a^{n_k} b^{n_k} c^{n_k} d^{n_k} \ldots c^{n_2} d^{n_2} c^{n_1} d^{n_1}$. The language generated by M_1 is $\mathcal{L}(M_1) = \{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \ldots a^{n_k} b^{n_k} c^{n_k} d^{n_k} \ldots c^{n_2} d^{n_2} c^{n_1} d^{n_1} \mid n_i \geq 0 \text{ for } 1 \leq i \leq k \text{ and } k \geq 0\}$ (see Example 4.2).
A sample derivation in M_1 is
$q_0(\sigma(\tau^2(\delta)\sigma(\tau(\delta)\delta))) \Rightarrow$
$q_1(\tau^2(\delta)) q_0(\sigma(\tau(\delta)\delta)) q_2(\tau^2(\delta)) \Rightarrow^*$
$a q_1(\tau(\delta)) b q_1(\tau(\delta)) q_0(\delta) q_2(\tau(\delta)) c q_2(\tau(\delta)) d \Rightarrow^*$
$aaq_1(\delta) bbaq_1(\delta) bcq_2(\delta) dccq_2(\delta) dd \Rightarrow^*$

$aabbabcdccdd$

(2) The yT transducer $M_2 = (\{q\}, \{\sigma, \delta\}, \{a\}, q, R_2)$, where σ has rank 1 and δ has rank 0, and with $R_2 = \{q(\sigma(x)) \to q(x)q(x), q(\delta) \to a\}$ generates the language $\mathcal{L}(M_2) = \{a^{(2^n)} \mid n \geq 0\}$.
A sample derivation is
$q(\sigma^3(\delta)) \Rightarrow$
$q(\sigma^2(\delta))q(\sigma^2(\delta)) \Rightarrow^*$
$q(\sigma(\delta))q(\sigma(\delta))q(\sigma(\delta))q(\sigma(\delta)) \Rightarrow^*$
$q(\delta)q(\delta)q(\delta)q(\delta)q(\delta)q(\delta)q(\delta)q(\delta) \Rightarrow^*$
$aaaaaaaa$
The translation realized by M_2 is $M_2 = \{\langle \sigma^m(\delta), a^{(2^m)} \rangle \mid m \geq 0\}$.

(3) Let $M_3 = (Q, \Sigma, \Delta, q_0, R_3)$ be a yT transducer, with $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b, \delta\}$, $\Sigma_0 = \{\delta\}$, $\Sigma_1 = \{a, b\}$, $\Delta = \{1, 2, 3\}$ and R_3 consists of the following rules :

$$\begin{array}{lll}
q_0(a(x)) \to 1q_1(x)2q_2(x) & q_0(b(x)) \to q_2(x)q_3(x) & q_0(\delta) \to \lambda \\
q_1(a(x)) \to 1q_1(x) & q_1(b(x)) \to q_2(x) & q_1(\delta) \to \lambda \\
q_2(a(x)) \to 2q_2(x) & q_2(b(x)) \to q_3(x) & q_2(\delta) \to \lambda \\
q_3(a(x)) \to 3q_3(x) & q_3(b(x)) \to q_1(x) & q_3(\delta) \to \lambda
\end{array}$$

M_3 translates a tree of the form $a^{n_1}ba^{n_2}b \dots a^{n_k}\delta$, with $n_i \geq 0$ for $1 \leq i \leq k$ and $k \geq 0$, into the string $1^{n_1}2^{n_2}3^{n_3}1^{n_4} \dots m^{n_k}2^{n_1}3^{n_2}1^{n_3}2^{n_4} \dots \ell^{n_k}$, where m and ℓ are symbols in Δ and $m = 1$ if (k mod 3) = 1, 2 if (k mod 3) = 2, 3 if (k mod 3) = 0, and $\ell = 1$ if (($k + 1$) mod 3) = 1, 2 if (($k + 1$) mod 3) = 2, and 3 if (($k + 1$) mod 3) = 0.
The language generated by M_3 is $\mathcal{L}(M_3) = \{1^{n_1}2^{n_2}3^{n_3}1^{n_4} \dots m^{n_k}2^{n_1}3^{n_2}1^{n_3}2^{n_4} \dots \ell^{n_k} \mid n_i \geq 0$ for $1 \leq i \leq k$ and $k \geq 0\}$ with m and ℓ as above.
A sample derivation is
$q_0(a^2bba^3ba\delta) \Rightarrow$
$1q_1(abba^3ba\delta)2q_2(abba^3ba\delta) \Rightarrow^*$
$11q_1(bba^3ba\delta)22q_2(bba^3ba\delta) \Rightarrow^*$
$11q_2(ba^3ba\delta)22q_3(ba^3ba\delta) \Rightarrow^*$
$11q_3(a^3ba\delta)22q_1(a^3ba\delta) \Rightarrow^*$
$11333q_3(ba\delta)22111q_1(ba\delta) \Rightarrow^*$
$11333q_1(a\delta)22111q_2(a\delta) \Rightarrow^*$
$113331q_1(\delta)221112q_2(\delta) \Rightarrow^*$
113331221112

(4) Consider the yT transducer $M_4 = (\{p, q\}, \{\tau, \delta\}, \{a, b\}, p, \{p(\tau(x)) \to q(x)q(x), p(\delta) \to \lambda, q(\tau(x)) \to aq(x), q(\tau(x)) \to bq(x), q(\delta) \to a, q(\delta) \to b\})$, with $\Sigma_0 = \{\delta\}$ and $\Sigma_1 = \{\tau\}$. The translation defined by M_4 is $M_4 = \{\langle \tau^m(\delta), u \rangle \mid m \geq 0,\ u \in \{a, b\}^*$ and $|u| = 2m\}$ and the language generated by M_4 is $\mathcal{L}(M_4) = \{u \mid u \in \{a, b\}^*$ and $|u|$ is even $\}$.
A sample derivation is
$p(\tau^4(\delta)) \Rightarrow$
$q(\tau^3(\delta))q(\tau^3(\delta)) \Rightarrow^*$
$bq(\tau^2(\delta))aq(\tau^2(\delta)) \Rightarrow^*$
$bbq(\tau(\delta))aaq(\tau(\delta)) \Rightarrow^*$

$bbbq(\delta)aaaq(\delta) \Rightarrow^*$
$bbbbaaaa$

Note that M_1, M_2 and M_3 are deterministic, but M_4 is not, because of the rules $q(\tau(x)) \to aq(x)$, $q(\tau(x)) \to bq(x)$ and $q(\delta) \to a$, $q(\delta) \to b$.

□

Until now, we could use any tree in T_Σ as an input tree. But if we want, for instance, to generate the language $\{w \in \{a,b\}^* \mid |w| = 2^{(2^n)}$ for some $n \geq 0\}$, then, intuitively, we need an input language of the form $\{\sigma^{(2^n)}(\delta) \mid n \geq 0\}$, because if we use $\sigma^m(\delta)$ (with $m \geq 1$) as an input tree, then we cannot be certain that $m = 2^n$ for some $n \geq 0$. Therefore in [ERS] a class of transducers is defined, that can have a specified subset of T_Σ as an input language.

Definition 5.2
Let $\mathcal{C}$ be a class of tree languages. A *top-down $\mathcal{C}$-tree transformation system* (yT($\mathcal{C}$) transducer) is a pair (M, L), where $M = (Q, \Sigma, \Delta, q_0, R)$ is a yT transducer, $L \subseteq T_\Sigma$ and $L \in \mathcal{C}$. (M, L) is *deterministic* if M is. □

The *language generated by* (M, L) is $M(L) = \{w \in \Delta^* \mid q_0(t) \Rightarrow^* w$ for some $t \in L\}$. $M(L)$ is called a *top-down $\mathcal{C}$-tree transformation language.* L is called the *input language.*

We will denote the class of (deterministic) top-down $\mathcal{C}$-tree transformation systems by yT($\mathcal{C}$) (yDT($\mathcal{C}$)), and the class of languages they generate by yTL($\mathcal{C}$) (yDTL($\mathcal{C}$)).

Remark : the 'y' in yT transducer means 'yield'. A top-down tree-to-string transducer is called a yT transducer because the $\mathcal{C}$-tree transformation languages are usually defined by taking the yield of the tree languages which are images of $\mathcal{C}$-tree languages under conventional top-down transducers ([Ro]).

Example 5.2
This is Example (3.1.4)(iii) from [ERS].
The language $\{w \in \{a,b\}^* \mid |w| = 2^{(2^n)}$ for some $n \geq 0\}$ is generated by the top-down EDT0L-tree (see [ERS]) transformation system (M, L), where $L = \{\sigma^{(2^n)}(\delta) \mid n \geq 0\}$, $M = (\{q\}, \{\sigma, \delta\}, \{a, b\}, q, R)$, with $\sigma \in \Sigma_1$ and $\delta \in \Sigma_0$, and R consists of the following rules :

$q(\sigma(x)) \to q(x)q(x)$
$q(\delta) \to a$
$q(\delta) \to b$

□

In [ERS] now restrictions on the derivations of yT transducers are introduced. In order to do so, the concept of *state-sequence of a derivation at a node of an input tree* is needed. Intuitively, it is the sequence of states in which the transducer starts to translate (the left to right sequence of copies of) the subtree which has the given node as its root.

In the following definition we will use sequences of states. These sequences are elements of Q^*, but for reasons of clarity we will write such a sequence between $\langle$ and $\rangle$.

Definition 5.3
Let $M = (Q, \Sigma, \Delta, q_0, R)$ be in yT. Let $\alpha : q(t) \Rightarrow^* w$ be a derivation of M with $q \in Q$, $t \in T_\Sigma$ and $w \in \Delta^*$. Let d be a node of t.
The *state-sequence of* α *at* d is a sequence $\langle q_1 \dots q_m \rangle$ of states of M (with $m \geq 0$) defined recursively as follows.

i. If $t = \sigma \in \Sigma_0$ and d is the unique node labelled by σ, then the state-sequence of α at d is $\langle q \rangle$.

ii. Assume that $t = \sigma(t_1 \dots t_k)$ with $\sigma \in \Sigma_k$ and $k \geq 1$. If d is the root of t then the state-sequence of α at d is $\langle q \rangle$. Now let d be a node of t_i for some i, $1 \leq i \leq k$. Consider the first step of the derivation $\alpha : q(\sigma(t_1 \dots t_k)) \Rightarrow r[t_1, \dots, t_k] \Rightarrow^* w$, where $q(\sigma(x_1 \dots x_k)) \rightarrow r$ is in R. If x_i does not occur in r, then the state-sequence of α at d is empty, i.e., $\langle\rangle$. Assume now that x_i occurs in r and let $r = u_1 q_1(x_i) u_2 q_2(x_i) \dots u_n q_n(x_i) u_{n+1}$ with $n \geq 1$ and $u_\ell \in (\Delta \cup Q(X_k \backslash \{x_i\}))^*$ for $1 \leq \ell \leq n+1$. It should be clear that there are unique derivations $\alpha_j : q_j(t_i) \Rightarrow^* v_j$ (with $v_j \in \Delta^*$ for $1 \leq j \leq n$) which are a 'part' of the derivation $r[t_1, \dots, t_k] \Rightarrow^* w$. Let s_j be the state-sequence of α_j at d. Then their concatenation $s_1 s_2 \dots s_n$ is the state-sequence of α at d.

□

In this definition we have used the obvious fact that if $w_1 q_1(t_1) w_2 \dots w_n q_n(t_n) w_{n+1} \Rightarrow^* w$ for some $w, w_i \in \Delta^*$, with $1 \leq i \leq n+1$, and $q_j \in Q$, $t_j \in T_\Sigma$, with $1 \leq j \leq n$, then there are unique derivations $q_j(t_j) \Rightarrow^* v_j$ such that $w = w_1 v_1 w_2 \dots w_n v_n w_{n+1}$ and the latter derivations are 'part' of the original one in the sense that some reordering of them is precisely the original derivation.

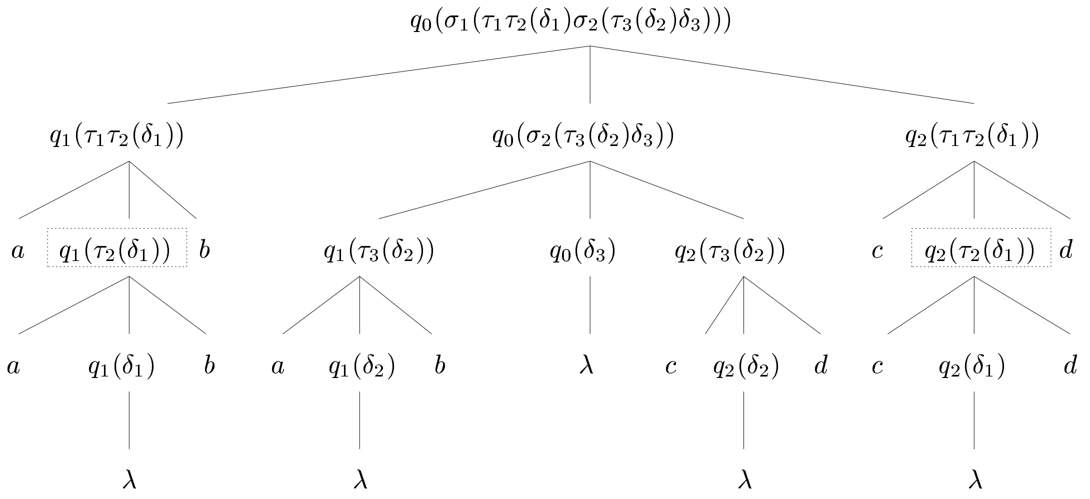
Example 5.3
(1) For the derivation and the input tree of the transducer M_1 given in Example 5.1, the state-sequence at each node labelled σ and at the right-most δ is $\langle q_0 \rangle$; at each node labelled τ and at all other nodes labelled δ the state-sequence is $\langle q_1 q_2 \rangle$.

(2) For the derivation and the input tree of the transducer M_2 given in Example 5.1, the state-sequence at the top-most node labelled σ (the root, that is) is $\langle q \rangle$, the second σ has state-sequence $\langle qq \rangle$, the third $\langle qqqq \rangle$ and the node labelled δ has state-sequence $\langle qqqqqqqq \rangle$.

(3) For the derivation and the input tree of the transducer M_3 given in that same example, the state-sequence at the root of the input tree (the first a) is $\langle q_0 \rangle$. The second a has state-sequence $\langle q_1 q_2 \rangle$, the first b $\langle q_1 q_2 \rangle$, the second b $\langle q_2 q_3 \rangle$, the third, fourth and fifth a $\langle q_3 q_1 \rangle$, the last b $\langle q_3 q_1 \rangle$, the last a $\langle q_1 q_2 \rangle$, and the δ also $\langle q_1 q_2 \rangle$.

(4) For the derivation and the input tree of the transducer M_4 given in Example 5.1, the topmost τ has state-sequence $\langle p \rangle$, the other nodes have state-sequence $\langle qq \rangle$. □

Since a state-sequence is not very easy to determine if we have to use the above definition, we will clarify the construction of state-sequences with the help of a 'derivation tree'. The 'derivation tree' t for the derivation of M_1 given in Example 5.3 (1) can be given as follows :



We have added subscripts to the input symbols in order to distinguish different nodes with the same label. The state-sequence of this derivation at, for instance, τ_2, can be found at the third level of t, by simply concatenating, in the order in which they appear in t, the states that are applied to the occurrences of the subtree with root τ_2 in t : $\langle q_1, q_2\rangle$. Note that, when we draw the 'derivation tree' of a derivation α as we did above, the state-sequence of α at a node of the input tree can always be found at one level of the 'derivation tree'.

We are now ready to define the restrictions on yT transducers.

Definition 5.4
Let $M = (Q, \Sigma, \Delta, q_0, R)$ be in yT, and let $k \geq 1$ be an integer. A derivation $\alpha : q_0(t) \Rightarrow^* w$ has *copying-bound* k if, for each node d of t, the length of the state-sequence of α at d is at most k.

Let L be a tree language. (M, L) has *copying-bound* k if for each $w \in M(L)$ there exist $t \in L$ and a derivation $q_0(t) \Rightarrow^* w$ with copying-bound k. (M, L) is *finite copying* if it has copying-bound k for some k. The same terminology holds for M if it is true of (M, T_Σ). □

Finite copying is denoted by a subscript 'fc', and a copying-bound is indicated by a subscript '(k)'. Thus for instance the classes of copying-bound k and finite copying top-down $\mathcal{C}$-tree transformation languages are denoted by $\mathrm{yT}_{\mathrm{fc}(k)}\mathrm{L}(\mathcal{C})$ and $\mathrm{yT}_{\mathrm{fc}}\mathrm{L}(\mathcal{C})$, respectively.

Example 5.4
In this example we will discuss the transducers of Example 5.1.
(1) M_1 has copying-bound 2, thus M_1 is finite copying.

(2) M_2 does not have a copying-bound, because the length of the state-sequences of nodes of an input tree grows as the depth of the input tree grows.
(3) M_3 is finite copying; it has copying-bound 2.
(4) M_4 has copying-bound 2, so $\mathcal{L}(M) \in \mathrm{yT}_{\mathrm{fc}(2)}\mathrm{L}$. □

5.3 Results

In this section we prove that ssag's generate the same languages as $\mathrm{yT}_{\mathrm{fc}}(\mathrm{REC})$ transducers.

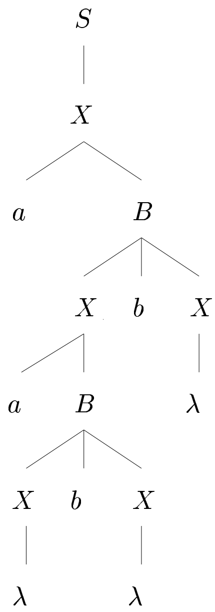
The states of a transducer generate translations of the subtrees of an input tree, while the attributes of an OnlyS-sag add meanings to the subtrees of a derivation tree of the underlying cfg. Therefore it seems natural to suppose that states in a transducer correspond to attributes in an OnlyS-sag, and that the derivation trees of an OnlyS-sag correspond to the input trees of a transducer. In the following lemma we will prove that indeed for every OnlyS-sag we can construct a yT(REC) transducer that generates the same language.

We label the productions of a cfg $G = (N, T, P, S_0)$ with p_1 through $p_{|P|}$. An *abstract syntax tree* of G is a derivation tree t of G in which the label of each nonterminal node is replaced by the label of the production applied at that node, and in which all the terminal nodes are removed.

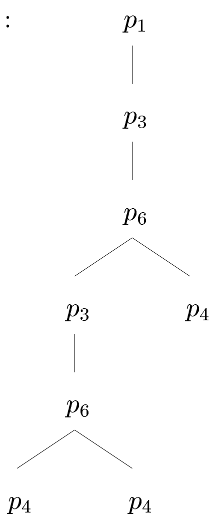
By encoding every derivation tree of G, we get a tree language over $\{p_1, \ldots, p_{|P|}\}$, and each p_i (for $1 \leq i \leq |P|$) has exactly one rank, being the number of nonterminals in the right-hand side of the production labelled p_i. Obviously, this tree language is in REC (see [ERS] p. 160).

Example 5.5
Consider the cfg given in Example 2.6 (2), of which we have labelled the productions p_1 through p_6 (so $S_0 \to X$ has label p_1 and $B \to XbX$ has label p_6), and the following derivation tree t for a^2b^2 of that cfg.

$t:$



$t':$



The encoded version of t is t'. □

Lemma 5.1
OnlyS-SAL $\subseteq$ yDTL(REC)
Proof
Let $G = (G_0, D, B, R)$ be an OnlyS-sag with $G_0 = (N, T, P, S_0)$, where P consists of $|P|$ productions that are labelled p_1 through $p_{|P|}$, $D = (\{\Gamma^*\}, \Phi)$ for some alphabet Γ and $B = (S, I, \alpha_0, W)$. Let $K = \{t \mid t \text{ is an abstract syntax tree of } G_0\}$, then $K \in$ REC. We will construct a yT transducer $M = (Q, \Sigma, \Delta, q_0, R')$, with $Q = S\text{-}Att$, $\Sigma = \{p_1, \ldots, p_{|P|}\}$ where, for $n \geq 0$, $\Sigma_n = \{p_i \mid 1 \leq i \leq |P|$ and the right-hand side of the production labelled p_i in P contains exactly n nonterminals $\}$, $\Delta = \Gamma$ and $q_0 = \alpha_0$, such that $M(K) = \mathcal{L}(G)$. Let $R' = \emptyset$.
For every terminating production $p : X \to w$ in P with semantic rules $\langle s_1, 0\rangle = v_1, \ldots, \langle s_m, 0\rangle = v_m$ where $m = d(X)$, we add the rules $s_1(p) \to v_1, \ldots, s_m(p) \to v_m$ to R'.
For every nonterminating production
$q : X_0 \to w_1 X_1 w_2 \ldots w_k X_k w_{k+1}$
in P, with semantic rules

$$\begin{aligned}
\langle s_1, 0\rangle &= u_{1,1}\langle s_{1,1}, j_{1,1}\rangle u_{1,2} \ldots u_{1,n_1} \ \langle s_{1,n_1}, j_{1,n_1}\rangle u_{1,n_1+1}\\
&\ \vdots \qquad\qquad\qquad\qquad\qquad \vdots\\
\langle s_\ell, 0\rangle &= u_{\ell,1}\langle s_{\ell,1}, j_{\ell,1}\rangle u_{\ell,2} \ldots u_{\ell,n_\ell} \ \langle s_{\ell,n_\ell}, j_{\ell,n_\ell}\rangle u_{\ell,n_\ell+1}
\end{aligned}$$

where $\ell = d(X_0)$, we add the rules

$$\begin{aligned}
s_1(q(x_1 \ldots x_k)) &\to u_{1,1}s_{1,1}(x_{j_{1,1}})u_{1,2} \ldots u_{1,n_1} \ s_{1,n_1}(x_{j_{1,n_1}})u_{1,n_1+1}\\
&\ \vdots \qquad\qquad\qquad\qquad\qquad \vdots\\
s_\ell(q(x_1 \ldots x_k)) &\to u_{\ell,1}s_{\ell,1}(x_{j_{\ell,1}})u_{\ell,2} \ldots u_{\ell,n_\ell} \ s_{\ell,n_\ell}(x_{j_{\ell,n_\ell}})u_{\ell,n_\ell+1}
\end{aligned}$$

to R'.
The resulting transducer will, with input language K, generate the same language as G does. □

Example 5.6
Consider the OnlyS-sag G, with the following productions and semantic rules :

p_1 :	$S_0 \to X$	$\alpha_0 = a\beta_1\beta_1\gamma_1$	
p_2 :	$X \to X$	$\beta_0 = a\beta_1$	$\gamma_0 = a\beta_1\beta_1\gamma_1$
p_3 :	$X \to \lambda$	$\beta_0 = \lambda$	$\gamma_0 = \lambda$

and with initial symbol S_0 and designated attribute α.
$\mathcal{L}(G) = \{a^{(n^2)} \mid n > 0\}$ (see also Example 3.2 (5)).
According to the construction given in the previous lemma, we can create the following yT(REC) transducer : $M = (Q, \Sigma, \Delta, q_0, R)$ with $Q = \{\alpha, \beta, \gamma\}$, $\Sigma = \{p_1, p_2, p_3\}$ where $\Sigma_1 = \{p_1, p_2\}$ and $\Sigma_0 = \{p_3\}$, $\Delta = \{a\}$, $q_0 = \alpha$ and R consists of the rules

$\alpha(p_1(x)) \to a\beta(x)\beta(x)\gamma(x)$
$\beta(p_2(x)) \to a\beta(x)$
$\gamma(p_2(x)) \to a\beta(x)\beta(x)\gamma(x)$
$\beta(p_3) \to \lambda$
$\gamma(p_3) \to \lambda$

M translates trees of the form $p_1p_2^mp_3$ into $a^{(m+1)^2}$. If $K = \{t \mid t$ is an abstract syntax tree of $G\}$, then $M(K) = \mathcal{L}(G)$. □

As described before Lemma 5.1, encoding a derivation tree t of an ag gives a tree that can be used as an input tree t' of a transducer. The root of every subtree of t' corresponds to exactly one occurrence of a nonterminal, say X, in t. If the ag under consideration is special, then we know that every (s-)attribute of X is used exactly once. Since, according to Lemma 5.1, there is a one-to-one correspondence between the attributes of X and the states that are applied to the subtree t'', that corresponds to X, of the input tree, there are exactly $d(X)$ states applied to t'', where d is the rank function on the nonterminals of the ag. This holds for every subtree of every input tree of the transducer. Since, in an ag, every nonterminal has a fixed and finite number of attributes, the number of states applied to every subtree of every input tree is at most $\max\{\ |S(X)| \mid X \in N\}$, where N is the set of nonterminals of the ag.

In terms of mcfg's, it is even easier to determine this copying bound : using the method of Lemma 5.1 to translate an m-mcfg to a transducer gives a $\mathrm{yT_{fc(\mathit{m})}}$(REC) transducer.

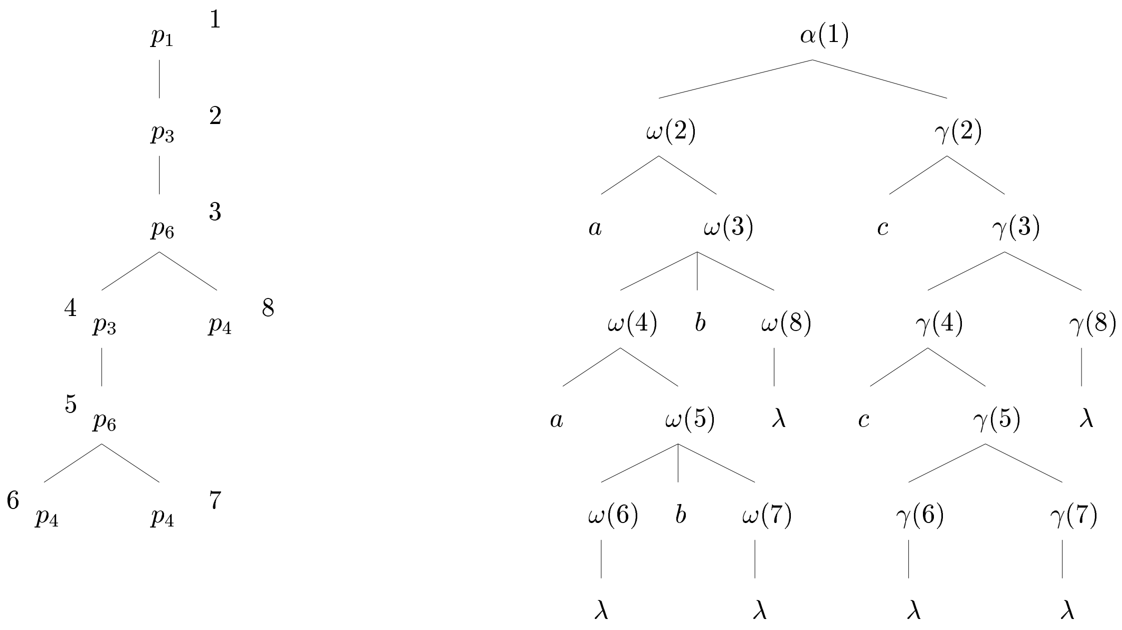
This proves the following lemma.

Lemma 5.2
SSAL $\subseteq$ $\mathrm{yT_{fc}L}$(REC).

Example 5.7
A $\mathrm{yT_{fc(2)}}$(REC) transducer that is equivalent to the ssag of Example 2.6 (2) (which generates the language $L = \{wc^n \mid w \in \{a,b\}^*, |w|_a = |w|_b = n\}$) is the following :

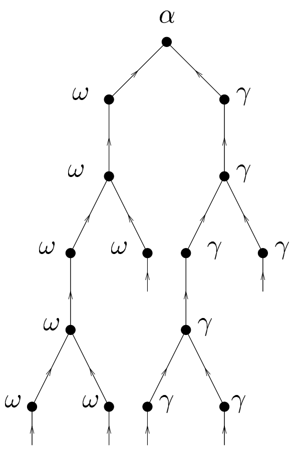
$\alpha(p_1(x)) \to \omega(x)\gamma(x)$	
$\omega(p_2(x)) \to b\omega(x)$	$\gamma(p_2(x)) \to c\gamma(x)$
$\omega(p_3(x)) \to a\omega(x)$	$\gamma(p_3(x)) \to c\gamma(x)$
$\omega(p_4) \to \lambda$	$\gamma(p_4) \to \lambda$
$\omega(p_5(xy)) \to \omega(x)a\omega(y)$	$\gamma(p_5(xy)) \to \gamma(x)\gamma(y)$
$\omega(p_6(xy)) \to \omega(x)b\omega(y)$	$\gamma(p_6(xy)) \to \gamma(x)\gamma(y)$

For reasons of clarity, we will label the nodes of the abstract syntax tree t' given in Example 5.5 as follows.



The rightmost tree is the 'derivation tree' of the derivation $\alpha(p_1p_3p_6(p_3p_6(p_4p_4)p_4)) \Rightarrow^* a^2b^2c^2$, in which we have replaced each subtree by the number referring to its root in the leftmost tree.

If we now draw the 'dependency graph' of this 'derivation tree', we get the inverted tree



which is exactly the same as the dependency graph of the derivation tree for $a^2b^2c^2$ given in Example 2.6 (2). This shows once more the similarities between ssag's and finite copying transducers, and in particular the one-to-one correspondence between attributes and states. □

We will use the following terminology concerning a transducer M : we will write 'state-sequence of M' instead of 'state-sequence of a derivation of M at a node of an input tree'. Furthermore, we call a rule $q(\sigma(x_1 \dots x_k)) \to \beta$, with $\beta \in (\Delta \cup Q(X_k))^*$, a 'rule for q' or a 'rule for σ' or a 'rule for q and σ', and we say that q 'is applied to' $\sigma(x_1 \dots x_k)$.

Let $M = (Q, \Sigma, \Delta, q_0, R)$ be in $\mathrm{yT_{fc}}$, with copying-bound c.
For a rule
$r \ : q(\sigma(x_1 \dots x_k)) \to w_1 q_1(x_{i_1}) w_2 \dots w_n q_n(x_{i_n}) w_{n+1}$
in R, we define, for $1 \le i \le k$, the *state-sequence of* x_i *in* r, $ss(r,i) \in Q^*$, as follows :
$ss(r,i) = \langle p_1 \dots p_\ell \rangle$
where $0 \le \ell \le c$, $p_1, \dots p_\ell \in Q$, and $p_1(x_i) \dots p_\ell(x_i)$ contains all occurrences of x_i in the right-hand side of r, in this order (so r can be written as $u_1 p_1(x_i) u_2 p_2(x_i) \dots u_\ell p_\ell(x_i) u_{\ell+1}$ with $u_j \in (\Delta \cup Q(X_k \backslash \{x_i\}))^*$ for $1 \le j \le \ell + 1$). Thus $ss(r,i)$ is the part of the state-sequence of the root of the i^{th} subtree of a tree of the form $\sigma(t_1 \dots t_k)$, with $t_1, \dots, t_k \in T_\Sigma$, that can be derived from the applied rule (r). Again, for reasons of clarity, we put the state-sequence between $\langle$ and $\rangle$.

Furthermore, if $ss(r,i) = \langle p_1 \dots p_\ell \rangle$, we define $ss^j(r,i)$ to be $\langle p_1 \dots p_{j-1} \overline{p_j} p_{j+1} \dots p_\ell \rangle$ with $1 \le j \le \ell$, which is an element of $Q^* \cdot \overline{Q} \cdot Q^*$, where $\overline{Q} = \{\overline{q} \mid q \in Q\}$ is a disjoint copy of Q.

Finally, for some $m \ge 1$, we define $ss(r_1 \dots r_m, i)$ to be an abbreviation of $ss(r_1, i) \cdot \ldots \cdot ss(r_m, i)$.

We will now show that for every $\mathrm{yT_{fc}}$ transducer M an ssag G can be constructed such that $\mathcal{L}(M) = \mathcal{L}(G)$.

If a yT transducer M has copying-bound c, this means that of each subtree of an input tree at least 0 and at most c copies or translations can be made, some of which may be equal. The state-sequence of a derivation at a node d of an input tree t shows exactly how many and which copies are made of the subtree with root d of t. Often, several subtrees of the input tree will have the same state-sequence $\langle s\rangle$, with $s \in Q^*$. All these subtrees together will be denoted by the nonterminal $X_{\langle s\rangle}$.

We will construct our OnlyS-sag such that all translations in $\langle s\rangle$ are made at the same time. This is possible by giving $X_{\langle s\rangle}$ a synthesized attribute q for every state q that occurs in $\langle s\rangle$. Unfortunately it is possible that the same state q occurs more than once (say r times) in $\langle s\rangle$. This would force us to use the attribute q r times, which prevents the resulting OnlyS-sag from being special. The solution that we have chosen to solve this problem is to construct a $\mathrm{yT}_{\mathrm{fc}(c)}$ transducer M' with $\mathcal{L}(M) = \mathcal{L}(M')$, such that there is no state-sequence of M' that contains more than one occurrence of each state.

In the following lemma we will prove that this is a normal form for deterministic transducers, since for deterministic transducers the proof is easier than for nondeterministic transducers. This is not a restriction, however, since from Lemma 3.2.3 from [ERS] we know that, for every $k \geq 1$, $\mathrm{yT}_{\mathrm{fc}(k)}\mathrm{L(REC)} = \mathrm{yDT}_{\mathrm{fc}(k)}\mathrm{L(REC)}$.

Lemma 5.3
Let M be in $\mathrm{yDT}_{\mathrm{fc}(c)}$.
It is possible to construct an equivalent transducer M' in $\mathrm{yDT}_{\mathrm{fc}(c)}$ such that every state-sequence of length z of M' $(2 \leq z \leq c)$ consists of z *different* states.

Proof
Let $M = (Q, \Sigma, \Delta, q_0, R)$, with copying-bound c. M' will be $(Q', \Sigma, \Delta, q_0', R')$, and the states of M' will be the state-sequences of M, in which the state that was applied in M is marked.

First, we will consider every rule for q_0 in R. These are of the form
$r\ : q_0(\sigma(x_1 \dots x_k)) \to w_1 q_1(x_{i_1}) w_2 \dots w_n q_n(x_{i_n}) w_{n+1}$.
For each of these rules we make the same rule in R', but we replace q_0 by $\langle\overline{q_0}\rangle$, and in the right-hand side we replace, for every x_i $(1 \leq i \leq k)$, the state that is applied to the u^{th} $(1 \leq u \leq c)$ occurrence of x_i in r by $ss^u(r,i)$. Then we add $\langle\overline{q_0}\rangle$ and the $ss^u(r,i)$ that we have introduced to Q'. After having done this for every rule for q_0 in R, we set $q_0' = \langle\overline{q_0}\rangle$.

We now have introduced new states in Q', for which we have to make the appropriate rules. Let $\langle p_1 \dots p_{j-1}\overline{p_j}p_{j+1} \dots p_m\rangle$ be such a new state, with $1 \leq j \leq m \leq c$. For every $\sigma \in \Sigma$, consider the rules

$$\begin{array}{lllllllll}
r_1: & p_1(\sigma(x_1 \dots x_k)) & \to & w_{1,1} & q_{1,1}(x_{i_{1,1}}) & w_{1,2} & \dots & w_{1,n_1} & q_{1,n_1}(x_{i_{1,n_1}}) \quad w_{1,n_1+1} \\
\vdots & & \vdots & & & & \vdots & & \\
r_m: & p_m(\sigma(x_1 \dots x_k)) & \to & w_{m,1} & q_{m,1}(x_{i_{m,1}}) & w_{m,2} & \dots & w_{m,n_m} & q_{m,n_m}(x_{i_{m,n_m}}) \; w_{m,n_m+1}.
\end{array}$$

Since M is deterministic these are all the possible rules for the p_h $(1 \leq h \leq m)$ and σ. (If there is a p_h for which there is no rule for σ, we cannot make, for this σ, a rule for $\langle p_1 \dots p_{j-1}\overline{p_j}p_{j+1} \dots p_m\rangle$ in R', nor for the other marked versions of $\langle p_1 \dots p_{j-1}p_j p_{j+1} \dots p_m\rangle$.) The rule that we can now add to R' for $\langle p_1 \dots p_{j-1}\overline{p_j}p_{j+1} \dots p_m\rangle$ is the same as r_j, but in the left-hand side we replace p_j by $\langle p_1 \dots p_{j-1}\overline{p_j}p_{j+1} \dots p_m\rangle$, and in the right-hand side we replace, for every x_i $(1 \leq i \leq k)$, the state that is applied to the ℓ^{th} occurrence of x_i in the right-hand side of r_j by

$ss(r_1 \dots r_{j-1}, i) \cdot ss^{\ell}(r_j, i) \cdot ss(r_{j+1} \dots r_m, i)$.
We have to execute this procedure for every newly introduced nonterminal.
The above construction makes sure that each state-sequence of M' consists of different states, because now a state-sequence of length z of M' ($0 \leq z \leq c$) is a sequence of z equal state-sequences of M, except that the j^{th} state-sequence of M (in the state-sequence of M') has a mark on its j^{th} element ($1 \leq j \leq z$). Since we only have renamed the states of M, M' has the same copying-bound as M.

Obviously, M' is deterministic. □

Example 5.8
Applying the construction of Lemma 5.3 to M_3 of Example 5.1 gives us the following $\mathrm{yDT_{fc(2)}}$ transducer in normal form :

$$\begin{array}{lll}
\langle\overline{q_0}\rangle(a(x)) \rightarrow 1\langle\overline{q_1}q_2\rangle(x)2\langle q_1\overline{q_2}\rangle(x) & \langle\overline{q_0}\rangle(b(x)) \rightarrow \langle\overline{q_2}q_3\rangle(x)\langle q_2\overline{q_3}\rangle(x) & \langle\overline{q_0}\rangle(\delta) \rightarrow \lambda \\
\langle\overline{q_1}q_2\rangle(a(x)) \rightarrow 1\langle\overline{q_1}q_2\rangle(x) & \langle\overline{q_1}q_2\rangle(b(x)) \rightarrow \langle\overline{q_2}q_3\rangle(x) & \langle\overline{q_1}q_2\rangle(\delta) \rightarrow \lambda \\
\langle q_1\overline{q_2}\rangle(a(x)) \rightarrow 2\langle q_1\overline{q_2}\rangle(x) & \langle q_1\overline{q_2}\rangle(b(x)) \rightarrow \langle q_2\overline{q_3}\rangle(x) & \langle q_1\overline{q_2}\rangle(\delta) \rightarrow \lambda \\
\langle\overline{q_2}q_3\rangle(a(x)) \rightarrow 2\langle\overline{q_2}q_3\rangle(x) & \langle\overline{q_2}q_3\rangle(b(x)) \rightarrow \langle\overline{q_3}q_1\rangle(x) & \langle\overline{q_2}q_3\rangle(\delta) \rightarrow \lambda \\
\langle q_2\overline{q_3}\rangle(a(x)) \rightarrow 3\langle q_2\overline{q_3}\rangle(x) & \langle q_2\overline{q_3}\rangle(b(x)) \rightarrow \langle q_3\overline{q_1}\rangle(x) & \langle q_2\overline{q_3}\rangle(\delta) \rightarrow \lambda \\
\langle\overline{q_3}q_1\rangle(a(x)) \rightarrow 3\langle\overline{q_3}q_1\rangle(x) & \langle\overline{q_3}q_1\rangle(b(x)) \rightarrow \langle\overline{q_1}q_2\rangle(x) & \langle\overline{q_3}q_1\rangle(\delta) \rightarrow \lambda \\
\langle q_3\overline{q_1}\rangle(a(x)) \rightarrow 1\langle q_3\overline{q_1}\rangle(x) & \langle q_3\overline{q_1}\rangle(b(x)) \rightarrow \langle q_1\overline{q_2}\rangle(x) & \langle q_3\overline{q_1}\rangle(\delta) \rightarrow \lambda
\end{array}$$

with initial state $\langle\overline{q_0}\rangle$. □

We use Theorem 3.2.1 from [ERS], which states that a specific recognizable input language can be coded as part of the transducer implying that we can consider arbitrary input languages.

Theorem 5.4
For each top-down tree transformation system (M, L) with $L \in \mathrm{REC}$, there exists a top-down yT transducer M' such that $M(L) = M'(T_{\Sigma'})$, where Σ' is the input alphabet of M'. The construction involved preserves determinism and copying-bound.

This theorem is the reason that, to reach the goal of this section, it is now sufficient to prove that for every $\mathrm{yT_{fc}}$ transducer there is an equivalent ssag. Again, we will give the proof for the translation from deterministic $\mathrm{yT_{fc}}$ transducers into ssag's, because that is easier than the non-deterministic case.

As mentioned before, we use in this proof nonterminals of the form $X_{\langle s\rangle}$, which denotes the set of all (subtrees of) input trees that can have state-sequence $\langle s\rangle$. Such a nonterminal $X_{\langle s\rangle}$ will have an s-attribute for every state that occurs in $\langle s\rangle$, and no other attributes.

Lemma 5.5
$\mathrm{yDT_{fc}L} \subseteq \mathrm{SSAL}$.
Proof
Let $M = (Q, \Sigma, \Delta, q_0, R)$, with copying-bound c, be a deterministic $\mathrm{yT_{fc}}$ transducer. We may assume that every state-sequence of M consists of different states (see Lemma 5.3). We will construct an OnlyS-sag G such that $\mathcal{L}(G) = \mathcal{L}(M)$. G will be (G_0, D, B, R'), with $G_0 = (N, T, P, S_0)$, $D = (\Omega, \Phi)$, and $B = (S, I, \alpha_0, W)$. Since G is OnlyS, we know

that $I(X)$ will be the empty set for every $X \in N$, and because G is an sag, we know that $\Omega = \{\Delta^*\}$, $W(s) = \Delta^*$ for all attributes s, and Φ consists of all derived functions of the free monoid Δ^*.
Let $N = T = P = \emptyset$.

We construct G from M as follows, starting with the rules for q_0. For every rule
$r : q_0(\sigma(x_1 \dots x_k)) \to w_1 q_1(x_{i_1}) w_2 \dots w_n q_n(x_{i_n}) w_{n+1}$
in R, we add the following production to P :
$p : X_{\langle q_0 \rangle} \to r X_{ss(r,1)} \dots X_{ss(r,k)}$,
with $R'(p) = \{\langle q_0, 0\rangle = w_1 \langle q_1, i_1\rangle w_2 \dots w_n \langle q_n, i_n\rangle w_{n+1}\}$. Note that we add a new terminal r to the production to be able to distinguish between productions that have the same nonterminals, but that are translations of different rules of R.
We add $X_{\langle q_0 \rangle}$, $X_{ss(r,1)}$, $\dots$, $X_{ss(r,k)}$ to N, r to T, and we set $S(X_{\langle q_0 \rangle}) = \{q_0\}$ and $S(X_{ss(r,i)}) = \{q \in Q \mid q \text{ occurs in } ss(r,i)\}$ for $1 \le i \le k$. Furthermore, to make sure that the initial nonterminal of G does not occur in the right-hand side of any production, we add the production $S_0 \to X_{\langle q_0 \rangle}$ to P, S_0 to N, $S(S_0) = \{\alpha_0\}$, and we set $R'(S_0 \to X_{\langle q_0 \rangle}) = \{\langle \alpha_0, 0\rangle = \langle q_0, 1\rangle\}$.

Now we have to make rules for the nonterminals (except for $X_{\langle q_0 \rangle}$) that we have introduced. Let $X_{\langle p_1 \dots p_\ell \rangle}$ be such a nonterminal $(1 \le \ell \le c)$. For every $\sigma \in \Sigma$, consider the ℓ rules in R

$$\begin{array}{lll}
r_1 : p_1(\sigma(x_1 \dots x_k)) \to & w_{1,1} q_{1,1}(x_{i_{1,1}}) w_{1,2} & \dots w_{1,n_1} q_{1,n_1}(x_{i_{1,n_1}}) w_{1,n_1+1} \\
\vdots & \vdots & \vdots \\
r_\ell : p_\ell(\sigma(x_1 \dots x_k)) \to & w_{\ell,1} q_{\ell,1}(x_{i_{\ell,1}}) w_{\ell,2} & \dots w_{\ell,n_\ell} q_{\ell,n_\ell}(x_{i_{\ell,n_\ell}}) w_{\ell,n_\ell+1}.
\end{array}$$

Note that there cannot be more than one rule for p_m and σ $(1 \le m \le \ell)$, because M is deterministic. Furthermore, if for some p_m there is no rule for σ, then we cannot make, for this σ, a production for $X_{\langle p_1 \dots p_\ell \rangle}$.
We use these ℓ rules to make the following production for $X_{\langle p_1 \dots p_\ell \rangle}$:
$p' : X_{\langle p_1 \dots p_\ell \rangle} \to r_1 \dots r_\ell X_{ss(r_1 \dots r_\ell, 1)} \dots X_{ss(r_1 \dots r_\ell, k)}$,
where $R'(p')$ consists of the following semantic rules :

$$\begin{array}{lll}
\langle p_1, 0\rangle = & w_{1,1} \langle q_{1,1}, i_{1,1}\rangle w_{1,2} & \dots w_{1,n_1} \langle q_{1,n_1}, i_{1,n_1}\rangle w_{1,n_1+1} \\
\vdots & \vdots & \vdots \\
\langle p_\ell, 0\rangle = & w_{\ell,1} \langle q_{\ell,1}, i_{\ell,1}\rangle w_{\ell,2} & \dots w_{\ell,n_\ell} \langle q_{\ell,n_\ell}, i_{\ell,n_\ell}\rangle w_{\ell,n_\ell+1}.
\end{array}$$

Then we add $X_{ss(r_1 \dots r_\ell, 1)}, \dots, X_{ss(r_1 \dots r_\ell, k)}$ to N, $r_1, \dots, r_\ell$ to T, and we set $S(X_{ss(r_1 \dots r_\ell, i)}) = \{q \in Q \mid q \text{ occurs in } ss(r_1 \dots r_\ell, i)\}$ for $1 \le i \le k$. When we have made, for every nonterminal, a production for every $\sigma \in \Sigma$ (if possible), then $\mathcal{L}(G) = \mathcal{L}(M)$.

Since every state-sequence of M consists of different states, and since there is a one-to-one correspondence between the s-attributes of G and the states of M, the OnlyS-sag resulting from this construction is special. □

Example 5.9
(1) Using the construction of Lemma 5.5 we make an ssag that is equivalent to the transducer in normal form of Example 5.8. To simplify notation, we will first rename the states of that transducer :

$$\begin{array}{lll}
\alpha(a(x)) \to 1\beta(x)2\gamma(x) & \alpha(b(x)) \to \epsilon(x)\zeta(x) & \alpha(\delta) \to \lambda \\
\beta(a(x)) \to 1\beta(x) & \beta(b(x)) \to \epsilon(x) & \beta(\delta) \to \lambda \\
\gamma(a(x)) \to 2\gamma(x) & \gamma(b(x)) \to \zeta(x) & \gamma(\delta) \to \lambda \\
\epsilon(a(x)) \to 2\epsilon(x) & \epsilon(b(x)) \to \eta(x) & \epsilon(\delta) \to \lambda \\
\zeta(a(x)) \to 3\zeta(x) & \zeta(b(x)) \to \vartheta(x) & \zeta(\delta) \to \lambda \\
\eta(a(x)) \to 3\eta(x) & \eta(b(x)) \to \beta(x) & \eta(\delta) \to \lambda \\
\vartheta(a(x)) \to 1\vartheta(x) & \vartheta(b(x)) \to \gamma(x) & \vartheta(\delta) \to \lambda
\end{array}$$

with initial state α.
We number these rules p_1 through p_{21} (from left to right and from top to bottom).
The corresponding ssag is the following :

$$\begin{array}{llll}
X_{\langle\alpha\rangle} & \to p_1 X_{\langle\beta\gamma\rangle} & \alpha_0 = 1\beta_1 2\gamma_1 & \\
X_{\langle\alpha\rangle} & \to p_2 X_{\langle\epsilon\zeta\rangle} & \alpha_0 = \epsilon_1\zeta_1 & \\
X_{\langle\alpha\rangle} & \to p_3 & \alpha_0 = \lambda & \\
X_{\langle\beta\gamma\rangle} & \to p_4 p_7 X_{\langle\beta\gamma\rangle} & \beta_0 = 1\beta_1 & \gamma_0 = 2\gamma_1 \\
X_{\langle\beta\gamma\rangle} & \to p_5 p_8 X_{\langle\epsilon\zeta\rangle} & \beta_0 = \epsilon_1 & \gamma_0 = \zeta_1 \\
X_{\langle\beta\gamma\rangle} & \to p_6 p_9 & \beta_0 = \lambda & \gamma_0 = \lambda \\
X_{\langle\epsilon\zeta\rangle} & \to p_{10} p_{13} X_{\langle\epsilon\zeta\rangle} & \epsilon_0 = 2\epsilon_1 & \zeta_0 = 3\zeta_1 \\
X_{\langle\epsilon\zeta\rangle} & \to p_{11} p_{14} X_{\langle\eta\vartheta\rangle} & \epsilon_0 = \eta_1 & \zeta_0 = \vartheta_1 \\
X_{\langle\epsilon\zeta\rangle} & \to p_{12} p_{15} & \epsilon_0 = \lambda & \zeta_0 = \lambda \\
X_{\langle\eta\vartheta\rangle} & \to p_{16} p_{19} X_{\langle\eta\vartheta\rangle} & \eta_0 = 3\eta_1 & \vartheta_0 = 1\vartheta_1 \\
X_{\langle\eta\vartheta\rangle} & \to p_{17} p_{20} X_{\langle\beta\gamma\rangle} & \eta_0 = \beta_1 & \vartheta_0 = \gamma_1 \\
X_{\langle\eta\vartheta\rangle} & \to p_{18} p_{21} & \eta_0 = \lambda & \vartheta_0 = \lambda
\end{array}$$

(2) Consider the $\mathrm{yDT}_{\mathrm{fc}(2)}$ transducer M_1 of Example 5.1. We label the rules p_1 through p_6. Since $\langle q_0\rangle$ and $\langle q_1 q_2\rangle$ are the only possible state-sequences of M_1, M_1 is already in the normal form described in Lemma 5.3. The construction of Lemma 5.5 gives the following ssag :

$$\begin{array}{llll}
S_0 & \to X_{\langle q_0\rangle} & \langle\alpha_0, 0\rangle = \langle q_0, 1\rangle & \\
X_{\langle q_0\rangle} & \to p_1 X_{\langle q_1 q_2\rangle} X_{\langle q_0\rangle} & \langle q_0, 0\rangle = \langle q_1, 1\rangle\langle q_0, 2\rangle\langle q_2, 1\rangle & \\
X_{\langle q_0\rangle} & \to p_2 & \langle q_0, 0\rangle = \lambda & \\
X_{\langle q_1 q_2\rangle} & \to p_3 p_5 X_{\langle q_1 q_2\rangle} & \langle q_1, 0\rangle = a\langle q_1, 1\rangle b & \langle q_2, 0\rangle = c\langle q_2, 1\rangle d \\
X_{\langle q_1 q_2\rangle} & \to p_4 p_6 & \langle q_1, 0\rangle = \lambda & \langle q_2, 0\rangle = \lambda
\end{array}$$

□

By Lemma's 5.2 and 5.5 we have now reached the goal of this section.

Theorem 5.6
$\mathrm{yT}_{\mathrm{fc}}(\mathrm{REC}) = \mathrm{SSAL}$.

Corollary 5.7
$\mathrm{yT}_{\mathrm{fc}}(\mathrm{REC}) = \mathrm{MCFL}$. □

Chapter 6

Conclusion and further research

In this thesis we have proven the equivalence of mcfg's, srg's, ssag's and $\text{yT}_{\text{fc}}(\text{REC})$ transducers. Additional results are PMCFL = OnlyS-SAL = RL $\subseteq$ yTL(REC), and some normal forms for the grammar formalisms under consideration.

Of course, this is only a small part of the work that could be done in this area. As interesting subjects for further research we propose :

- Comparison of mcfg's with local unordered scattered context grammars and control grammars (see [RS], [RS94], [W], [PS]).
- Comparison of pumping lemma's for mcfg's and $\text{yT}_{\text{fc}}(\text{REC})$ transducers (see [SMFK] and [ERS]).
- Carrying over normal forms for cfg's to ag's.
- Carrying over some of the results we proved for OnlyS-sag's (e.g., the information-lossless condition) to ag's with i- and s-attributes.
- Is there a generalization of the combinatory categorial grammar (see [VW]) that is equivalent to the mcfg?
- Is there a kind of relational grammar, that generates trees instead of strings (see [R], [GR]), that generates the same tree languages as the tree adjoining grammar ([VW])?

Bibliography

[A] K. Ajdukiewicz; *Die syntaktische Konnexität*; Studia Philosophica, **1**:1-27, 1935. English translation in: S. McCall, editor, *Polish Logic, 1920-1939*, 207-231, Oxford University Press, Oxford.

[B] G. V. Bochmann; *Semantic evaluation from left to right*, Communications of the ACM 19 (1976).

[DJL] P. Deransart, M. Jourdan, B. Lorho; *Attribute Grammars, Definitions, Systems and Bibliography*; Lecture Notes in Computer Science 323, 1-51 (1988).

[E86] J. Engelfriet; *The complexity of languages generated by attribute grammars*; SIAM J. Comput. 15 (1986), 70-86.

[E94] J. Engelfriet; Lecture notes for the course *Formele Talen en Automaten 2* (in Dutch), Leiden University, Fall 1994.

[ERS] J. Engelfriet, G. Rozenberg, G. Slutzki; *Tree transducers, L-systems, and two-way machines*; J. of Comput. Syst. Sciences 20 (1980), 150-202.

[Ga] G. Gazdar; *Applicability of indexed grammars to natural languages.* In: U. Reyle and C. Rohrer, editors; Natural Language Parsing and Linguistic Theories, 69-94, Reidel, Dordrecht, 1988.

[Gi] R. Giegerich; *Composition and evaluation of Attribute Coupled Grammars*; Acta Informatica 25, 355-423 (1988).

[GR] A. Grazon, J.-C. Raoult; *Equational relations on trees*; manuscript, Rennes, June 1992.

[HU] J. E. Hopcroft, J. D. Ullman; *Introduction to Automata Theory, Languages, and Computation*; Addison-Wesley (1980).

[J] A. K. Joshi; *How much context-sensitivity is necessary for characterizing structural descriptions - Tree adjoining grammars?.* In: D. Dowty, L. Karttunen, and A. Zwicky, editors; Natural Language Processing - Theoretical, Computational and Psychological Perspective, 206-250, Cambridge University Press, New York, 1985. (Originally presented in 1983.)

[JLT] A. K. Joshi, L. S. Levy, and M. Takahashi; *Tree adjunct grammars*; J. Comput. System Sciences, **19**(1):136-163, 1975.

[K] D. E. Knuth; *Semantics of context-free languages*; Mathematical Systems Theory 2 (1968), 127-145.

[P] C. Pollard; *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*; Ph.D. thesis, Stanford University, CA, 1984.

[PS] M. A. Palis, S. M. Shende; *Pumping lemmas for the control hierarchy*; to appear in Math. Systems Theory.

[RS] O. Rambow, G. Satta; *A two-dimensional hierarchy for parallel rewriting systems*; submitted to Theor. Comp. Science.

[RS94] O. Rambow, G. Satta; *A rank hierarchy for deterministic tree-walking transducers*; Proc. CAAP '94, Lecture Notes in Computer Science 787, 308-321.

[R] J. C. Raoult; *A survey of tree transductions*; in 'Tree automata and languages', M. Nivat, A. Podelski, editors, Elsevier, 1992.

[Ro] W. C. Rounds; *Mappings and Grammars on trees*; Math. Systems theory 4 (1970), 257-287.

[SMFK] H. Seki, T. Matsumura, M. Fujii, T. Kasami; *On multiple context-free grammars*; Theor. Comp. Science 88 (1991), 191-229.

[S85] M. J. Steedman; *Dependency and coordination in the grammar of Dutch and English*; Language, **61**: 523-568, 1985.

[S86] M. J. Steedman; *Combinators and grammars*; In: R. Oehrle, E. Bach, and D. Wheeler, editors; Categorial Grammars and Natural Language Structures, 417-442, Foris, Dordrecht, 1986.

[T73] J. W. Thatcher; *Tree automata: an informal survey*; in 'Currents in the Theory of Computing' (A. V. Aho, Ed.),143-172, Prentice Hall, Englewood Cliffs, 1973.

[T67] J. W. Thatcher; *Characterizing derivation trees of context-free grammars through a generalization of finite automata theory*; J. Comput. System Sci. 1 (1967), 317-322.

[TW] J. W. Thatcher, J. B. Wright; *Generalized finite automata theory with an application to a decision problem of second-order logic*; Math. Systems Theory 2 (1968), 57-81.

[V] K. Vijay-Shanker; *A study of tree adjoining grammars*; Ph.D.thesis, University of Pennsylvania, Philadelphia, PA, 1987.

[VW] K. Vijay-Shanker, D. J. Weir; *The equivalence of four extensions of context-free grammars*; Math. Systems Theory 27 (1994), 511-546.

[W] D. J. Weir; *A geometric hierarchy beyond context-free languages*; Theor. Comp. Science 104 (1992), 235-261.