P(figure 2.6)	=	1/20 · 1/4 · 1/4	= 1/320
P(figure 2.7)	=	1/20 · 1/4 · 1/2	= 1/160
P(figure 2.8)	=	2/20 · 1/4 · 1/8 · 1/4	= 1/1280

table 2.1

This table shows that a model which defines probabilities over parse trees by taking into account only one derivation, does not accommodate the frequencies of all subtrees that may contribute to the generation of a parse tree. By taking into account the probabilities of all derivations of a parse tree, no subtree that might possibly be of statistical interest is ignored. How can we compute the probability of a parse tree? The probability of a parse is equal to the probability that it is generated by any of its derivations. Since these derivations are mutually exclusive, the probability of a parse is the sum of the probabilities of all its derivations. (This marks the difference with normal "stochastic grammars", where no distinction is made between the probability of a parse tree and the probability of a derivation which generates that tree; cf. §3.3-4). The calculation of the probability of the above parse tree for "*Mary likes Susan*" is left to the reader. Finally, the probability of a sentence or string is the sum of the probabilities of all its parses.

We want to conclude this chapter with an emerging property of DOP1, which will turn out to be of interest for the rest of this thesis. In DOP1, the probability of a parse depends on all derivations that generate that parse; therefore, the more different ways in which a parse can be generated, the higher its probability tends to be; this implies that a parse which can (also) be generated by relatively large subtrees tends to be favored over a parse which can only be generated by relatively small subtrees. Thus, given a sentence, there is a preference for the parse which can be generated by the largest possible subtrees.

Chapter 3

Towards a Formal Language Theory of Stochastic Grammars

In this chapter, we develop a theory in which the properties of stochastic grammars can be formally articulated and compared. We describe DOP1 as a projection of a corpus of tree structures into a Stochastic Tree-Substitution Grammar (STSG), and we formally compare STSG with other stochastic grammars.

3.1 Formal Stochastic Language Theory

The notion of a stochastic grammar usually refers to a finite specification of possibly infinitely many strings and their analyses together with their probabilities. If we want to compare the formal properties of different stochastic grammars, we need a Formal Stochastic Language Theory. In such a theory, we are not so much concerned with weak and strong generative capacity (as is the case in traditional Formal Language Theory), but with weak and strong *stochastic* generative capacity. The following definitions are therefore convenient.

Definitions:

The **stochastic string language** generated by a stochastic grammar G is the set of pairs $<x, p(x)>$ where x is a string from the string language generated by G and $p(x)$ the probability of that string.

The **stochastic tree language** generated by a stochastic grammar G is the set of pairs $<x, p(x)>$ where x is a tree from the tree language generated by G and $p(x)$ the probability of that tree.

In analogy to weak and strong equivalence, we define the following equivalences for stochastic grammars:

Definitions:

Two stochastic grammars are called **weakly stochastically equivalent**[1], iff they generate the same stochastic string language.

Two stochastic grammars are called **strongly stochastically equivalent**[2], iff they generate the same stochastic tree language.

Note that if two stochastic grammars are weakly stochastically equivalent they are also weakly equivalent (i.e. they generate the same string language). Moreover, if two stochastic grammars are strongly stochastically equivalent they are also strongly equivalent (i.e. they generate the same tree language) and weakly stochastically equivalent.

[1] What we call weak stochastic equivalence is called simply "stochastic equivalence" in (Fu, 1974).
[2] In (Bod, 1993a), this type of equivalence is called "superstrong equivalence".

Now that we have mathematical notions for comparing the generative capacities of stochastic grammars, we want to exclude the pathological cases of *improper* and *infinitely ambiguous* grammars.

Definition Properness of Grammars

A grammar is called *proper* iff only such nonterminals can be generated whose further rewriting can eventually result in a string of terminals.

Example: the context free grammar $<\{S,A\}, \{a\}, S, \{S\rightarrow Sa, S\rightarrow a, S\rightarrow aA\}>$ is not proper, since there is a generation $S\rightarrow aA$ which can never result in a string of terminals.[3]

Definition Finite Ambiguity of Grammars

A grammar is called *finitely ambiguous* if there is no finite string that has infinitely many derivations.

Example: the context free grammar $<\{S\}, \{a\}, S, \{S\rightarrow S, S\rightarrow a\}>$ is not finitely ambiguous, since the string a has infinitely many derivations.

Convention
We will only deal with grammars that are proper and finitely ambiguous.

3.2 Stochastic Tree-Substitution Grammar

The way DOP1 combines subtrees into new trees and computes probabilities of derivations and parses may very well be described by what we will call a "Stochastic Tree-Substitution Grammar" (STSG):

[3] In (Jelinek et al., 1990), an algorithm is given that determines whether or not a grammar may be made proper by the elimination of rules (p. 31).

Definition Stochastic Tree-Substitution Grammar

A *Stochastic Tree-Substitution Grammar* G is a five-tuple $<V_N, V_T, S, R, P>$ where

V_N is a finite set of nonterminal symbols.

V_T is a finite set of terminal symbols.

$S \in V_N$ is the distinguished symbol.

R is a finite set of elementary trees whose top nodes and interior nodes are labeled by nonterminal symbols and whose yield nodes are labeled by terminal or nonterminal symbols.

P is a function which assigns to every elementary tree $t \in R$ a probability $p(t)$. For a tree t with a root α, $p(t)$ is interpreted as the probability of substituting t on α. We require, therefore, that $0 < p(t) \leq 1$ and $\sum_{t:root(t)=\alpha} p(t) = 1$.

Substitution

If t_1 and t_2 are trees such that the *leftmost nonterminal yield node* of t_1 is equal to the *root* of t_2, then $t_1 \circ t_2$ is the tree that results from substituting t_2 for this leftmost nonterminal yield node in t_1. The partial function $\circ$ is called *leftmost substitution*. We will write $(t_1 \circ t_2) \circ t_3$ as $t_1 \circ t_2 \circ t_3$, and in general $(..((t_1 \circ t_2) \circ t_3) \circ ..) \circ t_n$ as $t_1 \circ t_2 \circ t_3 \circ ... \circ t_n$. For reasons of conciseness we will use the term substitution for leftmost substitution. Notice that the value $p(t)$ for an elementary tree with root α is the probability of substituting t for any nonterminal leaf node α in any elementary tree in R.

Derivation

A *leftmost derivation* generated by an STSG G is a tuple of trees $<t_1,...,t_n>$ such that $t_1,...,t_n$ are elements of R, the root of t_1 is labeled by S and the yield of $t_1 \circ ... \circ t_n$

is labeled by terminal symbols. The set of leftmost derivations generated by G is thus given by $Derivations(G) = \{<t_1,...,t_n> \mid t_1,...,t_n \in R \wedge root(t_1) = S \wedge yield(t_1 \circ ... \circ t_n) \in V_T^+\}$. For convenience we will use the term derivation for leftmost derivation. A derivation $<t_1,...,t_n>$ is called a derivation of tree T, iff $t_1 \circ ... \circ t_n = T$. A derivation $<t_1,...,t_n>$ is called a derivation of string s, iff $yield(t_1 \circ ... \circ t_n) = s$. The probability of a derivation $<t_1,...,t_n>$ is defined as $p(t_1) \cdot ... \cdot p(t_n)$.

Parse tree

A *parse tree* generated by an STSG G is a tree T such that there is a derivation $<t_1,...,t_n> \in Derivations(G)$ for which $t_1 \circ ... \circ t_n = T$. The set of parse trees, or *tree language*, generated by G is given by $Parses(G) = \{T \mid \exists <t_1,...,t_n> \in Derivations(G) : t_1 \circ ... \circ t_n = T\}$. For reasons of conciseness we will often use the terms *parse* or *tree* for a parse tree. A parse whose yield is equal to string s, is called a parse of s. The probability of a parse is defined as the sum of the probabilities of all its derivations.

String

A *string* generated by an STSG G is an element of V_T^+ such that there is a parse generated by G whose yield is equal to the string. The set of strings, or *string language*, generated by G is given by $Strings(G) = \{s \mid \exists\ T : T \in Parses(G) \wedge s = yield(T)\}$. The probability of a string is defined as the sum of the probabilities of all its parses. This means that the probability of a string is also equal to the sum of the probabilities of all its derivations.

It may be evident that STSG is a generalization over DOP1: the model DOP1 projects a corpus of tree structures into an STSG, where the subtrees of DOP1 are the elementary trees of the STSG, and where the substitution probabilities of the subtrees of DOP1 are the probabilities of the corresponding elementary trees of the STSG.

3.3 A Comparison between Stochastic Tree-Substitution Grammar and Stochastic Context-Free Grammar

The oldest and most well-known of all stochastic enrichments of context-free grammars is the so-called "Stochastic Context-Free Grammar" or SCFG (Booth, 1969; Suppes, 1970). An SCFG enriches every rewrite rule of a CFG with a probability which corresponds to the application probability of this rule. In an SCFG, the stochastic dependences are limited to the scope of single rewrite rules. It may be clear that SCFGs run into serious trouble if faced with solving ambiguities that are beyond the scope of single rewrite rules. It is therefore almost evident that SCFGs are stochastically weaker than STSGs. However, as an example of how Formal Stochastic Language Theory may be used to formally articulate this, we will compare SCFG and STSG in the context of this theory. Let us start with the definition of SCFG[4].

Definition Stochastic Context-Free Grammar

A *Stochastic Context-Free Grammar G* is a five-tuple $<V_N, V_T, S, R, P>$ where

V_N is a finite set of nonterminal symbols.

V_T is a finite set of terminal symbols.

$S \in V_N$ is the distinguished symbol.

R is a finite set of productions each of which is of the form $\alpha \rightarrow \beta$, where $\alpha \in V_N$ and $\beta \in (V_N \cup V_T)^+$.

P is a function which assigns to every production $\alpha \rightarrow \beta \in R$ a probability $p(\alpha \rightarrow \beta)$, for which holds that $0 < p(\alpha \rightarrow \beta) \leq 1$ and $\Sigma_x \, p(\alpha \rightarrow x) = 1$.

[4] This definition follows (Booth, 1969), (Fu, 1974), (Levelt, 1974), (Wetherell, 1980), (Fujisaki et al., 1989), (Jelinek et al. 1990).

The probability of a leftmost derivation (and its corresponding parse tree) generated by an SCFG is equal to the product of the probabilities associated with the productions applied. Note that, contrary to STSG, every parse tree is generated by exactly one leftmost derivation. The probability of a string generated by an SCFG is equal to the sum of the probabilities of all its derivations.

We will now compare STSG with SCFG in terms of respectively weak and strong stochastic equivalence.

Proposition 1
For every STSG there exists a weakly stochastically equivalent SCFG.

Proof of Proposition 1
Given an STSG G, we convert every elementary tree $t \in R$ into a context-free production $root(t) \rightarrow yield(t)$. This may lead to multiple occurrences of the same production, since different elementary trees may have the same root and yield. To every such production a probability is assigned which is equal to the probability of the tree from which the production is derived. In order to eliminate multiple occurrences of productions, we collapse equivalent productions and add up their probabilities. The resulting SCFG G' generates the same string language as G. It is now easy to see that the sum of the probabilities of all derivations of a string in G is equal to the sum of the probabilities of all derivations of this string in G'. This means that G and G' assign the same probability to every string in their string language. Thus, G and G' are weakly stochastically equivalent.
□

Proposition 2
For every SCFG there exists a weakly stochastically equivalent STSG.

Proof of Proposition 2
Given an SCFG G, we convert every production $\alpha \rightarrow \beta \in R$ into a unique elementary tree t of depth one such that $root(t) = \alpha$ and $yield(t) = \beta$. To every such tree a probability is assigned which is equal to the probability of the production from which the tree is derived. The resulting STSG G' generates the same string language and tree language as G. Now it is easy to see that for every

derivation in G there is a unique derivation in G' with the same probability. Thus, the sum of the probabilities of all derivations of a string in G is equal to the sum of the probabilities of all derivations of this string in G'. This means that G and G' assign the same probability to every string in their string language. Thus, G and G' are weakly stochastically equivalent.

□

From the propositions 1 and 2 the following corollary can be deduced.

Corollary 1
The set of stochastic string languages generated by STSGs is equal to the set of stochastic string languages generated by SCFGs.

Corrollary 1 is significant in the sense that if we were only interested in the strings and not in the trees (for instance for the task of string prediction in speech recognition output), we might convert an STSG (and thus a DOP1 model) into a more succinct SCFG.

Proposition 3
For every SCFG there exists a strongly stochastically equivalent STSG.

Proof of Proposition 3
Consider the proof of proposition 2. Since G and G' generate the same tree language and every derivation in G corresponds to a unique derivation in G' with the same probability, G and G' are strongly stochastically equivalent.

□

Proposition 4
There exists an STSG for which there is no strongly equivalent SCFG.

Proof of Proposition 4
Consider the following STSG G consisting of one elementary tree with a probability equal to 1:

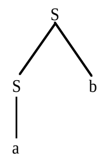

figure 3.1

The tree language generated by G is equal to the set containing only the above elementary tree. An SCFG is strongly equivalent with G if it generates only the above tree. An SCFG which generates the above tree should consist of the productions $S \rightarrow Sb$ and $S \rightarrow a$. But such an SCFG generates more than just the above tree. Contradiction.

□

Proposition 5
There exists an STSG for which there is no strongly stochastically equivalent SCFG.

Proof of Proposition 5
Consider the proof of proposition 4. Since strong stochastic equivalence implies strong equivalence there is no SCFG which is strongly stochastically equivalent with G.

□

From the propositions 3 and 5 the following corollary can be deduced.

Corollary 2
The set of stochastic tree languages generated by SCFGs is a proper subset of the set of stochastic tree languages generated by STSGs.

Though corollary 2 may seem a significant result, it mainly follows from the property that STSGs are not always strongly equivalent with SCFGs. In the context of stochastic language theory, however, we are not so much interested in tree languages as in *stochastic* tree languages. Thus, it is more interesting to compare stochastic tree languages of strongly equivalent grammars.

Proposition 6
There exists an STSG for which there is a strongly equivalent SCFG but no strongly stochastically equivalent SCFG.

Proof of Proposition 6
Consider the following STSG G consisting of three elementary trees that are all assigned with a probability of 1/3.[5]

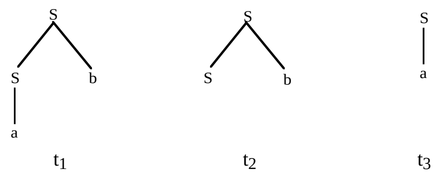

figure 3.2

The string language generated by G is $\{ab^*\}$. The only (proper) SCFG G' which is strongly equivalent with G consists of the following productions.

$S \rightarrow Sb$ (1)
$S \rightarrow a$ (2)

G' is strongly stochastically equivalent with G iff it assigns the same probabilities to the parse trees in the tree language as assigned by G. Let us consider the probabilities of two trees generated by G, i.e. the trees represented by t_1 and t_3.[6] The tree represented by t_3 has exactly one derivation: by selecting the elementary tree t_3. The probability of generating this tree is hence equal to *1/3*. The tree represented by t_1 has two derivations: by selecting elementary tree t_1, or by combining the elementary trees t_2 and t_3. The probability of generating this tree is

5 This STSG is also interesting because it can be projected from a DOP1-model whose corpus of sentence-analyses consists only of tree t_1.
6 Note that the trees t_1 and t_3 are both elements of the set of (elementary) trees R of G and of the tree language generated by G.

equal to the sum of the probabilities of its two derivations, which is equal to $1/3 + 1/3 \cdot 1/3 = 4/9$.

If G' is strongly stochastically equivalent with G, then it should assign the probabilities 4/9 and 1/3 to the trees represented by t_1 and t_3 respectively. The tree t_3 is exhaustively generated by production *(2)*; thus the probability of this production should be equal to 1/3: $p(S \rightarrow a) = 1/3$. The tree t_1 is exhaustively generated by applying productions *(1)* and *(2)*; thus the product of the probabilities of these productions should be equal to 4/9: $p(S \rightarrow Sb) \cdot p(S \rightarrow a) = 4/9$. By substitution we get $p(S \rightarrow Sb) \cdot 1/3 = 4/9$, which implies that $p(S \rightarrow Sb) = 4/3$. This means that the probability of production *(1)* should be larger than 1, which is not allowed. Thus, G' cannot be made strongly stochastically equivalent with G.
□

The (proof of) proposition 6 is an important result since it shows that STSGs are not only stronger than SCFGs because there are STSGs for which there is no strongly equivalent SCFG, but that STSGs are really *stochastically* stronger, also with respect to SCFGs that might be strongly equivalent to STSGs. It makes also clear why STSGs are stronger: SCFGs cannot attach a probability to a structure larger than one rewrite rule, while STSGs can.

3.4 Other Stochastic Grammars

In this section, we informally compare STSG with two other stochastic language models: Stochastic History-Based Grammar and Stochastic Tree-Adjoining Grammar. These grammars have been proposed as alternatives to SCFG to overcome the stochastic context-insensitiveness of SCFG.

3.4.1 Stochastic History-Based Grammar (SHBG)

Stochastic History-Based Grammars (SHBG[7]) are developed in (Black et al., 1993; Black, Garside & Leech, 1993), though introduced earlier in (Smith, 1973). In SHBG, the probability of applying a rewrite rule in a leftmost derivation is

7 My abbreviation.

made conditional on the rules that were used before in that derivation. In (Black et al., 1993), it is said that SHBG provides "a very rich if not the richest model of context ever attempted in a probabilistic parsing model". However, the limitation to a leftmost derivation for conditionalizing the probability of a rule means that still not all possible stochastic dependences are captured.

Let us illustrate this with the sentence *The emaciated man starved*, of which an analysis is given in figure 3.3. The numbers in the figure refer to the order of applications of the rules in a leftmost derivation of this sentence.

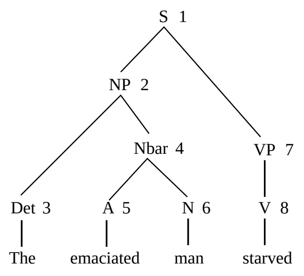

figure 3.3

Suppose that there is a strong stochastic dependence between the words *emaciated* and *starved*, appearing in sentences like the one above, and that these words are largely independent of the words surrounding them (in this case *The* and *man*). An adequate stochastic grammar should be able to account for this specific dependence between *emaciated* and *starved*. It turns out that SHBG is not able to do so. In order to show this, let us explain with somewhat more detail the probabilistic background of SHBG. Suppose that the probability of rule 1 in figure 3.3 (i.e. $S \rightarrow NP\ VP$) is given by $p(1)$. Since in SHGB the rule probability is made conditional on the former rules in the leftmost derivation, the conditional probability of rule 2 is given by $p(2|1)$. The conditional probability of rule 3 is given by $p(3|2,1)$ and so forth. The probability of the whole analysis is equal to the product of the conditional probabilities of the rules: $p(1) \cdot p(2|1) \cdot p(3|2,1) \cdot \ldots \cdot p(8|7,6,5,4,3,2,1)$.

While SHBG can thus capture a dependence between all lexical items *The, emaciated, man* and *starved* together, there is no way to account for the specific dependence between *emaciated* and *starved*, without *the* and *man*. What would be needed are conditional rule probabilities like $p(8|7,5,4,2,1)$ where the probability of rule 8 is made conditional on all former rules except 6 and 3. SHBG does not account for such probabilities, due to the restriction to a leftmost derivation for conditionalizing the probabilities of rewrite rules. Even if a so-called "finite Markov history" is used, SHBG can only describe the relations between items like *starved* and *man*, *emaciated* and *man*, or *emaciated, man* and *starved*, but not between *emaciated* and *starved* alone, since *man* is produced after *emaciated* and before *starved* in a leftmost derivation. Moreover, any restriction to another canonical derivation (rightmost, leftcorner etc.) would yield analogous limitations.

In STSG, on the other hand, the dependence between *emaciated* and *starved* can be captured by an elementary tree in which *emaciated* and *starved* are the only lexical items, and where *the* and *man* are left out, as is shown in figure 3.4.

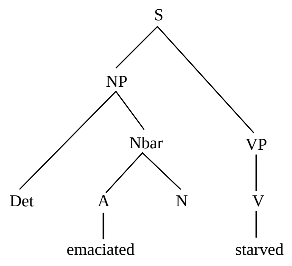

figure 3.4

This artificial example exemplifies a dependence which is strongly semantic in nature. An example which expresses a dependence of a more (semi-)idiomatic nature, is illustrated by the following sentence from the Air Travel Information System (ATIS) corpus (Hemphill et al., 1990): *Show me flights from Dallas to Atlanta.* The NP-construction *flights from X to Y* is almost idiomatic in the ATIS domain and occurs extremely frequently. It may be clear that, analogous to the previous example, SHBG can describe the dependences between all the words of such an NP, but it cannot attach a probability to the NP-construction where *Dallas* and *Atlanta* are left out. This is a serious shortcoming, since for the ambiguity resolution of a sentence which contains an NP like *flights from X to Y*, it is necessary to describe this NP as one statistical unit. STSG, on the other hand, can easily describe this NP as a statistical unit by taking the probability of this construction in the ATIS corpus.

3.4.2 Stochastic Tree-Adjoining Grammar (STAG)

Although STAG (Resnik, 1992; Schabes, 1992) is not a stochastic enrichment of a context-free grammar, but of a tree-adjoining grammar (Joshi, 1987) which belongs to the class of mildy context-sensitive grammars (Joshi et al., 1991), it is interesting to deal with STAG because of its similarity with STSG. An STAG assigns a probability to each elementary (initial or auxiliary) tree that corresponds to the probability that this elementary tree is combined by substitution or adjunction with another elementary tree. If we leave out the adjunction operation, STAG is formally equivalent with STSG. Thus, it looks as if STAG captures at least the stochastic dependences that can be captured by STSG. However, if we look at current instantiations of STAG, we find two serious shortcomings:

(1) Since STAG is linguistically motivated by tree-adjoining grammar (TAG), there are constraints on the form and use of elementary trees. For instance, modifiers are usually represented by separate auxiliary trees, which means that in analyzing the sentence *The emaciated man starved*, the modifier *emaciated* is inserted in the NP *the man* by means of adjunction. Linguistically this may be elegant, but statistically the dependence between *emaciated* and *starved* is lost, since they are not allowed to appear in one elementary tree.

(2) In current implementations of STAG, only the probability of a derivation is accounted for (cf. Resnik, 1992; Schabes, 1992), and not the probability of a resulting tree or of an interpretation. This is statistically inadequate, since, like in STSG, the probability of a derivation is different from the probability of a tree in STAG.

Thus, current instantiations of STAG seem still to be based on the assumption that the statistical dependences coincide with the linguistic dependences of the underlying competence model. In order to create an adequate performance model

based on TAG, it is not enough to attach probabilities to the competence units of this model. Instead, competence and performance need to be carefully distinguished, where the performance units must be taken as arbitrarily large trees from a corpus of analyzed language utterances.

3.5 Open problems

There are still many problems to be solved regarding the relations between stochastic grammars. So far, we have only designed the contours of a Formal Stochastic Language Theory which allowed us to formally compare STSG with SCFG. We believe that the following open problems need also to be treated within such a theory (whose solutions fall beyond the scope of this thesis).

* Does there exist a stochastic enrichment of CFG which is stochastically stronger than STSG? We haven't found one yet.

* Is there a stochastic hierarchy within the class of stochastic enrichments of CFGs, where SCFG is at the bottom, STSG at the top, and SHBG somewhere in between?

* If the former question can be answered positively, are there similar stochastic hierarchies in the other classes of the Chomsky hierarchy?