Expressive Power and Consistency Properties of State-of-the-Art Natural Language Parsers

Gabriel Infante-Lopez and Maarten de Rijke

Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{infante,mdr}@science.uva.nl

Abstract. We define Probabilistic Constrained W-grammars (PCW-grammars), a two-level formalism capable of capturing grammatical frameworks used in two state of the art parsers, namely bilexical grammars and stochastic tree substitution grammars. We provide embeddings of these parser formalisms into PCW-grammars, which allows us to derive properties about their expressive power and consistency, and relations between the formalisms studied.

1 Introduction

State of the art statistical natural language parsers, e.g., [1, 3, 4] are procedures for extracting the syntactic structure hidden in natural language sentences. Usually, statistical parsers have two clearly identifiable main components. One has to do with the nature of the set of syntactic analyses that the parser can provide. It is usually defined using a grammatical framework, such as probabilistic context free grammars (PCFGs), bilexical grammars, etc. The second component concerns the way in which the different parts in the grammatical formalism are learned. For example, PCFGs can be read from tree-banks and their probabilities estimated using maximum likelihood [11].

Clearly, the grammatical framework underlying a parser is a key component in the overall definition of the parser which determines important characteristics of the parser, either directly or indirectly. Among others, the grammatical framework defines the set of languages the parser can potentially deal with, a lower bound on the parser's complexity, and the type of items that should be learned by the second component mentioned above. Hence, a thorough understanding of the grammatical framework on which a parser is based provides a great deal of information about the parser itself. We are particularly interested in the following properties: (1) The expressive power of a grammar formalism. (2) Conditions under which the probability distribution defined over the set of possible syntactic analyses is consistent: if this is the case, the probabilities associated with an analysis can be used as meaningful probabilistic indicators both for further stages of processing [11] and for evaluation [7]. (3) The relation to other grammatical frameworks; this provides insights about the assumptions made by the various frameworks.

Since building a parser is a time consuming process, formal properties of the underlying grammatical framework are not always a priority. Also, comparisons between parser models are usually based on experimental evidence. In order to establish formal properties of parsers and to facilitate the comparison of parsers, we believe that a unifying grammatical framework, of which different parsers' grammars can be obtained as instances, is instrumental. Our main contribution in this paper is the introduction of a grammatical framework capable of capturing state of the art grammatical formalisms. Our framework is based on so-called W-grammars, due originally to Van Wijngaarden [13]. We constrain W-grammars to obtain CW-grammars, which are more suitable for statistical natural language parsing than W-grammars. PCW-grammars extend CW-grammars with probabilities. In this paper we provide embeddings of bilexical grammars [5] and stochastic tree substitution grammars [1] into PCW-grammars, and we use these embeddings to derive results on expressive power, consistency, and relations with other grammatical formalisms. Due to lack of space, embeddings of further grammatical formalisms have had to be omitted.

In Section 2 we present our grammatical framework and establish results on expressive power and conditions for inducing consistent distributions. In Section 3 we capture the models mentioned above in our framework, and derive consequences of the embeddings. In Section 4 we conclude.

2 Grammatical Framework

In this section we describe the grammatical framework we will be working with. We introduce constrained W-grammars, then present a probabilistic version, and also introduce technical notions needed in later sections.

2.1 Constrained W-Grammars

A *constrained W-grammar* (*CW-grammar*) is a 6-tuple (V, NT, T, S, $\xrightarrow{m}$, $\xrightarrow{s}$) such that:

- V is a set of symbols called *variables.* Elements in V are denoted with calligraphic characters, e.g., $\mathcal{A}, \mathcal{B}, \mathcal{C}$.
- NT is a set of symbols called *non-terminals*; elements in NT are denoted with upper-case letters, e.g., X, Y, Z.
- T is a set of symbols called *terminals*, denoted with lower-case letters, e.g.: a, b, c, such that V, T and NT are pairwise disjoint.
- S is an element of NT called *starting symbol.*
- $\xrightarrow{m}$ is a finite binary relation defined on $(V \cup NT \cup T)^*$ such that if $x \xrightarrow{m} y$, then $x \in V$. The elements of $\xrightarrow{m}$ are called *meta-rules.*
- $\xrightarrow{s}$ is a finite binary relation on $(V \cup NT \cup T)^*$ such that if $r \xrightarrow{s} s$ then $r \in NT$, $s \neq \epsilon$ and s does not have any variable appearing more than once. The elements of $\xrightarrow{s}$ are called *pseudo-rules.*

CW-grammars differ from Van Wijngaarden's original W-grammars in that pseudo-rules have been constrained. The original W-grammars allow pseudo-rules to have variables on the left-hand side as well as repeated variables on both the right- and left-hand side. The constrained version defined above yields a dramatic reduction in the expressive power of W-grammars, but, as we will see below, at the same time it allows us to capture state of the art parsers.

CW-Grammars are rewriting devices, and as such they consist of rewriting rules. They differ from the usual rewriting systems in that the rewriting rules do not exist a-priori. Using pseudo-rules and meta-rules one builds 'real' rules that can be used in the rewriting process. The rewriting rules produced are denoted by $\overset{w}{\Longrightarrow}$. These rules are built by first selecting a pseudo-rule, and then using meta-rules for instantiating all the variables the pseudo-rule might contain.

For example, let $W = (V, NT, T, S, \xrightarrow{m}, \xrightarrow{s})$ be a CW-grammar where $V = \{\mathcal{ADJ}\}$, $NT = \{S, Adj, Noun\}$, $T = \{ball, big, fat, red, green, \ldots\}$, while $\xrightarrow{m}$ and $\xrightarrow{s}$ are given by the following table:

meta-rules	pseudo-rules
$\mathcal{ADJ} \xrightarrow{m} \mathcal{ADJ}\, Adj$	$S \xrightarrow{s} \mathcal{ADJ}\, Noun$
$\mathcal{ADJ} \xrightarrow{m} Adj$	$Adj \xrightarrow{s} big$
	$Noun \xrightarrow{s} ball$

Suppose now that we want to build the rule $\mathcal{S} \overset{w}{\Longrightarrow} Adj\ Adj\ Noun$. We take the pseudo-rule $\mathcal{S} \xrightarrow{s} \mathcal{ADJ}\ Noun$ and instantiate the variable $\mathcal{ADJ}$ with $Adj\ Adj$ to get the desired rule. The rules defined by W have the following shape: $\mathcal{S} \overset{w}{\Longrightarrow} Adj^*\ Noun$. Trees for this grammar are flat, with a main node S and all the adjectives in it as daughters; see Figure 1.

The *string language* $L(W)$ generated by a CW-grammar W is the set $\{\beta \in T^+ : S \overset{w\,*}{\Longrightarrow} \beta\}$. In words, a string β belongs to the language $L(W)$ if there is a way to instantiate rules $\overset{w}{\Longrightarrow}$ that derive β from S. A *W-tree* yielding a string l is defined as the $\overset{w}{\Longrightarrow}$ derivation producing l. A W-tree 'pictures' the rules (i.e., pseudo-rules + variable instantiations) that have been used for deriving a string; Figure 1(a) shows an example. The way in which a rule has been obtained from pseudo-rules or the way in which its variables have been instantiated remains hidden. The *tree language* generated by a CW-grammar W is the set $T(G)$ defined by all W-trees generated by W yielding a string in $L(G)$.

Theorem 1. *CW-Grammars are weakly equivalent to context-free grammars.*

Proof. Let $W = (V, NT, T, S, \xrightarrow{m}, \xrightarrow{s})$ be a CW-grammar. Let $G_W = (NT', T', S', R')$ be a context-free grammar defined as follows (to avoid confusion we denote the rules in R by $\rightarrow$): $NT' = (V \cup NT)$; $T' = T$; S' is the starting symbol in W; and $X \rightarrow \alpha \in R$ iff $X \xrightarrow{m} \alpha$ or $X \xrightarrow{s} \alpha$. It can be shown that G_W is well-defined and generates the same language as W.

Given a CW-grammar W, the *context-free grammar underlying* W, notation $CFG(W)$, is the grammar G_W defined in the proof of Theorem 1. In Figure 1 we show a derivation in W and the corresponding one in $CFG(W)$.

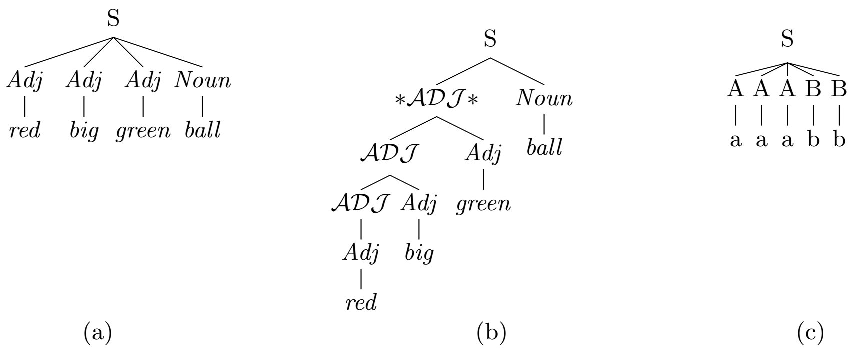

Fig. 1. (a) A tree generated by W. (b) The same tree with meta-rule derivations made visible. (c) A derivation tree for the string "*aaabb*"

Lemma 1. *Let W be a CW-grammar and let $G = CFG(W)$. For every τ in $T(G)$ there is a unique tree $v \in T(W)$ such that v is the product of hiding all meta derivations in τ.*

Proof. We sketch the proof using Figure 1. Suppose we want to derive the W-tree corresponding to the tree in Figure 1(a) from the one in Figure 1(b). Besides the G-tree we need to know which rules in G are meta-rules in W and which non-terminals in G are variables in W. To obtain a W-tree from a G-tree we replace all variables in CFG-rules (corresponding to pseudo-rules) by the yield of the CFG-derivation (corresponding to a meta-derivation). To illustrate this idea, consider the yield *red big green* below the variable $*\mathcal{ADJ}*$ in Figure 1(b): 'hide' the meta-derivation producing it, thus obtaining the tree in Figure 1(a), the W-tree. Since such a replacement procedure is uniquely defined, for every tree in $T(G)$ there is a unique way to hide meta-derivations, consequently for every G-tree there is a unique W-tree, as desired.

Next, we give an example to show that CW-grammars are *not* strongly equivalent to context-free grammars. In other words, trees generated by CW-grammars are different from trees generated by context-free grammars.

Example 1. Let $W = (V, NT, T, S, \xrightarrow{m\,*}, \xrightarrow{s\,*})$ a CW-grammar with $V = \{\mathcal{A}, \mathcal{B}, \mathcal{S}\}$, $NT = \{A, B\}$, $T = \{a, b\}$, $\xrightarrow{m} = \{\mathcal{A} \xrightarrow{m} \mathcal{A}A,\ \mathcal{A} \xrightarrow{m} A,\ \mathcal{B} \xrightarrow{m} \mathcal{B}B,\ \mathcal{B} \xrightarrow{m} B\}$, and $\xrightarrow{s} = \{A \xrightarrow{s} a,\ B \xrightarrow{s} b,\ \mathcal{S} \xrightarrow{s} \mathcal{A}\mathcal{B}\}$.

The grammar W generates the language $\{a^*b^*\}$ through instantiations of the variables A and B to strings in A^* and B^* respectively. The derivation $\overset{w}{\Longrightarrow}$ for a string *aaabb* is as follows: $S \overset{w}{\Longrightarrow} AAABB \overset{w}{\Longrightarrow} aAABB \overset{w}{\Longrightarrow} aaABB \overset{w}{\Longrightarrow} aaaBB \overset{w}{\Longrightarrow} aaabB \overset{w\,*}{\Longrightarrow} aaabb$. The tree representing this derivation (Figure 1(c)) has only one internal level (labeled $AAABB$), and its leaves form the accepted string. Observe that no CFG can generate the kind of flat structures displayed in Figure 1(c) since any context-free grammar producing the same language as W will have more than one intermediate level in its derivation trees.

2.2 Probabilistic CW-Grammars

Probabilistic CW-grammars (PCW-grammars, for short) are CW-grammars in which the rules are augmented with a probability value, such that the probabilities belonging to rules sharing the same left-hand side sum up to one. More formally, in a probabilistic CW-grammar $(V, NT, S, \xrightarrow{m}, \xrightarrow{s})$ we have that

- $\sum_{x \xrightarrow{m}_p y} p = 1$ for all meta-rules $x \xrightarrow{m}_p y$ having x in the left-hand side.
- $\sum_{x \xrightarrow{s}_p y} p = 1$ for all pseudo-rules $x \xrightarrow{s}_p y$ having x in the left-hand side.

Next, we need to define how we assign probabilities to derivations, rules, and trees. To start with derivations, if $\alpha' \xrightarrow{m\ *} \alpha$ then there are $\alpha_1, \ldots, \alpha_k$ such that $\alpha_i \xrightarrow{m} \alpha_{i+1}$, $\alpha_1 = \alpha'$ and $\alpha_k = \alpha$. We define the probability $P(\alpha' \xrightarrow{m\ *} \alpha)$ of a derivation $\alpha' \xrightarrow{m\ *} \alpha$ to be $\prod_{i=1}^{k-1} P(\alpha_i \xrightarrow{m} \alpha_{i+1})$.

Now, let $X \overset{w}{\Longrightarrow} \alpha$ be a rule. The probability $P(X \overset{w}{\Longrightarrow} \alpha)$ is defined as the product of $P(\alpha' \xrightarrow{m\ *} \alpha)$ and $\sum_{\alpha' \in A} P(X \xrightarrow{s} \alpha')$, where

$$A = \{\alpha' \in (V \cup NT \cup T)^+ : X \xrightarrow{s} \alpha', \alpha' \xrightarrow{m\ *} \alpha\}.$$

I.e., the probability of a 'real' rule is the sum of the probabilities of all meta derivations producing it.

The *probability of a tree* is defined as the product of the probabilities of the rules making up the tree, while the *probability of a string* $\alpha \in T^+$ is defined as the sum of the probabilities assigned to all trees yielding α.

Theorem 2. *Let W be a CW-grammar, let G be CFG(W), and let W' be a PCW-grammar that extends W by assigning probability values to all rules in W. There is a way to extend G into a PCFG G' such that W' and G' assigning the same probability mass to all strings in the language accepted by G (which coincides with the language accepted by W).*

Proof. Let $G = (NT', T', S', R')$ be a PCFG with NT', T', S' as defined in the proof of Theorem 1 and R' such that $X \to \alpha \in R$ iff $X \xrightarrow{m} \alpha$ or $X \xrightarrow{s} \alpha$. Note that a $\overset{w}{\Longrightarrow}$ derivation τ might be the product of many different derivations using rules in R' (G-derivations for short); call this set $D(\tau)$. From the definitions it is clear that $p(\tau) = \sum_{v \in D(\tau)} p(v)$. To prove the theorem we need to show (1) that for τ and τ' two different $\overset{w}{\Longrightarrow}$ derivations of the string α, it holds that $D(\tau) \cap D(\tau') = \emptyset$, and (2) that for every G-derivation v there is a $\overset{w}{\Longrightarrow}$ derivation τ such that $v \in D(\tau)$. Both results follow from Lemma 1.

For a given PCW-grammar W, the PCFG defined in the proof of Theorem 2 is called *the PCFG underlying W*. As an immediate consequence of the construction of the PCFG given in Theorem 2 we get that a PCW-grammar is *consistent* iff its underlying PCFG is consistent.

2.3 Learning CW-Grammars from Tree-Banks

PCW-grammars are induced from tree-banks in almost the same way as PCFGs are. The main difference is that the former require an explicit decision on the nature of the hidden derivations. As we will see below, the two different approaches to natural language parsing that we present in this paper differ substantially in the assumptions they make in this respect.

2.4 Some Further Technical Notions

Below we will use PCW-grammars to “capture” models underlying a number of state of the art parsers. The following will prove useful. Let F and G be two grammars with tree languages $T(G)$ and $T(F)$ and languages $L(F)$ and $L(G)$, respectively. Then, F is *f-equivalent* to G if $L(F) = L(G)$ and there is a bijective function $f : T(F) \to T(G)$. Given two grammatical formalisms A and B, we say that A is *f-transformable* to B, if for every grammar F in A there is a grammar G in B such that F is f-equivalent to G.

3 Capturing State of the Art Parsers

In this section we use PCW-grammars to capture the models underlying two state of the art parsers.

3.1 Bilexical Grammars

Bilexical grammar [4, 5] is a formalism in which lexical items, such as verbs and their arguments, can have idiosyncratic selectional influences on each other. They can be used for describing bilexical approaches to dependency and phrase-structure grammars, and a slight modification yields link grammars.

Background. A *split unweighted bilexical grammar* B is a 3-tuple $(W, \{r_w\}_{w \in W}, \{l_w\}_{w \in W})$ where:

- W is a set, called the (terminal) *vocabulary*, which contains a distinguished symbol ROOT
- For each word $w \in W$, a pair of regular grammars l_w and r_w, having starting symbols S_{l_w} and S_{r_w}, respectively. Each grammar accepts some regular subset of W^*.

A *dependency tree* is a tree whose nodes (internal and external) are labeled with words from W; the root is labeled with the symbol ROOT. The children (‘dependents’) of a node are ordered with respect to each other and the node itself, so that the node has both *left children* that precede it and *right children* that follow it. A dependency tree T is *grammatical* if for every word token w that appears in the tree, l_w accepts the (possibly empty) sequence of w’s left children (from right to left), and r_w accepts the sequence of w’s right children (from left to right). *Weighted bilexical grammars* are like unweighted bilexical grammars but all of their automata assign weights to the strings they generate. Lemma 2 implies that weighted bilexical grammars are a subset of PCW-grammars.

Bilexical Grammars as CW-Grammars. With every bilexical grammar B we associate a CW-grammar W_B as follows.

Definition 1. Let $B = (W, \{l_w\}_{w \in L}, \{r_w\}_{w \in L})$ be a split bilexical grammar. Let $W_B = (V, NT, T, S, \xrightarrow{m}, \xrightarrow{s})$ be the CW-grammar defined as follows:

- The set of variables V is given by the set of starting symbols S_{l_w} and S_{r_w} from regular grammars l_w and r_w respectively, and w in W.
- The set of non-terminals NT is some set in 1-1-correspondence with W, e.g., it can be defined as $NT = \{w' : w \in W\}$.
- The set of terminals T is the set of words W.
- The set of meta-rules is given by the union of $\{w' \xrightarrow{m} w : w \in W\}$ and the rules in all of the right and left regular grammars in B.
- The set of pseudo-rules is given by $X' \xrightarrow{s} S_{\bar{l}_w} x S_{r_w}$ where $\bar{l}_w$ denotes the regular expression inverting (reading backwards) all strings in $L(l_w)$.

Below, we establish the (weak) equivalence between a bilexical grammar B and its CW-grammar counterpart W_B. The idea is that the set of meta-rules, producing derivations that would remain hidden in the tree, are used for simulating the regular automata. Pseudo-rules are used as a nexus between a hidden derivation and a visible one: For each word w in the alphabet, we define a pseudo-rule having w as a terminal, and two variables S_{l_w} and S_{r_w} marking the left and right dependents, respectively. These variables correspond to the starting symbols for the left and right automata l_w and r_w, respectively. Instantiating the pseudo-rule associated to w would use a left and a right derivation using the left and the right automata, respectively, via meta-rules. The whole derivation remains hidden in the $\overset{w}{\Longrightarrow}$ derivation, as in bilexical grammars.

Lemma 2. *Bilexical grammars are f-transformable to CW-grammars.*

Proof. We have to give a function $f : T(B) \to T(W_B)$, where B is a bilexical grammar and W_B the grammar defined in Definition 1, such that f is invertible. A bilexical tree yielding the string $s = w_1, \ldots, w_n$ can be described as a sequence $u_1, \ldots, u_n$ of 3-tuples $\langle \alpha_i, w_i, \beta_i \rangle$ such that l_{w_i} accepts α_i and r_{w_i} accepts β. The desired function f transforms a dependency tree in a W-tree by transforming the sequence of tuples into a $\overset{w}{\Longrightarrow}$ derivation. We define f as $f(\langle \alpha, w_i, \beta \rangle) = W_i \overset{w}{\Longrightarrow} \alpha w_i \beta$. The rule corresponding to $\langle \alpha, w_i, \beta \rangle$ is the one produced by using the pseudo rule $W_i' \xrightarrow{s} S_{\bar{l}_w} x S_{r_w}$ and instantiating $S_{\bar{l}_w}$ and S_{r_w} with α and β respectively. Since the sequence of tuples forms a dependency tree, the sequence of W-rules builds up a correct W-tree.

Expressive Power and Consistency. By Lemma 2 bilexical grammars are weakly equivalent to context-free grammars. Moreover, the idea behind Example 1 can be used to prove that bilexical grammars are not strongly equivalent to CFGs. Briefly, bilexical grammars can create flat structures of the kind produced by the grammar in Example 1; such structures cannot be produced by CFGs.

As a consequence of Lemma 2, learning bilexical grammars is equivalent to learning PCW-grammars, which, in turn, is equivalent to learning the PCFGs underlying the PCW-grammars. Eisner [4] assumed that all hidden derivations were produced by Markov chains. Under the PCW-paradigm, his methodology is equivalent to transforming all trees in the training material by making all their hidden derivations visible, and inducing the underlying PCFG from the transformed trees. Variables in the equivalent PCW-grammar are defined according to the level degree of the Markov chain (we assume that the reader is familiar with Markov models and Markov chains [11]). In particular, if the Markov chain used is of degree one, variables are in one-to-one correspondence with the set of words, and the consistency result follows from the fact that inducing a degree one Markov chain in a bilexical grammar is the same as inducing the underlying PCFG in the equivalent PCW-grammar using maximum likelihood, plus the fact that using maximum likelihood estimation for inducing PCFGs produces consistent grammars [2, 8].

3.2 Stochastic Tree Substitution Grammars

Data-oriented parsing (DOP) is a memory-based approach to syntactic parsing. The basic idea is to use the subtrees from a syntactically annotated corpus directly as a stochastic grammar. The DOP-1 model [1] was the first version of DOP, and most later versions of DOP are variations on it. The underlying grammatical formalism is stochastic tree substitution grammars (STSG), which is the grammatical formalism we capture here.

Background. The model itself is extremely simple and can be described as follows: for every sentence in a parsed training corpus, extract every subtree. Then we use these trees to form an stochastic tree substitution grammar. Formally, a *stochastic tree-substitution grammar* (STSG) G is a 5-tuple $\langle V_N, V_T, S, R, P\rangle$ 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 node symbol $root(t) = \alpha$, $P(t)$ is interpreted as the probability of substituting t on a node α. We require, therefore, for a given α that $\sum_{\{t:root(t)=\alpha\}} = 1$ and that $0 < P(t) \leq 1$ (where t's root node symbol is α).

If t_1 and t_2 are elementary trees such that the leftmost non-terminal frontier node symbol of t_1 is equal to the root node symbol of t_2, then $t_1 \circ t_2$ is the tree that results from substituting t_2 in this leftmost non-terminal frontier node symbol in t_1. The partial function $\circ$ is called *leftmost substitution* or simply *substitution*. Trees are derived using left most substitution.

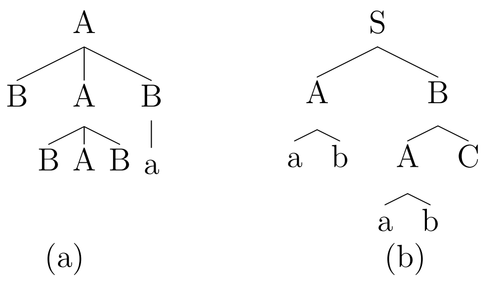

Fig. 2. (a) A derivation tree (b) An elementary tree

STSGs as CW-Grammars. STSGs are not quite context-free grammars. The main difference, and the hardest to capture in a CFG-like setting, is the way in which probabilities are computed for a given tree. The probability of a tree is given by the sum of the probabilities of all derivations producing it. CW-grammars offer a similar mechanism: the probability of the body of a rule is the sum of the probabilities of all meta-derivations producing it. The idea of the equivalence is to associate to every tree produced by a STSG a 'real' rule of the PCW-grammar in such a way that the body of the rule codifies the whole tree.

To implement this idea, we need to code up trees as strings. The simplest way to achieve this is to visit the nodes in a depth first left to right order and for each inner node use the applied production, while for the leaves we type the symbol itself if the symbol is a terminal and a primed version of it if the symbol is a non-terminal. For example, the derivation describing the tree in Figure 2(a) is $(A, BAB)B'(A, BAB)B'A'B'(B, a)a$.

The first step in capturing STSGs is to build rules capturing elementary trees using the notation just introduced. Specifically, let t be an elementary tree belonging to a STSG. Let S be its root and α its string representation. The CF-like rule $S' \rightarrow \alpha$ is called the *elementary rule* of t. Elementary rules store all information about the elementary tree. They have primed non-terminals where a substitution can be carried out. E.g., if t is the elementary tree pictured in Figure 2(b), its elementary rule is $S' \rightarrow (A, B)(A, ab)ab(B, AC)(A, ab)abC'$. Note the primed version of C in the frontier of the derivation.

Definition 2. Let $H = (V_N, V_T, S, R, P)$ be a STSG. Let $W_H = (V, NT, T, S', \xrightarrow{m}, \xrightarrow{s})$ be the following CW-grammar.

- V is the primed version of V_T.
- (A, α) is in NT iff $(A, \alpha) \rightarrow \epsilon$ appears in some elementary tree.
- T is exactly as V_T.
- S' is a new symbol.
- The set of meta-rules is built by transforming each elementary tree to its corresponding elementary rule.
- The set of pseudo-rules is given by $(A, \alpha) \xrightarrow{s} \epsilon$ if $A \rightarrow \alpha$ appears in a elementary tree, plus rules $S' \xrightarrow{s} S$.

Two remarks are in order. First, all generative capacity is encoded in the set of meta-rules. In the CW-world, the body of a rule (i.e., an instantiated pseudo-rule) encodes a derivation of the STSG. Second, the probability of a 'real' rule is the sum of the probabilities of meta-derivations yielding the rule's body.

Lemma 3. *STSGs are f-transformable to CW-grammars, with f invertible.*

Proof. Let $H = (V_N, V_T, S, R, P)$ be a STSG and let W_H be the CW-grammar given in Definition 2. Let t be a tree produced by H. We prove the lemma using induction on the length of the derivation producing t. If t has length 1, there is an elementary tree t_1 such that S is the root node and yields α, which implies that there is a meta-rule obtained from the elementary rule corresponding to the elementary tree t_1. The relation is one-to-one as, by definition, meta-rules are in one-to-one correspondence with elementary trees.

Suppose the lemma is true for derivation lengths less than or equal to n. Suppose t is generated by a derivation of length $n + 1$. Assume there are trees t_1, t_2 with $t_1 \circ t_2 = t$. By definition there is a unique meta-rule r_1 corresponding with t_1 and by inductive hypothesis there is a unique derivation for t_2.

Corollary 1. *Let $H = (V_N, V_T, S, R, P)$ be an STSG, and let W_H be the CW-grammar given in Definition 2. There is a one-to-one correspondence between derivations in H and W_H.*

Lemma 4. *Let $H = (V_N, V_T, S, R, P)$ be an STSG, and let W_H be the CW-grammar given in Definition 2. Both grammars assign the same probability mass to trees related through the one-to-one mapping described in Lemma 1.*

Proof. A tree has a characteristic W-rule, defined by its shape. Moreover probability of a W-rule, according to the definition of PCW-grammars, is given by the sum of the probabilities of all derivations producing the rule's body, i.e., all STSG derivations producing the same tree. As a consequence, a particular STSG tree, identified from the body of the corresponding W-rule, has the same probability assigned by the equivalent CW-Grammar.

Expressive Power and Consistency. By Corollary 3, STSGs are weakly equivalent to context-free grammars. The consistency of an STSG depends on the methodology used for computing the probabilities assigned to its elementary trees. DOP-1 is one particular approach to computing these probabilities. Under the DOP-1 perspective, a tree t contributes all its possible subtrees to a new tree-bank from which the probabilities of elementary trees are computed. Probabilities of an elementary tree are computed using maximum likelihood. Since the events in the new tree-bank are not independently distributed, the resulting probabilities are inconsistent and biased [9]. Solutions taking into account the dependence between trees in the resulting tree-banks have been suggested [12].

Even though consistency conditions cannot be derived for the DOP-1 estimation procedure given that it does not attempt to learn the underlying PCFG, our formalism suggests that probabilities should be computed differently from the way it is done in DOP-1. By our embedding, a tree t in the tree-bank corresponds to the body of a pseudo-rule instantiated through meta-derivations; t is the final "string" and does not have any information on the derivation that took place. But viewing t as a final string changes the problem definition! Now, we have as input a set of elementary rules and a set of accepted trees. The problem is to compute probabilities for these rules: an unsupervised problem that can be solved using any unsupervised technique. The consistency of the resulting STSG depends on the consistency properties of the unsupervised method.

4 Discussion and Conclusion

We introduced probabilistic constrained W-grammars, a grammatical framework capable of capturing a number of models underlying state of the art parsers. We established expressive power properties for two formalisms (bilexical grammars, and stochastic tree substitution grammars) together with some conditions under which the inferred grammars are consistent. We should point out that, despite their similarities, there is a fundamental difference between PCW-grammars and PCFGs, and this is the two-level mechanism of the former formalism. This mechanism allows us to capture two state of the art natural language parsers, which cannot be done using standard PCFGs only.

We showed that, from a formal perspective, STSGs and bilexical grammars share certain similarities. Bilexical grammars suppose that rule bodies are obtained by collapsing hidden derivations. That is, for Eisner, a rule body is a regular expression. Similarly, Bod's STSGs take this idea to the extreme by taking the whole sentence to be the yield of a hidden derivation. PCW-grammars naturally suggest different levels of abstraction; in [6] we have shown that these levels can be used to reduce the size of grammars induced from tree-banks, and, hence, to optimize parsing procedures.

From a theoretical point of view, the concept of f-transformable grammars, which we use heavily in our proofs, is a very powerful one that relaxes the known equivalence notions between grammars. Since arbitrary functions f can be defined between arbitrary tree languages and CFG-like trees, they can be used to map other formalisms to context-free trees. Examples include Collins' first model (based on Markov rules) [3], Tree Adjoining Grammars [10] or Categorial Grammars [14]. As part of our future research, we aim to capture further grammatical formalisms, and to characterize the nature of the functions f used to achieve this.

Acknowledgments

Both authors were supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. In addition, Maarten de Rijke was also supported by grants from NWO, under project numbers 365-20-005, 612.069.006, 612.000.106, 612.000.207, and 612.066.302.

References

1. R. Bod. *Beyond Grammar—An Experience-Based Theory of Language.* Cambridge University Press, Cambridge, England, 1999.
2. Z. Chi and S. Geman. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305, 1998.
3. M. Collins. Three generative, lexicalized models for statistical parsing. In *Proc. ACL'97 and EACL'97*, pages 16–23, Madrid, Spain, 1997.
4. J. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-96*, pages 340–245, Copenhagen, Denmark, 1996.
5. J. Eisner. Bilexical grammars and their cubic-time parsing algorithms. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers, October 2000.
6. G. Infante-Lopez and M. de Rijke. Alternative approaches for generating bodies of grammar rules. In *Proc. ACL'04*, Barcelona, Spain, 2004.
7. G. Infante-Lopez and M. de Rijke. Comparing the ambiguity reduction abilities of probabilistic context-free grammars. In *Proc. of LREC'04*, 2004.
8. S. Joan-Andreu and J.-M. Benedí. Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):1052–1055, 1997.
9. M. Johnson. The DOP estimation method is biased and inconsistent. *Computational Linguistics*, 28(1):71–76, 2002.
10. A.K. Joshi. Tree Adjoining Grammars: How much context sensitivity is required to provide a reasonable structural description. In I. Karttunen D. Dowty and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250, Cambridge, U.K., 1985. Cambridge University Press.
11. C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, MA, 1999.
12. K. Sima'an and L. Buratto. Backoff parameter estimation for the DOP model. In *Proc. of ECML*, 2003.
13. A. van Wijngaarden. Orthogonal design and description of a formal language. Technical Report MR76, Mathematisch Centrum., Amsterdam, 1965.
14. M. Wood. *Categorial Grammars.* Routledge., London, 1993.