A New General Grammar Formalism for Parsing

Gabriel Infante-Lopez and Martín Ariel Domínguez

Grupo de Procesamiento de Lenguaje Natural
Universidad Nacional de Córdoba - Argentina
{gabriel,mdoming}@famaf.unc.edu.ar

Abstract. We introduce Probabilistic Constrained W-grammars (PCW-grammars), a two-level formalism capable of capturing grammatical frameworks used in three different state of the art grammar formalism, namely Bilexical Grammars, Markov Rules, and Stochastic Tree Substitution Grammars. For each of them we provide an embedding into PCW-grammars, which allows us to derive properties about their expressive power and consistency, and relations between the formalisms studied.

Keywords: Parsing, PCW-grammars, Bilexical Grammars.

1 Introduction

State of the art statistical parsers, e.g., [5,8,3,15] 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 the nature of the set of syntactic analysis 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, for supervised parsers, PCFGs can be read from tree-banks and their probabilities estimated using maximum likelihood [16]. In contrast, for unsupervised dependency parsing [14,1,18], the grammar formalism is inferred from a non-annotated corpus of sentences.

Clearly, the grammatical framework underlying a parser is a key component in the overall definition of the parser. Important characteristics of the parser are determined, either directly or indirectly, through its grammatical formalism; among others, the grammatical framework defines the set of languages the parser can potentially deal with, an upper bound for the parser's complexity, and the type of items that should be learned by the second component. 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 analysis is consistent: if this is the case, the probabilities associated to an analysis can be used as meaningful probabilistic indicators both for further stages of processing [16] and for evaluation [10]. (3) The relation to other grammatical frameworks; this provides insights on 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 is the introduction of a grammatical framework capable of capturing three state of the art grammatical formalisms, namely Bilexical Grammars [9,7,18], Markov Rules [5], and Stochastic Tree Substitution Grammars [3,1]. Our framework is based on so-called W-grammars, due originally to [19]. 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. For each the three formalisms, we provide an embedding in PCW-grammars and use this embedding to derive results on expressive power, consistency, and relations among the formalisms.

Section 2 presents our grammatical framework and proves expressive power and conditions to induce consistent distributions. In Section 3 we capture the models mentioned above in our framework, and derive consequences of the embeddings. In Section 4 we discuss the results and 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 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 noted 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*.

PCW-Grammars are rewriting devices, and as such they consist of rewriting rules. They differ from usual rewriting systems in that the rewriting rules do not exist a-priori. Pseudo-rules and meta-rules provide a mechanisms for building 'real' rules that can be used in the rewriting process. The rewriting rules are denoted by $\xRightarrow{w}$. These rules are built by first selecting a pseudo-rule, and then using meta-rules for instantiating all the variables the pseudo-rule might contains.

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

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

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) has an example. The way in which the rule has been obtained from pseudo-rules or the way the variables have been instantiated remains hidden. The *tree language* generated by 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 Theorem 1. In Figure 1 we show a derivation in W and the corresponding one in $CFG(W)$.

Lemma 1. *Let W be a CW-grammar and let $G = CFG(W)$. For every τ in $T(G)$ there is a unique tree $\upsilon \in T(W)$.*

Proof. We sketch the proof using Figure 1. From the picture it is clear that there is a unique way to hide meta-derivations. The proof follows from applying this simple idea to all trees in $T(G)$.

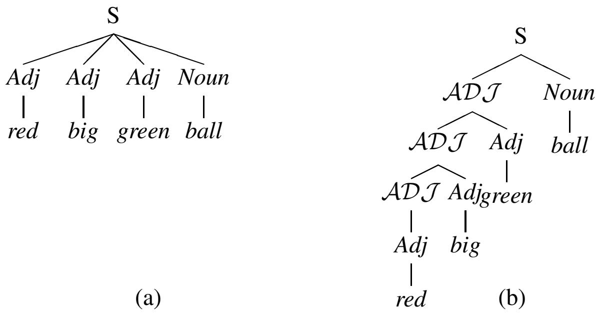


Fig. 1. (a) A tree generated by W. (b) The same tree with meta-rule derivations made visible

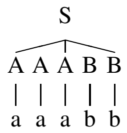


Fig. 2. A derivation tree for the string "$aaabb$"

A derivation produced by the CFG underlying W has all meta and pseudo derivations visible. Clearly for every derivation in the CFG underlying v, there is a unique $\overset{w}{\Longrightarrow}$ derivation, and it is obtained by replacing all internal nodes corresponding to head symbols of meta-rules by the yield of the meta-rule production.

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 $W = (V, NT, T, S, \overset{m\ *}{\longrightarrow}, \overset{s\ *}{\longrightarrow})$ a CW-grammar with $V = \{\mathcal{A}, \mathcal{B}, \mathcal{S}\}$, $NT = \{A, B\}$, $T = \{a, b\}$, $\overset{m}{\longrightarrow} = \{\mathcal{A} \overset{m}{\longrightarrow} \mathcal{A}A, \mathcal{A} \overset{m}{\longrightarrow} A, \mathcal{B} \overset{m}{\longrightarrow} \mathcal{B}B, \mathcal{B} \overset{m}{\longrightarrow} B\}$, and $\overset{s}{\longrightarrow} = \{A \overset{s}{\longrightarrow} a, B \overset{s}{\longrightarrow} b, \mathcal{S} \overset{s}{\longrightarrow} \mathcal{A}\mathcal{B}\}$.
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, pictured in Figure 2, only has one internal level (labeled $AAABB$), and its leaves form the accepted string.

No context-free grammar can generate the kind of flat structures displayed in Figure 2 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 where the rules are augmented with a probability value, such that the probabilities belonging to rules sharing the some left-hand side sum up to one. More formally, in a probabilistic CW-grammar $(V, NT, S, \overset{m}{\longrightarrow}, \overset{s}{\longrightarrow})$ 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 the former, 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. *For every PCW-grammar there is a PCF grammar assigning the same probability mass to all strings in the language.*

Proof. Let $G = (NT', T', S', R')$ be a PCFG with NT', T', S' as defined in the proof of Theorem 2 and R' such that $X \rightarrow \alpha \in R$ 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); let us 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 $\overset{w}{\Longrightarrow}$ 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 PCFG are. The main difference is that the former requires an explicit decision on the nature of the hidden derivations. As we will see, the three different approaches we present, differ substantially in the assumptions they made in this respect.

2.4 Some Further Technical Notions

In later sections we will be using 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. We say that F is *f-equivalent* to G if $L(G) = L(T)$ and there is a bijective function $f : T(F) \rightarrow T(G)$. 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 a number of state of the art parsers.

3.1 Bilexical Grammars

Bilexical grammars [8,9,7,18] 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_{r_w} and S_{l_w}. 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).

Example 2. Let $B = (W, \{l_w\}_{w \in L}\}, \{r_w\}_{w \in L})$ be a split regular grammar defined as follows: $W = \{a, b\}$, $l_a = b^*$, $r_a = \epsilon$, $l_b = \epsilon$, $l_{ROOT} = a$, $r_{ROOT} = \epsilon$ and $r_b = (a|b)^*$. This grammar accepts the string "bbabaa" with the analysis tree pictured in Figure 3. Because l_{ROOT} accepts a, l_a accepts "bbb", l_a accepts ϵ, l_b accepts ϵ, and r_b accepts "a".

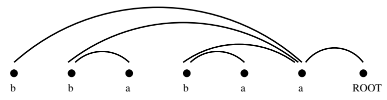


Fig. 3. An example of a dependency tree

Bilexical Grammars as CW-Grammars

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 regulars 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.*
- *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 $f : T(B) \rightarrow T(W_B)$, 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 $alpha_i$ 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.

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 subclass of PCW-grammars.

Properties. By Lemma 2 bilexical grammars are weakly equivalent to context free grammars. Moreover, Example 1 can be used to show that bilexical grammars are not strongly equivalent to CFGs; hence, the probabilistic version of bilexical grammars cannot be captured by PCFGs.

As a consequence of Lemma 2, learning bilexical grammars is equivalent to learning PCW-grammars, and since every PCW-grammar has an underlying PCFG, learning a PCW-grammar is equivalent to learning its underlying PCFG. [8] 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 [13]). In particular, if the Markov chain used is of degree one, variables are in one-to-one correspondence with the set of words, and it can be shown that the resulting bilexical grammar is consistent. The 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 PCWG using maximum likelihood, plus the fact that using maximum likelihood for inducing PCFG produces consistent grammars [11,4].

3.2 Markov Rules

In this subsection we capture one of the models presented by Collins, his so-called first model. The main idea behind [5,6] is to extend what he calls a "simple" CFG to a lexicalized backed-off grammar.

Background. Collins' first model may be viewed as a way to describe the probabilities assigned to CF-like rules. A rule has the following shape:

$$P(h) \rightarrow L_n(l_n) \ldots L_1(l_1) H(h) R_1(r_1) \ldots R_m(r_m), \tag{1}$$

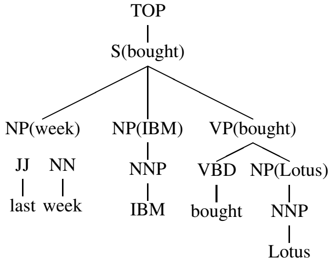
where H is the head-child of the phrase, which inherits the head word h from its parent P, and where $L_n(l_n), \ldots, L_1(l_1)$ and $R_1(r_1), \ldots, R_m(r_m)$ are left and right modifiers of H, respectively. Either or both of n and m may be zero, and $n = m = 0$ for unary rules. Figure 4 shows a tree with its respective rules.

Collins defines the probability of a rule (such as (1)) as the probability of its right-hand side, conditioned on the probability of its left-hand side, which is then decomposed as follows:

$$\begin{aligned}&\mathcal{P}(L_n(l_n) \ldots L_1(l_1) H(h) R_1(r_1) \ldots R_m(r_m) | P(h)) = \\ &\mathcal{P}_h(H|P(h)) \times \prod_{i=1,\ldots,n+1} \mathcal{P}_l(L_i(l_i)|L_1(l_1), \ldots, L_{i-1}(l_{i-1}), P(h), H) \times \\ &\prod_{j=1,\ldots,m+1} \mathcal{P}_r(R_j(r_j)|L_1(l_1), \ldots, L_{n+1}, R_1(r_1), \ldots, R_{j-1}(r_{j-1}), P(h), H).\end{aligned}$$

Collins approximates the probabilities using Markov independence assumptions for each order. In particular, the generation of the right-hand side of a rule as in (1), given the left-hand side, is decomposed into three steps:

1. Generate the head constituent label of the phrase, with probability $\mathcal{P}_H(H|P, h)$.
2. Generate modifiers to the left of the head with probability $\prod_{i=1\ldots n+1} \mathcal{P}_L(L_i(l_1)|P, h, H)$, where $L_{n+1}(l_{n+1}) = \texttt{STOP}$. The STOP symbol is a non-terminal, and the model stops generating left modifiers as soon as it has been generated.



```
      TOP → S(bought)
S(bought) → NP(week) NP(IBM) VP(bought)
 NP(week) → JJ(last) NN(week)
  NP(IBM) → NNP(IBM)
VP(bought) → VBD(bought) NP(Lotus)
 NP(Lotus) → NNP(Lotus)
```

Fig. 4. A lexicalized parse tree and the rules it contains; POS tags omitted for brevity

3. Generate modifiers to the right of the head with probability $\prod_{i=i\dots m+1} \mathcal{P}_R(R_i(r_i)|P,h,H)$. $R_{m+1}(r_{m+1})$ is defined as STOP.

We can think of the probabilities $\mathcal{P}_R(R_i(r_i)|P,h,H)$ and $\mathcal{P}_L(L_i(r_i)|P,h,H)$ as being produced by a zero order Markov chain.

Markov Rules as CW-grammars. For capturing bilexical grammars, we first described the formalism using regular language and later added probabilities. To capture Collins' first model we proceed in the opposite direction. We use the zero-order Markov model he builds to define regular languages and use these to build a CW grammar corresponding to Collins' model.

Independent of their order, *Markov models* or *Markov chains* describe a regular language. Let $M = (S, P, F, I)$ be a Markov chain, where S is a sequence of states, P the transition matrix, $F \subseteq S$ the set of absorbing states, and I the initial distribution of probabilities. We can transform the Markov model into an automaton A_M directly by taking S as the states of the automaton, F as the set of final states, and the initial states as the state that receives an initial probability mass.

Let NT be the set of possible phrase names, e.g., NP, PP, etc.; let W be the set of words in the lexicon; we assume that both sets are finite. It can be seen from the definition of rules, that for each pair $(H,w) \in NT \times W$ there are two Markov chains $r_{(H,w)}$, $l_{(H,w)}$.

A bit of notation: following Collins, we rewrite rules such as (1) as follows:

$$(P,h) \rightarrow (L_n,l_n)\dots(L_1,l_1)(H,h)(R_1,r_1)\dots(R_m,r_m)$$

The probability of strings $(H,h)(L_1,l_1)\dots(L_n,l_n)$ and $(H,h)(R_1,r_1)\dots(R_m,r_m)$ is given by the probabilities assigned to paths $(L_1,l_1)\dots(L_n,l_n)$STOP and $(R_1,r_1)\dots(R_m,r_m)$STOP in the Markov chains $l_{(H,h)}$ and $r_{(H,h)}$ respectively.

Definition 2. *Let $B = (NT, W, \{l_{(H,w)}\}_{w\in W,h\in NT}, \{r_{(H,w)}\}_{w\in W,h\in NT})$ be a grammar based Markov rules. 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 $S_{l_{(\bar{H},w)}}$ and $S_{(H,w)}$ from the regular grammars $l_{(\bar{H},w)}$ and $r_{(H,w)}$ 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 probabilistic meta-rules is given by the union of the rules in each of the right and left regular grammars (i.e., the set given by $\mathcal{A} \xrightarrow{m} \alpha$ iff $A \rightarrow \alpha \in l_{(H,w)}$ or $\mathcal{A} \rightarrow \alpha \in r_{(H,w)}$ for some H and some w) plus the set $\{w' \xrightarrow{m} w : w \in W\}$.*
- *The set of pseudo-rules is given by $(P,h) \xrightarrow{s}_{\mathcal{P}_H(H|(P,h))} S_{l_{(H,h)}}(H,h)S_{r_{(H,h)}}$*

An important technical detail: a pair (H,h) should not be viewed as separate variables, but as a single entity; consequently, they are treated as one for replacing.

Lemma 3. *Markov rules as used in Collins' first model are f-transformable to CW-grammars.*

Properties. By Lemma 3 Collins' first model is weakly equivalent to context free grammars. Moreover, Example 1 can be used to show that the model is not strongly equivalent to CFGs; consequently, its probabilistic version can not be captured using PCFGs. As a consequence of Lemma 3, learning Markov rules-based grammars is equivalent to learning PCW-grammars and since every PCW grammar has an underlying PCFG, learning a PCW-grammar is equivalent to learning its underlying PCFG.

[5] 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. If the Markov chain used is of degree zero, there is only one variable (the Markov chain contains a unique state), and the induced Markov rules-based grammar is consistent. The result follows from the fact that inducing a degree zero Markov chain is the same as inducing the underlying PCFG in the equivalent PCW-grammar using maximum likelihood, plus the fact that using maximum likelihood for inducing PCFGs produces consistent grammars [11,4].

The embedding of Collins' and Eisner's model into CW-grammars shows that even though the first deals with phrase structures and the second with dependency structures, they only differ in the number of variables they assume to exist in the underlying PCW-grammars.

3.3 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 [2] was the first version of DOP, and most later versions of DOP are variations on it. Recently, in [1] the authors used Tree Substitution Grammars in Unsupervised dependency parsing and they obtained the state of the art results. The underlying grammatical formalism in all previous models is Stochastic Tree Substitution Grammars (STSG), 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. Now we use these trees to form an Stochastic Tree Substitution Grammar.

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).

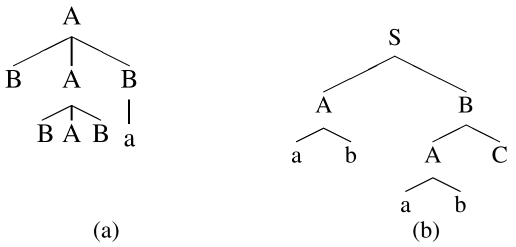


Fig. 5. (a) A derivation tree. (b) An elementary tree.

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.

STSGs as CW-grammars. DOP-1 is not quite a context-free grammar. 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 the rule body. The idea of the equivalence is to associate to every tree produced by DOP 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 tree the in Figure 5.(a) is $(A, BAB)B'a(A, BAB)(B, a)a$.

We start capturing STSG by building 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 $S' \to \alpha$ 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. For example, let t be the elementary tree pictured in Figure 5.(b), its corresponding elementary rule is $S' \to (A, B)(A, ab)ab(B, AC)(A, ab)abC'$. Note the primed version of C in the frontier of the derivation.

Definition 3. *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 $(A, \alpha) \to \epsilon$ 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. 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 DOP-1 model. Second, the probability of a ‘real’ rule is the sum of the probabilities of meta-derivations yielding the rule’s body.

Lemma 4. *Let $H = (V_N, V_T, S, R, P)$ be a STSG and let W_H be the CW-grammar given in Definition 3. There is a one-to-one correspondence between derivations in H and meta-rule derivations in W_H.*

Proof. 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 $n + 1$. We can assume that there are trees t_1, t_2 such that $t_1 \circ t_2 = t$. 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.

Lemma 5. *Let $H = (V_N, V_T, S, R, P)$ be an STSG, and let W_H be the CW-grammar given in Definition 3. Then W_H accepts the same set of strings as H, i.e., STSGs and PCW-grammars are weakly equivalent.*

Proof. Let α be a string in $L(H)$. There is at least one tree derivation $t_1 \circ \ldots \circ t_k$ yielding α. From Lemma 4 we know that there is a CW-Rule $S' \overset{w}{\Longrightarrow} \alpha$ such that after applying rules $(A, \beta) \xrightarrow{s} \epsilon$, α is obtained.

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 3. There is a one-to-one correspondence between derivations in H and W_H.*

Corollary 2. *STSGs are f-transformable to CW-grammars.*

Lemma 6. *Let $H = (V_N, V_T, S, R, P)$ be an STSG, and let W_H be the CW-grammar given in Definition 3. Both grammars assign the same probability mass to trees related through the one-to-one mapping described in Corollary 1.*

Proof. A tree has a characteristic W-rule, defined by its shape. I.e., the 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.

Properties. By Corollary 2, 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 it, 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 [12]. Solutions that take into account the dependence between trees in the resulting tree-banks have been suggested [17].

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. 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 completely. 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 resulting STSG's consistency depends on the consistency properties 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 art parsers. We provided expressive power properties for three formalisms (Bilexical Grammars, Markov Rules, and Stochastic Tree Substitution Grammars) together with some conditions under which the inferred grammars are consistent.

We showed that, from a formal perspective, Bilexical Grammars and Markov rules do not differ in a principled way: both are based on approximating bodies of rules using Markov models. We also found that STSGs and Markov rules also have certain similarities. Markov rules suppose that rule bodies are obtained by collapsing hidden derivations. That is, for Collins a rule body is a regular expression (equivalent to Markov models). Similarly, Bod's DOP model takes this idea to the extreme by taking the whole sentence to be the yield of a hidden derivation. PCW-grammars suggest that we explore intermediate levels of abstraction, levels in between having everything hidden and only having productions hidden.

References

1. Blunsom, P., Cohn, T.: Unsupervised induction of tree substitution grammars for dependency parsing. In: Proceedings of EMNLP 2010 (2010)
2. Bod, R.: Enriching linguistics with statistics. Ph.D. thesis, Univ. Amsterdam (1995)
3. Bod, R.: Beyond Grammar—An Experience-Based Theory of Language. Cambridge University Press (1999)
4. Chi, Z., Geman, S.: Estimation of probabilistic context-free grammars. Compu. Ling. 305, 24:299–24:305 (1998)

5. Collins, M.: Three generative, lexicalized models for statistical parsing. In: ACL 1997 (1997)
6. Collins, M.: Head-driven statistical models for natural language parsing. Ph.D. thesis, Univ. Pennsylvania (1999)
7. Domníguez, M.A., Infante-Lopez, G.: Searching for Part of Speech Tags That Improve Parsing Models. In: Nordström, B., Ranta, A. (eds.) GoTAL 2008. LNCS (LNAI), vol. 5221, pp. 126–137. Springer, Heidelberg (2008)
8. Eisner, J.: Three new probabilistic models for dependency parsing: An exploration. In: Proceedings of International Conference on Computational Linguistics (COLING-1996), Copenhagen (1996)
9. Eisner, J.: Bilexical grammars and their cubic-time parsing algorithms. In: Advances in Probabilistic and Other Parsing Technologies, pp. 29–62 (2000)
10. Infante-Lopez, G., de Rijke, M.: Comparing the ambiguity reduction abilities of probabilistic context-free grammars. In: Proc. LREC 2004 (2004)
11. Joan-Andreu, S., Bened, J.M.: Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations. IEEE Trans. on Pattern Analysis and Machine Intelligence (1997)
12. Johnson, M.: The dop estimation method is biased and inconsistent. Comp. Ling 76, 28:71–28:76 (2002)
13. Jurafsky, D., Martin, J.: Speech and Language Processing. Prentice Hall (2000)
14. Klein, D., Manning, C.: Corpus-based induction of syntactic structure: Models of dependency and constituency. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (2004)
15. Koo, T., Carreras, X., Collins, M.: Simple semi-supervised dependency parsing. In: Proc. ACL/HLT (2008)
16. Manning, C., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press (1999)
17. Sima'an, K., Buratto, L.: Backoff Parameter Estimation for the DOP Model. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 373–384. Springer, Heidelberg (2003)
18. Spitkovsky, V.I., Alshawi, H., Jurafsky, D., Manning, C.D.: Viterbi training improves unsupervised dependency parsing. In: Proc. CoNLL 2010 (2010)
19. van Wijngaarden, A.: Orthogonal design and description of a formal language. In: Technical Report MR76, Mathematisch Centrum, Amsterdam (1965)