A Note on the Expressive Power of Probabilistic Context Free Grammars

Gabriel Infante-Lopez (gabriel@famaf.unc.edu.ar)
Facultad de Matemática, Astronomía y Física
Universidad Nacional de Córdoba
Argentina

Maarten de Rijke (mdr@science.uva.nl)
Informatics Institute
University of Amsterdam
The Netherlands

Abstract. We examine the expressive power of probabilistic context free grammars (PCFGs), with a special focus on the use of probabilities as a mechanism for reducing ambiguity by filtering out unwanted parses. Probabilities in PCFGs induce an ordering relation among the set of trees that yield a given input sentence. PCFG parsers return the trees bearing the maximum probability for a given sentence, discarding all other possible trees. This mechanism is naturally viewed as a way of defining a new class of tree languages. We formalize the tree language thus defined, study its expressive power, and show that the latter is beyond context freeness. While the increased expressive power offered by PCFGs helps to reduce ambiguity, we show that, in general, it cannot be decided whether a PCFG removes all ambiguities.

1. Introduction

Probabilities have been used in many aspects of natural language processing. In the context of context free grammars (CFGs), probabilities have been used to define a probability distribution over the set of trees defined by a CFG. The resulting formalism, probabilistic context free grammars (PCFGs), extends CFGs by assigning probabilities to the production rules of the grammar. PCFGs have been successfully used as the formalism underlying many approaches to natural language parsing, see e.g., (Eisner, 1996; Charniak, 1995; Collins, 1999; Eisner, 2000; Bod et al., 2002; Klein and Manning, 2003; Infante-Lopez and de Rijke, 2004; Infante-Lopez, 2005), to name just a few. In these approaches to parsing, probabilities have a very specific role: they are used as a filtering mechanism. To see just how, we can think of the parsing procedure for a CFG as a two-fold procedure. Suppose a sentence is given. First, a set of candidate trees is proposed, and, second, a tree is non-deterministically chosen from this set of candidate trees. The selected tree, then, is returned as output.

CFGs define the set of candidate trees for a given input sentence as the set of all trees that yield the sentence. In contrast, in the context of parsing, PCFGs are used to define the set of candidate trees as the set of trees that yield the input sentence *with maximum probability*. In other words, in the context of parsing, probabilities are used to reduce the set of candidate trees suggested by the bare CFG. It is well-known that ambiguity is a serious challenge for parsers. The size of the set of candidate trees gives an indication of the ambiguity a given input sentence: in some cases, the number of parse trees assigned to a sentence may grow exponentially in the length of the sentence (Wich, 2000; Wich, 2001). Hence, it is vital to deploy strategies that help reduce the size of the set of candidate trees. Such strategies, which we call *filtering mechanisms*, filter out undesired trees from the set of candidate trees. Probabilities can play a very important role as a filtering mechanism.

While there are many studies covering the role that probabilities play in the context of formal languages, the focus of these studies is rarely on the ability to reduce ambiguity. Properties of formal languages regarding consistency (Booth and Thompson, 1973; Wetherell, 1980; Chaudhuri and Rao, 1986), learnability conditions (Horning, 1969), parameter estimation (Manning and Schütze, 1999), etc. are very well-known. However, little is known about the power of probabilities as a mechanism for reducing ambiguity, and this is where our focus lies. How important and how powerful are probabilities as a filtering mechanism?

To give an example of the kind of issues that we are after, suppose that a PCFG G is given and that its probabilities are used to select a subset T' of trees in the tree language generated by G. It might be the case that there exists a non-probabilistic CFG G' such that its tree language is equal to T. Clearly, if such a grammar G' exists for all PCFGs G, then probabilities are not really needed from a formal language perspective. In contrast, if this is not the case, the role that probabilities play in parsing becomes essential, and we need to address questions that differ from those answered in (Booth and Thompson, 1973; Wetherell, 1980; Chaudhuri and Rao, 1986).

In this paper we focus on answers to fundamental questions regarding the use of probabilities as a filtering mechanism. We pay special attention to the following questions:

1. Is it possible to select the set of candidate trees produced by a PCFG using a vanilla context free grammar? The question is relevant because, if the answer is affirmative, this means that for a given PCFG there is a CFG that specifies the same set of candidate

trees as the PCFG for all sentences; as a consequence, probabilities would not be essential.

2. Can we decide whether a given PCFG filters out all but one tree for all sentences in the language? An affirmative answer to this question is equivalent to saying that the given PCFG has solved all ambiguities in the language accepted by the grammar.

In this paper, we answer both questions. Specifically, we show the following:

- Probabilities cannot be mimicked by rules, i.e., their use is fundamental from a formal language perspective: whenever used as a filtering mechanism, probabilities can define a set of trees that can *not* captured by CFGs.
- It is not possible to decide whether a PCFG filters all trees but one for all sentences, i.e., it is not possible to decide whether the filtering mechanism resolves all ambiguities.

The techniques used to obtain these results are standard. Our emphasis, though, is on their implications: probabilities are valuable not only from a machine learning perspective, but also from a formal language perspective.

The remainder of the paper is organized as follows. In Section 2 we present some background notation. In Sections 3 and 4 we address questions 1 and 2 above, respectively. Finally, we discuss related work in Section 5, and in Section 6 we conclude the paper.

2. Maximum Probability Tree Grammars

In this section we define the basic concepts we need in our discussion. A *context free grammar* (CFG) is defined as quadruple $\langle T, NT, S, R\rangle$, consisting of a terminal vocabulary T, a non-terminal vocabulary NT, a distinguished symbol $S \in NT$, usually called the *start symbol* or *axiom*, and a set of productions or rewrite rules R. The sets T, NT, and R are finite; T and NT are disjoint ($T \cap NT = \emptyset$). For our purposes, CFGs play two roles. The first is to provide sentences with a syntactic explanation. As is well-known, this syntactic explanation is given by the way in which a sentence is rewritten from the start symbol. Formally, let G be a CFG and x a sentence in T^*. A *left-derivation* $t(x)$ is a sequence of rules $\langle r_1, \ldots, r_m\rangle$ such that

$$S \stackrel{r_1}{\Longrightarrow} \alpha_1 \stackrel{r_2}{\Longrightarrow} \alpha_2 \stackrel{r_3}{\Longrightarrow} \ldots \stackrel{r_m}{\Longrightarrow} \alpha_m,$$

where $\alpha_i \in (T \cup NT)^*$, $\alpha_m = x$, and r_i is a rule in R that rewrites the left-most non-terminal of α_{i-1}. The set of all left-most derivations, called the *tree language*, is denoted by $T(G)$, and x is called the *yield* of $t(x)$.

The second main role of CFGs is to define the set of sentences that are considered to be grammatical by the grammar. More precisely, let G be a CFG. The *language accepted* by G (notation: $L(G)$) is $L(G) = \{x : T(x) \in T(G)\}$.

Let G be a CFG and x a sentence. A *complete parser* is a procedure that computes the following set:

$$T(x) = \{t(x) \in T(G)\}.$$

A *parser* is a procedure that, besides computing $T(x)$, chooses one tree non-deterministically from it. Formally,

$$Parser(x) = random(T(x)),$$

where $random(X)$ is a function that selects an element from the set X, assigning to each element the same probability of being chosen.[1] For a given grammar G and a sentence x in $L(G)$, there may be multiple trees yielding the same sentence x.

In some cases the size of the set $T(x)$ grows exponentially in the length of x (Wich, 2000), and many of the trees in $T(x)$ are trees that we do not want as a result. These undesired trees need to be filtered out from the set $T(x)$. One widely used way of achieving this is to use probabilistic context free grammars. A *probabilistic context free grammar* (PCFG) is a pair (G, p), where G is a CFG and p a positive function defined over the set of rules such that for all A in NT:

$$\sum_{\alpha \in (N \cup NT)^*} p(A \rightarrow \alpha) = 1.$$

Even though probabilities in a PCFG are defined over the set of rules, they are used to define a probability distribution p over the set of derivations $T(G)$. The probability of a tree in $T(G)$ is defined as follows. Let (G, p) be a PCFG, and let $t(x) = \langle r_1, \ldots, r_m \rangle$ be a tree in $T(G)$. The *probability* assigned to $t(x)$ is $p(t(x)) = p(r_1) \cdot \ldots \cdot p(r_m)$. That is, the probability assigned to a tree is the product of the probabilities assigned to the rules building up the tree (Manning and Schütze, 1999).

[1] Note that, formally peaking, *non-deterministic choice* and *uniformly distributed choice* are not the same concept: non-determinism implies that there is no information about the underlying distribution, which clearly is not the case when using a uniform distribution. In this paper we do not distinguish between the two terms.

The distribution generated by the probabilities is used to select a subset of trees from the set of all possible trees yielding a sentence. A procedure that computes this subset is called a probabilistic parser. More formally, let G be a PCFG and let x be a sentence in $L(G)$. A *probabilistic complete parser* for a grammar G is a procedure that takes as input a sentence in the language and computes the following function:

$$PParser(x) = \{t(x) \in T(G) : \arg\max_{t(x)}\{p(t(x))\}\}.$$

Finally, a *probabilistic parser* is defined as the non-deterministic choice of a tree returned by a probabilistic complete parser. We can view a probabilistic parser as a two-step algorithm, with one step implementing a probabilistic complete parser and a second implementing a non-deterministic choice function.

2.1. Filtering Out Trees Using Probabilities

In this subsection, we consider the class of languages induced by probabilities when we view the latter as a filtering mechanism. Let G be a PCFG. The set of *most probable trees* produced by G (noted $M(G)$) is a subset of $T(G)$ that is defined as follows:

$$M(G) = \bigcup_{x \in L(G)} PParser(x).$$

Note that there may be more than one tree bearing maximum probability for a given sentence. We allow for this: $M(G)$ contains them all. Note also that $M(G)$ is a subset of $T(G)$. Furthermore, the sets $M(G)$ and $T(G)$ are the same set if, and only if, for every sentence all trees in $L(G)$ share the same probability mass.

Based on the set of most probable trees $M(G)$, we define a new class of tree languages. The idea behind this class of languages is that they are like PCFGs but instead of taking the full set of trees they only take the most probable ones. More formally, a *maximum probability tree grammar* (MPTG) is a PCFG where its tree language is defined as the set of most probable trees, while the set of strings accepted by the MTPG remains the same.

Note that all state-of-the-art parsers based on PCFGs filter trees out in the way we have just defined (Eisner, 1996; Charniak, 1995; Collins, 1999; Eisner, 2000; Klein and Manning, 2003; Infante-Lopez and de Rijke, 2004; Infante-Lopez, 2005). They return the trees yielding a given sentence that bear maximum probability, and thus, they implicitly define an MPTG.

$$
\begin{array}{ll}
S \rightarrow_{2/3} AB & S \rightarrow_{1/3} S_2 \\
A \rightarrow_{1/2} aAb & S_2 \rightarrow_{1/2} aS_2d \\
A \rightarrow_{1/2} ab & S_2 \rightarrow_{1/2} aCd \\
B \rightarrow_{1/2} cBd & C \rightarrow_{1/2} bCc \\
B \rightarrow_{1/2} cd & C \rightarrow_{1/2} bc
\end{array}
$$

Figure 1. An MPTG for an inherently ambiguous language.

3. Expressive Power

We turn to the first of the research questions formulated in Section 1: What kind of expressiveness do MPTGs have? Can they be captured by CFGs? More concretely, does the mechanism of retaining only trees with maximum probability defined in MPTGs define tree languages that cannot be captured by CFGs? We show that the set of trees identified by a probabilistic CFG plus a maximization procedure cannot be generated or specified directly by any CFG. In other words, probabilistic CFGs plus a maximization step define tree languages that are beyond the expressive power of CFGs. To put it more formally: MPTGs are not strongly equivalent to CFGs.

Theorem 1 *MPTGs are not strongly equivalent to CFGs.*

Our strategy for proving Theorem 1 is based on a context free inherently ambiguous language. Recall that a *context free inherently ambiguous* language L is a language such that all CFGs generating it have at least one string in L that has two trees yielding it (Hopcroft and Ullman, 1979; Parikh, 1966). What we present below is an inherently ambiguous language generated by a PCFG that assigns a unique tree bearing maximum probability to each string in the language. As a consequence, the MPTG induced by the grammar is unambiguous, but the tree language cannot be captured by a CFG because the language is inherently ambiguous, i.e., for all CFGs generating it, there is at least one sentence that is the yield of two different trees.

Let us make matters concrete now. The grammar $G = (T, NT, S, R)$ is defined as follows. Put $T = \{a, b, c, d\}$, $NT = \{S, S_2, A, B, C\}$, and let the set of rules R be as described in Figure 1. In order to better understand the complexity of G we split it into two different grammars $G_1 = (T_1, NT_1, S, R_1)$ and $G_2 = (T_2, NT_2, S, R_2)$, where

1. $T_1 = \{a, b, c, d\}$,
2. $NT_1 = \{S, A, B\}$,

3. $T_2 = \{a, b, c, d\}$,

4. $NT_2 = \{S, S_2, C\}$, and

5. R_1 and R_2 are as defined by Figure 2.

Observe that G generates the following string language:

$$L = \{a^n b^n c^m d^m : n, m \in \mathbb{N}\} \cup \{a^n b^m c^m d^n : n, m \in \mathbb{N}\}.$$

L can be described as the union of the two context free languages generated by G_1 and G_2, respectively, namely $L_1 = \{a^n b^n c^m d^m : n, m \in \mathbb{N}\}$ and $L_2 = \{a^n b^m c^m d^n : n, m \in \mathbb{N}\}$.

We will now describe the tree language generated by G. Remember that G is a MPTG, and not just a PCFG. The set of its derivations is the set of trees bearing maximum probability. Define $L_3 = \{a^n b^n c^n d^n : n \in \mathbb{N}\}$, and let $T(L_3)$ denote the set of trees in $T(G)$ that yield a sentence in L_3. Our first observation is that all trees in $(T(G_1) \cup T(G_2)) \setminus T(L_3)$ have a unique derivation. Hence, they obviously belong to $M(G)$. For the trees in L_3 there are two possible derivations: one generated by G_1 and the other generated by G_2. In order to fully characterize $M(G)$, we need to determine which of the two derivations (if not both) belongs to $M(G)$.

Below, we show that only the derivations produced by G_1 belong to $M(G)$. In other words, all derivations produced by G_2 are filtered out. To obtain this characterization, we first characterize the trees in $T(L_3)$.

Lemma 2 *Let x be a string in L_3, and let $t_1(x)$ be a tree in $T(G_1)$ and $t_2(x)$ a tree in $T(G_2)$. Then, the number of rules appearing in $t_1(x)$ is the same as the number of rules appearing in $t_2(x)$. Moreover, the rule $S \rightarrow_{2/3} AB$ appears once in $t_1(x)$, while the rule $S \rightarrow_{1/3} S_2$ appears once in $t_2(x)$.*

R_1	R_2
$S \rightarrow_{2/3} AB$	$S \rightarrow_{1/3} S_2$
$A \rightarrow_{1/2} aAb$	$S_2 \rightarrow_{1/2} aS_2d$
$A \rightarrow_{1/2} ab$	$S_2 \rightarrow_{1/2} aCd$
$B \rightarrow_{1/2} cBd$	$C \rightarrow_{1/2} bCc$
$B \rightarrow_{1/2} cd$	$C \rightarrow_{1/2} bc$

Figure 2. A decomposition of G.

PROOF. Let x be a string in $L_3 = \{a^n b^n c^n d^n : n \in \mathbb{N}\}$. We prove the lemma by induction on n, the superscript in the definition of L_3. For the base case, let n be 1; then the lemma follows from the fact that the two possible trees, both pictured in Figure 3, have the same number of rules, and it is clear that the rule $S \rightarrow_{2/3} AB$ appears once in $t_1(x)$, while the rule $S \rightarrow_{1/3} S_2$ appears exactly once in $t_2(x)$.

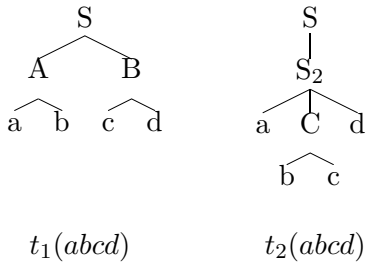

Figure 3. Two derivations for *abcd*.

Let the statement be true for $k < n$, and let us show it for $x = a^n b^n c^n d^n$. Note, first of all, that for a word in L_3, the two possible trees follow the schema of Figure 4. Now, using this observation for the first part of the

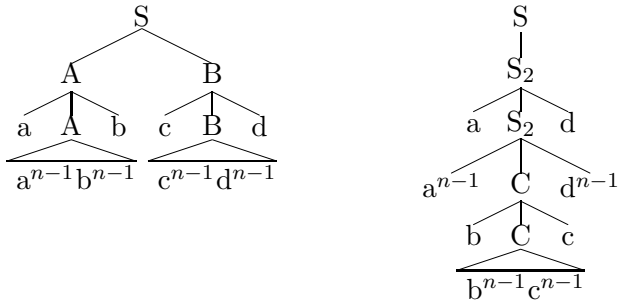

Figure 4. Derivations for $a^n b^n c^n d^n$.

lemma, the string $a^{n-1}b^{n-1}c^{n-1}d^{n-1}$ can be derived by collapsing the A, B, S_2 and C non-terminals, respectively. According to the inductive hypothesis, the resulting trees have the same number of rules, and in the process of collapsing we have eliminated the same number of rules from the two trees, proving the first part of the lemma. For the second part of the lemma, consult Figure 4 again, where it can be seen that the rules $S \rightarrow_{2/3} AB$ and $S \rightarrow_{1/3} S_2$ appear once in $t_1(x)$ and $t_2(x)$, respectively. ⊣

Lemma 2 says that the probabilities of two derivations in L_3 yielding the same sentence are determined by the first rule in each grammar. That

is, we can distinguish between the two possible probabilities assigned to the two derivations for a sentence in L_3 by simply observing the probability of the first rule in each of the derivations.

Lemma 3 *Let x be a string in L_3, $t_1(x)$ in $T(G_1)$ and $t_2(x)$ in $T(G_2)$. Then $p(t_1(x)) > p(t_2(x))$.*

PROOF. The proof is immediate from Lemma 2. All derivations for a given string x in L_3 have the same number of rule applications, and except for the first rule applied (either $S \rightarrow_{2/3} AB$ or $S \rightarrow_{1/3} S_2$), all rules have equal probabilities associated with them. This means that the tree using the G_1-rule $S \rightarrow_{2/3} AB$ has the higher probability, as desired. ⊣

Lemma 4 *$M(G)$ is equal to $T(G_1) \cup \{t(x) \in T(G_2) : x \in L_2 \setminus L_3\}$.*

PROOF. The lemma is a direct consequence of Lemma 3. ⊣

Finally, with this characterization we can prove Theorem 1.

PROOF. (Of Theorem 1) Note first that for every string in $L(G)$ there is a unique tree in $M(G)$: if the string belongs to $L_1 \setminus L_3$ or to $L_2 \setminus L_3$ this is because the grammars G_1 and G_2 assign a unique derivation tree to each string. If, however, $x \in L_3$, then, by maximizing probabilities, we discard the tree belonging to $T(G_2)$, thus only leaving the tree in $T(G_1)$, as shown in Lemma 4.

In sum, using probabilities we have obtained a unique tree for every sentence in an inherently ambiguous language. It is a well-known fact that CFGs cannot disambiguate an inherently ambiguous language (Hopcroft and Ullman, 1979). Hence, since L is inherently ambiguous (Parikh, 1966), there cannot be a CFG that generates all and only the trees in $MPTG(G)$. ⊣

Theorem 1 provides an answer to question 1 from the Introduction, showing that there is no way to mimic probabilities plus maximization using rules. Hence, probabilities (plus maximization) add not only a statical perspective but also expressive power to CFGs. This increase in expressive power is due to a probabilistic parser's implicit (but global) requirement that it sees all rules building up a tree for choosing the one with maximum probability.

4. Undecidability

While probabilities plus maximization buy us additional expressive power on top of CFGs, they do not buy us everything. Specifically, given that probabilities help to disambiguate a grammar's language, it is natural to ask if we can predict (that is, determine before parsing) whether a PCFG is capable of fully disambiguating its language. We show that this is not possible. We establish the result by transforming an arbitrary CFG into a PCFG such that the given CFG is unambiguous if, and only if, the corresponding PCFG has only one tree in the candidate list of each sentence. Our result then follows from the well-known fact that determining whether a CFG is unambiguous is undecidable.

For each sentence x, probabilities single out a set $PParser(x)$ of trees bearing maximum probability. An *ideal* grammar is one that filters out all trees but one for each sentence in the language. In other words, an ideal PCFG defines for each sentence x, its set $PParser(x)$ with cardinality equal to 1.

We want to prove that it is undecidable to determine whether a PCFG is ideal. To this end, we first prove that for every context free grammar G there is a way to extend it with probabilities such that the resulting set $M(G)$ contains the same set of trees as G. In other words, for any CFG we build a probabilistic version that does not filter out any tree. Our undecidability result, then, follows from the fact that our question is equivalent to determining whether a CFG is unambiguous.

We have to build the probabilistic counterpart of a CFG, in such a way that all trees associated to a given sentence bear the same probability. In this case, the set of trees with maximal probability is exactly the set of trees. We show the result for grammars in Chomsky Normal Form whose definition we now recall.

A context free grammar $G = (T, NT, R, S)$ is said to be in Chomsky Normal Form (CNF) if, and only if, every rule in R is of one of the following forms:

- $A \to a$ for some $A \in NT$ and some $a \in T$.
- $A \to BC$, for some $A \in NT$ and $B, C \in NT - \{S\}$.

Our strategy is to show that any grammar in CNF assigns the same probability to all trees yielding the same string. To this end we show that all trees yielding the same string in CNF use the same number of rules; we then build a grammar assigning the same probability to all rules, and we obtain what we are looking for.

We now present the lemmas needed.

Lemma 5 *Let $G = (T, NT, S, R)$ be a grammar in CNF. All trees yielding a k-length sub-string of NT^* use the same number of rules.*

PROOF. Let us define a sequence $A_0, \ldots, A_n, \ldots$ of subsets of NT^* as follows: $A_0 = \{S\}$, A_1 consists of elements α in NT^* such that α is derived from S in one step, and, in general, α is in A_i if there is an element α' in A_{i-1} such that $\alpha' \Rightarrow \alpha$. The lemma is immediate from the fact that that all sets are pairwise disjoint, i.e., $A_i \cap A_j = \emptyset$ for every $i \neq j$. ⊣

Corollary 6 *Let G be a CFG. Every derivation producing a string x of length k in $L(G)$ has the same number of rules.*

Lemma 7 *Let G be a context free grammar. G can be transformed into a probabilistic context free grammar G' with the special property that all rules have exactly the same probability value.*

PROOF. Let G be a grammar in CNF, and let R be its set of rules. Let X be the most frequent non-terminal in the left-hand sides of rules. Let n be the number of times X is the left hand-side of a rule. Let $Z_1, \ldots, Z_n$ be brand new non-terminal symbols. For every non-terminal Y we add as many rules $Y \rightarrow Z_i$ as needed to have the number of rules having Y in the left-hand side equal to n. We add probability $1/n$ to each of these new rules. The resulting grammar is a well-defined, though not necessarily consistent, probabilistic context free grammar, and all rules have exactly the same probability values as required. ⊣

The PCF grammar G' obtained from a grammar G as described in Lemma 7, is called the *uniform version* of G.

Note that the uniform grammar produced by Lemma 7 need not be consistent, given that some probability mass is going to non-terminating derivations — derivations that end up in the dummy non-terminal. Still, what is important to us is that the set of trees accepted by the PCFG remains the same, and, even more importantly, that every derivation producing the same sentence has the same probability value.

Lemma 8 *Let G be a context free grammar, and let G' be its uniform version. Let x be a string in $L(G)$. Then all left-most derivations producing x have the same probability.*

PROOF. Since every string in the language has the same set of trees as G, the dummy rule is not used in any derivation of final strings. According to Lemma 5, every tree has the same number of rules. And since every rule has the same probability value, all trees that yields

a sentence x have the same probability value. Finally, the set of trees bearing maximum probability is exactly the set of trees in the original grammar G. ⊣

As this lemma proves, trees defined through maximum probability tree grammars include the class of trees defined via CFGs. As a direct consequence, we have the following result.

Theorem 9 *Determining whether a PCFG disambiguates a tree language is undecidable.*

PROOF. We have built a grammar that assigns the same probability mass to all possible trees for a given string. As a consequence, the PCFG is unambiguous if, and only if, the non-probabilistic grammar is. Deciding whether the PCFG is unambiguous is equivalent to deciding whether a CFG in CNF is unambiguous, which is known to be undecidable (Hopcroft and Ullman, 1979). ⊣

Theorem 9 answers question 2 from the Introduction, saying that it is not possible to decide whether a PCFG has completely managed to solve all ambiguities. Note that the results in the present section combined with the results in the previous one imply that the class of tree languages described by PCFGs is a strict subclass of the tree languages described by MPTGs. The inclusion is implied by the present section while the proper inclusion is implied by the previous one.

5. Related Work

Before concluding, we briefly compare the work in this paper to the literature. Ours is not the first study of the expressive power of weighted formal languages. Cortes and Mohri (2000) show that the expressive power of weighted automata is beyond regular languages. This result has in common with the result we present in Section 3 that both show that weighted systems accept a wider set of languages than bare, unweighted systems. The two results also use the same strategy; they present a language that does not belong to the bare grammatical formalism but that can be captured by the weighted version.

Our approach differs from that of Cortes and Mohri (2000) in that we use probabilities to select subtrees as a side-product of filtering. Cortes and Mohri (2000) found a well-known context free language to be accepted by a weighted automata under a general definition of acceptance. The approaches also differ in that ours is concerned with the tree language and theirs with the string language. Finally, the two

results differ in that our proof is standard from a technical point of view, while the other is rather involved.

The main result presented in Section 3 is not directly linked to statistics. As discussed by Abney (1996), probabilities can help in many aspects of syntax (e.g., disambiguation, degrees of grammaticality, error tolerance, naturalness, structural preferences, learning, lexical acquisition). We respond to the more principled question "What can we do with probabilities (weights)?" rather than than "How can we compute the probabilities (weights)?". Abney argues that, intuitively, probabilities can help disambiguation, but in Section 3 we have shown that they provide a mechanism that simply cannot be mimicked with rules. We went beyond mere intuitions and presented technical facts.

6. Conclusions

This paper has focused on questions related to the importance of probabilities in the context of parsing and, in particular, on their usage as a filtering mechanism. We have shown that probabilities, when used as a filtering mechanism, can add expressive power to grammars, defining a class of tree languages beyond the expressivity of context free grammars.

The proofs we used to establish this result are standard — this is not where the main contribution of the paper is. Our main contribution is the first rigorous proof for the widely held belief that PCFGs (with a maximization procedure) are more expressive than bare CFGs. Theorem 1 suggests that any grammatical formalism that suffers from inherent ambiguity can increase its expressive power by means of probabilities. At the same time, Theorem 9 implies that probabilities are not a final answer to ambiguity: determining whether they will manage to remove all ambiguities is undecidable.

Acknowledgements

Gabriel Infante-Lopez was partially supported by the Netherlands Organization for Scientific Research (NWO) under project number 220-80-001. Maarten de Rijke was supported by NWO under project numbers 017.001.190, 220-80-001, 264-70-050, 354-20-005, 612-13-001, 612.000.-106, 612.000.207, 612.066.302, 612.069.006, and 640.001.501.

References

Abney, S.: 1996, 'Statistical Methods and Linguistics'. In: J. Klavans and P. Resnik (eds.): *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*. Cambridge, MA: The MIT Press.

Bod, R., R. Scha, and K. Sima'an (eds.): 2002, *Data Oriented Parsing*. CSLI.

Booth, T. and R. Thompson: 1973, 'Applying Probability Measures to Abstract Languages'. *IEEE Transaction on Computers* **C-33**(5), 442–450.

Charniak, E.: 1995, 'Parsing with Context-Free Grammars and Word Statistics'. Technical Report CS-95-28, Department of Computer Science, Brown University, Providence.

Chaudhuri, R. and A. N. V. Rao: 1986, 'Approximating Grammar Probabilities: Solution of a Conjecture.'. *Journal of the ACM* **33**(4), 702–705.

Collins, M.: 1999, 'Head-Driven Statistical Models for Natural Language Parsing'. Ph.D. thesis, University of Pennsylvania, PA.

Cortes, C. and M. Mohri: 2000, 'Context-free Recognition with Weighted Automata'. *Grammars* **2-3**(3).

Eisner, J.: 1996, 'Three New Probabilistic Models for Dependency Parsing: An Exploration'. In: *Proceedings of 16th International Conference on Computational Linguistics (COLING)*. Copenhagen, Denmark, pp. 340–245.

Eisner, J.: 2000, 'Bilexical Grammars and Their Cubic-Time Parsing Algorithms'. In: H. Bunt and A. Nijholt (eds.): *Advances in Probabilistic and Other Parsing Technologies*. Kluwer Academic Publishers, pp. 29–62.

Hopcroft, J. and J. Ullman: 1979, *Introduction to Automata Theory, Lanaguges, and Computation*. Reading, MA: Addison Wesley.

Horning, J. J.: 1969, 'A Study of Grammatical Inference'. Ph.D. thesis, Stanford University.

Infante-Lopez, G.: 2005, 'Two-level Probabilistic Grammars for Natural Language Parsing'. Ph.D. thesis, Universiteit van Amsterdam.

Infante-Lopez, G. and M. de Rijke: 2004, 'Alternative Approaches for Generating Bodies of Grammar Rules'. In: *Proceedings of the 42nd Annual Meeting of the ACL*. Barcelona.

Klein, D. and C. Manning: 2003, 'Accurate Unlexicalized Parsing'. In: *Proceedings of the 41st Annual Meeting of the ACL*.

Manning, C. and H. Schütze: 1999, *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Parikh, R. J.: 1966, 'On Context-Free Languages'. *Journal of the ACM* **13**, 570–581.

Wetherell, C. S.: 1980, 'Probabilistic Languages: A Review and Some Questions.'. *ACM Computer Surveys* **4**(12), 361–379.

Wich, K.: 2000, 'Exponential Ambiguity of Context-free Grammars'. In: *Proceedings of the 4th International Conference on Developments in Language Theory*. pp. 125–138.

Wich, K.: 2001, 'Characterization of Context-free Languages with Polynomially Bounded Ambiguity'. In: *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS)*.