Prefix Probabilities for Linear Context-Free Rewriting Systems

Mark-Jan Nederhof
School of Computer Science
University of St Andrews
North Haugh, St Andrews, Fife
KY16 9SX
United Kingdom

Giorgio Satta
Dept. of Information Engineering
University of Padua
via Gradenigo, 6/A
I-35131 Padova
Italy

Abstract

We present a novel method for the computation of prefix probabilities for linear context-free rewriting systems. Our approach streamlines previous procedures to compute prefix probabilities for context-free grammars, synchronous context-free grammars and tree adjoining grammars. In addition, the methodology is general enough to be used for a wider range of problems involving, for example, several prefixes.

1 Introduction

There are a number of problems related to probabilistic grammatical formalisms that involve summing an infinite number of values. For example, if P is a probability distribution over strings defined by a probabilistic grammar, and w is a string, then the **prefix probability** of w is defined to be:

$$\sum_{v} P(wv)$$

In words, all possible suffixes v that may follow prefix w are considered, and the probabilities of the concatenations of v and w are summed.

Prefix probabilities can be exploited to predict the next word or part of speech, for incremental processing of text or speech from left to right (Jelinek and Lafferty, 1991). They can also be used in speech processing to score partial hypotheses in beam search (Corazza et al., 1991).

At first sight, it is not clear that prefix probabilities can be effectively computed, as the number of possible strings v is infinite. It was shown however by Jelinek and Lafferty (1991) that in the case of probabilistic context-free grammars, the infinite sums can be isolated from any particular w, and these sums can be computed off-line by solving linear systems of equations. For any particular w, the prefix probability can then be computed in cubic time in the length of w, on the basis of the values computed off-line. Whereas Jelinek and Lafferty (1991) consider parsing in the style of the Cocke-Kasami-Younger algorithm (Younger, 1967; Harrison, 1978), prefix probabilities for probabilistic context-free grammars can also be computed in the style of the algorithm by Earley (1970), as shown by Stolcke (1995).

This approach is not restricted to context-free grammars. It was shown by Nederhof et al. (1998) that prefix probabilities can also be effectively computed for probabilistic tree adjoining grammars. That effective computation is also possible for probabilistic synchronous context-free grammars was shown by Nederhof and Satta (2011b), which departed from earlier papers on the subject in that the solution was divided into a number of steps, namely a new type of transformation of the grammar, followed by elimination of epsilon and unit rules, and the computation of the inside probability of a string.

In this paper we focus on a much more general formalism than those mentioned above, namely that of probabilistic linear context-free rewriting systems (PLCFRS). This formalism is equivalent to the probabilistic simple RCGs discussed by Maier and Søgaard (2008) and by Kallmeyer and Maier (2010), and probabilistic extensions of multiple context-free grammars, such as those considered by Kato et al. (2006). Nonterminals in a PLCFRS can generate discontinuous constituents. For this reason, (P)LCFRSs have recently been used to model discontinuous phrase structure treebanks as well as non-projective dependency treebanks; see (Maier and Lichte, 2009; Kuhlmann and Satta, 2009; Kallmeyer and Maier, 2010).

The main contribution of this paper is a method for computing prefix probabilities for PLCFRSs. We are not aware of any existing algorithm in the literature for this task. Our method implies existence of algorithms for the computation of prefix probabilities for probabilistic versions of formalisms that are special cases of LCFRSs, such as the generalized multitext grammars of Melamed et al. (2004), which are used to model translation, and the already mentioned formalism proposed by Kuhlmann and Satta (2009) to model non-projective dependency structures.

We follow essentially the same approach as Nederhof and Satta (2011b), and reduce the problem of computing the prefix probabilities for PLCFRSs to the well-known problem of computing inside probabilities for PLCFRSs. The reduction is obtained by the composition of a PLCFRS with a special finite-state transducer. Most importantly, this composition is independent of the specific input string for which we need to solve the prefix probability problem, and can therefore be computed off-line. We also show how off-line application of a generic form of epsilon and unit rule elimination for PLCFRS can be exploited, which simplifies the computation of the inside probability.

The rest of this paper is organized as follows. In Section 2 we introduce PLCFRS and finite-state transducers. In Section 3 we discuss a general algorithm for composing a PLCFRS with a finite-state transducer. This construction will be used in several places in later sections. Section 4 shows an effective way of computing the inside probabilities for PLCFRSs, and Section 5 presents a method for the elimination of epsilon and unit rules in a PLCFRS. Section 6 combines all of the previous techniques, resulting in an algorithm for the computation of prefix probabilities via a reduction to the computation of inside probabilities. We then conclude in Section 7 with some discussion.

2 Definitions

This section summarizes the terminology and notation of linear context-free rewriting systems, and their probabilistic extension. For more detailed definitions on linear context-free writing systems, see Vijay-Shanker et al. (1987).

For an integer $n \geq 1$, we write $[n]$ to denote the set $\{1, \dots, n\}$ and $[0] = \emptyset$. We write $[n]_0$ to denote $[n] \cup \{0\}$. A **linear context-free rewriting system** (LCFRS for short) is a tuple $G = (N, \Sigma, P, S)$, where N and Σ are finite, disjoint sets of nonterminal and terminal symbols, respectively. Each $A \in N$ is associated with an integer value $\phi(A) \geq 1$, called its **fan-out**. The nonterminal S is the **start symbol**, with $\phi(S) = 1$. Finally, P is a set of **rules**, each of the form:

$$\pi : A \rightarrow g(A_1, A_2, \dots, A_r)$$

where $A, A_1, \dots, A_r \in N$, and:

$$g : (\Sigma^*)^{\phi(A_1)} \times \cdots \times (\Sigma^*)^{\phi(A_r)} \rightarrow (\Sigma^*)^{\phi(A)}$$

is a linear, non-erasing function. In other words, g takes r tuples of strings as input, the j-th tuple being of size $\phi(A_j)$, and provides as output a tuple of strings of size $\phi(A)$. Each of the output strings is the concatenation of a sequence of elements, each element being an input string or a terminal symbol. Each input string is used precisely once in such a sequence. The number r is called the **rank** of the rule, and is denoted by $\rho(\pi)$ or $\rho(g)$.

The symbol π is the label of the rule, and each rule is uniquely identified by its label. For technical reasons, we allow the existence of multiple rules that are identical apart from their labels. Again for technical reasons, we assume that each function g is uniquely identified with one rule π.

The rank of LCFRS G, written $\rho(G)$, is the maximum rank among all rules of G. The fan-out of LCFRS G, written $\phi(G)$, is the maximum fan-out among all nonterminals of G.

Let a rule π be:

$$\begin{aligned} &\pi : A \rightarrow g(A_1, A_2, \dots, A_r), \text{ where} \\ &\quad g(\langle x_{1,1}, \dots, x_{1,\phi(A_1)} \rangle, \\ &\qquad \dots, \\ &\qquad \langle x_{r,1}, \dots, x_{r,\phi(A_r)} \rangle \,) \;= \\ &\qquad\qquad \langle\; y_{1,1} \cdots y_{1,m_1}, \\ &\qquad\qquad\quad \dots, \\ &\qquad\qquad\quad y_{\phi(A),1} \cdots y_{\phi(A),m_{\phi(A)}} \;\rangle \end{aligned}$$

where for each $k \in [\phi(A)]$ and $l \in [m_k]$, $y_{k,l}$ is from the set $\Sigma \cup \{x_{i,j} \mid i \in [r],\ j \in [\phi(A_i)]\}$. We say π is **monotone** if for each $i \in [r]$, for each $j_1, j_2 \in [\phi(A_i)]$ such that $j_1 < j_2$, and for each $k_1, k_2 \in [\phi(A)]$, $l_1 \in [m_{k_1}]$ and $l_2 \in [m_{k_2}]$ such that $y_{k_1,l_1} = x_{i,j_1}$ and $y_{k_2,l_2} = x_{i,j_2}$, we have that either $k_1 < k_2$, or $k_1 = k_2$ and $l_1 < l_2$. In other words, the order of variables associated with each of the right-hand side nonterminals is preserved in the output of function g. In this paper we will assume all rules in a LCFRS are monotone. Restriction to monotone rules preserves the generative power of LCFRS (Michaelis, 2001; Kracht, 2003; Kallmeyer, 2010). Furthermore, known techniques to extract (probabilistic) LCFRSs from treebanks all provide grammars with monotone rules (Maier and Lichte, 2009; Kuhlmann and Satta, 2009).

For each nonterminal A in a LCFRS G, there is a set of **derivations** for A, denoted $D_G(A)$, or $D(A)$ when G is understood. The simultaneous definition of $D_G(A)$ for all A is inductively as follows. Let:

$$\pi : A \to g(A_1, A_2, \dots, A_r)$$

be a rule of G, and let $\zeta_1, \dots, \zeta_r$ be derivations for $A_1, \dots, A_r$. Then the expression $g(\zeta_1, \dots, \zeta_r)$ is a derivation for A, with an *uninterpreted* function symbol g. When $r = 0$, we write the derivation as $g()$.

The **yield** of a derivation ζ for nonterminal A in grammar G is the $\phi(A)$-tuple of strings resulting from evaluating all occurrences in ζ of function symbols. It will be denoted as $\mathit{yield}_G(\zeta)$, with once more G omitted when the grammar is understood. The language **generated** by G, denoted $L(G)$, is the set of all strings w such that $\langle w \rangle$ is the yield of a derivation for S; formally, $L(G) = \{w \mid \langle w \rangle \in \mathit{yield}_G(\zeta),\ \zeta \in D_G(S)\}$.

Example 1 Consider the LCFRS G defined by the rules:

$$\begin{aligned}
&\pi_1 : S \to g_1(A), \text{ where} \\
&\qquad g_1(\langle x_{1,1}, x_{1,2} \rangle) = \langle x_{1,1} x_{1,2} \rangle \\
&\pi_2 : A \to g_2(A), \text{ where} \\
&\qquad g_2(\langle x_{1,1}, x_{1,2} \rangle) = \langle a x_{1,1} b, c x_{1,2} d \rangle \\
&\pi_3 : A \to g_3(), \text{ where} \\
&\qquad g_3() = \langle \varepsilon, \varepsilon \rangle
\end{aligned}$$

We have $\phi(S) = 1$, $\phi(A) = \phi(G) = 2$, $\rho(\pi_3) = 0$ and $\rho(\pi_1) = \rho(\pi_2) = \rho(G) = 1$. G generates the language $\{a^n b^n c^n d^n \mid n \in \mathbb{N}\}$. For instance, the string $a^3 b^3 c^3 d^3$ is obtained through the yield $\langle a^3 b^3 c^3 d^3 \rangle$ of the derivation $g_1(g_2(g_2(g_2(g_3()))))$ for S. □

Let $\pi : A \to g(A_1, A_2, \dots, A_r)$ be a rule in G and let d_π be the number of occurrences of terminal symbols in the definition of the associated function g. We define the size of π as:

$$|\pi| = d_\pi + \phi(A) + \sum_{i \in [r]} \phi(A_i)$$

It is not difficult to see that we can encode rule π and its associated function g using space linear in $|\pi|$. We define the size of G as $|G| = \sum_{\pi \in P} |\pi|$, where P is the set of all rules in G.

A LCFRS is said to be **reduced** if the function g from each rule occurs in some derivation from $D(S)$. Because each function uniquely identifies a rule, this means that also all rules are useful for obtaining some derivation for S. A procedure for transforming a LCFRS to make it reduced will be discussed in Section 3.

A **probabilistic** LCFRS (PLCFRS for short) is a pair $\mathcal{G} = (G, p)$ where $G = (N, \Sigma, P, S)$ is a LCFRS and p is a function from P to real numbers in $[0, 1]$. We say that $\mathcal{G}$ is **proper** if for each A:

$$\sum_{\pi : A \to g(A_1, A_2, \dots, A_{\rho(g)})} p(\pi) \;=\; 1$$

The probability of a derivation ζ in G, denoted $L_{\mathcal{G}}(\zeta)$, is obtained by multiplying the probability of the rule corresponding to each occurrence of a function symbol in ζ. The probability of a string w, denoted $L_{\mathcal{G}}(w)$, is the sum of the probabilities of all derivations in $D(S)$ of which the yield is $\langle w \rangle$. A PLCFRS $\mathcal{G}$ is **consistent** if $\sum_w L_{\mathcal{G}}(w) = 1$. It is not difficult to see that if a PLCFRS $\mathcal{G}$ is reduced, proper and consistent, then for each A, $\sum_{\zeta \in D(A)} L_{\mathcal{G}}(\zeta) = 1$.

Example 2 Consider the extension of the LCFRS from the previous example with function p defined by $p(\pi_1) = 1$, $p(\pi_2) = 0.3$ and $p(\pi_3) = 0.7$. The resulting PLCFRS $\mathcal{G}$ is proper, as the sum of probabilities of all rules with left-hand side S is 1, and so is the sum of probabilities of all rules with left-hand side A. The probability of derivation $g_1(g_2(g_2(g_2(g_3()))))$ is $1 \cdot (0.3)^3 \cdot 0.7$. This is also the probability of the string $a^3 b^3 c^3 d^3$. It can be easily shown that $\mathcal{G}$ is consistent, as:

$$\sum_{j=0}^{\infty} 1 \cdot (0.3)^j \cdot 0.7 \;=\; 1$$

□

In the following sections, we need to consider PLCFRSs $\mathcal{G}$ that are not necessarily proper or consistent, and our discussion will involve computation of values $L_{\mathcal{G}}(A)$ for each nonterminal A, defined by:

$$L_{\mathcal{G}}(A) \;=\; \sum_{\zeta \in D(A)} L_{\mathcal{G}}(\zeta)$$

In the literature, $L_{\mathcal{G}}$ is also called the **partition function** of the grammar. We can relate these values by means of the following equations, one for each nonterminal A:

$$L_{\mathcal{G}}(A) = \sum_{\pi:A\rightarrow g(A_1,\ldots,A_{\rho(g)})} p(\pi) \cdot \prod_{j\in[\rho(g)]} L_{\mathcal{G}}(A_j)$$

The above relations specify a system of polynomial, nonlinear equations. The smallest nonnegative solution to this system gives us the values of $L_{\mathcal{G}}(A)$ for all A.

The sought solutions for the nonlinear system described above can be irrational and non-expressible by radicals, even if we assume that all the rules of our LCFRS have probabilities that are rational numbers, as observed by Etessami and Yannakakis (2009). Hence values $L_{\mathcal{G}}(A)$ can only be approximated.

Approximate solutions can be obtained using the fixed-point iteration method, as described for example in (Abney et al., 1999). This method is easy to implement but shows very slow convergence in the worst case, resulting in exponential time behaviour (Etessami and Yannakakis, 2009).

Alternatively, the more powerful Newton's method can be exploited; see again (Etessami and Yannakakis, 2009). It has been shown by Kiefer et al. (2007) that, after a certain number of initial iterations, each new iteration of Newton's method adds a fixed number of bits to the precision of the approximate solution, resulting in polynomial time convergence in the size of the grammar and the number of bits in the desired approximation. However, Kiefer et al. (2007) also show that, in some degenerate cases, the number of iterations needed to compute the first bit of the solution can be at least exponential in the size of the system. For further discussion of practical issues in the computation of values $L_{\mathcal{G}}(A)$, we refer the reader to Wojtczak and Etessami (2007) and Nederhof and Satta (2008).

Despite the computational limitations discussed above, there are cases in which values $L_{\mathcal{G}}(A)$ can be computed exactly, and simpler methods can be exploited. This happens when there are no cyclic dependencies in the probabilistic grammar. This is the situation we will deal with in Section 4.

In this paper, we will consider a type of **finite-state transducer** (FST) that has a single final state and that consumes exactly one input symbol in each transition. Although such restricted FSTs cannot describe all rational transductions (Berstel, 1979), they are sufficient for our purposes, and the restrictions greatly simplify the definitions in the following sections.

Hence, our type of FST can be defined by a tuple $M = (\Sigma_1, \Sigma_2, Q, q_s, q_f, T)$, where Σ_1 and Σ_2 are finite sets of input and output symbols, respectively, Q is a finite set of **states**, of which q_s is the **start state** and q_f is the **final state**, and T is a finite set of **transitions**.

Each transition has the form $s \overset{a,u}{\mapsto} s'$, where $s, s' \in Q$, $a \in \Sigma_1$ and $u \in \Sigma_2 \cup \{\epsilon\}$. This means that we can jump from s to s' by reading a from the input, and generating u as output.

The transduction generated by FST M, denoted $L(M)$, is the set of all pairs $\langle w, v\rangle$ such that there is a sequence of n transitions $s_0 \overset{a_1,u_1}{\mapsto} s_1, \ldots, s_{n-1} \overset{a_n,u_n}{\mapsto} s_n$, such that $s_0 = q_s$, $s_n = q_f$, $a_1 \cdots a_n = w$, and $u_1 \cdots u_n = v$. The definition allows $n = 0$, in which case q_f must be q_s and the transduction includes $\langle \varepsilon, \varepsilon\rangle$. We say a FST M is **unambiguous** if each pair $\langle w, v\rangle$ is obtained by at most one sequence of transitions as above.

3 Composition

Seki et al. (1991, Thm 3.9(3), pg. 203) have shown that the class of languages generated by LCFRSs are closed under intersection with regular languages.[1] The proof is a generalization of the proof that context-free languages are closed under intersection with regular languages (Bar-Hillel et al., 1964). We slightly extend this result by showing that if we point-wise map the language generated by a LCFRS G by means of a FST M, then the resulting language is generated by another LCFRS G', or formally, $L(G') = \{v \mid w \in L(G) \wedge \langle w, v\rangle \in L(M)\}$. As we will show below, LCFRS G' can be effectively constructed from G and M. The construction is denoted by operator $\circ$, hence $G' = G \circ M$. The construction can be extended to take as input a PLCFRS $\mathcal{G}$ and a FST M and the output is then another PLCFRS $\mathcal{G}'$, and we denote $\mathcal{G}' = \mathcal{G} \circ M$. If the FST is unambiguous, then the probabilities are preserved, or formally:

$$L_{\mathcal{G}'}(v) = \sum_{\langle w,v\rangle\in L(M)} L_{\mathcal{G}}(w)$$

This follows directly from the fact that rule probabilities from $\mathcal{G}$ are copied unchanged to $\mathcal{G}'$.

Without loss of generality, we will assume that $G = (N, \Sigma, P, S)$ and $M = (\Sigma_1, \Sigma_2, Q, q_s, q_f, T)$, with $\Sigma = \Sigma_1$. The construction produces $G' = (N', \Sigma_2, P', S')$.

[1]The result by Seki et al. (1991) is obtained for a syntactic variant of LCFRSs called multiple context-free grammars.

The nonterminals in N' are of the form $A(\langle s_1, s'_1\rangle, \ldots, \langle s_{\phi(A)}, s'_{\phi(A)}\rangle)$, where $A \in N$ and $s_1, s'_1, \ldots, s_{\phi(A)}, s'_{\phi(A)} \in Q$. The intuition behind this definition is discussed in what follows. From the derivations for A in G, we take the subset of those that have yield $\langle w_1, ..., w_{\phi(A)}\rangle$ such that for each $j \in [\phi(A)]$ it must be possible to take the automaton from state s_j to state s'_j while consuming input w_j. In addition, input symbols are replaced by the corresponding output strings, according to the relevant transitions of the automaton. The start symbol S' is naturally $S(\langle q_s, q_f\rangle)$, as the composition ultimately needs to match strings that are generated by G against those input strings that take the automaton from the start state to the final state.

The rules of G' are constructed by exhaustively applying the following procedure. Choose a rule from G of the form:

$$\begin{aligned}
&\pi : A \rightarrow g(A_1, A_2, \ldots, A_r), \text{ where}\\
&\quad g(\ \langle x_{1,1}, \ldots, x_{1,\phi(A_1)}\rangle,\\
&\qquad \ldots,\\
&\qquad \langle x_{r,1}, \ldots, x_{r,\phi(A_r)}\rangle\ ) =\\
&\qquad\quad \langle\ y_{1,1} \cdots y_{1,m_1},\\
&\qquad\qquad \ldots,\\
&\qquad\qquad y_{\phi(A),1} \cdots y_{\phi(A),m_{\phi(A)}}\ \rangle
\end{aligned}$$

where for each $k \in [\phi(A)]$ and $l \in [m_k]$, $y_{k,l}$ is from the set $\Sigma \cup \{x_{i,j} \mid i \in [r],\ j \in [\phi(A_i)]\}$. Further choose two states $s_{i,j}, s'_{i,j}$ from M for each $i \in [r]$ and each $j \in [\phi(A_i)]$, and choose two states $q_{k,l}, q'_{k,l}$ from M for each $k \in [\phi(A)]$ and each $l \in [m_k]$, under the following constraints, which if satisfied define $z_{k,l}$ for $k \in [\phi(A)]$ and $l \in [m_k]$:

- if $y_{k,l} = x_{i,j}$, then $q_{k,l}$ must be $s_{i,j}$ and $q'_{k,l}$ must be $s'_{i,j}$, and we let $z_{k,l} = x_{i,j}$,
- if $y_{k,l} = a \in \Sigma$ then we must be able to choose a transition $q_{k,l} \overset{a,u}{\mapsto} q'_{k,l}$, and we let $z_{k,l} = u$,
- for each $k \in [\phi(A)]$ and each $l \in [m_k - 1]$, $q'_{k,l} = q_{k,l+1}$

In words, all variables coming from right-hand side nonterminals are associated with two states, whose intended meaning was explained before. In addition, each terminal occurrence in the output of the function g must correspond to a transition between two states. Lastly, an output component of the form $y_{k,1} \cdots y_{k,m_k}$ must match a contiguous path through the automaton, with each $y_{k,l}$ representing a segment of that path from state $q_{k,l}$ to state $q'_{k,l}$.

This determines a rule in G', which is of the form:

$$\begin{aligned}
&\pi' : A' \rightarrow g'(A'_1, A'_2, \ldots, A'_r), \text{ where}\\
&\quad g'(\ \langle x_{1,1}, \ldots, x_{1,\phi(A_1)}\rangle,\\
&\qquad \ldots,\\
&\qquad \langle x_{r,1}, \ldots, x_{r,\phi(A_r)}\rangle\ ) =\\
&\qquad\quad \langle\ z_{1,1} \cdots z_{1,m_1},\\
&\qquad\qquad \ldots,\\
&\qquad\qquad z_{\phi(A),1} \cdots z_{\phi(A),m_{\phi(A)}}\ \rangle
\end{aligned}$$

Here A' is:

$$A(\langle q_{1,1}, q'_{1,m_1}\rangle, \ldots, \langle q_{\phi(A),1}, q'_{\phi(A),m_{\phi(A)}}\rangle)$$

and for each $i \in [r]$, A'_i is:

$$A_i(\langle s_{i,1}, s'_{i,1}\rangle, \ldots, \langle s_{i,\phi(A_i)}, s'_{i,\phi(A_i)}\rangle)$$

A new label π' and new function name g' are produced for each rule in G' that is constructed as above.

If we are dealing with probabilities, then the rules in $\mathcal{G}' = \mathcal{G} \circ M$ are constructed in the same manner, and in addition, the probability of each rule π is copied to each rule π' constructed from it.

Example 3 Assume the following is a rule in G:

$$\begin{aligned}
&\pi : A \rightarrow g(A_1, A_2), \text{ where}\\
&\quad g(\langle x_{1,1}, x_{1,2}\rangle, \langle x_{2,1}, x_{2,2}, x_{2,3}\rangle) =\\
&\qquad\quad \langle x_{1,1} a x_{2,1}, x_{2,2} x_{1,2}, x_{2,3}\rangle
\end{aligned}$$

Further assume the existence of states $s_1, \ldots, s_9$ in M and the existence of a transition $s_2 \overset{a,b}{\mapsto} s_3$ for M. Then we add to G' the rule:

$$\begin{aligned}
&\pi' : A(\langle s_1, s_4\rangle, \langle s_5, s_7\rangle, \langle s_8, s_9\rangle) \rightarrow g'(\\
&\quad A_1(\langle s_1, s_2\rangle, \langle s_6, s_7\rangle),\\
&\quad A_2(\langle s_3, s_4\rangle, \langle s_5, s_6\rangle, \langle s_8, s_9\rangle)), \text{ where}\\
&\qquad g'(\langle x_{1,1}, x_{1,2}\rangle, \langle x_{2,1}, x_{2,2}, x_{2,3}\rangle) =\\
&\qquad\quad \langle x_{1,1} b x_{2,1}, x_{2,2} x_{1,2}, x_{2,3}\rangle
\end{aligned}$$

□

A LCFRS $G' = G \circ M$ (or PLCFRS $\mathcal{G}' = \mathcal{G} \circ M$) is generally not reduced. We can make it reduced by a process almost identical to the process of reduction for context-free grammars (Sippu and

Soisalon-Soininen, 1988). This consists of three phases. First, there is a bottom-up phase to identify the 'generating' nonterminals, that is, those that have derivations. Second, restricting attention to the generating nonterminals, a top-down phase identifies the possibly smaller set of nonterminals that are also reachable from the start symbol. Lastly, all rules are removed except those that are generating and reachable.

Computation Let us first relate $|G'|$ to $|G|$, where $G' = G \circ M$. To simplify the discussion, consider a rule of G of the form $\pi : A \rightarrow g(A_1, A_2, \ldots, A_r)$ without terminal symbols, or in other words $d_\pi = 0$. Let Q be the set of states of M. Then the composition construction defines a new rule in G' for each possible choice of a number of states in Q equal to $\phi(A) + \sum_{i \in [r]} \phi(A_i)$. This results in $|Q|^{|\pi|}$ new rules in G' that are derived from π, where each new rule has size $\mathcal{O}(|\pi|)$. Thus the target grammar has size exponential in $|G|$.

Our algorithm for composition can be easily implemented to run in time $\mathcal{O}(|G'|)$, that is, in linear time in the size of the output grammar. Because of the above discussion, the algorithm runs in exponential time in the size of the input. Exponential time for the composition construction is not unexpected: the problem at hand is a generalization of the parsing problem for LCFRS, and the latter problem is known to be NP-hard when the grammar is part of the input (Satta, 1992).

The critical term in the above analysis is $|\pi|$. If we can cast our LCFRS in a form in which each rule has length bounded by some constant, then composition can be carried out in polynomial time. The process of reducing the length of rules in a LCFRS is called **factorization**. It is known that not all LCFRSs can be factorized in such a way that each rule has length bounded by some constant (Rambow and Satta, 1999). However, in the context of natural language parsing, it has been observed that the vast majority of rules in real world applications can be factorized to some small length, and that excluding the worst-case rules which cannot be handled in this way does not significantly affect accuracy; see for instance (Huang et al., 2009) and (Kuhlmann and Satta, 2009) for discussion. Efficient algorithms for factorization of LCFRSs have been presented by Kuhlmann and Satta (2009), Gómez-Rodríguez and Satta (2009) and Sagot and Satta (2010).

Finally, the procedure for reducing a LCFRS that we outlined in this section can be easily implemented to run in time linear in the size of the source grammar.

4 Effective LCFRS Parsing

String parsing with LCFRS G and input $w = a_1 \cdots a_n \in \Sigma^*$ can be described in terms of composition as follows. We construct a FST M_w with states $s_0, \ldots, s_n$, of which s_0 is the start state and s_n is the final state, and transitions $s_{i-1} \overset{a_i, a_i}{\mapsto} s_i$ for each $i \in [n]$. In words, the FST describes a mapping from only one string w to itself. The composition $G' = G \circ M_w$ restricts the derivations from G to only those that have $\langle w \rangle$ as yield.

This approach is consistent with the observation by Lang (1994) that parsing is a form of intersection. In the case of very ambiguous grammars, the approach has the advantage that the set of all parse trees is represented in a compact manner by the intersection grammar, or $G' = G \circ M_w$ in our case.

Because M_w is unambiguous, the composition also has favourable properties for PLCFRS $\mathcal{G}$. In particular, the probability $L_{\mathcal{G}}(w)$ of a string w can be determined by summing the probabilities of all derivations of PLCFRS $\mathcal{G}' = \mathcal{G} \circ M_w$, or formally:

$$L_{\mathcal{G}}(w) = \textstyle\sum_v \; L_{\mathcal{G}'}(v) = L_{\mathcal{G}'}(S(\langle s_0, s_n \rangle))$$

We have thus reduced the problem of computing the inside probability of w under $\mathcal{G}$ to the problem of computing the values of the partition function for $\mathcal{G}'$. Because $L_{\mathcal{G}}(w)$ can be less than 1, it is clear that $\mathcal{G}'$ need not be consistent, even if we assume that $\mathcal{G}$ is.

As we have discussed in Section 2, if $\mathcal{G}'$ is any PLCFRS that may not be proper or consistent, then the values $L_{\mathcal{G}'}(A)$ for the different nonterminals of $\mathcal{G}'$ can be expressed in terms of a system of equations. Solving such equations can be computationally expensive.

A more efficient way to compute the values $L_{\mathcal{G}'}(A)$ is possible however if there are no cyclic dependencies in $\mathcal{G}'$, that is, in the system of equations specified in Section 2, no value is defined in terms of itself. This can be ensured if the functions g of the rules of the grammar are such that the multiset of non-empty strings in the output are never the same as the multiset of non-empty strings in the input arguments. This in turn is ensured if the grammar contains no epsilon rules and no unit rules.

Analogously to the theory of context-free grammars, we define an **epsilon rule** to be of the form:

$$\pi : A \rightarrow g(), \text{ where } g() = \langle \varepsilon, \ldots, \varepsilon \rangle$$

We define a **unit rule** to be of the form:

$$\pi : A \rightarrow g(A_1), \text{ where}$$
$$g(\langle x_1, \ldots, x_{\phi(A_1)} \rangle) = \langle x_1, \ldots, x_{\phi(A_1)} \rangle$$

Note that we assume that $\phi(A_1) = \phi(A)$. The reason why we do not need to consider a reordering of variables is because we have assumed that all LCFRSs are monotone, as mentioned in Section 2. In Section 5 we will show that a LCFRS can be transformed to eliminate all epsilon rules and unit rules, while preserving the language as well as the assignment of probabilities to strings. In the remainder of the present section, we discuss the computation of the values $L_{\mathcal{G}'}(A)$ under the assumption that we do not have to deal with cyclic dependencies.

We define a binary relation $\prec$ over sequences of strings by letting $\langle w_1, \ldots, w_k \rangle \prec \langle v_1, \ldots, v_m \rangle$ if and only if $|w_1 \cdots w_k| < |v_1 \cdots v_m|$ or $|w_1 \cdots w_k| = |v_1 \cdots v_m|$ and $m < k$. In words, a sequence is smaller than another if it contains fewer occurrences of symbols, and when the two sequences contain the same number of occurrences of symbols, then the first is smaller if the symbol occurrences are distributed over more strings.

Let w be any string. If LCFRS G does not contain epsilon rules or unit rules, then LCFRS $G' = G \circ M_w$ has the following property. If $\zeta = g'(\zeta_1, \ldots, \zeta_r)$ is a derivation for some nonterminal A' in G', then *yield*$(\zeta_i) \prec$ *yield*(ζ) for every $i \in [r]$. Note that *yield*$(\zeta) = g'($*yield*$(\zeta_1), \ldots,$ *yield*$(\zeta_r))$.

If we extend this to the probabilistic case, with $\mathcal{G}' = \mathcal{G} \circ M_w$, $\mathcal{G} = (G, p)$ and $\mathcal{G}' = (G', p')$, then $L_{\mathcal{G}'}(\zeta) = p'(\pi') \cdot \prod_i L_{\mathcal{G}'}(\zeta_i)$, where π' is the label of the rule in which g' occurs. If we use the fact that multiplication distributes over addition, we derive:

$$L_{\mathcal{G}'}(A') = \sum_{\pi' : A' \rightarrow g'(A'_1, \ldots, A'_{\rho(g')})} p'(\pi') \cdot \prod_{i \in [\rho(g')]} L_{\mathcal{G}'}(A'_i)$$

Recall that M_w has the set of states $\{s_0, \ldots, s_n\}$, where $n = |w|$. Using the notation in Section 3, A' is therefore of the form:

$$A(\langle s_{b_{1,1}}, s_{b'_{1,m_1}} \rangle, \ldots, \langle s_{b_{\phi(A),1}}, s_{b'_{\phi(A),m_{\phi(A)}}} \rangle),$$

where for each $k \in [\phi(A)]$ and $l \in [m_k]$, $b_{k,l}$ and $b'_{k,l}$ are integers in $[n]_0$. Furthermore, for each $i \in [\rho(g')]$, A'_i is of the form:

$$A_i(\langle s_{c_{i,1}}, s_{c'_{i,1}} \rangle, \ldots, \langle s_{c_{i,\phi(A_i)}}, s_{c'_{i,\phi(A_i)}} \rangle),$$

where for each $j \in [\phi(A_i)]$, $c_{i,j}$ and $c'_{i,j}$ are integers in $[n]_0$.

By associating each pair $\langle s_{b_{k,l}}, s_{b'_{k,l}} \rangle$ (or $\langle s_{c_{i,j}}, s_{c'_{i,j}} \rangle$) with a corresponding substring $a_{b_{k,l}+1} \cdots a_{b'_{k,l}}$ (or $a_{c_{i,j}+1} \cdots a_{c'_{i,j}}$, respectively) of w, we obtain sequences of strings for A'_i that are smaller than those for A', by relation $\prec$. Because $\prec$ is acyclic, this means we can compute $L_{\mathcal{G}'}(A'_i)$ strictly before computing $L_{\mathcal{G}'}(A')$. All values of $L_{\mathcal{G}'}$ for different nonterminals can be obtained by enumerating all sequences of substrings from w in an order that is consistent with $\prec$. We refer to this procedure as the **inside algorithm** for PLCFRSs.

Computation It is not difficult to implement the inside algorithm in such a way that all values $L_{\mathcal{G}'}(A)$ can be computed in time linear in the size of $G' = G \circ M_w$. This is essentially a generalization of the well-known inside algorithm for probabilistic context-free grammars (Manning and Schütze, 1999) to a special kind of non-recursive PLCFRS.

5 Grammar Transformations

In this section we introduce grammar transformations that remove cyclic dependencies from LCFRS, and discuss their application to a special class of PLCFRSs.

We say a nonterminal A is **nullable** in grammar G if there is a derivation $\zeta \in D_G(A)$ with *yield*$_G(\zeta) = \langle \varepsilon, \ldots, \varepsilon \rangle$. The set E_G of nullable nonterminals in G can be found as a straightforward generalization of the algorithm to find nullable nonterminals in a context-free grammar (Sippu and Soisalon-Soininen, 1988). Similarly, we can construct the set of all nonterminals A for which at least one derivation $\zeta \in D_G(A)$ has a yield containing at least one non-empty string. We denote this set by $\overline{E}_G$.

A LCFRS G can now be transformed into a LCFRS G' without epsilon rules, by applying the following exhaustively.[2] Take a rule from G:

$$\begin{aligned}\pi : A \rightarrow g(A_1, \ldots, A_r),\ \text{where}\\ g(\ \langle x_{1,1}, \ldots, x_{1,\phi(A_1)}\rangle,\\ \ldots,\\ \langle x_{r,1}, \ldots, x_{r,\phi(A_r)}\rangle\ ) =\\ \langle \alpha_1, \ldots, \alpha_{\phi(A)}\rangle\end{aligned}$$

and determine:

$$\begin{aligned}\mathcal{U}_\pi &= \{i \mid i \in [r],\ A_i \in E_G\}\\ \overline{\mathcal{U}}_\pi &= \{i \mid i \in [r],\ A_i \in \overline{E}_G\}\end{aligned}$$

and then choose any set $\mathcal{U}$ with $(\mathcal{U}_\pi \setminus \overline{\mathcal{U}}_\pi) \subseteq \mathcal{U} \subseteq \mathcal{U}_\pi$. That is, the set must contain all indices of right-hand side nonterminals that only generate yields with empty strings, and it may additionally contain the indices of other nullable nonterminal occurrences. Now construct the rule:

$$\begin{aligned}\pi_{\mathcal{U}} : A \rightarrow g_{\mathcal{U}}(B_1, \ldots, B_{r'}),\ \text{where}\\ g_{\mathcal{U}}(\ \langle x'_{1,1}, \ldots, x'_{1,\phi(B_1)}\rangle,\\ \ldots,\\ \langle x'_{r',1}, \ldots, x'_{r',\phi(B_{r'})}\rangle\ ) =\\ \langle \alpha'_1, \ldots, \alpha'_{\phi(A)}\rangle\end{aligned}$$

where $B_1, \ldots, B_{r'}$ is obtained from $A_1, \ldots, A_r$ by omitting A_i if $i \in \mathcal{U}$, where $r' = r - |\mathcal{U}|$. Similarly, $g_{\mathcal{U}}$ has only r' arguments, by omitting its i-th argument if $i \in \mathcal{U}$. Lastly, for each $k \in [\phi(A)]$, α'_k is obtained from α_k by omitting any variables of the form $x_{i,j}$ where $i \in \mathcal{U}$. The constructed rule $\pi_{\mathcal{U}}$ now becomes a rule in the transformed grammar G' if it is not an epsilon rule.

Example 4 Assume a rule in G of the form:

$$\begin{aligned}\pi : A \rightarrow g(A_1, A_2, A_3),\ \text{where}\\ g(\langle x_1\rangle, \langle x_2\rangle, \langle x_3\rangle) = \langle x_3 a x_1 x_2\rangle\end{aligned}$$

Further assume that E_G includes A_1 and A_2 but not A_3, and $\overline{E}_G$ includes A_1 and A_3 but not A_2. Then G' will contain:

$$\begin{aligned}\pi_{\{2\}} : A \rightarrow g_{\{2\}}(A_1, A_3),\ \text{where}\\ g_{\{2\}}(\langle x_1\rangle, \langle x_3\rangle) = \langle x_3 a x_1\rangle\\ \pi_{\{1,2\}} : A \rightarrow g_{\{1,2\}}(A_3),\ \text{where}\\ g_{\{1,2\}}(\langle x_3\rangle) = \langle x_3 a\rangle\end{aligned}$$

$\square$

In the case of a PLCFRS $\mathcal{G} = (G, p)$ being transformed to $\mathcal{G}' = (G', p')$, we have:

$$p'(\pi_{\mathcal{U}}) = p(\pi) \cdot \prod_{i \in \mathcal{U}} \sum_{\substack{\zeta \in D_G(A_i) :\\ \textit{yield}(\zeta) = \langle \varepsilon, \ldots, \varepsilon\rangle}} L_{\mathcal{G}}(\zeta)$$

The summation of all derivations with yields consisting only of empty strings is difficult in general, and involves the solution of a system of polynomial, nonlinear equations, a topic which we addressed in Section 2.

However, there is a special case that can be easily dealt with, namely that $E_G \cap \overline{E}_G = \emptyset$ and for each $A \in E_G$:

$$\sum_{\substack{\zeta \in D_G(A) :\\ \textit{yield}(\zeta) = \langle \varepsilon, \ldots, \varepsilon\rangle}} L_{\mathcal{G}}(\zeta) = 1$$

This means that there is precisely one possible set $\mathcal{U}$ for each rule π, and $p'(\pi_{\mathcal{U}})$ will always be identical to $p(\pi)$. In this case the overall construction of elimination of nullable rules from PLCFRS $\mathcal{G}$ can be implemented in time linear in $|G|$. It is this special case that we will encounter in Section 6.

Now we turn to elimination of unit rules from a PLCFRS $\mathcal{G}$. We investigate sequences of applications of unit rules, multiplying the probabilities of all rule occurrences, and adding the probabilities of all such sequences between each pair of nonterminals. Formally, for each pair A and B of nonterminals, we have a value $\Delta_{\mathcal{G}}(A, B)$, and we write:

$$\begin{aligned}\Delta_{\mathcal{G}}(A, B) &= \delta(A = B)\ +\\ &+ \sum_{\substack{\pi : A \rightarrow g(A_1),\ \text{where}\\ g(\langle x_1, \ldots, x_{\phi(A_1)}\rangle) =\\ \langle x_1, \ldots, x_{\phi(A_1)}\rangle}} p(\pi) \cdot \Delta_{\mathcal{G}}(A_1, B)\end{aligned}$$

where $\delta(A = B)$ is defined to be 1 if $A = B$ and 0 otherwise. This forms a system of linear equations in the unknown variables $\Delta_{\mathcal{G}}(\cdot, \cdot)$. Such a system can be solved in polynomial time in the number of variables, for example using Gaussian elimination.

In order to construct the transformed grammar $\mathcal{G}'$, we exhaustively choose a rule from $\mathcal{G}$:

$$\pi : A \rightarrow g(A_1, A_2, \ldots, A_r)$$

[2]Removal of epsilon rules is also investigated by Seki et al. (1991, Lemma 2.2(N2), pg. 197) for multiple context-free grammars, a syntactic variant of LCFRSs. Here we extend the result to the probabilistic version of LCFRSs.

and choose B such that $\Delta_{\mathcal{G}}(B, A) > 0$, and let $\mathcal{G}'$ contain the rule:

$$\pi_B : B \rightarrow g_B(A_1, A_2, \ldots, A_r)$$

where g_B is defined to be identical to g. The probability of π_B is $p'(\pi_B) = p(\pi) \cdot \Delta_{\mathcal{G}}(B, A)$.

The size of $\mathcal{G}'$ is quadratic in $\mathcal{G}$. The time complexity is dominated by the computation of the solution of the linear system of equations. This computation takes cubic time in the number of variables. The number of variables in this case is $\mathcal{O}(|\mathcal{G}|^2)$, which makes the running time $\mathcal{O}(|\mathcal{G}|^6)$.

6 Prefix Probabilities

In this section we gather all the constructions that have been presented in the previous sections, and provide the main result of this paper.

Let $\mathcal{G}$ be a reduced, proper and consistent PLCFRS. We assume $\mathcal{G}$ does not contain any epsilon rules, and therefore the empty string is not in the generated language.

The first step towards the computation of prefix probabilities is the construction of a FST M_{pref} that maps any string w over an assumed input alphabet to any non-empty prefix of w. In other words, its transduction is the set of pairs $\langle wv, w\rangle$, for any non-empty string w and any string v, both over the alphabet Σ of $\mathcal{G}$. The reason why we do not consider empty prefixes w is because this simplifies the definition of M_{pref} below, assuming there can be only one final state. The restriction is without loss of generality, as the computation of the prefix probability of the empty string is easy: it is always 1.

The transducer M_{pref} has two states, q_s and q_f, which are also the start and final states, respectively. The transitions have the following forms, for each $a \in \Sigma$:

$$q_s \overset{a,a}{\mapsto} q_s$$
$$q_s \overset{a,a}{\mapsto} q_f$$
$$q_f \overset{a,\varepsilon}{\mapsto} q_f$$

In words, symbols are copied unchanged from input to output as long as the automaton is in the start state. After it jumps to the final state, no more output can be produced.

Note that if we consider any pair of states s and s', such that the automaton can reach s' from s, then precisely one of the following two cases is possible:

- On any path through the automaton, the empty string is produced in the output. This applies when $s = s' = q_f$.
- On any path through the automaton, a non-empty string is produced in the output. This applies when $s = s' = q_s$, or $s = q_s$ and $s' = q_f$.

Note that in the first case, for $s = s' = q_f$, any string can be consumed in the input, in exactly one way. This has an important implication for the composition $\mathcal{G}' = \mathcal{G} \circ M_{pref}$, namely that all nullable nonterminals in $\mathcal{G}'$ must be of the form $A' = A(\langle q_f, q_f\rangle, \ldots, \langle q_f, q_f\rangle)$. By the construction of $\mathcal{G}'$, and by the assumption that $\mathcal{G}$ is reduced, proper and consistent we have:

$$\sum_{\substack{\zeta \in D_{\mathcal{G}'}(A') : \\ \mathit{yield}(\zeta) = \langle \varepsilon, \ldots, \varepsilon\rangle}} L_{\mathcal{G}'}(\zeta) = \sum_{\zeta \in D_{\mathcal{G}(A)}} L_{\mathcal{G}}(\zeta) \;=\; 1$$

Therefore, we can apply a simplified procedure to eliminate epsilon rules, maintaining the probability of a rule where we eliminate nullable nonterminals from its right-hand side, as explained in Section 5.

After also unit rules have been eliminated, by the procedure in Section 5, we obtain a grammar $\mathcal{G}''$ without epsilon rules and without unit rules. For a given prefix w we can now construct $\mathcal{G}'' \circ M_w$, and apply the inside algorithm, by which we obtain $L_{\mathcal{G}''}(w)$, which is the required prefix probability of w by $\mathcal{G}$.

Computation Consider the PLCFRS $\mathcal{G}' = \mathcal{G} \circ M_{pref}$; assume $\mathcal{G}'$ is subsequently reduced. Due to the special topology of M_{pref} and to the assumption that G is monotonic, it is not difficult to see that all nonterminals of G' have the form $A(\langle s_1, s_2\rangle, \ldots, \langle s_{2\phi(A)-1}, s_{2\phi(A)}\rangle)$ such that, for some $k \in [2\phi(A)]_0$, we have $s_i = q_s$ for each $i \in [k]$ and $s_i = q_f$ for each $i \in [2\phi(A)] \setminus [k]$. Related to this, for each rule π of G, there are only $\mathcal{O}(|\pi|)$ many rules in G' (as opposed to a number exponential in $|\pi|$, as in the general case discussed in Section 3). Since each new rule in G' constructed from π has size proportional to $|\pi|$, we may conclude that $|G'| = \mathcal{O}(|G|^2)$. Furthermore, $\mathcal{G}'$ can be computed in quadratic time.

The simplified procedure for eliminating epsilon rules and the procedure for eliminating unit rules both take polynomial time, as discussed in

Section 5. This results in a PLCFRS $\mathcal{G}''$ such that $|G''|$ is polynomially related to $|G|$, where G is the source LCFRS. Note that $\mathcal{G}''$ can be computed off-line, that is, independently of the specific input string for which we need to compute the prefix probability.

Finally, computation of $\mathcal{G}'' \circ M_w$, for a given prefix w, can take exponential time in $|w|$. However, the exponential time behaviour of our algorithm seems to be unavoidable, since the problem at hand is more general than the problem of parsing of w under a LCFRS model, which is known to be NP-hard (Satta, 1992). As already discussed in Section 3, the problem can be solved in polynomial time in case we can cast the source LCFRS G in a normal form where each rule has length bounded by some constant.

7 Discussion

Our findings subsume computation of the prefix probability:

- for probabilistic context-free grammars in time $\mathcal{O}(n^3)$ (Jelinek and Lafferty, 1991),
- for probabilistic tree-adjoining grammars in time $\mathcal{O}(n^6)$ (Nederhof et al., 1998), and
- for probabilistic synchronous context-free grammars (Nederhof and Satta, 2011b).

The latter becomes clear once we see that a synchronous context-free grammar can be expressed in terms of a restricted type of LCFRS. All nonterminals of this LCFRS, except the start symbol, have fan-out 2, and all generated strings are of the form $w\$v$, where w functions as input string, v functions are output string, and \$ is a separator between the two, which does not occur in the input and output alphabets of the synchronous context-free grammar. The rules of the LCFRS are of two forms. The first form is a single rule with the start symbol $S^\dagger$ in the left-hand side:

$$\pi^\dagger : S^\dagger \rightarrow g^\dagger(S), \text{ where}$$
$$g^\dagger(\langle x_{1,1}, x_{1,2}\rangle) = \langle x_{1,1}\$x_{1,2}\rangle$$

The other rules all have a form that ensures that variables associated with the input string are never combined with variables associated with the output string:

$$\pi : A \rightarrow g(A_1, \ldots, A_r), \text{ where}$$
$$g(\langle x_{1,1}, x_{1,2}\rangle, \ldots \langle x_{r,1}, x_{r,2}\rangle) =$$
$$\langle y_{1,1} \cdots y_{1,m_1}, y_{2,1} \cdots y_{2,m_2}\rangle$$

and each $y_{1,l}$, $l \in [m_1]$, is either a terminal (from the input alphabet) or a variable of the form $x_{1,j}$, and each $y_{2,l}$, $l \in [m_2]$, is either a terminal (from the output alphabet) or a variable of the form $x_{2,j}$.

Let $\mathcal{G}$ be a PLCFRS mimicking a probabilistic synchronous CFG as outlined above. We can define an unambiguous FST M of which the transduction is the set of pairs $\langle w_1v_1\$w_2v_2, w_1\$w_2\rangle$ for any strings w_1, v_1 over the input alphabet and any strings w_2, v_2 over the output alphabet of the synchronous CFG, where w_1 and w_2 are non-empty. The FST has states s_0, s_1, s_2, s_3, of which s_0 and s_3 are the start and the final state, respectively. The transitions are of the form:

$$\begin{array}{ll} s_0 \overset{a,a}{\mapsto} s_0 & s_0 \overset{a,a}{\mapsto} s_1 \\ s_1 \overset{a,\varepsilon}{\mapsto} s_1 & s_1 \overset{\$,\$}{\mapsto} s_2 \\ s_2 \overset{a,a}{\mapsto} s_2 & s_2 \overset{a,a}{\mapsto} s_3 \\ s_3 \overset{a,\varepsilon}{\mapsto} s_3 & \end{array}$$

We can now proceed by constructing $\mathcal{G} \circ M$, and eliminating its epsilon and unit rules, to give $\mathcal{G}'$, much as in Section 6. Let w_1 and w_2 be two strings over the input and output alphabets, respectively. The prefix probability can now be effectively computed by constructing $\mathcal{G}' \circ M_{w_1\$w_2}$ and computing the inside algorithm.

We can use a similar idea to compute prefix probabilities for probabilistic extensions of the generalized multitext grammars of Melamed et al. (2004), a formalism used to model translation for which no prefix probability algorithm was previously known.

A seemingly small variation of the problem considered in this paper is to compute **infix probabilities** (Corazza et al., 1991; Nederhof and Satta, 2008). It is straightforward to define a FST M of which the transduction is the set of pairs $\langle v_1wv_2, w\rangle$. However, it does not seem possible to construct an *unambiguous* FST that achieves this. Therefore $\mathcal{G} \circ M$ does not have the required probabilistic properties that would allow a correct computation by means of the inside algorithm. The problem of infix probabilities for probabilistic context-free grammars was also considered by Nederhof and Satta (2011a).

References

S. Abney, D. McAllester, and F. Pereira. 1999. Relating probabilistic grammars and automata.

In *37th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 542–549, Maryland, USA, June.

Y. Bar-Hillel, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Y. Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*, chapter 9, pages 116–150. Addison-Wesley, Reading, Massachusetts.

J. Berstel. 1979. *Transductions and Context-Free Languages*. B.G. Teubner, Stuttgart.

A. Corazza, R. De Mori, R. Gretter, and G. Satta. 1991. Computation of probabilities for an island-driven parser. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):936–950.

J. Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, February.

K. Etessami and M. Yannakakis. 2009. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1–66.

C. Gómez-Rodríguez and G. Satta. 2009. An optimal-time binarization algorithm for linear context-free rewriting systems with fan-out two. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 985–993, Suntec, Singapore, August.

M.A. Harrison. 1978. *Introduction to Formal Language Theory*. Addison-Wesley.

Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35:559–595, December.

F. Jelinek and J.D. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323.

L. Kallmeyer and W. Maier. 2010. Data-driven parsing with probabilistic linear context-free rewriting systems. In *The 23rd International Conference on Computational Linguistics*, pages 537–545, Beijing, China, August.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer-Verlag.

Y. Kato, H. Seki, and T. Kasami. 2006. RNA pseudoknotted structure prediction using stochastic multiple context-free grammar. *IPSJ Digital Courier*, 2:655–664.

S. Kiefer, M. Luttenberger, and J. Esparza. 2007. On the convergence of Newton's method for monotone systems of polynomial equations. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 217–266.

M. Kracht. 2003. *The Mathematics of Language*. Mouton de Gruyter, Berlin.

M. Kuhlmann and G. Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 478–486, Athens, Greece.

B. Lang. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10(4):486–494.

W. Maier and T. Lichte. 2009. Characterizing discontinuity in constituent treebanks. In P. de Groote, M. Egg, and L. Kallmeyer, editors, *Proceedings of the 14th Conference on Formal Grammar*, volume 5591 of *Lecture Notes in Artificial Intelligence*, Bordeaux, France.

W. Maier and A. Søgaard. 2008. Treebanks and mild context-sensitivity. In P. de Groote, editor, *Proceedings of the 13th Conference on Formal Grammar*, pages 61–76, Hamburg, Germany. CSLI Publications.

C.D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

I. Dan Melamed, Giorgio Satta, and Ben Wellington. 2004. Generalized multitext grammars. In *Proceedings of ACL-04*.

J. Michaelis. 2001. *On Formal Properties of Minimalist Grammars*. Ph.D. thesis, Potsdam University.

M.-J. Nederhof and G. Satta. 2008. Computing partition functions of PCFGs. *Research on Language and Computation*, 6(2):139–162.

M.-J. Nederhof and G. Satta. 2011a. Computation of infix probabilities for probabilistic context-free grammars. In *Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1213–1221, Edinburgh, Scotland, July.

M.-J. Nederhof and G. Satta. 2011b. Prefix probability for probabilistic synchronous context-free grammars. In *49th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 460–469, Portland, Oregon, June.

M.-J. Nederhof, A. Sarkar, and G. Satta. 1998. Prefix probabilities from stochastic tree adjoining grammars. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, volume 2, pages 953–959, Montreal, Quebec, Canada, August.

O. Rambow and G. Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223:87–120.

B. Sagot and G. Satta. 2010. Optimal rank reduction for linear context-free rewriting systems with fan-out two. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 525–533, Uppsala, Sweden, July.

G. Satta. 1992. Recognition of linear context-free rewriting systems. In *30th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 89–95, Newark, Delaware, USA, June–July.

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

S. Sippu and E. Soisalon-Soininen. 1988. *Parsing Theory, Vol. I: Languages and Parsing*, volume 15 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):167–201.

K. Vijay-Shanker, D.J. Weir, and A.K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 104–111, Stanford, California, USA, July.

D. Wojtczak and K. Etessami. 2007. PReMo: an analyzer for Probabilistic Recursive Models. In *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference*, volume 4424 of *Lecture Notes in Computer Science*, pages 66–71, Braga, Portugal. Springer-Verlag.

D.H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208.