Probabilistic Parsing Strategies

Mark-Jan Nederhof*
Faculty of Arts
University of Groningen
P.O. Box 716
NL-9700 AS Groningen, The Netherlands
markjan@let.rug.nl

Giorgio Satta
Dip. di Elettronica e Informatica
Università di Padova
via Gradenigo, 6/A
I-35131 Padova, Italy
satta@dei.unipd.it

Abstract

We present new results on the relation between purely symbolic context-free parsing strategies and their probabilistic counter-parts. Such parsing strategies are seen as constructions of push-down devices from grammars. We show that preservation of probability distribution is possible under two conditions, viz. the correct-prefix property and the property of strong predictiveness. We improve existing results on the size of probabilistic parsers, and present a negative result on LR parsing.

1 Introduction

Context-free grammars and push-down automata are two equivalent formalisms to describe context-free languages. While a context-free grammar can be thought of as a purely declarative specification, a push-down automaton is considered to be an operational specification that determines which steps are performed for a given string in the process of deciding its membership of the language. By a *parsing strategy* we mean a particular mapping from context-free grammars to equivalent push-down automata. Different parsing strategies may have different space and time complexities, usually expressed in terms of the size of the resulting push-down automata, and in terms of the time complexity of recognition in the case of tabular simulation of nondeterminism, following the work by [10] and [2].

*Supported by the Royal Netherlands Academy of Arts and Sciences. Secondary affiliation is the German Research Center for Artificial Intelligence (DFKI).

This paper deals with the probabilistic extensions of context-free grammars and push-down automata, i.e. probabilistic context-free grammars, discussed by [3], and probabilistic push-down automata, discussed by [18] and [1]. These formalisms are obtained by adding probabilities to the rules and transitions of context-free grammars and push-down automata, respectively. More specifically, we will investigate the problem of "extending" parsing strategies to *probabilistic* parsing strategies. These are mappings from probabilistic context-free grammars to probabilistic push-down automata that preserve the induced probability distributions on the generated/accepted languages. We will show that

- no parsing strategy that lacks the correct-prefix property (CPP) can be extended to become a probabilistic parsing strategy;
- all parsing strategies that possess the correct-prefix property and the strong predictiveness property (SPP) can be extended to become probabilistic parsing strategies.

The above results generalize previous findings reported in [1], where only a few specific parsing strategies were considered in isolation. Our findings have important implications for well-known parsing strategies such as generalized LR parsing, henceforth simply called 'LR parsing'.[1] LR parsing has the CPP, but lacks the SPP, and as we will show, LR parsing cannot be extended to become a probabilistic parsing strategy.

Our findings also require revision of a conjecture by [1] concerning the succinctness of so-called shift-reduce parsing strategies. We show that, contrary to this conjecture, probabilistic push-down automata need not be any larger than their non-probabilistic counter-parts, provided the automata are obtained by strategies possessing both CPP and SPP.

In this paper we will consider push-down transducers instead of push-down automata. This means that the devices not only compute derivations of the grammar while processing an input string, but they also explicitly produce output strings from which these derivations can be obtained. We use transducers for two reasons. First, constraints on the output strings allow us to restrict our attention to "reasonable" parsing strategies. Those strategies that cannot be formalized within these constraints are unlikely to be of practical interest. Secondly, mappings from input strings to derivations, as those realized by push-down devices, turn out to be a very powerful abstraction and allow direct proofs of several general results.

Our work is motivated by the widespread interest in probabilistic parsing techniques that has arisen in the last decade in the area of natural language processing; see e.g. [5].

The paper is organized as follows. After giving standard definitions in Section 2, we give our formal definition of 'parsing strategy' in Section 3. The CPP

[1] Generalized (or nondeterministic) LR parsing allows for more than one action for a given LR state and input symbol.

and the SPP are discussed in Sections 4 and 5. Sections 6 and 7 provide examples of parsing strategies with and without the SPP. We end this paper with conclusions.

2 Preliminaries

In this section we briefly recall some standard notions from formal language theory. For more details we refer the reader to standard textbooks, as e.g. [8].

A context-free grammar (CFG) $\mathcal{G}$ is a 4-tuple (Σ, N, S, R), where Σ is a finite set of *terminals*, called the *alphabet*, N is a finite set of *nonterminals*, including the *start symbol* S, and R is a finite set of *rules*, each of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$. Without loss of generality, we assume that there is only one rule $S \to \sigma$ with the start symbol in the left-hand side, and that furthermore $\sigma \neq \epsilon$, where ϵ denotes the empty string.

For a fixed CFG $\mathcal{G}$, we define the relation $\Rightarrow$ on triples consisting of two strings $\alpha, \beta \in (\Sigma \cup N)^*$ and a rule $\pi \in R$ by: $\alpha \overset{\pi}{\Rightarrow} \beta$ if and only if α is of the form $wA\delta$ and β is of the form $w\gamma\delta$, for some $w \in \Sigma^*$ and $\delta \in (\Sigma \cup N)^*$, and $\pi = (A \to \gamma) \in R$. A *partial left-most derivation* is a string $d = \pi_1 \cdots \pi_m$, $m \geq 0$, such that $S \overset{\pi_1}{\Rightarrow} \cdots \overset{\pi_m}{\Rightarrow} \alpha$, for some $\alpha \in (\Sigma \cup N)^*$. A *left-most derivation* of a string $w \in \Sigma^*$ is a partial left-most derivation $d = \pi_1 \cdots \pi_m$ such that $S \overset{\pi_1}{\Rightarrow} \cdots \overset{\pi_m}{\Rightarrow} w$. We will identify a (partial) left-most derivation with the sequence of strings over $\Sigma \cup N$ that arise in that derivation.

For a CFG $\mathcal{G}$ we define the language $L(\mathcal{G})$ it generates as the set of strings of which there is a left-most derivation. We say a CFG is *reduced* if for each rule $\pi \in R$ there is a left-most derivation in which it occurs.

A *probabilistic* context-free grammar (PCFG) is a pair $(\mathcal{G}, p)$ consisting of a CFG $\mathcal{G} = (\Sigma, N, S, R)$ and a probability function p from R to real numbers in the interval $[0, 1]$. We say a PCFG is *proper* if $\Sigma_{\pi=(A\to\gamma)\in R}\ p(\pi) = 1$ for each $A \in N$.

For a PCFG $(\mathcal{G}, p)$, we define the probability $p(d)$ of a (partial) left-most derivation $d = \pi_1 \cdots \pi_m$ as $\Pi_{i=1}^{m}\ p(\pi_i)$. The probability $p(w)$ of a string w as defined by $(\mathcal{G}, p)$ is the sum of the probabilities of all left-most derivations of that string. We say a PCFG $(\mathcal{G}, p)$ is *consistent* if $\Sigma_{w\in\Sigma^*}\ p(w) = 1$.

A push-down transducer (PDT) $\mathcal{A}$ is a 6-tuple $(\Sigma_1, \Sigma_2, Q, X_{init}, X_{final}, \Delta)$, where Σ_1 is the input alphabet, Σ_2 is the output alphabet, Q is a finite set of *stack symbols* including the *initial stack symbol* X_{init} and the *final stack symbol* X_{final}, and Δ is the set of *transitions*. Each transition can have one of the following three forms: $X \mapsto XY$ (a push transition), $YX \mapsto Z$ (a pop transition), or $X \overset{x,y}{\mapsto} Y$ (a swap transition); here $X, Y, Z \in Q$, $x \in \Sigma_1 \cup \{\epsilon\}$ and $y \in \Sigma_2 \cup \{\epsilon\}$. Note that in our notation, stacks grow from left to right, i.e. the top-most stack symbol will be found at the right end.

Without loss of generality, we assume that any PDT is such that for a given stack symbol $X \neq X_{final}$, there are either one or more push transitions $X \mapsto XY$, or one or more pop transitions $YX \mapsto Z$, or one or more swap transitions $X \overset{x,y}{\mapsto} Y$,

but no combinations of different types of transition. If a PDT does not satisfy this normal form, it can easily be brought in this form by introducing for each stack symbol X three new stack symbols X_{push}, X_{pop} and X_{swap} and new swap transitions $X \overset{\epsilon,\epsilon}{\mapsto} X_{push}$, $X \overset{\epsilon,\epsilon}{\mapsto} X_{pop}$ and $X \overset{\epsilon,\epsilon}{\mapsto} X_{swap}$. In each existing transition that operates on top-of-stack X, we then replace X by one from X_{push}, X_{pop} or X_{swap}, depending on the type of that transition. We also assume that X_{final} does not occur in the left-hand side of a transition, again without loss of generality.

A *configuration* of a PDT is a triple (α, w, v), where $\alpha \in Q^*$ is a stack, $w \in \Sigma_1^*$ is the remaining input, and $v \in \Sigma_2^*$ is the output generated so far. For a fixed PDT $\mathcal{A}$, we define the relation $\vdash$ on triples consisting of two configurations and a transition τ by: $(\gamma\alpha, xw, v) \overset{\tau}{\vdash} (\gamma\beta, w, vy)$ if and only if τ is of the form $\alpha \mapsto \beta$, where $x = y = \epsilon$, or of the form $\alpha \overset{x,y}{\mapsto} \beta$. A *computation* on an input string w is a string $c = \tau_1 \cdots \tau_m$, $m \geq 0$, such that $(X_{init}, w, \epsilon) \overset{\tau_1}{\vdash} \cdots \overset{\tau_m}{\vdash} (\alpha, w', v)$. A *complete* computation on a string w is a computation with $w' = \epsilon$ and $\alpha = X_{final}$. The string v is called the *output* of the computation.

We will identify a computation with the sequence of configurations that arise in that computation. We also write $(\alpha, w, v) \vdash^* (\beta, w', v')$ or $(\alpha, w, v) \overset{c}{\vdash}{}^* (\beta, w', v')$, for $\alpha, \beta \in Q^*$, $w, w' \in \Sigma_1^*$ and $v, v' \in \Sigma_2^*$, to indicate that (β, w', v') can be obtained from (α, w, v) by applying a sequence c of zero or more transitions; we refer to such a sequence c as a *subcomputation*.

For a PDT $\mathcal{A}$, we define the language $L(\mathcal{A})$ it accepts as the set of strings on which there is a complete computation. We say a PDT is *reduced* if each transition $\tau \in \Delta$ occurs in some complete computation.

A *probabilistic* push-down transducer (PPDT) is a pair $(\mathcal{A}, p)$ consisting of a PDT $\mathcal{A}$ and a probability function p from the set Δ of transitions of $\mathcal{A}$ to real numbers in the interval $[0, 1]$. We say a PPDT $(\mathcal{A}, p)$ is *proper* if

- $\Sigma_{\tau=(X \mapsto XY) \in \Delta}\ p(\tau) = 1$ for each $X \in Q$ such that there is at least one transition $X \mapsto XY$, $Y \in Q$;
- $\Sigma_{\tau=(X \overset{x,y}{\mapsto} Y) \in \Delta}\ p(\tau) = 1$ for each $X \in Q$ such that there is at least one transition $X \overset{x,y}{\mapsto} Y$, $x \in \Sigma_1 \cup \{\epsilon\}, y \in \Sigma_2 \cup \{\epsilon\}, Y \in Q$; and
- $\Sigma_{\tau=(YX \mapsto Z) \in \Delta}\ p(\tau) = 1$, for each $X, Y \in Q$ such that there is at least one transition $YX \mapsto Z$, $Z \in Q$.

For a PPDT $(\mathcal{A}, p)$, we define the probability $p(c)$ of a (sub)computation $c = \tau_1 \cdots \tau_m$ as $\Pi_{i=1}^m\ p(\tau_i)$. The probability $p(w)$ of a string w as defined by $(\mathcal{A}, p)$ is the sum of the probabilities of all complete computations on that string. We say a PPDT $(\mathcal{A}, p)$ is *consistent* if $\Sigma_{w \in \Sigma^*}\ p(w) = 1$.

3 Parsing strategies

We define a *parsing strategy* to be a function $\mathcal{S}$ that maps a reduced CFG $\mathcal{G} = (\Sigma_1, N, S, R)$ to a pair $\mathcal{S}(\mathcal{G}) = (\mathcal{A}, f)$ consisting of a reduced PDT $\mathcal{A} = (\Sigma_1, \Sigma_2, Q,$

X_{init}, X_{final}, Δ), and a function f that maps a subset of Σ_2^* to a subset of R^*, with the following properties:

- $R \subseteq \Sigma_2$.
- For each string $w \in \Sigma_1^*$ and each complete computation c on w with output $v \in \Sigma_2^*$, $f(v) = d$ is a left-most derivation of w. Furthermore, each symbol from R occurs as often in v as it occurs in d.
- Conversely, for each string $w \in \Sigma_1^*$ and each left-most derivation d of w, there is precisely one complete computation on w with output v such that $f(v) = d$.

If c is a complete computation with output v, we will write $f(c)$ to denote $f(v)$. The conditions above then imply that f is a bijection from complete computations to left-most derivations.

Note that output strings of (complete) computations may contain symbols that are not in R, and the symbols that are in R may occur in a different order in v than in $f(v) = d$. The purpose of the symbols in $\Sigma_2 - R$ is to help this process of reordering of symbols in R.

For a string $v \in \Sigma_2^*$ we let $\overline{v}$ refer to the maximal subsequence of symbols from v that belong to R, or in other words, symbols from $\Sigma_2 - R$ are removed.

A *probabilistic parsing strategy* is defined to be a function $\mathcal{S}$ that maps a reduced, proper and consistent PCFG $(\mathcal{G}, p_\mathcal{G})$ to a triple $\mathcal{S}(\mathcal{G}, p_\mathcal{G}) = (\mathcal{A}, p_\mathcal{A}, f)$, where $(\mathcal{A}, p_\mathcal{A})$ is a reduced, proper and consistent PPDT, with the same properties as a (non-probabilistic) parsing strategy, and in addition:

- For each left-most derivation d and each complete computation c such that $f(c) = d$, $p_\mathcal{G}(d)$ equals $p_\mathcal{A}(c)$.

In other words, a complete computation has the same probability as the left-most derivation that it is mapped to by function f. An implication of this property is that for each string $w \in \Sigma_1^*$, the probabilities assigned to that string by $(\mathcal{G}, p_\mathcal{G})$ and $(\mathcal{A}, p_\mathcal{A})$, respectively, are equal.

We say that probabilistic parsing strategy $\mathcal{S}'$ is an *extension* of parsing strategy $\mathcal{S}$ if for each reduced CFG $\mathcal{G}$ and probability function $p_\mathcal{G}$ we have $\mathcal{S}(\mathcal{G}) = (\mathcal{A}, f)$ if and only if $\mathcal{S}'(\mathcal{G}, p_\mathcal{G}) = (\mathcal{A}, p_\mathcal{A}, f)$ for some $p_\mathcal{A}$.

In the following sections we will investigate which parsing strategies can be extended to become probabilistic parsing strategies.

4 Correct-prefix property

For a given PDT, we say a computation c is *dead* if $(X_{init}, w_1, \epsilon) \overset{c}{\vdash}^* (\alpha, \epsilon, v_1)$, for some $\alpha \in Q^*$, $w_1 \in \Sigma_1^*$ and $v_1 \in \Sigma_2^*$, and there are no $w_2 \in \Sigma_1^*$ and $v_2 \in \Sigma_2^*$ such that $(\alpha, w_2, \epsilon) \vdash^* (X_{final}, \epsilon, v_2)$. Informally, a dead computation is a computation that cannot be continued to become a complete computation.

We say that a PDT has the *correct-prefix property* (CPP) if it does not allow any dead computations. We say that a parsing strategy has the CPP if it maps each reduced CFG to a PDT that has the CPP.

In this section we show that the correct-prefix property is a necessary condition for extending a parsing strategy to a probabilistic parsing strategy. For this we need two lemmas.

Lemma 1 *For each reduced CFG $\mathcal{G}$, there is a probability function $p_\mathcal{G}$ such that PCFG $(\mathcal{G}, p_\mathcal{G})$ is proper and consistent, and $p_\mathcal{G}(d) > 0$ for all left-most derivations d.*

Proof. Since $\mathcal{G}$ is reduced, there is a finite set L consisting of left-most derivations d, such that for each rule π in $\mathcal{G}$ there is at least one $d \in L$ in which π occurs. Let $n_{\pi,d}$ be the number of occurrences of rule π in derivation $d \in L$, and let n_π be $\Sigma_{d \in L}\ n_{\pi,d}$, the total number of occurrences of π in L. Let n_A be the sum of n_π for all rules π with A in the left-hand side. A probability function $p_\mathcal{G}$ can be defined through 'maximum-likelihood estimation' to be such that $p_\mathcal{G}(\pi) = \frac{n_\pi}{n_A}$ for each rule $\pi = A \rightarrow \alpha$.

For all nonterminals A, $\Sigma_{\pi=A\rightarrow\alpha}\ p_\mathcal{G}(\pi) = \Sigma_{\pi=A\rightarrow\alpha}\ \frac{n_\pi}{n_A} = \frac{n_A}{n_A} = 1$, which means that the PCFG $(\mathcal{G}, p_\mathcal{G})$ is proper. Furthermore, [6] has shown that a PCFG $(\mathcal{G}, p_\mathcal{G})$ that is such that $p_\mathcal{G}$ was obtained by maximum-likelihood estimation is consistent. Finally, since $n_\pi > 0$ for each π, also $p_\mathcal{G}(\pi) > 0$ for each π, and $p_\mathcal{G}(d) > 0$ for all left-most derivations d. ∎

We say a computation is a *shortest* dead computation if it is dead and none of its proper prefixes is dead. Note that each dead computation has a prefix that is a shortest dead computation. For a PDT $\mathcal{A}$, let $\mathcal{T}_\mathcal{A}$ be the union of the set of all complete computations and the set of all shortest dead computations.

Lemma 2 *For each proper PPDT $(\mathcal{A}, p_\mathcal{A})$, $\Sigma_{c \in \mathcal{T}_\mathcal{A}}\ p_\mathcal{A}(c) \leq 1$.*

Proof. The proof is a trivial variant of the proof that for a proper PCFG $(\mathcal{G}, p_\mathcal{G})$, the sum of $p_\mathcal{G}(d)$ for all derivations d cannot exceed 1, which is shown by [3]. ∎

From this, the main result of this section follows.

Theorem 3 *No parsing strategy that does not have the CPP can be extended to become a probabilistic parsing strategy.*

Proof. Take a parsing strategy $\mathcal{S}$ that does not have the CPP. Then there is a reduced CFG $\mathcal{G} = (\Sigma_1, N, S, R)$, with $\mathcal{S}(\mathcal{G}) = (\mathcal{A}, f)$ for some $\mathcal{A}$ and f, and a shortest dead computation c allowed by $\mathcal{A}$.

It follows from Lemma 1 that there is a probability function $p_\mathcal{G}$ such that $(\mathcal{G}, p_\mathcal{G})$ is a proper and consistent PCFG and $p_\mathcal{G}(d) > 0$ for all left-most derivations d. Assume we also have a probability function $p_\mathcal{A}$ such that $(\mathcal{A}, p_\mathcal{A})$ is a proper and consistent PPDT that assigns the same probabilities to strings over Σ_1 as $(\mathcal{G}, p_\mathcal{G})$. Since $\mathcal{A}$ is reduced, each transition τ must occur in some complete

computation c'. Furthermore, for each complete computation c' there is a left-most derivation d such that $f(c') = d$, and $p_{\mathcal{A}}(c') = p_{\mathcal{G}}(d) > 0$. Therefore, $p_{\mathcal{A}}(\tau) > 0$ for each transition τ, and $p_{\mathcal{A}}(c) > 0$, where c is the above-mentioned dead computation.

Due to Lemma 2, $1 \geq \Sigma_{c' \in \mathcal{T}_{\mathcal{A}}} p_{\mathcal{A}}(c') \geq \Sigma_{w \in \Sigma_1^*} p_{\mathcal{A}}(w) + p_{\mathcal{A}}(c) > \Sigma_{w \in \Sigma_1^*} p_{\mathcal{A}}(w) = \Sigma_{w \in \Sigma_1^*} p_{\mathcal{G}}(w)$. This is in contradiction with the consistency of $(\mathcal{G}, p_{\mathcal{G}})$. Hence, a probability function $p_{\mathcal{G}}$ with the properties we required above cannot exist, and therefore $\mathcal{S}$ cannot be extended to become a probabilistic parsing strategy. ■

5 Strong predictiveness

We say that a PDT has the *strong predictiveness property* (SPP) if the existence of two transitions $XY \mapsto Z$ and $XY' \mapsto Z'$ implies $Z = Z'$. Informally this means that no information can flow from higher stack elements to lower stack elements; the stack symbol $Z = Z'$ resulting from some pop transition may reflect which stack symbol X it replaced, but the elements that have been located on top of X are forgotten.[2]

We say that a parsing strategy has the SPP if it maps each reduced CFG to a PDT with the SPP.

In the previous section it was shown that we may restrict ourselves to parsing strategies that have the CPP. Here we show that if, in addition, a parsing strategy has the SPP, then it can always be extended to become a probabilistic parsing strategy.

Theorem 4 *Any parsing strategy that has the CPP and the SPP can be extended to become a probabilistic parsing strategy.*

Proof. Take a parsing strategy $\mathcal{S}$ that has the CPP and the SPP, and take a reduced PCFG $(\mathcal{G}, p_{\mathcal{G}})$, where $\mathcal{G} = (\Sigma_1, N, S, R)$, and let $\mathcal{S}(\mathcal{G}) = (\mathcal{A}, f)$, for some PDT $\mathcal{A}$ and function f. We will show that there is a probability function $p_{\mathcal{A}}$ such that PPDT $(\mathcal{A}, p_{\mathcal{A}})$ assigns the same probabilities to strings over Σ_1 as $(\mathcal{G}, p_{\mathcal{G}})$.

For each stack symbol X, consider the set of transitions that are applicable with top-of-stack X. Remember that our normal form ensures that all such transitions are of the same type. Suppose this set consists of m swap transitions $\tau_i = X \overset{x_i, y_i}{\mapsto} Y_i$, $1 \leq i \leq m$. For each i, consider all subcomputations of the form $(X, x_i w, \epsilon) \overset{\tau_i}{\vdash} (Y_i, w, y_i) \vdash^* (Y', \epsilon, v)$ such that there is at least one pop transition of the form $ZY' \mapsto Z'$ or $Y' = X_{final}$, and define L_{τ_i} as the set of strings v output by these subcomputations. We also define $L_X = \cup_{j=1}^{m} L_{\tau_j}$, the set of all strings output by subcomputations starting with top-of-stack X, and ending just

[2] There is a property of push-down devices called *faiblement prédictif* (weakly predictive) [20]. Contrary to what this name may suggest however, this property is incomparable with the complement of our notion of SPP.

before a pop transition that decreases the height of the stack below that at the beginning, or ending with the final stack symbol X_{final}.

Now define for each i ($1 \leq i \leq m$):

$$p_{\mathcal{A}}(\tau_i) = \frac{\Sigma_{v \in L_{\tau_i}} p_{\mathcal{G}}(\overline{v})}{\Sigma_{v \in L_X} p_{\mathcal{G}}(\overline{v})} \quad (1)$$

In other words, the probability of a transition is the normalized probability of the set of subcomputations starting with that transition, seen as parts of derivations of the PCFG.

These definitions are well-defined. Since $\mathcal{A}$ is reduced and has the CPP, the sets L_{τ_i} are non-empty and thereby the denominator in the definition of $p_{\mathcal{A}}(\tau_i)$ is non-zero. Furthermore, $\Sigma_{i=1}^{m} p_{\mathcal{A}}(\tau_i)$ is clearly 1.

Now suppose the set of transitions for X consists of m push transitions $\tau_i = X \mapsto XY_i$, $1 \leq i \leq m$. For each i, consider all subcomputations of the form $(X, w, \epsilon) \overset{\tau_i}{\vdash} (XY_i, w, \epsilon) \vdash^* (X', \epsilon, v)$ such that there is at least one pop transition of the form $ZX' \mapsto Z'$ or $X' = X_{final}$, and define L_{τ_i}, L_X and $p_{\mathcal{A}}(\tau_i)$ as we have done above for the swap transitions.

Finally, suppose the set of transitions for X consists of m pop transitions $\tau_i = Y_i X \mapsto Z_i$, $1 \leq i \leq m$. Define $L_X = \{\epsilon\}$, and $p_{\mathcal{A}}(\tau_i) = 1$ for each i. Note that if $Y_i = Y_j$ ($1 \leq i, j \leq m$) then $Z_i = Z_j$ and therefore $i = j$, due to the SPP. We also define $L_{X_{final}} = \{\epsilon\}$.

Take a subcomputation $(X, w, \epsilon) \overset{c}{\vdash^*} (Y, \epsilon, v)$ such that there is at least one pop transition of the form $ZY \mapsto Y'$ or $Y = X_{final}$. Below we will prove by induction on the length that:

$$p_{\mathcal{A}}(c) = \frac{p_{\mathcal{G}}(\overline{v})}{\Sigma_{v \in L_X} p_{\mathcal{G}}(\overline{v})} \quad (2)$$

Since a complete computation c with output v is of this form, with $X = X_{init}$ and $Y = X_{final}$, we obtain the following, where D denotes the set of all left-most derivations of CFG $\mathcal{G}$:

$$\begin{aligned} p_{\mathcal{A}}(c) &= \frac{p_{\mathcal{G}}(\overline{v})}{\Sigma_{v \in L_{X_{init}}} p_{\mathcal{G}}(\overline{v})} && (3) \\ &= \frac{p_{\mathcal{G}}(f(c))}{\Sigma_{v \in L_{X_{init}}} p_{\mathcal{G}}(f(v))} && (4) \\ &= \frac{p_{\mathcal{G}}(f(c))}{\Sigma_{d \in D} p_{\mathcal{G}}(d)} && (5) \\ &= p_{\mathcal{G}}(f(c)) && (6) \end{aligned}$$

We have used two properties of f here. The first is that it preserves the frequencies of symbols from R, if considered as a mapping from output strings to derivations. The second property is that it can be considered as bijection from complete

computations to derivations. Lastly we have used consistency of PCFG $(\mathcal{G}, p_\mathcal{G})$, meaning that $\Sigma_{d \in D}\ p_\mathcal{G}(d) = 1$. It follows that $p_\mathcal{A}(w) = p_\mathcal{G}(w)$ for all $w \in \Sigma_1^*$.

For the proof by induction, we distinguish between three cases.

Case 1: Consider a subcomputation c with output $v = \epsilon$ consisting of zero transitions, with only configuration (X, ϵ, ϵ), where there is at least one pop transition of the form $ZX \mapsto Z'$ or $X = X_{final}$. We trivially have $p_\mathcal{A}(c) = 1$ and $\frac{p_\mathcal{G}(\overline{v})}{\Sigma_{v \in L_X}\ p_\mathcal{G}(\overline{v})} = \frac{p_\mathcal{G}(\epsilon)}{\Sigma_{v \in \{\epsilon\}}\ p_\mathcal{G}(\overline{v})} = 1$.

Case 2: Consider a subcomputation $c = \tau_i c'$, where $(X, x_i w, \epsilon) \overset{\tau_i}{\vdash} (Y_i, w, y_i) \overset{c'}{\vdash^*} (Y', \epsilon, y_i v)$, such that there is at least one pop transition of the form $ZY' \mapsto Z'$ or $Y' = X_{final}$. The induction hypothesis states that:

$$p_\mathcal{A}(c') = \frac{p_\mathcal{G}(\overline{v})}{\Sigma_{v \in L_{Y_i}}\ p_\mathcal{G}(\overline{v})} \tag{7}$$

If we combine this with the definition of $p_\mathcal{A}$, we obtain:

$$\begin{aligned}
p_\mathcal{A}(c) &= p_\mathcal{A}(\tau_i) \cdot p_\mathcal{A}(c') && (8)\\
&= \frac{\Sigma_{v \in L_{\tau_i}}\ p_\mathcal{G}(\overline{v})}{\Sigma_{v \in L_X}\ p_\mathcal{G}(\overline{v})} \cdot \frac{p_\mathcal{G}(\overline{v})}{\Sigma_{v \in L_{Y_i}}\ p_\mathcal{G}(\overline{v})} && (9)\\
&= \frac{p_\mathcal{G}(y_i) \cdot \Sigma_{v \in L_{Y_i}}\ p_\mathcal{G}(\overline{v})}{\Sigma_{v \in L_X}\ p_\mathcal{G}(\overline{v})} \cdot \frac{p_\mathcal{G}(\overline{v})}{\Sigma_{v \in L_{Y_i}}\ p_\mathcal{G}(\overline{v})} && (10)\\
&= \frac{p_\mathcal{G}(y_i) \cdot p_\mathcal{G}(\overline{v})}{\Sigma_{v \in L_X}\ p_\mathcal{G}(\overline{v})} && (11)\\
&= \frac{p_\mathcal{G}(\overline{y_i v})}{\Sigma_{v \in L_X}\ p_\mathcal{G}(\overline{v})} && (12)
\end{aligned}$$

Case 3: Consider a subcomputation c of the form $(X, w, \epsilon) \overset{\tau_i}{\vdash} (XY_i, w, \epsilon) \vdash^* (X'', \epsilon, v)$ such that there is at least one pop transition of the form $ZX'' \mapsto Z'$ or $X'' = X_{final}$. We may write this as the subcomputation $c = \tau_i c' \tau c''$ consisting of an application of a push transition $\tau_i = X \mapsto XY_i$, a subcomputation $(Y_i, w_1, \epsilon) \overset{c'}{\vdash^*} (Y_i', \epsilon, v_1)$, an application of a pop transition $\tau = XY_i' \mapsto X'$, and a subcomputation $(X', w_2, \epsilon) \overset{c''}{\vdash^*} (X'', \epsilon, v_2)$, where $w = w_1 w_2$ and $v = v_1 v_2$.

We can now use the induction hypothesis twice, resulting in:

$$p_\mathcal{A}(c') = \frac{p_\mathcal{G}(\overline{v_1})}{\Sigma_{v_1 \in L_{Y_i}}\ p_\mathcal{G}(\overline{v_1})} \tag{13}$$

and

$$p_\mathcal{A}(c'') = \frac{p_\mathcal{G}(\overline{v_2})}{\Sigma_{v_2 \in L_{X'}}\ p_\mathcal{G}(\overline{v_2})} \tag{14}$$

If we combine this with the definition of $p_{\mathcal{A}}$, we obtain:

$$p_{\mathcal{A}}(c) = p_{\mathcal{A}}(\tau_i) \cdot p_{\mathcal{A}}(c') \cdot p_{\mathcal{A}}(\tau) \cdot p_{\mathcal{A}}(c'') \quad (15)$$

$$= \frac{\Sigma_{v \in L_{\tau_i}} p_{\mathcal{G}}(\overline{v})}{\Sigma_{v \in L_X} p_{\mathcal{G}}(\overline{v})} \cdot \frac{p_{\mathcal{G}}(\overline{v_1})}{\Sigma_{v_1 \in L_{Y_i}} p_{\mathcal{G}}(\overline{v_1})} \cdot 1 \cdot \frac{p_{\mathcal{G}}(\overline{v_2})}{\Sigma_{v_2 \in L_{X'}} p_{\mathcal{G}}(\overline{v_2})} \quad (16)$$

Since $\mathcal{A}$ has the SPP, X' is unique to X and the output strings in L_{τ_i} are precisely those that can be obtained by concatenating an output string in L_{Y_i} and an output string in $L_{X'}$. Therefore $\Sigma_{v \in L_{\tau_i}} p_{\mathcal{G}}(\overline{v}) = \Sigma_{v_1 \in L_{Y_i}} \Sigma_{v_2 \in L_{X'}} p_{\mathcal{G}}(\overline{v_1 v_2})$ $= \Sigma_{v_1 \in L_{Y_i}} p_{\mathcal{G}}(\overline{v_1}) \cdot \Sigma_{v_2 \in L_{X'}} p_{\mathcal{G}}(\overline{v_2})$, and

$$p_{\mathcal{A}}(c) = \frac{p_{\mathcal{G}}(\overline{v_1}) \cdot p_{\mathcal{G}}(\overline{v_2})}{\Sigma_{v \in L_X} p_{\mathcal{G}}(\overline{v})} \quad (17)$$

$$= \frac{p_{\mathcal{G}}(\overline{v_1 v_2})}{\Sigma_{v \in L_X} p_{\mathcal{G}}(\overline{v})} \quad (18)$$

$$= \frac{p_{\mathcal{G}}(\overline{v})}{\Sigma_{v \in L_X} p_{\mathcal{G}}(\overline{v})} \quad (19)$$

This concludes the proof. ■

It has been shown by [7] that there is a family of languages that is such that the sizes of the smallest CFGs generating those languages are at least quadratically larger than the sizes of the smallest equivalent push-down automata. (A push-down automaton (PDA) is like a PDT, except that no output is generated.) This result depends on the push-down automata *not* having the SPP. In fact, push-down automata with both CPP and SPP can be transformed to equivalent CFGs of linear size.

This construction is as follows. Let $\mathcal{A} = (\Sigma, Q, X_{init}, X_{final}, \Delta)$ be a PDA with both CPP and SPP, where Σ is the input alphabet, and Q, X_{init}, X_{final} and Δ are as in the case of PDTs. Now construct the CFG $\mathcal{G} = (\Sigma, Q, X_{init}, R)$, where R contains the following rules.

- $X \to YZ$ for each transition $X \mapsto XY$, where Z is the unique stack symbol such that there is at least one transition $XY' \mapsto Z$;
- $X \to xY$ for each transition $X \overset{x}{\mapsto} Y$;
- $Y \to \epsilon$ for each stack symbol Y such that there is at least one transition $XY \mapsto Z$ or such that $Y = X_{final}$.

It is easy to see that there exists a bijection from complete computations of $\mathcal{A}$ to left-most derivations of $\mathcal{G}$. Apart from an additional derivation step by rule $X_{final} \to \epsilon$, the left-most derivations also have the same length as the corresponding complete computations.

This construction can straightforwardly be extended to probabilistic PDAs (PPDAs). Let $(\mathcal{A}, p_{\mathcal{A}})$ be a PPDA with both CPP and SPP. Then we construct $\mathcal{G}$ as above, and further define $p_{\mathcal{G}}$ such that $p_{\mathcal{G}}(\pi) = p_{\mathcal{A}}(\tau)$ for rules $\pi = X \to YZ$ or

$\pi = X \to xY$ that we construct out of transitions $\tau = X \mapsto XY$ or $\tau = X \overset{x}{\mapsto} Y$, respectively, in the first two items above. We also define $p_{\mathcal{G}}(Y \to \epsilon) = 1$ for rules $Y \to \epsilon$ obtained in the third item above. If $(\mathcal{A}, p_{\mathcal{A}})$ is reduced, proper and consistent then so is $(\mathcal{G}, p_{\mathcal{G}})$.

This shows that parsing strategies with the CPP and the SPP as well as their probabilistic counter-parts can also be described as grammar transformations. This is closely connected to the work on *covers* by [13, 11].

It is always possible to transform a PDA with the CPP but without the SPP to an equivalent PDA with both CPP and SPP, by a construction that increases the size of the description considerably (at least quadratically, in the light of the above construction and [7]). However, the new PDA can lead to a very different time complexity of recognition, or in other words, such transformations in general do not preserve parsing strategies and therefore are of minor interest to the issues discussed in this paper.

6 Example of strategy with SPP

Many well-known parsing strategies with the CPP also have the SPP, such as top-down parsing, left-corner parsing [14] and PLR parsing [16]. For space reasons we can only give detailed treatment of left-corner parsing.

In order to simplify the presentation, we allow a new type of transition, without increasing the power of PDTs, viz. a combined swap/push transition of the form $X \overset{x,y}{\mapsto} ZY$. Such a transition can be seen as short-hand for two transitions, the first of the form $X \overset{x,y}{\mapsto} Z_Y$, where Z_Y is a new symbol not already in Q, and the second of the form $Z_Y \mapsto Z_Y Y$. We also assume the existence of a transition $Z_Y Y' \mapsto X'$ for each transition $ZY' \mapsto X'$ that is actually specified.

For a fixed CFG $\mathcal{G} = (\Sigma, N, S, R)$, we define the binary relation $\angle$ over $\Sigma \cup N$ by: $X \angle A$ if and only if there is an $\alpha \in (\Sigma \cup N)^*$ such that $(A \to X\alpha) \in R$, where $X \in \Sigma \cup N$. We define the binary relation $\angle^*$ to be the reflexive and transitive closure of $\angle$. This implies that $a \angle^* a$ for all $a \in \Sigma$.

We now define $\mathcal{S}_{LC}(\mathcal{G}) = (\mathcal{A}, f)$. Here $\mathcal{A} = (\Sigma, R, Q, [S \to \bullet\, \sigma], [S \to \sigma\, \bullet], \Delta)$, where Q contains stack symbols of the form $[A \to \alpha \bullet \beta]$ where $(A \to \alpha\beta) \in R$ such that $\alpha \neq \epsilon \vee A = S$, and stack symbols of the form $[A \to \alpha \bullet Y\beta; X]$ where $(A \to \alpha Y\beta) \in R$ and $X, Y \in \Sigma \cup N$ such that $\alpha \neq \epsilon \vee A = S$ and $X \angle^* Y$. The latter type of stack symbol indicates that left-corner X of goal Y in the right-hand side of rule $A \to \alpha Y \beta$ has just been recognized. The transitions in Δ are:

- $[A \to \alpha \bullet Y\beta] \overset{a,\epsilon}{\mapsto} [A \to \alpha \bullet Y\beta; a]$ for each rule $A \to \alpha Y\beta$ and $a \in \Sigma$ such that $\alpha \neq \epsilon \vee A = S$ and $a \angle^* Y$;
- $[A \to \alpha \bullet B\beta] \overset{\epsilon,\pi}{\mapsto} [A \to \alpha \bullet B\beta; C]$ for each pair of rules $A \to \alpha B\beta$ and $\pi = C \to \epsilon$ such that $\alpha \neq \epsilon \vee A = S$ and $C \angle^* B$;
- $[A \to \alpha \bullet B\beta; X] \overset{\epsilon,\pi}{\mapsto} [A \to \alpha \bullet B\beta; C]\ [C \to X \bullet \gamma]$ for each pair of rules $A \to \alpha B\beta$ and $\pi = C \to X\gamma$ such that $\alpha \neq \epsilon \vee A = S$ and $C \angle^* B$;

- $[A \to \alpha \bullet B\beta; C]\ [C \to \gamma \bullet] \mapsto [A \to \alpha \bullet B\beta; C]$ for each pair of rules $A \to \alpha B\beta$ and $C \to \gamma$ such that $\alpha \neq \epsilon \vee A = S$, $\gamma \neq \epsilon$ and $C \angle^* B$;
- $[A \to \alpha \bullet Y\beta; Y] \mapsto [A \to \alpha Y \bullet \beta]$ for each rule $A \to \alpha Y\beta$ such that $\alpha \neq \epsilon \vee A = S$.

The function f has to rearrange the output string to obtain a left-most derivation; this issue is discussed in [13].

It can be easily seen that the PDTs that result from the above construction have both CPP and SPP, and therefore left-corner parsing can be extended to become a probabilistic parsing strategy. A direct construction of probabilistic left-corner parsers from PCFGs has been presented by [18].

Since at most two rules occur in each of the items above, the size of a (probabilistic) left-corner parser is $\mathcal{O}(|\mathcal{G}|^2)$. This is in contrast to a construction of 'shift-reduce' PPDAs out of PCFGs from [1], which were of size $\mathcal{O}(|\mathcal{G}|^5)$. Our findings require a revision of the "conjecture that there is no *concise* translation of PCFGs into shift-reduce PPDAs" from [1]. It must be noted however that the 'shift-reduce' model adhered to by [1] is more restrictive than our PDT model, which we feel is more natural.

7 Example of strategy without SPP

In this section we show that the absence of the strong predictiveness property may mean that a parsing strategy with the CPP cannot be extended to become a probabilistic parsing strategy. We illustrate this for LR(0) parsing, formalized as a parsing strategy $\mathcal{S}_{LR}$, which has the CPP but not the SPP. (Related difficulties with probabilistic ELR parsing have been pointed out before by [19].) We assume the reader is familiar with LR parsing; see [15].

We take a PCFG $(\mathcal{G}, p_\mathcal{G})$ defined by:

$$\begin{array}{lll}
\pi_S &= S \to AB, & p_\mathcal{G}(\pi_S) = 1 \\
\pi_{A_1} &= A \to aC, & p_\mathcal{G}(\pi_{A_1}) = \frac{4}{10} \\
\pi_{A_2} &= A \to aD, & p_\mathcal{G}(\pi_{A_2}) = \frac{6}{10} \\
\pi_{B_1} &= B \to bC, & p_\mathcal{G}(\pi_{B_1}) = \frac{6}{10} \\
\pi_{B_2} &= B \to bD, & p_\mathcal{G}(\pi_{B_2}) = \frac{4}{10} \\
\pi_C &= C \to xc, & p_\mathcal{G}(\pi_C) = 1 \\
\pi_D &= D \to xd, & p_\mathcal{G}(\pi_D) = 1
\end{array}$$

Because of space limitations, we cannot present the entire LR automaton $\mathcal{A}$, with $\mathcal{S}_{LR}(\mathcal{G}) = (\mathcal{A}, f)$ for some f, but we merely mention two of its key transitions, which represent shift actions over c and d:

$$\tau_c = \{C \to x \bullet c, D \to x \bullet d\} \overset{c,\epsilon}{\mapsto} \{C \to x \bullet c, D \to x \bullet d\}\ \{C \to xc \bullet\}$$
$$\tau_d = \{C \to x \bullet c, D \to x \bullet d\} \overset{d,\epsilon}{\mapsto} \{C \to x \bullet c, D \to x \bullet d\}\ \{D \to xd \bullet\}$$

(We denote LR states by their sets of kernel items, as usual.)

Take a probability function $p_{\mathcal{A}}$ such that $(\mathcal{A}, p_{\mathcal{A}})$ is a proper PPDT. It can be easily seen that $p_{\mathcal{A}}$ must assign 1 to all transitions except τ_c and τ_d, since that is the only pair of distinct transitions that can be applied for one and the same top-of-stack symbol, viz. $\{C \to x \bullet c, D \to x \bullet d\}$.

However, $\frac{p_{\mathcal{G}}(axcbxd)}{p_{\mathcal{G}}(axdbxc)} = \frac{p_{\mathcal{G}}(\pi_{A_1}) \cdot p_{\mathcal{G}}(\pi_{B_2})}{p_{\mathcal{G}}(\pi_{A_2}) \cdot p_{\mathcal{G}}(\pi_{B_1})} = \frac{\frac{4}{10} \cdot \frac{4}{10}}{\frac{6}{10} \cdot \frac{6}{10}} = \frac{4}{9}$ but $\frac{p_{\mathcal{A}}(axcbxd)}{p_{\mathcal{A}}(axdbxc)} = \frac{p_{\mathcal{A}}(\tau_c) \cdot p_{\mathcal{A}}(\tau_d)}{p_{\mathcal{A}}(\tau_d) \cdot p_{\mathcal{A}}(\tau_c)} = 1 \neq \frac{4}{9}$. This shows that there is no $p_{\mathcal{A}}$ such that $(\mathcal{A}, p_{\mathcal{A}})$ assigns the same probabilities to strings over Σ as $(\mathcal{G}, p_{\mathcal{G}})$. It follows that the LR strategy cannot be extended to become a probabilistic parsing strategy.

Note that for $\mathcal{G}$ above, $p_{\mathcal{G}}(\pi_{A_1})$ and $p_{\mathcal{G}}(\pi_{B_1})$ can be freely chosen, and this choice determines the other values of $p_{\mathcal{G}}$, so we have two free parameters. For $\mathcal{A}$ however, there is only one free parameter in the choice of $p_{\mathcal{A}}$. This is in conflict with an underlying assumption of existing work on probabilistic LR parsing, by e.g. [4] and [9], viz. that LR parsers would allow more fine-grained probability distributions than CFGs. For practical cases however, [17] has shown that LR parsers do allow more accurate probability distributions than the CFGs from which they were constructed, if probability functions are estimated from corpora.

There is a construction proposed by [22, 21, 12] of probabilistic LR parsers with the same probability distribution as given PCFGs. Such an LR parser in general cannot have the same structure as the LR parser that would be constructed in the non-probabilistic case, as we have shown above. Instead, the probabilistic LR parser may contain several copies of one and the same state from the non-probabilistic LR parser. These different copies allow the probability function on transitions to distinguish cases that would normally be treated as identical. A serious problem with this approach is however that the required number of copies of each LR state is potentially infinite.

8 Conclusions

We have formalized the notion of parsing strategy as a mapping from context-free grammars to push-down transducers, and have investigated the extension to probabilities. We have shown that the question of which strategies can be extended to become probabilistic heavily relies on two properties, the correct-prefix property and the strong predictiveness property. The CPP is a necessary condition for extending a strategy to become a probabilistic strategy. The CPP and SPP together form a sufficient condition. Furthermore, if a strategy has both CPP and SPP then the resulting probabilistic automata have the same sizes as their non-probabilistic counter-parts. Finally, we have shown that there is at least one strategy of practical interest with the CPP but without the SPP that cannot be extended to become a probabilistic strategy.

Acknowledgements

We gratefully acknowledge correspondence with David McAllester, Virach Sornlertlamvanich and Eric Villemonte de la Clergerie.

References

[1] S. Abney, D. McAllester, and F. Pereira. 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 1999.

[2] S. Billot and B. Lang. The structure of shared forests in ambiguous parsing. In *27th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 143–151, Vancouver, British Columbia, Canada, June 1989.

[3] T.L. Booth and R.A. Thompson. Applying probabilistic measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450, May 1973.

[4] T. Briscoe and J. Carroll. Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, 19(1):25–59, 1993.

[5] E. Charniak. Immediate-head parsing for language models. In *39th Annual Meeting and 10th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, pages 116–123, Toulouse, France, July 2001.

[6] Z. Chi and S. Geman. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305, 1998.

[7] J. Goldstine, K. Price, and D. Wotschke. A pushdown automaton or a context-free grammar — which is more economical? *Theoretical Computer Science*, 18:33–40, 1982.

[8] M.A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, 1978.

[9] K. Inui et al. Probabilistic GLR parsing. In H. Bunt and A. Nijholt, editors, *Advances in Probabilistic and other Parsing Technologies*, chapter 5, pages 85–104. Kluwer Academic Publishers, 2000.

[10] B. Lang. Deterministic techniques for efficient non-deterministic parsers. In *Automata, Languages and Programming, 2nd Colloquium*, Lecture Notes in Computer Science, volume 14, pages 255–269, Saarbrücken, 1974. Springer-Verlag.

[11] R. Leermakers. How to cover a grammar. In *27th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 135–142, Vancouver, British Columbia, Canada, June 1989.

[12] S.-K. Ng and M. Tomita. Probabilistic LR parsing for general context-free grammars. In *Proc. of the Second International Workshop on Parsing Technologies*, pages 154–163, Cancun, Mexico, February 1991.

[13] A. Nijholt. *Context-Free Grammars: Covers, Normal Forms, and Parsing*, Lecture Notes in Computer Science, volume 93. Springer-Verlag, 1980.

[14] D.J. Rosenkrantz and P.M. Lewis II. Deterministic left corner parsing. In *IEEE Conference Record of the 11th Annual Symposium on Switching and Automata Theory*, pages 139–152, 1970.

[15] S. Sippu and E. Soisalon-Soininen. *Parsing Theory, Vol. II: LR(k) and LL(k) Parsing*, EATCS Monographs on Theoretical Computer Science, volume 20. Springer-Verlag, 1990.

[16] E. Soisalon-Soininen and E. Ukkonen. A method for transforming grammars into LL(k) form. *Acta Informatica*, 12:339–369, 1979.

[17] V. Sornlertlamvanich et al. Empirical support for new probabilistic generalized LR parsing. *Journal of Natural Language Processing*, 6(3):3–22, 1999.

[18] F. Tendeau. Stochastic parse-tree recognition by a pushdown automaton. In *Fourth International Workshop on Parsing Technologies*, pages 234–249, Prague and Karlovy Vary, Czech Republic, September 1995.

[19] F. Tendeau. *Analyse syntaxique et sémantique avec évaluation d'attributs dans un demi-anneau.* PhD thesis, University of Orléans, 1997.

[20] E. Villemonte de la Clergerie. *Automates à Piles et Programmation Dynamique — DyALog: Une application à la Programmation en Logique.* PhD thesis, Université Paris VII, 1993.

[21] J. Wright, A. Wrigley, and R. Sharman. Adaptive probabilistic generalized LR parsing. In *Proc. of the Second International Workshop on Parsing Technologies*, pages 100–109, Cancun, Mexico, February 1991.

[22] J.H. Wright and E.N. Wrigley. GLR parsing with probability. In M. Tomita, editor, *Generalized LR Parsing*, chapter 8, pages 113–128. Kluwer Academic Publishers, 1991.