Optimal Probabilistic Evaluation Functions for Search Controlled by Stochastic Context-Free Grammars

Anna Corazza, Renato De Mori, Roberto Gretter, and Giorgio Satta

Abstract—**Recently, the possibility of using Stochastic Context-Free Grammars (SCFG's) in Language Modeling (LM) has been considered. When these grammars are used, search can be directed by evaluation functions based on the probabilities that a SCFG generates a sentence, given only some words in it. Expressions for computing the evaluation function have been proposed by Jelinek and Lafferty for the recognition of word sequences in the case in which only the prefix of a sequence is known. Corazza *et al.* have proposed methods for probability computation in the more general case in which partial word sequences interleaved by gaps are known. This computation is too complex in practice unless the lengths of the gaps are known. This paper proposes a method for computing the probability of the best parse tree that can generate a sentence only part of which (consisting of islands and gaps) is known. This probability is the minimum possible, and thus the most informative, upper-bound that can be used in the evaluation function. The computation of the proposed upper-bound has cubic time complexity even if the lengths of the gaps are unknown. This makes possible the practical use of SCFG for driving interpretations of sentences in natural language processing.**

Index Terms—**Stochastic language models, stochastic context-free grammars, syntax-controlled signal interpretation, stochastic parsing, probabilistic scores for linguistic interpretation, island driven stochastic parsers.**

I. Introduction

INTERPRETING patterns under the control of Stochastic Context-Free Grammars (SCFG's) returns to attract the attention of researchers, especially because of potential applications to Automatic Speech Recognition (ASR) and Automatic Speech Understanding (ASU) [1]. In fact, the approach proposed in this paper originates from an ASU problem.[1] Nevertheless, its application is more general and the framework can be applied whenever the possible interpretations can be modeled by a SCFG.

Assume a pattern interpretation γ is based on a pattern description A. A most plausible interpretation γ^* for A can be found by the following decision criterion:

$$\gamma^* = argmax_\gamma \Pr(A \mid \gamma) \Pr(\gamma). \tag{1}$$

In the case of ASR and ASU, A is a description of an acoustic pattern. In ASR, γ represents a word sequence, since the goal is speech transcription. In ASU, the syntactic derivations associated with the word sequence need also be considered, since in most cases different derivations correspond to different meanings. Therefore, in ASU γ represents a pair $\langle \delta, \tau \rangle$, where τ is a derivation and δ is the unique word sequence associated with τ.

The term $\Pr(A \mid \gamma)$ in (1) can be obtained by an acoustic processor based on Hidden Markov Models (HMM). In case γ is a sequence-derivation pair, the hypothesis can be made that the acoustic signal depends only on the word sequence itself and not on the derivation. This is true if an acoustic description is chosen that mostly depends on phonemes and their context. The computation of $\Pr(\gamma)$ is based on a probabilistic Language Model (LM), which in this paper is assumed to be a SCFG. If γ is a word sequence, all its derivations in the grammar have to be considered in order to compute this probability. If $\gamma = \langle \delta, \tau \rangle$, $\Pr(\gamma)$ only depends on the derivation τ. Hence, if no other linguistic constraint is imposed, in the latter case the maximization of $\Pr(\gamma)$ implies that only the most probable derivation will be considered.

The search for an optimal interpretation γ^* considers a collection of sets of interpretations of A, with the requirement that the set of all possible interpretations of A is covered by the collection itself. Each set in the collection is called a Partial Interpretation (PI) and is scored by an upper-bound of the following evaluation function computed on all elements of the set:

$$f_A(\gamma) = \Pr(A \mid \gamma) \Pr(\gamma). \tag{2}$$

Manuscript received July 24, 1992; revised February 28, 1994. This work was supported by the following grants: ARO DAAL 03-89-C-0031, DARPA N00014-90-J-1863, NSF IRI 90-16592 , and Ben Franklin 91S.3078C-1. R. De Mori was supported by the Natural Sciences and Research Council of Canada (grant no. OGP 0009124) and the Institute for Robotics and Intelligent Systems, a Canadian Network of Centers of Excellence. This work was partially conducted while G. Satta was a post-doctoral fellow at the Institute for Research in Cognitive Science at the University of Pennsylvania.

A. Corazza and R. Gretter are with the Istituto per la Ricerca Scientifica e Tecnologica, I-38050 Povo/Trento, Italy.

R. De Mori is at the School of Computer Science, McGill University, 3480 University str, Montreal, PQ, H3A2A7, Canada.

G. Satta is with the Università di Venezia, Scienze dell'Informazione, via Torino 155, 30172 Mestre — Venezia, Italy.

IEEE Log Number 9405249.

[1] The research described in this paper is part of an effort having the purpose of providing spoken dialog capabilities to person-robot interaction. These capabilities are developed in the MAIA project at IRST (Trento, Italy) and at the Institute for Robotics and Intelligent Systems, a Canadian Network of Centers of Excellence.

0162-8828/94$04.00 © 1994 IEEE

In ASR and ASU, a PI is represented by one or more substrings interleaved by gaps, which correspond to parts of the input data not yet analyzed and, in ASU, by an associated partial derivation. Each PI σ is associated to a set composed of all possible completions of σ, i.e., of all complete interpretations obtained by filling the gaps of σ with substrings. The way in which each set is partitioned depends on the parsing strategy that drives the search. If the input is analyzed from left to right, all PI's are represented by prefixes of the input followed by a gap which is filled from left to right. Alternatively, search can start from points in the middle of the input sentence and PI's are expanded outward; this strategy is called island-driven (see for instance [3]).

Upper-bounds of function f can be computed as the product of upper-bounds of each term in the right-hand side of (2), called the acoustic and the syntactic term respectively. As far as the computation of upper-bounds of the syntactic term is concerned, a possible approach is the following. Given a PI σ, corresponding to a set of complete interpretations, we can compute the sum of $\Pr(\gamma)$ for all γ in σ, which is an upper-bound satisfying the above requirements. In [2], [4], this approach has been applied to ASR. Unfortunately, it is found that the computational complexity is unacceptable if search is driven by an island-driven strategy. Furthermore, in this approach the score does not result in the tightest upper-bound for a PI.

In this paper, a new scoring method is proposed for a PI σ, based on the computation of the maximum $\Pr(\gamma)$ for all γ in σ, that will be denoted by $\mathrm{Pm}(\sigma)$; this upper-bound can be used for ASU. Therefore, this score is the probability of the most likely derivation among all possible derivations for a word sequence in σ. This score is the minimum possible upper-bound, and thus the most informative one, that can be used for the evaluation function. This paper introduces algorithms for computing such an upper-bound for PI's obtained by a left-to-right or an island-driven parsing strategy. We show that in both cases these upper-bounds can be computed in cubic time with respect to the length of the input, that is with the same asymptotical complexity of well-known general purpose context-free parsing algorithms.

The remaining part of this paper is organized as follows. In Section II the notation and the basic definitions which will be used throughout this work are introduced. The presentation of our algorithms is divided into two parts: the computation that can be performed independently of PI's is described in Section III. The remaining computations are described in Section IV. Section V contains some concluding remarks. Finally, correctness proofs for the given algorithms are reported in the Appendix.

II. Definitions and Notation

In this section, basic definitions and notation are introduced that will be used throughout the paper.

Let Δ be a generic (finite) set of symbols. A *string* = u over Δ is a finite sequence of symbols from Δ; $|u|$ denotes the length of u. The *null string* ε is the (unique) string whose length equals zero. Let $u = a_1 \cdots a_n$, $n \geq 1$; $k{:}u$ and $u{:}k$, $(0 < k < n)$, denote the *prefix* string $a_1 \cdots a_k$ and the *suffix* string $a_{n-k+1} \cdots a_n$ respectively.

The set of all strings over Δ is denoted $\Delta^*(\varepsilon$ included); the set of all strings over Δ whose length is k, $(k \geq 0)$, is denoted Δ^k. Let u and v be two strings in Δ^*; uv denotes the *concatenation* of u and v (u before v). The concatenation is extended to sets of strings in the following way. Let $L_1, L_2 \subseteq \Delta^*$; uL_1 denotes the set $\{x \mid x = uy, y \in L_1\}$, $L_1 u$ denotes the set $\{x \mid x = yu, y \in L_1\}$ and $L_1 L_2$ denotes the set $\{x \mid x = yz, y \in L_1, z \in L_2\}$.

Some definitions are now briefly introduced; a more comprehensive discussion of them can be found in the literature on stochastic context-free languages (see for instance [5], [6]). A SCFG is a 4-tuple $G_s = (\mathrm{N}, \Sigma, P_s, S)$, where N is a finite set of *nonterminal* symbols, Σ is a finite set of *terminal* symbols such that $\mathrm{N} \cap \Sigma = \emptyset$, $S \in \mathrm{N}$ is a special symbol called *start symbol* and P_s is a finite set of pairs $\langle p, \Pr(p)\rangle$, where p is a *production* and $\Pr(p) \neq 0$ is the probability associated with p in G_s. Productions are represented as $H \rightarrow \gamma$, $H \in \mathrm{N}$, $\gamma \in (\mathrm{N} \cup \Sigma)^*$, and symbol P denotes the set of all productions in P_s. As a convention, it is assumed that productions not belonging to P have zero probability. G_s is assumed to be a *proper* SCFG, that is the following relation holds for every nonterminal H:

$$\sum_{\gamma \in (N \cup \Sigma)^*} \Pr(H \rightarrow \gamma) = 1. \tag{3}$$

The grammar G_s is in *Chomsky Normal Form* (CNF) if every production in P has the form $H \rightarrow F_1 F_2$ or $H \rightarrow a$, where $H, F_1, F_2 \in \mathrm{N}$, $a \in \Sigma$. Without any loss of generality, it is assumed in the following that G_s is in CNF. As a convention, it is also assumed that $\mathrm{N} = \{H_1, \cdots, H_{|N|}\}$, $H_1 = S$.

Let γ be a string in $(\mathrm{N} \cup \Sigma)^*$. A derivation in G_s of γ from nonterminal H will be represented in the following by a *derivation tree*, indicating all productions that have been used in the derivation (with repetitions). In such a case, γ is also called the yield of the tree and H is the root. The *height* of a derivation tree τ, written $\mathrm{hg}(\tau)$, is the length of the longest path from the root of τ to a leaf; if the yield of τ contains a single occurrence of a nonterminal symbol and $k > 0$ is the length of the path from the root node in τ to that nonterminal, we say that τ has a *spine* of length k and write $\mathrm{sp}(\tau) = k$. If the yield of a derivation tree does not contain nonterminal symbols, then the derivation tree is *complete*; otherwise it is *incomplete*.

Let τ be a derivation tree in G_s; $\Pr(\tau)$ is the probability of τ, and is obtained as the product of the probabilities of all productions used in the derivation represented by τ (with repetitions). This quantity represents the probability that the string of symbols composing the yield of τ has been derived by grammar G_s in the way prescribed by τ itself.[2] Let γ be a string in $(\mathrm{N} \cup \Sigma)^*$; the set of all derivation trees in G_s with root

[2]To be more precise, the definition of probability of a derivation tree should include both *finite* and *infinite* complete trees. The probability of an incomplete derivation tree τ should be defined as the sum of the probabilities of all finite and infinite complete derivation trees that can be obtained from τ. Under the assumption that the grammar is proper, it is possible to show that this infinite summation converges to the product of the probabilities of all productions involved in τ, which justifies the definition in the text above. Infinite derivation trees are not considered in this paper because the complexity of the mathematics required for dealing with them is beyond the scope of this paper.

H and yield γ will be denoted $H\langle\gamma\rangle$. Accordingly, $\Pr(H\langle\gamma\rangle)$ represents the sum of the probabilities of all derivation trees in such a set. In the following we will use the operator Pm defined as:

$$\mathrm{Pm}(H\langle\gamma\rangle) = \max_{\tau \in H\langle\gamma\rangle} \{\Pr(\tau)\}. \tag{4}$$

We remark that, when G_s is a natural language grammar and γ is a sentence, different derivation trees for γ may correspond to different meanings. In this case $\mathrm{Pm}(S\langle\gamma\rangle)$ is the probability of the most likely interpretation of γ.

Let $L \subseteq (\mathrm{N} \cup \Sigma)^*$: we extend the previous notation and write $H\langle L\rangle$ to represent the set of all derivation trees with root H and yield in L; accordingly, $\mathrm{Pm}(H\langle L\rangle)$ is the maximum among all the probabilities of derivation trees in $H\langle L\rangle$. The *language* generated by G_s, denoted $L(G_s)$, is the set of all strings in Σ^* that can be derived by the productions in G_s from the start symbol S. Finally, we will write $T(G_s)$ to denote the set of all derivation trees obtained by the grammar G_s, that is the set of all derivation trees in $\cup_{H_i \in \mathrm{N}} H_i\langle\Sigma^*\rangle$.

In this paper algorithms are introduced for the computation of quantities $\mathrm{Pm}(S\langle L\rangle)$, taking into consideration cases in which $L = \{u\}$, $L = u\Sigma^*$ and $L = \Sigma^* u \Sigma^*$, where u is a given string defined over Σ. Furthermore, we discuss how these algorithms can be extended to the computation of quantities $\mathrm{Pm}(S\langle L\rangle)$ for L of the form $\Sigma^* u_1 \Sigma^* \cdots \Sigma^* u_d \Sigma^*$, $d > 1$, that is for sequences composed of an arbitrary number of "islands" interleaved by "gaps." As already discussed in the introduction, these quantities can be used as scores in driving the search for the most likely complete interpretation of a pattern description, according to the model at hand. The computation can be divided into two steps: some auxiliary quantities are first computed that do not depend on the input string ("off-line" computation) and then combined with other quantities that do depend on the input ("on-line" computation). As a convenient notation, all quantities to be computed will be recorded in mono- or bi-dimensional arrays, whose indexes are related to nonterminal symbols of the grammar. Arrays with off-line quantities will be denoted by symbol L, arrays with on-line quantities will be denoted by symbol $M(\sigma)$, where the dependence on the given string sequence σ is explicitly represented by the functional notation. Furthermore, symbols L and $M(\sigma)$ are paired with subscripts to recall different classifications of the computation (*g*ap, *p*refix, *s*uffix or *i*sland).

We close this section by introducing some additional notations which will be used later. Let Δ_1 and Δ_2 be two symbol sets; $\Delta_1 - \Delta_2$ denotes the *asymmetric set-difference* between Δ_1 and Δ_2, that is the set $\{x \mid x \in \Delta_1, x \notin \Delta_2\}$.

Let M_1 be an $m \times n$ array and let M_2 be an $n \times p$array; a binary operation $\otimes$ is defined as follows:

$$[M_1 \otimes M_2][i,j] = \max_{1 \le k \le n} \{M_1[i,k] \cdot M_2[k,j]\}. \tag{5}$$

Let M be a $1 \times n$ array and let $\mathcal{I}$ be a subset of $\{1, \cdots, n\}$; we define $\mathrm{argmax}\{M, \mathcal{I}\}$ to be the set $\{i \mid i \in \mathcal{I}$ and $M[i] \ge M[k]$ for every $k \in \mathcal{I}\}$, that is the set of indices of all maximal elements of M, restricted to set $\mathcal{I}$. If M is an $m \times n$ array, we extend the notation to $M[j]$, the jth column of M, in the following way. Let $\mathcal{I} \subseteq \{1, \cdots, m\} \times \{1, \cdots, n\}$; we define $\mathrm{argmax}\{M[j], \mathcal{I}\}$ to be the set $\{(i,j) \mid (i,j) \in \mathcal{I}$ and $M[i,j] \ge M[h,j]$ for every h such that $(h,j) \in \mathcal{I}\}$.

III. Off-Line Computations

In order to compute the probability of the most likely derivation trees that include given fragments of a sentence in the language $L(G_s)$, some quantities are used that only depend on the grammar and not on the fragments themselves. It follows that these quantities can be computed off-line and once for all; this section deals with such a computation. The basic idea is to use a dynamic programming technique as in well-known methods for removing useless symbols from a context-free grammar (e.g., [7]).

A. Computation of Upper-Bounds for a Gap

A first quantity that can be computed off-line is the probability of the best complete derivation tree having root H_i and generating a string in Σ^*. Note that there is one such quantity for every nonterminal in the grammar. These probabilities will be used in the next section to compute the probability contribution of parts of the sentence that have not yet been analyzed; we will therefore call these quantities *gap upper-bounds*.

Gap upper-bounds are defined by means of an $|\mathrm{N}| \times 1$ array L_g in the following way:

$$L_g[i] = \mathrm{Pm}(H_i\langle\Sigma^*\rangle). \tag{6}$$

Before specifying the details of the computation of L_g, an intuitive description of the method is provided. We will argue that the computation can be limited to derivation trees with height not greater than $|\mathrm{N}|$. Let T_k be the set of all complete derivation trees with height not greater than k, $1 \le k \le |\mathrm{N}|$. Lemma 1 in Appendix can be used to show that a set of k' trees $\{\tau_1, \tau_2, \cdots, \tau_{k'}\}$, $k' \ge k$, can *always* be found within T_k, such that the roots of these trees are labeled by distinguishable symbols $H_{i_1}, H_{i_2}, \cdots, H_{i_{k'}}$ and the probabilities of these trees are the desired gap upper-bounds $L_g[i_1], L_g[i_2], \cdots, L_g[i_{k'}]$ for these symbols. A tabular iterative method can then be devised to compute the elements of array L_g, where, at the kth iteration, set T_k is considered. As a consequence, the method gives all elements of L_g in at most $|\mathrm{N}|$ iterations. In fact, at each iteration at least one new element of L_g is obtained. At the kth iteration, the set of indices of elements of L_g that still need to be computed, called the residue elements, is indicated as $\mathcal{I}_k$. Initially, set $\mathcal{I}_0$ contains all possible indices, and at each step, the new residue set can be obtained from the preceding one by removing the indices of the gap upper-bounds found at that iteration. The formal relations introduced in the next definition describe the core of such a tabular method (Fig. 1 shows a schematic representation of these relations).

Definition 1: A family of $|\mathrm{N}| \times 1$ arrays $L_g^{(k)}$, $k \ge 1$, is specified by the following relations:

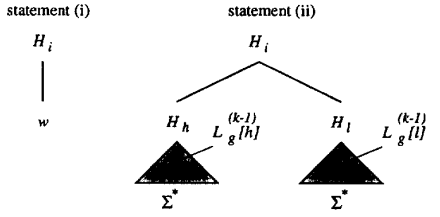


Fig. 1. Schematic representation of the relations used for the definition of arrays $L_g^{(k)}$.

1) let $\mathcal{I}_0 = \{1, \cdots, |\mathrm{N}|\}$:

$$L_g^{(1)}[i] = \max_w \{\Pr(H_i \rightarrow w)\}, i \in \mathcal{I}_0;$$
$$\mathcal{I}_1 = \mathcal{I}_0 - \operatorname{argmax}\{L_g^{(1)}, \mathcal{I}_0\};$$

2) for every $k \geq 2$:

$$L_g^{(k)}[i] = \begin{cases} \max\{L_g^{(k-1)}[i], \max_{h,l}\{\Pr(H_i \rightarrow H_h H_l) \cdot \\ L_g^{(k-1)}[h] L_g^{(k-1)}[l]\}\}, & i \in \mathcal{I}_{k-1}; \\ L_g^{(k-1)}[i], & i \in \mathcal{I}_0 - \mathcal{I}_{k-1}; \end{cases}$$
$$\mathcal{I}_k = \mathcal{I}_{k-1} - \operatorname{argmax}\{L_g^{(k)}, \mathcal{I}_{k-1}\}.$$

Theorem 1 in the Appendix proves that arrays $L_g^{(k)}$ converge to the desired array L_g in no more than $|\mathrm{N}|$ steps. As far as computational complexity is concerned, gap upper-bounds can be computed in time $O(|\mathrm{N}||P|)$. In fact, at most $|\mathrm{N}|$ iterations are needed, each one requiring two multiplications for every production in P.

B. Computation of factors for prefix and suffix upper-bounds

In this subsection, a second family of probabilities is considered whose computation only depends on the productions of the grammar G_s. These probabilities are associated with "optimal" derivation trees whose root is H_i and whose yield is composed of a nonterminal H_j followed by a string in Σ^*. Note that each of these probabilities depends on a pair of nonterminal symbols. These quantities will be used in the next section to compute optimal upper-bounds for the probability that a single derivation produces a given prefix string.

Let us define an $|\mathrm{N}| \times |\mathrm{N}|$ array L_p in the following way:

$$L_p[i, j] = \mathrm{Pm}(H_i \langle H_j \Sigma^* \rangle). \tag{7}$$

Elements of the array L_p can be computed using a method quite similar to the one introduced for the computation of L_g, as described in the following. For any integer k, $1 \leq k \leq |\mathrm{N}|$, let T_k be the set of all derivation trees in $H_i \langle H_j \Sigma^* \rangle$ having spine of length not greater than k, that is the path from the root node H_i to the left-corner node H_j has length not greater than k. At the kth iteration in the computation, elements in T_k having highest probability are considered, for every $H_i \in \mathrm{N}$. Lemma 1 in Appendix can be used to show that, proceeding in this way, at least $|\mathrm{N}|$ new elements of L_p become available at each iteration. Thus, array L_p can be obtained in no more than $|\mathrm{N}|$ iterations. As before, at the kth iteration the elements of L_p that still need to be computed have indices belonging to set $\mathcal{I}_k$.

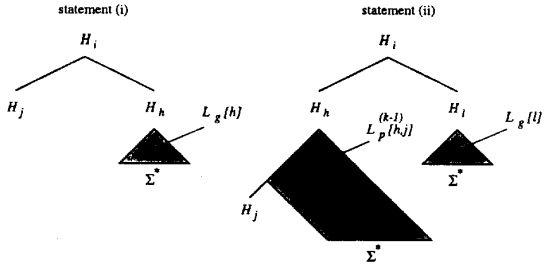


Fig. 2. Schematic representation of the relations used for the definition of arrays $L_p^{(k)}$.

In analogy with the preceding subsection, a family of auxiliary arrays $L_p^{(k)}$, $k \geq 1$, is defined in the following and its use in the computation of array L_p is discussed. Fig. 2 reports a schematic representation of the relations introduced in the definition; we recall the reader that $L_p^{(k)}[j]$ denotes the jth column of array $L_p^{(k)}$.

Definition 2: A family of $|\mathrm{N}| \times |\mathrm{N}|$ arrays $L_p^{(k)}$, $k \geq 1$, is specified by the following relations:

1) let $\mathcal{I}_0 = \{(h, k) \mid 1 \leq h, k \leq |\mathrm{N}|\}$:

$$L_p^{(1)}[i, j] = \max_h \{\Pr(H_i \rightarrow H_j H_h) L_g[h]\}, (i, j) \in \mathcal{I}_0;$$
$$\mathcal{I}_1 = \mathcal{I}_0 - \bigcup_{1 \leq j \leq |\mathrm{N}|} \operatorname{argmax}\{L_p^{(1)}[j], \mathcal{I}_0\};$$

2) for every $k \geq 2$:

$$L_p^{(k)}[i, j] = \begin{cases} \max\{L_p^{(k-1)}[i, j], \max_{h,l}\{\Pr(H_i \rightarrow H_h H_l) \cdot \\ L_p^{(k-1)}[h, j] L_g[l]\}\}, & (i, j) \in \mathcal{I}_{k-1}; \\ L_p^{(k-1)}[i, j], & (i, j) \in \mathcal{I}_0 - \mathcal{I}_{k-1}; \end{cases}$$
$$\mathcal{I}_k = \mathcal{I}_{k-1} - \bigcup_{1 \leq j \leq |\mathrm{N}|} \operatorname{argmax}\{L_p^{(k)}[j], \mathcal{I}_{k-1}\}.$$

A tabular method can be easily devised starting from the above relations, which computes auxiliary arrays $L_p^{(k)}$. Theorem 2 in Appendix relates arrays $L_p^{(k)}$ to the desired array L_p, by proving convergence in at most $|\mathrm{N}|$ iterations.

Let us consider now the highest probabilities of derivation trees whose root is H_i and whose yield is a string in the set $\Sigma^* H_j$. Once again, these quantities are grouped in an $|\mathrm{N}| \times |\mathrm{N}|$ array L_s whose generic element is defined as:

$$L_s[i, j] = \mathrm{Pm}(H_i \langle \Sigma^* H_j \rangle).$$

Symmetrical relations with respect to the ones introduced for arrays $L_p^{(k)}$ can be obtained for a family of arrays $L_s^{(k)}$, leading to a method for the computation of array L_s. Details of the method are omitted here for the sake of brevity, although elements of array L_s will be used in the next section.

Arrays L_p and L_s can be computed with time complexity $O(|\mathrm{N}|^2 |P|)$. In fact each of the $|\mathrm{N}|$ defined iterations requires $|\mathrm{N}||P|$ elementary steps.

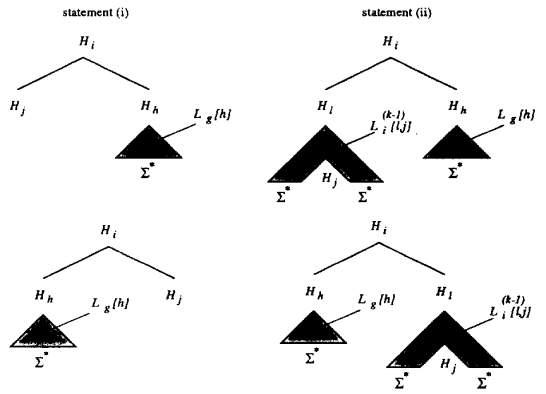


Fig. 3. Schematic representation of the relations used for the definition of arrays $L_i^{(k)}$.

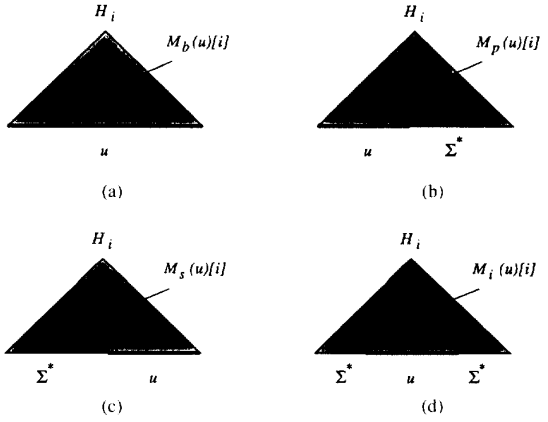


Fig. 4. Schematic representation of the probabilities to be computed on-line.

C. Computation of Factors for Island Upper-Bounds

Let us consider the maximum probability of the derivation trees whose root is H_i and whose yield is a nonterminal H_j surrounded by two strings in Σ^*; these probabilities will be used in the next section for the computation of syntactic upper-bounds for the probability of the derivation of an "island." Let us define an $|\mathrm{N}| \times |\mathrm{N}|$ array L_i as:

$$L_i[i,j] = \mathrm{Pm}(H_i \langle \Sigma^* H_j \Sigma^* \rangle).$$

In this case too, a tabular method can be considered for the computation of L_i requiring no more than $|\mathrm{N}|$ iterations. The recursive relations introduced in the following are a straightforward adaptation of the relations developed for array L_p (see Fig. 3).

Definition 3: A family of $|\mathrm{N}| \times |\mathrm{N}|$ arrays $L_i^{(k)}$ is defined by the following relations:

1) let $\mathcal{I}_0 = \{(h,k) \mid 1 \le h,k \le |\mathrm{N}|\}$:

$$L_i^{(1)}[i,j] = \max_h \{\max\{\Pr(H_i \to H_j H_h), \Pr(H_i \to H_h H_j)\} L_g[h]\}, \quad (i,j) \in \mathcal{I}_0;$$

$$\mathcal{I}_1 = \mathcal{I}_0 - \bigcup_{1 \le i \le |\mathrm{N}|} \mathrm{argmax}\{L_i^{(1)}[i], \mathcal{I}_0\};$$

2) for every $k \ge 2$:

$$L_i^{(k)}[i,j] = \begin{cases} \max\{L_i^{(k-1)}[i,j], \max_{h,l}\{L_i^{(k-1)}[l,j] L_g[h] \cdot \max\{\Pr(H_i \to H_h H_l), \Pr(H_i \to H_l H_h)\}\}\}, & (i,j) \in \mathcal{I}_{k-1}; \\ L_i^{(k-1)}[i,j], & (i,j) \in \mathcal{I}_0 - \mathcal{I}_{k-1}; \end{cases}$$

$$\mathcal{I}_k = \mathcal{I}_{k-1} - \bigcup_{1 \le i \le |\mathrm{N}|} \mathrm{argmax}\{L_i^{(k)}[i], \mathcal{I}_{k-1}\}.$$

Theorem 3 in the Appendix establishes the convergence of arrays $L_i^{(k)}$ to the desired array L_i in at most $|\mathrm{N}|$ steps. As in the case of arrays L_p and L_s, a closer inspection of the relations in Definition 3 reveals that the computation of array L_i has time complexity $O(|\mathrm{N}|^2 |P|)$.

IV. On-Line Computations

This section presents efficient methods for the computation of the probabilities of the most likely trees that derive sentences in $L(G_s)$, subject to the constraint that some fragments of these sentences are fixed. We focus primarily on cases in which a prefix or an island are specified. We then show how the developed theory can be generalized to the cases in which an arbitrary number of islands have already been recognized. The main idea is to adapt well-known tabular methods for the recognition of context-free languages. It is crucial to keep all those computation steps that do not depend on the input constraints, namely the computation of those quantities presented in the previous section. This results in an obvious performance improvement.

A. Best Derivation Probabilities

The probability of the most likely derivation of a given string $u$according to a grammar G_s can be computed using a probabilistic version of the Cocke–Kasami–Younger (CKY) recognizer (see, for instance, [8], [9]) based on the Viterbi algorithm, as shown in [4].

Let $u = a_1 \cdots a_n$ be a string in Σ^n and let \MBU{u} be an $|\mathrm{N}| \times 1$ array such that:

$$M_b(u)[i] = \mathrm{Pm}(H_i \langle u \rangle).$$

The functional notation adopted here for arrays $M_b(u)$ expresses the dependency of the defined quantities upon the given string u. A schematic representation of these probabilities is depicted in Fig. 4(a).

It is easy to prove, by induction on the length of u, that the following recursive relation characterizes $M_b(u)$ (recall that $k{:}u$ and $u{:}k$, $0 < k < n$, are prefixes and suffixes respectively of string u, having length k):

$$M_b(u)[i] = \begin{cases} \Pr(H_i \to u), & |u| = 1; \\ \max_{0<t<|u|,h,l} \{\Pr(H_i \to H_h H_l) \cdot M_b(t{:}u)[h] M_b(u{:}(|u|-t))[l]\}, & |u| > 1. \end{cases}$$

From the above relations it can be seen how the computation of $M_b(u)$ is recursively based on the computation of arrays

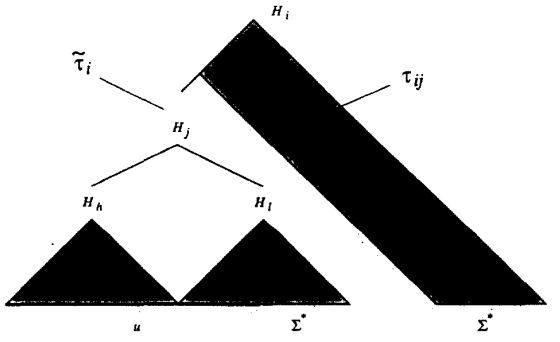


Fig. 5. Schematic representation of the relations used for the computation of arrays $M_p(u)$.

$M_b(u_s)$ for every string u_i that is a proper infix of u. The resulting method has time complexity $O(|P||u|^3)$, which is also the time complexity of the original CKY algorithm.

B. Prefix and Suffix Upper-Bounds

We recall from Section II that, for a given string u, set $H_i\langle u\Sigma^*\rangle$ contains all possible derivation trees having root node labeled by nonterminal H_i and, as a yield, a string of terminal symbols with u as a prefix. In this subsection an efficient computation is introduced for the probability of the most likely trees in $H_i\langle u\Sigma^*\rangle$. In the case $H_i = S$, these probabilities are optimal syntactic upper-bounds for partial interpretations starting with u; therefore, in the following, these probabilities will be referred to as *prefix upper-bounds*.

Let $u = a_1 \cdots a_n$ be a string in Σ^n. Prefix upper-bounds for u are grouped together in an $|\mathrm{N}| \times 1$ array $M_p(u)$ as follows:

$$M_p(u)[i] = \mathrm{Pm}(H_i\langle u\Sigma^*\rangle).$$

The notation adopted in (11) makes explicit the dependency of the probabilities upon the given prefix u. Prefix upper-bounds can be schematically represented as in Fig. 4(b).

The computation of each element $M_p(u)[i]$ can be carried out on the basis of the following observation. Let τ_i be an optimal derivation tree in the set $H_i\langle u\Sigma^*\rangle$. In the general case, τ_i can be decomposed into trees $\tilde{\tau}_i$ and τ_{ij} such that $\tilde{\tau}_i$ is the least (minimum height) subtree of τ_i whose root generates a string containing the complete prefix u and τ_{ij} belongs to $H_i\langle H_j\Sigma^*\rangle$ for some $H_j \in \mathrm{N}$; this is schematically represented in Fig. 5. It follows that τ_{ij} is an optimal derivation tree in $H_i\langle H_j\Sigma^*\rangle$, otherwise τ_i will not be optimal in $H_i\langle u\Sigma^*\rangle$. Since the probability of τ_{ij} is already available as element $L_p[i,j]$ defined in the previous section, it remains to consider the computation of probabilities of $\tilde{\tau}_i$ for a given prefix u and for every root node H_j. The following definition is the basis for this computation.

Definition 4: An $|\mathrm{N}| \times 1$ auxiliary array $\widetilde{M}_p(u)$ is associated with a string u in the following way:

$$\widetilde{M}_p(u)[j] = \begin{cases} \Pr(H_j \rightarrow u), & |u| = 1; \\ \max\limits_{0<t<|u|,h,l} \{\Pr(H_j \rightarrow H_h H_l) \cdot \\ M_b(t{:}u)[h] M_p(u{:}(|u|-t))[l]\}, & |u| > 1. \end{cases}$$

Following Fig. 5, $M_b(t{:}u)[h]$ is the probability of the best derivation from H_h of the first t symbols of u, while $M_p(u{:}(|u|-t))[l]$ is the probability of the best derivation from H_l of a string having the last $|u|-t$ symbols of u as a prefix and an unknown tail. Apart from the case $|u| = 1$, it is easy to show by induction on $|u|$ that the element $\widetilde{M}_p(u)[i]$ corresponds to the probability of an optimal derivation tree in $H_i\langle u\Sigma^*\rangle$ such that each immediate constituent of the root H_i spans a proper substring of u.

We now relate $\widetilde{M}_p(u)$ to the desired array $M_p(u)$. Let us introduce an auxiliary $|\mathrm{N}| \times |\mathrm{N}|$ array L'_p such that $L'_p[i,j] = L_p[i,j]$ for $i \neq j$ and $L'_p[i,i] = 1$ for $1 \leq i \leq |\mathrm{N}|$. Theorem 4 in Appendix shows that $M_p(u)$ can be computed from $\widetilde{M}_p(u)$ by means of the following relation (operator $\otimes$ has been introduced in (5)):

$$M_p(u) = L'_p \otimes \widetilde{M}_p(u).$$

As far as computational complexity is concerned, array $\widetilde{M}_p(u)$ depends upon arrays $M_p(u_s)$ for every proper suffix u_s of u. Both arrays can then be computed recursively, first considering $u = a_n$, then using $M_p(a_n)$ for the computation of the arrays associated with $u = a_{n-1}a_n$, and so on. As discussed before, the computation of $M_b(u_p)$ for every proper prefix u_p of u requires a cubic amount of time with respect to $|u|$. Starting from all these quantities, the computation of $\widetilde{M}_p(u_s)$ can be carried out with complexity $O(|P||u|^2)$, as it appears from a careful analysis of the above relations. In addition, $|u|$ matrix operations $\otimes$ must be performed for the computation of all possible $M_p(u_s)$, where each such operation requires $|\mathrm{N}|^2$ elementary steps. Summing up, the entire computation can be carried out in time $O(\max\{|P||u|^3, |\mathrm{N}|^2|u|\})$: this is roughly the same amount of time required by practical algorithms for the recognition of the string hypothesized so far. Hence, the computation of upper-bounds in a search through competitive hypotheses can be interleaved with the context-free language recognizer without any asymptotic increase in time complexity.

In a symmetrical way with respect to array $M_p(u)$, an $|\mathrm{N}| \times 1$ array $M_s(u)$ can be defined (see again Fig. 4(c)):

$$M_s(u)[i] = \mathrm{Pm}(H_i\langle \Sigma^* u\rangle).$$

Elements of $M_s(u)$ will be called *suffix upper-bounds*. Relations similar to the one discussed above can be derived for the computation of suffix upper-bounds, based on an auxiliary array $\widetilde{M}_s(u)$ defined in a symmetrical way with respect to array $\widetilde{M}_p(u)$. In analogy with (12), the following relation can in fact be established:

$$M_s(u) = L'_s \otimes \widetilde{M}_s(u),$$

where L'_s is an $|\mathrm{N}| \times |\mathrm{N}|$ array obtained from L_s by replacing the diagonal with unit elements. Further details are omitted because of the similarity with the relations developed for $M_p(u)$.

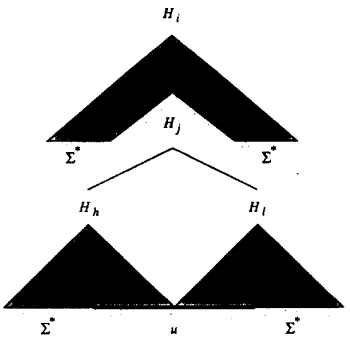


Fig. 6. Schematic representation of the relations used for the computation of arrays $M_i(u)$.

C. Island Upper-Bounds

An efficient way of computing the highest probability among all derivation trees in the set $H_i\langle\Sigma^* u \Sigma^*\rangle$ is now introduced. This probability will be called *island upper-bound*. If $H_i = S$, island upper-bounds represent optimal syntactic upper-bounds in the search for the most likely derivation of a sentence in $L(G_s)$, starting from the already analyzed islands. As the theory is similar to the one presented in Section IV-B, some details will be omitted.

Given a string $u = a_1 \cdots a_n$ in Σ^n, $M_i(u)$ is the $|\mathrm{N}| \times 1$ array whose elements are defined as follows:

$$M_i(u)[i] = \mathrm{Pin}(H_i\langle\Sigma^* u \Sigma^*\rangle).$$

Island upper-bound probabilities can be schematically represented as in Fig. 4(d). Array $M_i(u)$ can be computed starting from the auxiliary array introduced in the next definition (see Fig. 6 for a graphical interpretation of the relations reported in the definition).

Definition 5: An $|\mathrm{N}| \times 1$ auxiliary array $\widetilde{M}_i(u)$ is associated with string u in the following way:

$$\widetilde{M}_i(u)[j] = \begin{cases} \Pr(H_j \to u), & |u| = 1; \\ \max\limits_{0<t<|u|,h,l} \{\Pr(H_j \to H_h H_l) \cdot \\ \quad M_s(t{:}u)[h] M_p(u{:}(|u|-t))[l]\}, & |u| > 1. \end{cases}$$

Observe that array $\widetilde{M}_i(u)$ is computed using elements of arrays $M_s(u_p)$ and $M_p(u_s)$, for every proper prefix and suffix of u such that $u_p u_s = u$.

Let L'_i be an $|\mathrm{N}| \times |\mathrm{N}|$ array such that $L'_i[i,j] = L_i[i,j]$ for $i \neq j$ and $L'_i[i,i] = 1$ for $1 \leq i \leq |\mathrm{N}|$. In analogy with prefix upper-bounds, array $M_i(u)$ can be eventually computed by means of the following relation:

$$M_i(u) = L'_i \otimes \widetilde{M}_i(u).$$

It can be easily seen that island upper-bounds can be computed in an amount of time asymptotically equivalent to the one found for the computation of prefix upper-bounds.

D. Multiple Islands and Gaps

The theory developed so far is now generalized to the case of arbitrary sequences of islands and gaps. It will be shown that this general case can be dealt with with the same asymptotical time complexity derived in previous subsections.

Let $L = \Sigma^* u_1 \Sigma^* u_2 \Sigma^* \cdots \Sigma^* u_d \Sigma^*$, $d \geq 2$, where each u_k is an island of known words. We are concerned with the computation of the probability of the most likely derivation trees in $S\langle L\rangle$. Language L can be encoded by means of a string sequence $\sigma = (u_1, u_2, \cdots, u_d)$. We set $|\sigma| = \sum_{k=1}^{d} |u_k|$. We define an $|\mathrm{N}| \times 1$ array as follows:

$$M(\sigma)[i] = \mathrm{Pin}(H_i\langle L\rangle).$$

A word in some island u_k of σ is called a *left* (*right*) *boundary* if it is at the beginning (end) of u_k. In all other cases, the word is called *internal*. Let $q < r$, be two non-negative integers. We denote as ${}_q\sigma_r$ the sequence obtained from σ by considering only words in positions t, for $q+1 \leq t \leq r$. Let also a and b be the $(q+1)$th and the rth words in σ respectively. We say that ${}_q\sigma_r$ is a *generalized prefix* if a is a left boundary and b is internal. Conversely, ${}_q\sigma_r$ is a *generalized suffix* if a is internal and b is a right boundary. If a and b are left and right boundary, respectively, ${}_q\sigma_r$ is a *generalized island*; if a and b are both internal, ${}_q\sigma_r$ is a *generalized string*. Generalizing the approach of the previous subsections, we now introduce an auxiliary array that will be used in the computation of array $M(\sigma)$ (note that $\sigma = {}_0\sigma_n$, $n = |\sigma|$).

Definition 6: An $|\mathrm{N}| \times 1$ auxiliary array $\widetilde{M}({}_q\sigma_r)$ is associated with sequence ${}_q\sigma_r$ and is specified as follows:

$$\widetilde{M}({}_q\sigma_r)[i] = \begin{cases} \Pr(H_i \to {}_q\sigma_r), & |{}_q\sigma_r| = 1; \\ \max\limits_{0<t<|{}_q\sigma_r|,h,l} \{\Pr(H_i \to H_h H_l) \cdot \\ \quad M({}_q\sigma_{q+t})[h] M({}_{q+t}\sigma_r)[l]\}, & |{}_q\sigma_r| > 1. \end{cases}$$

Using quantities L'_p, L'_s and L'_i introduced in the preceding subsections, array $M(\sigma)$ can be eventually computed using the following relations:

$$M({}_q\sigma_r) = \begin{cases} L'_p \otimes \widetilde{M}({}_q\sigma_r) & {}_q\sigma_r \text{ a generalized prefix;} \\ L'_s \otimes \widetilde{M}({}_q\sigma_r) & {}_q\sigma_r \text{ a generalized suffix;} \\ L'_i \otimes \widetilde{M}({}_q\sigma_r) & {}_q\sigma_r \text{ a generalized island;} \\ \widetilde{M}({}_q\sigma_r) & {}_q\sigma_r \text{ a generalized string.} \end{cases}$$

The recursive relations introduced above generalize the cases we have seen in previous subsections. A proof of the correctness of these relations with respect to the (17) is similar to the other proofs of this section. The proposed method is closely related to the CKY algorithm. This can be easily seen by observing that each array $M({}_q\sigma_r)$, $0 < q < r < n$, is recursively defined on the basis of arrays $M({}_{q'}\sigma_{r'})$, $q < q' < r' < r$. The whole computation can then be carried out using a recognition table T which is an $n \times n$ array, whose entries correspond to arrays $M({}_q\sigma_r)$:

$$T[q,r] = M({}_q\sigma_r), \qquad 0 < q < r < n. \tag{8}$$

Observe also that by introducing the auxiliary arrays $\widetilde{M}({}_q\sigma_r)$, we have been able to factor out in the computation those steps

that do not depend on the input sequence σ and can therefore be computed off-line. Computational complexity analysis can be carried out as in the preceding cases and gives the same asymptotical results.

V. Conclusion

Stochastic grammars are a useful tool for tasks such as driving the interpretation of a written or a spoken sentence. Pioneer work in stochastic syntax analysis is described in the book by Fu [10] and has its roots in papers by Salomaa [11], Lee and Fu [12], Persoon and Fu [13], Lu and Fu [14].

When these grammars are used, there is a need for computing the probability that they generate a sentence, given only some of the words in it. Expressions for the calculation of probabilities of strings of this type have been proposed by Jelinek and Lafferty [1] for the case in which only the prefix of a sentence is known. This case is also considered by Persoon and Fu [13] for scoring partial parses of strings which cannot be further parsed because they have been affected by an error. Corazza *et al.* [2] have proposed methods for probability computation in the more general case in which partial word strings interleaved by gaps are given. This computation is too complex in practice unless the lengths of the gaps are known.

This paper addresses a new problem and proposes a method for computing the probability of the most likely trees that can generate a sentence only part of which is known. This probability is the lowest possible upper-bound for an evaluation function to be used in driving the search for the best interpretation of an input pattern description. Even in the most general case of an input sequence consisting of an arbitrary number of strings separated by gaps of unknown length, we have shown that the above probability can be computed using (deterministic) polynomial time. This makes it possible to use SCFG's in practice for driving interpretations of sentences in natural language.

In the developed theory, no assumption about the gap length has been considered. However, there are applications in which the gap length can be estimated with a good approximation, as for example in Optical Character Recognition (OCR). The theory developed so far can be easily extended to cases in which the gap length (or a probability distribution of it) is known. Also in this case, the overall time complexity is cubic in the length of the input sequence. More details about computations in this case can be found in [15]

The theory proposed in this paper has affinity with that of error-correcting parsing. Syntactic errors can be represented by stochastic regular expressions. Probabilities of most likely errors can be precomputed generalizing in this way the minimum-distance criterion of [16].

Appendix

A. Correctness of the Methods

In this appendix, we provide formal proofs of the correctness of the methods reported in Sections III and IV for the computation of the proposed syntactic upper-bounds.

We start with a technical lemma that presents a property that has been used in the proposed methods; in order to do this, we need to introduce some notation. Let τ be a derivation tree; the set of all (*derivation*) *subtrees* of τ is defined as follows. Every internal (nonleaf) node n in τ along with *all* its successor nodes uniquely identify the subtree of τ rooted at n. Note that leaves are not considered subtrees. We say that τ' is a *proper* subtree of τ if τ' is a subtree of τ and $\tau' \neq \tau$. Let M be a predicate defined on a set of derivation trees $\mathcal{A}$. The set $\mathcal{G}_M$ is composed of all and only the derivation trees $\tau \in \mathcal{A}$ such that $M(\tau)$ holds and there is no $\tau' \in \mathcal{A}$, τ' a proper subtree of τ, for which M holds. The set $\mathcal{G}_M$ is called the *ground* set of M.

Lemma 1: If M is a predicate defined on a set $\mathcal{A}$ of derivation trees and $\mathcal{G}_M$ is the corresponding ground set, then

$$\max_{\tau \in \mathcal{A}, M(\tau)} \Pr(\tau) = \max_{\tau \in \mathcal{G}_M} \Pr(\tau). \tag{A.1}$$

Proof: Let τ^* be one of the trees having maximum probability among all trees in $\mathcal{A}$ for which M holds:

$$\Pr(\tau^*) = \max_{\tau \in \mathcal{A}, M(\tau)} \Pr(\tau).$$

If $\tau^* \in \mathcal{G}_M$ then the lemma is proved. Otherwise, from the definition of $\mathcal{G}_M$, there exists a proper subtree τ of τ^* such that $\tau \in \mathcal{G}_M$; it follows that $\Pr(\tau) \geq \Pr(\tau^*)$. But this implies $\Pr(\tau) = \Pr(\tau^*)$ and then

$$\Pr(\tau^*) = \max_{\tau \in \mathcal{G}_M} \Pr(\tau),$$

which proves the lemma. □

Lemma 1 shows that the search for the probability of the "best" derivation trees in some set $\mathcal{A}$, for which some property M holds, can be restricted to the set $\mathcal{G}_M$. This is useful in all cases in which $\mathcal{G}_M$ is properly included in the set of all trees in $\mathcal{A}$ that satisfy M. Note that no assumption is made on set $\mathcal{A}$: in the following this result is applied to sets of complete derivation trees as well as to sets of incomplete derivation trees.

We turn now to the results regarding the off-line computations discussed in Section III. In the following we prove a characterization of array L_g in terms of the arrays $L_g^{(k)}$ introduced in Definition 1

Theorem 1: For every k, $1 \leq k \leq |\mathrm{N}|$, the following statements hold:

a) $L_g^{(k)}[i] = \max_{\tau \bullet H_i\langle\Sigma^*\rangle, \mathrm{hg}(\tau) \leq k} \Pr(\tau), \quad i \in \mathcal{I}_0;$

b) $L_g^{(k)}[i] = \mathrm{Pm}(H_i\langle\Sigma^*\rangle), \quad i \in \mathcal{I}_0 - \mathcal{I}_k.$

Proof: Statement a) follows from the definition of $L_g^{(k)}$ and is easily proved by finite induction on k.

Statement b) is proved by finite induction on k. In the base case ($k = 1$) it has to be proved that, for every $i \in \mathrm{argmax}\{L_g^{(1)}, \mathcal{I}_0\}$, $L_g^{(1)}[i]$ equals the probability of the "optimal" tree(s) in $H_i\langle\Sigma^*\rangle$. Let Q_1 be a predicate that is true for every complete tree in $T(G_s)$, that is every complete tree derived in G_s; note that the ground set G_{Q_1} of Q_1 is the set of all trees with height equal to one. Lemma 1 states that $\max_{\tau \in T(G_s)} \Pr(\tau) = \max_{\tau \in \mathcal{G}_{Q_1}} \Pr(\tau)$. Since from statement (i) of the theorem $L_g^{(1)}$ is defined by the highest probabilities of

trees in $\mathcal{G}_{Q_1}$, the base case directly follows from the definition of the operator argmax.

Let us consider the inductive step for $k > 1$. Assuming the inductive hypothesis, it suffices to show that $L_g^{(k)}[i] = \mathrm{Pm}(H_i\langle\Sigma^*\rangle)$ for every i in $\mathcal{I}_{k-1} - \mathcal{I}_k$. We proceed as in the base case, by defining a unary predicate Q_k on derivation trees. For every τ, $Q_k(\tau)$ holds if and only if the following conditions are both satisfied:

- the root of τ is H_i, $i \in \mathcal{I}_{k-1}$;
- for every τ' a proper subtree of τ, if the root of τ' is H_j, $j \in \mathcal{I}_0 - \mathcal{I}_{k-1}$, then $\Pr(\tau') = \mathrm{Pm}(H_j\langle\Sigma^*\rangle)$.

It can be easily shown that, for any derivation tree τ having root H_i, $i \in \mathcal{I}_{k-1}$, if $Q_k(\tau)$ is not true, then τ cannot be considered an optimal tree in $H_i\langle\Sigma^*\rangle$; therefore attention can be restricted to the tree set characterized by Q_k. From a careful examination of the definition of Q_k and the definition of ground set, it is possible to conclude that a derivation tree τ is in $\mathcal{G}_{Q_k}$, if and only if the root of τ has label H_i, $i \in \mathcal{I}_{k-1}$, and *each one* of its children is a tree in $H_j\langle\Sigma^*\rangle$, $j \in \mathcal{I}_0 - \mathcal{I}_{k-1}$, with the highest probability, i.e., a tree composed of "optimal" subtrees already considered at previous steps. Using the inductive hypothesis, it can be seen that the max operator in statement b) of Definition 1 is applied to a superset of $\mathcal{G}_{Q_k}$. Therefore, applying Lemma 1 and the definition of operator argmax, one gets:

$$L_g^{(k)}[i] = \mathrm{Pm}(H_i\langle\Sigma^*\rangle), \qquad \in \mathcal{I}_{k-1} - \mathcal{I}_k.$$

This concludes the proof.[3] □

Since $\mathcal{I}_{|\mathrm{N}|} = \emptyset$, it directly follows that arrays $L_g^{(k)}$ converge to L_g in no more than $|\mathrm{N}|$ steps.

The next theorem proves the correctness of the method for the computation of the array L_p through the construction of arrays $L_p^{(k)}$, $1 \le k \le |\mathrm{N}|$ (see Definition 2). Since the proof of this theorem is similar to the proof of Theorem 1, only an outline will be given here. Recall that, given a tree $\tau \in H_i\langle H_j\Sigma^*\rangle$, $\mathrm{sp}(\tau)$ represents the length of the spine of τ.

Theorem 2: For every k, $1 \le k \le |\mathrm{N}|$, the following statements hold:

a) $L_p^{(k)}[i,j] = \max_{\tau \in H_i\langle H_j\Sigma^*\rangle, \mathrm{sp}(\tau) \le k} \Pr(\tau)$, $(i,j) \in \mathcal{I}_0$;

b) $L_p^{(k)}[i,j] = \mathrm{Pm}(H_i\langle H_j\Sigma^*\rangle)$, $\quad (i,j) \in \mathcal{I}_0 - \mathcal{I}_k$.

Outline of the proof: Statement a) directly follows from the definition of $L_p^{(k)}$. Statement b) can be proved by finite induction on k, in analogy with Theorem 1 and considering the set $\mathcal{L}$ of all derivation trees in $H_i\langle H_j\Sigma^*\rangle$, $(i,j) \in \mathcal{I}_0$. In the base case, a predicate Q_1 is defined that is true for every element in $\mathcal{L}$; the proof then proceeds as in Theorem 1. The inductive step ($k > 1$) has to show that for every j, $1 \le j \le |\mathrm{N}|$, the equality $L_p^{(k)}[i,j] = \mathrm{Pm}(H_i\langle H_j\Sigma^*\rangle)$ holds for every i such that $(i,j) \in \mathrm{argmax}\{L_p^{(k)}[j], \mathcal{I}_{k-1}\}$. For each value of j, one can proceed as in the inductive step of Theorem 1 and define a predicate $Q_{k,j}$ as follows. For every $\tau \in \mathcal{L}$, $Q_{k,j}(\tau)$ holds if and only if the following conditions are both satisfied:

- the root node and the nonterminal node in the yield of τ are labeled by symbols H_i and H_j respectively, such that $(i,j) \in \mathcal{I}_{k-1}$;
- for every τ' a proper subtree of τ, if $\tau' \in H_h\langle H_j\Sigma^*\rangle$, $(h,j) \in \mathcal{I}_0 - \mathcal{I}_{k-1}$, then $\Pr(\tau') = \mathrm{Pm}(H_h\langle H_j\Sigma^*\rangle)$.

For every tree τ in $H_i\langle H_j\Sigma^*\rangle$, if $Q_{k,j}(\tau)$ is false then τ is not an optimal tree in that set. One can then observe that in the definition of the jth column of array $L_p^{(k)}$ (statement b) of Definition 2 the max operator is applied to a superset of $\mathcal{G}_{Q_{k,j}}$. Therefore, a direct application of Lemma 1 proves the inductive step. □

Again, from the above result it directly follows that arrays $L_p^{(k)}$ converge to L_p in no more than $|\mathrm{N}|$ steps.

The next theorem states a similar result for array L_i introduced in Definition 3. The proof is similar to the one of Theorem 2, and is therefore omitted.

Theorem 3: For every k, $1 \le k \le |\mathrm{N}|$, the following statements hold:

a) $L_i^{(k)}[i,j] = \max_{\tau \in H_i\langle\Sigma^* H_j\Sigma^*\rangle, \mathrm{sp}(\tau) \le k} \Pr(\tau)$, $\quad (i,j) \in \mathcal{I}_0$;

b) $L_i^{(k)}[i,j] = \mathrm{Pm}(H_i\langle\Sigma^* H_j\Sigma^*\rangle)$, $\quad (i,j) \in \mathcal{I}_0 - \mathcal{I}_k$.

Finally, we prove the correctness of the method presented in Section IV-B for the on-line computation of the array $M_p(u)$. We remark that a similar proof can be used to show the correctness of the method reported in Section IV-B for the computation of the array $M_i(u)$.

Theorem 4: Let L'_p be an $|\mathrm{N}| \times |\mathrm{N}|$ array such that $L'_p[i,j] = L_p[i,j]$ for $i \ne j$ and $L'_p[i,i] = 1$ for $1 \le i \le |\mathrm{N}|$. Then, we have:

$$M_p(u) = L'_p \otimes \widetilde{M}_p(u).$$

Proof: For every $1 \le i \le |\mathrm{N}|$, let τ_i be an optimal derivation tree in $H_i\langle u\Sigma^*\rangle$, that is $\Pr(\tau_i) = M_p(u)[i]$. If the left child of the root node in τ_i does not completely generate prefix u, then $\Pr(\tau_i) = \widetilde{M}_p(u)[i]$. Since τ_i is optimal, $\Pr(\tau_i) = L'_p[i,i] \times \widetilde{M}_p(u)[i] \ge L'_p[i,k] \times \widetilde{M}_p(u)[k]$ for every $1 \le k \le |\mathrm{N}|$. By the definition of $\otimes$ it follows that $M_p(u)[i] = [L'_p \otimes \widetilde{M}_p(u)][i]$. In the general case (see Fig. 5 in Section III) let $\widetilde{\tau}_i$ be the least proper subtree of τ_i that generates the complete prefix u; let also H_j be the root node of $\widetilde{\tau}_i$ for some $1 \le j \le |\mathrm{N}|$. The derivation tree τ_i can then be decomposed into subtrees $\widetilde{\tau}_i$ and τ_{ij} such that τ_{ij} belongs to $H_i\langle H_j\Sigma^*\rangle$. The equality $\Pr(\widetilde{\tau}_i) = \widetilde{M}_p(u)$ must hold, otherwise τ_i cannot be an optimal tree. For the same reason, the equality $\Pr(\tau_{ij}) = L_p[i,j]$ also holds. Again from the definition of $\otimes$, $M_p(u)[i] = [L'_p \otimes \widetilde{M}_p(u)][i]$. □

ACKNOWLEDGMENT

Authors wish to thank an anonymous referee whose comments and suggestions have been very helpful in the development of Section IV-D.

[3] From the proof of the inductive step it follows that the computation based on statement b) in Definition 1 can be improved by restricting the indices h and l of the max operator to those belonging to the set $\mathcal{I}_0 - \mathcal{I}_{k-1}$ (in this case, statement b) of the theorem is no longer true).

REFERENCES

[1] F. Jelinek and J. D. Lafferty, "Computation of the probability of initial substring generation by stochastic context free grammars," *Computat. Linguist.*, vol. 17, no. 3, pp. 315–323, 1991.

[2] A. Corazza, R. De Mori, R. Gretter, and G. Satta, "Computation of probabilities for a stochastic island-driven parser," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, no. 9, pp. 936–950, 1991.

[3] W. A. Woods, "Optimal search strategies for speech understanding control," *Artificial Intell.*, vol. 18, no. 3, pp. 295–326, 1982.

[4] F. Jelinek, J. D. Lafferty, and R. L. Mercer, "Basic methods of probabilistic context free grammars," in *Speech Recognition and Understanding*, R. D. Mori and P. Laface, Eds. Berlin, Germany: Springer-Verlag, 1992.

[5] R. C. Gonzales and M. G. Thomason, *Syntactic Pattern Recognition*. Reading, MA: Addison-Wesley, 1978.

[6] C. S. Wetherell, "Probabilistic languages: a review and some open questions," *Computing Surveys*, vol. 12, no. 4, pp. 361–379, 1980.

[7] M. A. Harrison, *Introduction to Formal Language Theory*. Reading, MA: Addison-Wesley, 1978.

[8] D. H. Younger, "Recognition and parsing of context-free languages in time n^3," *Inform. Contr.*, vol. 10, pp. 189–208, 1967.

[9] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation and Compiling*, vol. 1. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[10] K. S. Fu, *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[11] A. Salomaa, "Probabilistic and weighted grammars," *Inform. Contr.*, vol. 15, pp. 529–544, 1969.

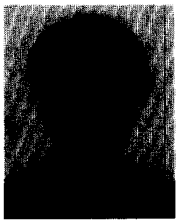
[12] H. C. Lee and K. S. Fu, "A stochastic syntax analysis procedure and its application to pattern classification," *IEEE Trans. Comput.*, vol. C-4, no. 3, pp. 660–666, 1972.

[13] E. Persoon and K. S. Fu, "Sequential classification of strings generated by stochastic context-free grammars," *Int. J. Comput. Inform. Sci.*, vol. 4, no. 3, pp. 205–218, 1975.

[14] S. Y. Lu and K. S. Fu, "Stochastic error-correcting syntax analysis for recognition of noisy patterns," *IEEE Trans. Comput.*, vol. C-26, no. 12, pp. 1268–1276, 1977.

[15] R. Gretter, "Upper-bounds for theories composed by islands and gaps and generated by stochastic context-free grammars—General case," manuscript, IRST, Trento, Italy, 1993.

[16] E. Tanaka and K. S. Fu, "Error-correcting parsers for formal languages," *IEEE Trans. Comput.*, vol. C-27, no. 7, pp. 605–616, 1978.



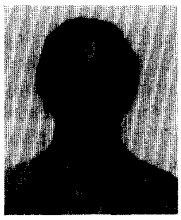
Anna Corazza was born in Verona, Italy, on July 17, 1964. She received the Doctoral degree in electronic engineering from the University of Padua, Italy, in 1989.

Since then, she is joining the Speech Processing group at the Istituto per la Ricerca Scientifica e Tecnologica (IRST), in Trento, Italy. Her research activity regards stochastic modeling of language for applications in speech recognition and understanding.

Renato De Mori was born in Milan, Italy in 1941. He received the doctorate degree in electronic engineering from Politecnico di Torino, Torino, Italy, in 1967.

He became full Professor in Italy in 1975. Since 1986, he has been Professor and the Director of the School of Computer Science at McGill University, Montreal, PQ, Canada. In 1991, he became Associate of the Canadian Institute for Advanced Research and project leader of the Institute for Robotics and Intelligent Systems, a Canadian Center of Excellence. His research interests are now stochastic parsing techniques,automatic speech understanding, connectionist models, and reverse engineering.

Dr. De Mori is the author of many publications in the areas of computer systems, pattern recognition, artificial intelligence, and connectionist models. He is a Fellow of the IEEE Computer Society, has been member of various committees in Canada, Europe, and the United States; he is on the board of the following international journals: the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, *Signal Processing*, *Speech Communication*, *Pattern Recognition Letters*, *Computer Speech and Language*, and *Computational Intelligence*.



Roberto Gretter was born in Trento, Italy, on October 12, 1961. He received the doctoral degree in electronic engineering from the University of Padua in 1987.

Since 1988, he has been Researcher at the Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy, where he is working in the area of automatic speech recognition. His present research interests are focused on large vocabulary speech recognition and morphological analysis.

Giorgio Satta was born in Battipaglia, Italy, on July 15, 1960. He received the Doctoral degree in electronic engineering from the University of Padua, Italy, in 1985 and the Ph.D. degree in computer science from the Department of Pure and Applied Mathematics at University of Padua, in 1990.

He is currently joining the Department of Computer Science at University of Venice, Italy. His research interests are in natural language parsing, mathematics of language, and formal language theory.