Computation of Upper-Bounds for Stochastic Context-Free Languages*

A. Corazza  
Istituto per la Ricerca Scientifica e Tecnologica,  
38050 Povo di Trento, Italy  
corazza@irst.it

R. De Mori  
School of Computer Science,  
McGill University,  
3480 University str, Montreal, Quebec, Canada, H3A2A7  
renato@chuck.cs.mcgill.ca

G. Satta  
Institute for Research in Cognitive Science,  
University of Pennsylvania, Philadelphia, PA, USA  
gsatta@linc.cis.upenn.edu

dedicated by the input acoustic features $A$:

$$s(\sigma_p) = s_1(A | \sigma_p)s_2(\sigma_p)$$  

where $s_1(A | \sigma_p)$ is an upper-bound on $Pr(A | \sigma)$ and can be obtained as will be briefly discussed in the last section, while $s_2(\sigma_p)$ is the probability of the best complete derivation tree that can derive $\sigma_p$.

In the following sections the computation of $s_2$ is studied for partial solutions derived by left-to-right as well as middle-out parsing strategies. In the last section we argue that this upper-bound is optimal and we
compare it with upper-bounds proposed in the literature for ASR, showing that our approach results in a more feasible computation.

Background

In this section basic definitions and notation adopted in this paper are introduced. Let $\Delta$ be a generic set of symbols. A string $u$ over $\Delta$ is a finite sequence of symbols from $\Delta$; we write $|u|$ to denote the length of $u$. The null string $\varepsilon$ is the (unique) string whose length equals zero. Let $u = w_1 \ldots w_n$, $n \geq 1$; we write $k:u$ and $u:k$ for convenience, we also assume $N = \{H_1, \ldots, H_N\}$, to denote the prefix string $w_1 \ldots w_k$ and the suffix string $w_{n-k+1} \ldots w_n$ respectively.

The set of all strings over $\Delta$ is denoted $\Delta^*$ ($\varepsilon$ included). Let $u$ and $v$ be two strings in $\Delta^*$; $uv$ 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 $\{x \mid x = yu, y \in L_1\}$, $L_1u$ denotes $\{x \mid x = yu, y \in L_1\}$ and $L_1L_2$ denotes $\{x \mid x = yz, y \in L_1, z \in L_2\}$.

A SCFG is a 4-tuple $G_\gamma = (N, C, P, S)$, where $N$ is a finite set of nonterminal symbols, $C$ is a finite set of terminal symbols such that $N \cap C = \emptyset$, $S \in N$ is a special symbol called start symbol and $P$ is a finite set of pairs $(p, Pr(p))$, where $p$ is a production and $Pr(p)$ is the probability associated to $p$ in $G_\gamma$. Productions are represented with the form $H \rightarrow \gamma$, $H \in N$, $\gamma \in (N \cup \Sigma)^*$, and symbol $P$ denotes the set of all productions in $P$. As a convention, productions which do not belong to $P$ have zero probability. We assume $G_\gamma$ to be a proper SCFG, that is the following relation holds for every $H$ in $N$:

$$\sum_{\gamma \in (N \cup \Sigma)^*} Pr(H \rightarrow \gamma) = 1. \quad (4)$$

The grammar $G_\gamma$ is in Chomsky Normal Form (CNF) if all productions in $P$ have the form $H \rightarrow F_1F_2$ or $H \rightarrow w$, where $H, F_1, F_2 \in N$, $w \in \Sigma$. Without loss of generality, we assume in the following that $G_\gamma$ is in CNF; for convenience, we also assume $N = \{H_1, \ldots, H \mid \Sigma\}$, $H_1 = S$.

The definition of derivation in a SCFG, can be found in the literature on formal languages (see for example [Gonzales and Thomasen, 1978], [Wetherell, 1980]). Let $\gamma$ be a string in $(N \cup \Sigma)^*$. A derivation in $G_\gamma$ of $\gamma$ from a nonterminal $H$ is represented as a derivation tree, indicating all productions that have been used in the derivation (with repetitions). We write $H(\gamma)$ to denote the set of all such trees. Let $\tau$ be a derivation tree in $H(\gamma)$; $Pr(\tau)$ is the probability of $\tau$, and is obtained as the product of probabilities of all productions involved in $\tau$ (with repetitions). The operator $Prm$, defined as follows, is introduced:

$$Prm(H(L)) = \max_{\tau \in H(L)} \{Pr(\tau)\}. \quad (5)$$

Let $L$ be a string set, $L \subseteq (N \cup \Sigma)^*$. We generalize the preceding notation and write $H(L)$ to represent the set of all derivation trees with root $H$ and yield in $L$, $Prm(H(L))$ is the maximum among probabilities of all $\tau$ in $H(L)$. In this paper we will develop a framework for the computation of $Prm(H(L))$ for $L = \{u\}$, $L = u\Sigma^*$ and $L = \Sigma^*u\Sigma^*$, where $u$ is a given string over $\Sigma$.

As discussed in the introduction, these quantities can be used as upper-bounds in the search for the most likely complete derivation for the input signal. Due to space constraints, only the highlights of the theory are presented in this paper. A complete description along with formal proofs can be found in [Corazza et al., 1991].

Let $A$ and $B$ be two sets; we write $A - B$ to denote the asymmetric set-difference between $A$ and $B$, that is the set $\{x \mid x \in A, x \notin B\}$. 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.

$$[M_1 \otimes M_2][i, j] = \max_{1 \leq k \leq n} \{M_1[i, k] \cdot M_2[k, j]\}. \quad (6)$$

Let $M$ be a $1 \times n$ array and $T$ be a subset of $\{1, \ldots, n\}$; we define $argmax\{M[T]\}$ to be the set $\{i \mid M[i] \geq M[k], k \in T\}$. If $M$ is an $m \times n$ array, we extend the notation to $M[i]$, the $i$-th row of $M$, in the following way. Let $T \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\}$; we define $argmax\{M[T]\}$ to be the set $\{(i, j) \mid M[i, j] \geq M[i, h], (i, h) \in T\}$.

Off-line computations

Some of the expressions required for computing the "best" derivation probability of a sentence partial interpretation do not depend on the input string, but only on the grammar; therefore these terms can be computed once for all. Methods for the computation of these quantities are studied in the following. The basic idea is to use a “dynamic programming” technique which is reminiscent of the well-known methods for removing useless symbols from a context-free grammar (see for example [Harrison, 1978]). A summary of these quantities is schematically depicted in Figure 1.

![Figure 1: Summary of the quantities computed off-line.](image-url)
Let us define an \( |N| \times 1 \) array \( M_g \) such that
\[
M_g[i] = \Pr(H_i(\Sigma^*)).
\] (7)
The elements of \( M_g \) can be computed as follows. Let \( \tau \) be a derivation tree such that a path from its root to one of its leaves contains more than one occurrence of a nonterminal \( H_j \). Let also \( \tau' \) be the derivation tree obtained from \( \tau \) by removing the subtree rooted in the occurrence of \( H_j \) closer to the root of \( \tau \) and replacing it with the subtree rooted in the occurrence of \( H_j \) closer to the yield of \( \tau \). It is possible to show that the probability of \( \tau' \) is strictly less than the probability of \( \tau \). The process can be iterated, showing that every optimal tree has height not greater than \( |N| \). As a second point, let \( T_k \) be the set of all derivation trees with height not greater than \( k \). By induction on \( k \), it is possible to show that within \( T_k \) a set of at least \( k \) trees \( \{ \tau_1, \tau_2, \ldots, \tau_k \} \) is always found, whose roots are labeled by different symbols \( H_1, H_2, \ldots, H_k \) and whose probabilities are optimal, that is these probabilities define the elements \( M_g[i_1], M_g[i_2], \ldots, M_g[i_k] \) (this result is proved in Corazza et al., 1991) and is exploited in the following to speed-up the computation.

A tabular method can then be used to compute array \( M_g \), where at the \( k \)-th iteration we consider set \( T_k \). As a consequence of the two observations above, the method converges to \( M_g \) in at most \( |N| \) iterations and at each iteration at least \( |N| \) new elements of \( M_g \) are available (again see Corazza et al., 1991 for a formal proof of this statement); the remaining elements are recorded using sets \( I_k \): this enables us to speed-up the computation.

We specify in the following the relations that can be used in the computation of \( M_g \) (note how elements of array \( M_g \) are exploited):
\begin{align*}
(i) & \quad \text{for } (i, j) \in I_0, I_0 = \{ (h, k) \mid 1 \leq h, k \leq |N| \}:
\quad L_p^{(1)}[i, j] = \max_h \{ \Pr(H_i \rightarrow H_j H_h) M_g[h] \} \\
& \quad I_1 = I_0 - \argmax \{ M_g^{(1)}, I_0 \} \\
(ii) & \quad \text{for every } k \geq 2:
\quad L_p^{(k)}[i, j] = \left\{ \begin{array}{ll}
\max_{h,i} \{ L_p^{(k-1)}[i, j], \max_h \Pr(H_i \rightarrow H_h H_i) \} \\
M_g^{(k-1)}[h, i], i \in I_0 - I_{k-1} \\
M_g^{(k-1)}[i], i \in I_{k-1} - I_{k-2} \\
I_k = I_{k-1} - \argmax \{ M_g^{(k-1)}, I_{k-1} \}.
\end{array} \right.
\end{align*}

A closer inspection of the above relations reveals that the computation can be carried out in an amount of time \( O(|N|^2 |P_3|) \).

Computation of factors for prefix and suffix upper-bounds

In the next section, we will need to know the probabilities of the "optimal" derivation trees whose root is \( H_i \) and whose yield is a string in \( \Sigma^* \). Let us define an \( |N| \times |N| \) array \( L_p \) such that (see Figure 1)
\[
L_p[i, j] = \Pr(H_i(\Sigma^* H_j)).
\] (8)
Elements of the array \( L_p \) can be computed using a tabular technique very similar to the one employed in the computation of \( M_g \), as described in the following. Let \( T_k \) be the set of all derivation trees in \( H_i(\Sigma^* H_j) \) such that the path from the root node \( H_i \) to the left-corner node \( H_j \) has length not greater than \( k \). At the \( k \)-th iteration in the computation, we consider the "best" probabilities of elements of \( T_k \) and we are guaranteed to converge to the desired array \( L_p \) in no more than \( |N| \) iterations. Moreover, at each iteration in the computation we are guaranteed that at least \( |N| \) new elements of \( L_p \) are available (again see Corazza et al., 1991 for a formal proof of this statement); the remaining elements are recorded using sets \( I_k \): this enables us to speed-up the computation.

We specify in the following the relations that can be used in the computation of \( L_p \) (note how elements of array \( M_g \) are exploited):
\begin{align*}
(i) & \quad \text{for } (i, j) \in I_0, I_0 = \{ (h, k) \mid 1 \leq h, k \leq |N| \}:
\quad L_p^{(1)}[i, j] = \max_h \{ \Pr(H_i \rightarrow H_j H_h) M_g[h] \} \\
& \quad I_1 = I_0 - \argmax \{ L_p^{(1)}[i], I_0 \} \\
(ii) & \quad \text{for every } k \geq 2:
\quad L_p^{(k)}[i, j] = \left\{ \begin{array}{ll}
\max_{i,j} \{ L_p^{(k-1)}[i, j], \max_h \Pr(H_i \rightarrow H_h H_i) \} \\
M_g^{(k-1)}[i, j], (i, j) \in I_{k-1} - I_{k-2} \\
M_g^{(k-1)}[i], i \in I_{k-1} - I_{k-2} \\
I_k = I_{k-1} - \argmax \{ L_p^{(k-1)}[i], I_{k-1} \}.
\end{array} \right.
\end{align*}

A symmetrical result can be obtained for the computation of array \( L_s \). The details are omitted. As a final remark, we observe that the computation described above can be carried out in an amount of time \( O(|N|^2 |P_3|) \).

Computation of factors for island upper-bounds

Finally, we 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 \( \Sigma^* \); these
probabilities will be employed in the next section in the computation of syntactic island upper-bounds. Let us define an \([N] \times [N]\) array \(L_i\) such that (see Figure 1)

\[
L_i[i,j] = \text{Prm}(H_i(\Sigma^*H_j\Sigma^*)).
\]  

(10)

In this case too, we use a tabular method and organize the computation in such a way that the method converges to \(L_i\) after no more than \([N]\) iterations; the recursive relations are a straightforward adaptation of the relations studied for array \(L_p\):

(i) for \((i,j) \in \mathcal{I}_0, \mathcal{I}_0 = \{(h,k) \mid 1 \leq h,k \leq [N]\}:

\[
L_i^{(1)}[i,j] = \max \{ \max \{ \text{Pr}(H_i \rightarrow H_hH_j)M_p[h], \}
\]

\[
\max \{ \text{Pr}(H_i \rightarrow H_HH_j)M_p[i,j] \}, \}
\]

\[
\mathcal{I}_0 = \mathcal{I}_0 - \bigcup \argmax_{1 \leq i \leq [N]} \{L_i^{(1)}[i], \mathcal{I}_0\};
\]

(ii) for every \(k \geq 2\):

\[
L_i^{(k)}[i,j] = \begin{cases} 
\max \{ L_i^{(k-1)}[i,j], \\
\max \{ \text{Pr}(H_i \rightarrow H_hH_j)M_p[h], \\
\max \{ \text{Pr}(H_i \rightarrow H_HH_j)M_p[i,j] \} \} \} , \\
(i,j) \in \mathcal{I}_{k-1}; \\
L_i^{(k-1)}[i,j], \\
(i,j) \in \mathcal{I}_0 - \mathcal{I}_{k-1}; \\
\end{cases}
\]

\[
\mathcal{I}_k = \mathcal{I}_{k-1} - \bigcup \argmax_{1 \leq i \leq [N]} \{L_i^{(k-1)}[i], \mathcal{I}_{k-1}\}.
\]

As in the previous case, the corresponding computation can be carried out in an amount of time \(O([N]^2[P_d])\).

### On-line computations

As already discussed in the introduction, we are interested in finding the probability of an "optimal" derivation in \(G_i\) of a sentence which includes, as a prefix or as an island, an already recognized word sequence \(u\). In this section we present the main result of this paper, namely an efficient computation that results in such a probability. Using the adopted notation, we will restate the problem as one of finding the probability of the "optimal" derivations of sentences in the languages \(\Sigma^*u\Sigma^*\). The studied computations make use of some expressions introduced in the previous section. A summary of the computed quantities is schematically depicted in Figure 2.

### Best derivation probabilities

In what follows, we will need the probability of the most likely derivation of a given string \(u\). Such a quantity can be computed using a probabilistic version of the Kasami-Younger-Cocke (CYK) recognizer (see for example[Aho and Ullman, 1972]) based on the Viterbi algorithm, as shown in [Jelinek and Lafferty, 1991]. The following relations describe this algorithm using the notation adopted in this paper.

Let \(u = w_1 \ldots w_n\) be a string in \(\Sigma^n\) and \(M_p(u)\) be an \([N] \times 1\) array such that

\[
M_p(u)[i] = \text{Pr}(\max : H_i(u)).
\]  

(11)

It is easy to prove, by induction on the length of \(u\), that the following recursive relation characterizes \(M_p(u)\):

\[
M_p(u)[i] = \begin{cases} 
\text{Pr}(H_i \rightarrow u), & |u| = 1; \\
\max_{n \leq |u|} \{ \text{Pr}(H_i \rightarrow H_nH_i) \\
M_p(u)[n]M_p(u(|u| - n)) \}, & |u| > 1.
\end{cases}
\]

The computation of \(M_p(u)\) requires \(O([P_d][u]^3)\) time; this is also the running time for the original CYK algorithm.

### Prefix upper-bounds

Let \(u = w_1 \ldots w_n\) be a string in \(\Sigma^n\). We define an \([N] \times 1\) array \(M_p(u)\) as follows:

\[
M_p(u)[i] = \text{Prm}(H_i(u\Sigma^*)).
\]  

(12)

The definition states that every element \(M_p(u)[i]\) equals the probability of an optimal derivation tree \(\tau_i\) whose root is labeled by \(H_i\) and whose yield includes \(u\) as a prefix. The computation of such an element can be carried out on the basis of the following observation. The derivation tree \(\tau_i\) can always be decomposed into trees \(\tau_i\) and \(\tau_{ij}\) such that \(\tau_i\) is the least subtree of \(\tau_i\) that completely dominates prefix \(u\) and \(\tau_{ij}\) belongs to \(H_i(\Sigma^*)\); this is shown in Figure 3. It turns out that both \(\tau_i\) and \(\tau_{ij}\) are the optimal trees that satisfy such a requirement. Since we already dispose of the probability of \(\tau_{ij}\), which is element \(L_p[i,j]\) defined in the previous section, we are left with the computation of probabilities of \(\tau_i\) for a given \(u\) and for every root node \(H_j\). We present some relations that can be used in such a computation.

The following \([N] \times 1\) auxiliary array is associated with a string \(u\) using the probability of a SCFG \(G_i\) in
Chomsky Normal Form:

\[ \tilde{M}_p(u)[i] = \begin{cases} 
Pr(H_i \rightarrow u), & |u| = 1; \\
\max_{0 < t < |u|, h, i} \{ Pr(H_i \rightarrow H_h H_i) M_B(t; u)[h] M_p(u([u] - t))[i] \}, & |u| > 1.
\]

Apart from the case \( |u| = 1 \), element \( \tilde{M}_p(u)[i] \) corresponds to the probability of optimal derivation trees in \( H_i(u \Sigma^*) \) such that each immediate constituent of the root \( H_i \) spans a proper substring of \( u \) (see again Figure 3). The following result, which is proved in [Corazza et al., 1991], relates array \( M_p(u) \) to arrays \( E_p(u) \) and \( L_p \). Let the array \( L'_p[i, j] = L_p[i, j], i \neq j, \) and \( L'_p[i, i] = 1, 1 \leq i \leq |N|; \) then we have:

\[ M_p(u) = L'_p \otimes \tilde{M}_p(u). \] (13)

We observe that array \( \tilde{M}_p(u) \) depends upon arrays \( M_p(u_s) \) for every proper suffix \( u_s \) of \( u \). Both arrays can then be computed recursively, first considering \( u = w_n \), then using \( M_p(w_n) \) for the computation of the arrays associated with \( u = w_{n-1} w_n \), and so on. A careful analysis of the above relations reveals that the overall computation can be carried out in time \( O(\max\{|P_d||u|^2, |\Sigma|^2|u|\}) \). This is roughly the same amount of time independently required by practical algorithms for the (partial) recognition of the string hypothesized so far.

In a symmetrical way with respect to array \( M_p(u) \), we define an \(|N| \times 1\) array \( M_i(u) \):

\[ M_i(u)[i] = Pr(H_i \rightarrow u). \] (14)

Relations very close to the one discussed above have been studied for the computation of an auxiliary array \( \tilde{M}_i(u) \) such that \( M_i(u) = L'_i \otimes \tilde{M}_i(u) \) (\( L'_i \) is the same as \( L_i \) with unitary elements on the diagonal).

**Island upper-bounds**

We conclude with the problem of computing the probability that a nonterminal \( H_i \) derives in an optimal way a string of terminal symbols that includes a given sequence \( u \) as an island. The relations reported below have been obtained using the same recursive technique that has been applied in the previous subsection.

Given the string \( u = w_1 \ldots w_n \) in \( \Sigma^* \), \( M_i(u) \) is the \(|N| \times 1\) array defined as:

\[ M_i(u)[i] = Pr(H_i(u \Sigma^* u \Sigma^*)). \] (15)

Let us consider the \(|N| \times 1\) auxiliary array \( \tilde{M}_i(u) \) defined as follows (see Figure 4):

\[ \tilde{M}_i(u)[i] = \begin{cases} 
Pr(H_i \rightarrow u), & |u| = 1; \\
\max_{0 < t < |u|, h, i} \{ Pr(H_i \rightarrow H_h H_i) M_B(t; u)[h] M_p(u([u] - t))[i] \}, & |u| > 1.
\]

Note that \( \tilde{M}_i(u) \) is recursively computed using arrays \( M_s(u_p) \) and \( M_p(u_s) \) for every proper prefix and suffix of \( u \) such that \( u_p u_s = u \). Finally, the matrix \( M_i(u) \) can be obtained as:

\[ M_i(u) = L'_i \otimes \tilde{M}_i(u). \] (16)

where \( L'_i \) is the same as \( L_i \) with unitary elements on the diagonal. The same computational requirements discussed for the prefix upper-bounds are found for the island upper-bounds.

**Discussion and conclusions**

Several proposals have been advanced in the ASR literature for the use of \( A^* \)-like algorithms, based on the integration of acoustic and syntactic upper-bounds to drive the search process toward both acoustically and syntactically promising analyses. Different degrees of approximation can be used in the computation of \( s_1(A|\sigma_p) \) in (3) depending on the available constraints on the sequences of words which can fill in the gaps. Of course, tighter constraints allow one to conceive a more informative heuristic function resulting in a more efficient \( A^* \) algorithm.

Syntactic upper-bounds have been recently proposed in [Jelinek and Lafferty, 1991] and [Corazza et al., 1991a] that can be used in ASR to find the most plausible word sequence \( \sigma \) that matches the input signal.
This requires the maximization of the probability of the set of all the possible derivation trees spanning \( \sigma \). In ASU, however, it is more interesting to find the most likely tree spanning \( \tau \), which represents the best syntactic interpretation. In fact, different syntactic interpretations can support different system’s responses.

In this paper a theoretical framework has been introduced for the computation of the latter type of syntactic upper-bounds, in case of partial analyses obtained by a monodirectional left-to-right parser or by a bidirectional island-driven parser. Motivations for the use of island-driven parsing strategies in automatic speech processing have been presented in [Woods, 1982]. In fact island-driven flexibility allows the introduction of optimal heuristics that, when used with monodirectional parsing strategies, do not guarantee admissibility.

Given the optimization function corresponding to the greatest probability of a derivation, the proposed syntactic scoring function is optimal. To see this, note that the analysis process can be intended as a search on a tree in which every internal node corresponds to a partial derivation and every leaf node corresponds to a complete derivation of a sentence in the language; for each internal node, its children represent the derivations that can be obtained from it in one parsing step. It can be shown (see [Nilsson, 1982]) that the number of nodes explored in such a search is the minimal one whenever the scoring function employed defines an upper-bound which is as tight as possible. The syntactic scoring function we have proposed is the best conceivable one: in fact, for any internal node it results in the largest value of the optimization function computed on all possible solutions that can be reached from that node. In comparison with ASR, note that the ASR scoring functions proposed in [Jelinek and Lafferty, 1991] and [Corazza et al., 1991a] are defined by the sum of the values obtained by the optimization function on the reachable solutions: therefore, in practical cases these scores are far from being the tightest ones. Unfortunately, for the optimization functions required in ASR cases, better scoring functions present serious computational problems.

As far as efficiency is concerned, two different steps must be distinguished in the computation of the studied scoring function. In a previous section some relations have been introduced that can be computed off-line; the computation requires an amount of time that is quadratic in the number of productions in \( G \). Relations introduced in a following section must be computed on-line, because they depend on the analyzed string \( u \). The best derivation probability for \( u \) can be computed in an \( O(|P|u^3) \) amount of time, while the computation of prefix, suffix and island probabilities takes \( O(\max \{|P|u^3, |p|^2|u|\}) \) time. In one-word extension of a previously analyzed string \( u \), the score updating takes an amount of time which is \( O(\max \{|P|u^2, |P|^2|u|\}) \).

Finally, the proposed framework can be straightforwardly adapted to compute upper-bounds when the number of words necessary to complete the sentence is given. In this case, upper-bounds may be closer to the right values.

Acknowledgements
The authors are indebted to Roberto Gretter for his valuable comments on this paper and for his contribution to this research.

References