A Tabular Method for Island-Driven Context-Free Grammar Parsing

Giorgio Satta and **Oliviero Stock**
Istituto per la Ricerca Scientifica e Tecnologica
38050 Povo, Trento (Italy)
e-mail: satta@irst.it

Abstract

Island-driven parsing is of great relevance for speech recognition/understanding and other natural language processing applications. A bidirectional algorithm is presented that efficiently solves this problem, allowing both any possible determination of the starting words in the input sentence and flexible control. In particular, a mixed bottom-to-top and top-down approach is followed, without leading to redundant partial analyses. The algorithm performance is discussed.

Introduction

As opposed to traditional monodirectional parsing strategies, island-driven strategies (see [Woods, 1982; Woods, 1985]) start the analysis of the input sentence from several (dynamically determined) positions within it, and then proceed outward from them in both directions.

Island-driven strategies have been long applied in speech understanding systems, starting with the early work in [Walker, 1978] to more recent ones (for instance [Giachin and Rullent, 1989]); moreover, island-driven parsing is also encouraged within the speech area by recent results for word-spotting techniques. Strong motivations for the use of island-driven strategies in speech understanding applications are presented in [Woods, 1982], based on the fact that island-driven flexibility allows the employment of optimal heuristics that, when used with monodirectional strategies, do not guarantee admissibility. Island-driven parsing is also defended in [Stock *et al.*, 1989], where the predictive power of bidirectional strategies is discussed, along with its advantages for speech understanding systems. In this perspective, another interesting application for island-driven parsing is (written) ill-formed input, in which the parser can take advantage of partial analyses surrounding an incomplete (or missing) constituent.

This paper presents an algorithm for island-driven parsing of context-free languages. The algorithm provides efficient solutions to two interesting parsing problems, namely the proliferation of (redundant) analyses caused by the bidirectional behaviour of the parser, and the interaction between bottom-up (more exactly bottom-to-top) and top-down strategies (see the discussion below).

The formal notation used throughout the paper is the conventional one in the parsing theory literature (see for example [Aho and Ullman, 1972]). A context-free grammar is a quadruple $G = (N, \Sigma, P, S)$; the r-th production in P has the form $D_r \rightarrow C_{r,1} \dots C_{r,\pi_r}$, π_r being the length of its right-hand side. Finally, a string $x_i \dots x_j$ will always denote the null-string ε for $j < i$.

Island-driven parsing

Let G be a context-free grammar and w be a string in Σ^*. Assume also that a number $M \geq 1$ of items in different positions within w are selected, to be called *islands*. We define *island-driven* any recognition/parsing strategy that starts analysis for w from the selected islands and then proceeds outward (this is adapted from the *middle-out* strategy defined in [Woods, 1985]). A particular case of island-driven strategy is the *head-driven* parsing strategy (see for example [Proudian and Pollard, 1985; Kay, 1989; Satta and Stock, 1989]). In head-driven strategies, the "starting places" of the analysis are independent of w and are determined by the grammar G, so that each production p has only one (always the same) element within its right-hand side from which partial analysis for p is "triggered". In island-driven strategies such a condition is relaxed; as will be discussed in the following, this requires additional computational effort.

Bidirectionality and island-driven strategy

In island-driven context-free parsing, the analysis of a constituent C within the input sentence can be triggered by the completion of one or more constituents (dominating the selected islands) in the right-hand side of a production for C. In [Cheatham, 1967; Aho and Ullman, 1972] a parsing strategy called *left-corner* is discussed, in which the analysis of each production p of the grammar is triggered by the presence within the input sentence of the leftmost constituent of the right-hand side of p (called the *left-corner*).[1] One can then achieve the desired parsing behaviour by weakening the left-corner condition in such a way that

[1]This strategy should not be confused with the *bottom-up* parsing strategy, in which a production p is "declared" by the parser only after all constituents in its right-hand side have been completely analyzed.

any set of symbols in the right hand-side of p can trigger the analysis of p itself. This move forces the parser to expand partial analyses for p on both sides, i.e. to analyze the input in a bidirectional fashion.

In island-driven parsing one must also deal with cases in which no island has been selected within the portion of the input where a constituent C is required by the surrounding analyses. Hence the parser must employ top-down prediction to be sure that no constituent is lost. This in turn forces either left-to-right or right-to-left analysis, depending on whether the prediction for C took place at its left or at its right boundary.

The bidirectional and the top-down monodirectional requirements for the parser present at least two problems. First of all, the capability of expanding partial analyses on both sides results in analysis proliferation, as discussed in [Woods, 1982; Satta and Stock, 1989]. More precisely, let p be a production of the form $A \rightarrow \alpha$, $|\alpha| = m$. Starting the analysis for p from a constituent in α, and given that a partial analysis for p can be expanded in two different ways at each step, there are $O(2^m)$ different ways of obtaining a complete analysis for p. Even if such a process can be carried out within $O(m^2)$ steps by using tabular (dynamic) methods (see below) this should be compared with the monodirectional methods in which the same process takes $O(m)$ steps. Furthermore, if more than one island has been selected within α, the completion of the analysis for p takes $O(m^3)$ steps, since in the worst case a partial analysis of p can be extended in $O(m)$ different ways at each step, by merging it with the already found partial analyses.

A second problem originates from the interaction between island-driven analyses and top-down analyses, which must be carried out in order to recover constituents that fall apart from the selected islands. The overall result is that, in addition to the (redundant) island-outward recognition discussed above, every constituent gets analyzed twice more by top-down prediction at its boundaries. Although in this case the order of analysis redundancy is less than the one resulting from the bidirectional requirement, in practical cases it affects parsing performance as much as the latter, because in natural language applications the average length of the production's right-hand side is not greater than three or four.

Island-covers

Tabular methods are among the most efficient algorithms for recognition/parsing of general context-free languages (see [Graham and Harrison, 1976] for an overview). These methods process an input string $w = a_1 \dots a_n$ by filling in the entries of an $(n+1) \times (n+1)$ zero-indexed matrix T so that $A \in t_{i,j}$ *if and only if* $S \overset{*}{\Rightarrow} \alpha A \beta \overset{*}{\Rightarrow} \alpha a_{i+1} \dots a_j \beta$ *and* $R_T(\alpha, \beta, w, i, j)$, where the predicate R_T represents the filtering capabilities of the algorithm at hand.

As recently pointed out in [Leermakers, 1989], almost all tabular parsing methods for context-free grammars process the input string according to a covering grammar G_c that is built on the fly from the input grammar G.[2] As an example, in the well known chart parsing techniques ([Kay, 1986]) one can see all possible edges as the nonterminals of a context-free grammar, whose binary productions express the allowed combinations between active and inactive edges (in the following this observation should be of some help to the reader who is familiar with chart parsing).

To simplify the exposition of the studied algorithm and the expression of its formal properties in the next sections, we define here the specific cover used by the method. The following definition introduces the symbols that will be used in representing partial analyses for productions of the input grammar.

Definition 1 *Let* $G = (\mathrm{N}, \Sigma, P, S)$ *be a context-free grammar. The* i-items *associated to* G *are all and only the elements defined as follows:*

(i) *for every production* $D_r \rightarrow C_{r,1} \dots C_{r,\pi_r}$ *in* P*:*

(a) $I_r^{(s,t)}$ *is an i-item,* $0 \leq s < t \leq \pi_r, (s,t) \neq (0, \pi_r)$*;*

(b) $I_r^{(0,0)}$ *and* $I_r^{(\pi_r,\pi_r)}$ *are i-items;*

(ii) *for every* $A \in \mathrm{N}, I_A$ *is an i-item.*

In the following, the symbol $I_G^{(I)}$ will denote the set of all i-items.

In the next definition we will cover an input grammar in such a way that the bidirectional requirement is met. For the sake of simplicity and without loss of generality (see final section) we assume G to be expressed in a null-production free form; moreover, we assume that only one production in G rewrites the start symbol S.

Definition 2 *Let* $G = (\mathrm{N}, \Sigma, P, S)$ *be a context-free grammar. The* island cover *(*i-cover *for short)* $G_I = (I_G^{(I)}, \Sigma, P_I, I_S)$ *for* G *is a context-free grammar whose productions* P_I *are partitioned into the following three sets.*

(i) *for every production* p_r *in* P*, the* prediction production set $P_I^{(0)}$ *contains the productions* $I_r^{(0,0)} \rightarrow \varepsilon$ *and* $I_r^{(\pi_r,\pi_r)} \rightarrow \varepsilon$*;*

(ii) *the* projection production set $P_I^{(1)}$ *is defined as follows:*

(a) *for every production* $D_r \rightarrow C_{r,1}$ *in* P*, the production* $I_{D_r} \rightarrow X_I$ *belongs to* $P_I^{(1)}$*, where* $X_I = I_{C_{r,1}}$ *if* $C_{r,1} \in \mathrm{N}, X_I = C_{r,1}$ *if* $C_{r,1} \in \Sigma$*;*

(b) *for every production* $D_r \rightarrow C_{r,1} \dots C_{r,\pi_r}$ *in* P*,* $\pi_r > 1$*, and for every* s*,* $1 \leq s \leq \pi_r$*, the production* $I_r^{(s-1,s)} \rightarrow X_{I,s}$ *belongs to* $P_I^{(1)}$*, where* $X_{I,s} = I_{C_{r,s}}$ *if* $C_{r,s} \in \mathrm{N}, X_{I,s} = C_{r,s}$ *if* $C_{r,s} \in \Sigma$*;*

(iii) *the* expansion production set $P_I^{(2)}$ *is defined as follows. For every production* $D_r \rightarrow C_{r,1} \dots C_{r,\pi_r}$ *in* $P, \pi_r > 1$*:*

[2]Let G and G_c be two CFG's such that $L(G) = L(G_c) = L$; G_c is said to be a *cover* for G if there exists a (many-to-one) function from parses of $w \in L$ in G_c to parses of w in G. This means that one can express all parses of w in G in terms of all parses of w in G_c.

(a) *for every* s, t, k, $0 \leq s < t \leq \pi_r$, $(s,t) \neq (0, \pi_r)$, $s < k < t$, *the productions* $I_r^{(s,t)} \rightarrow I_r^{(s,t-1)} X_{I,t}$, $I_r^{(s,t)} \rightarrow Y_{I,s} I_r^{(s+1,t)}$ *and* $I_r^{(s,t)} \rightarrow I_r^{(s,k)} I_r^{(k,t)}$ *belong to* $P_I^{(3)}$, *where* $X_{I,t} = I_{C_{r,t}}$ *if* $C_{r,t} \in \mathrm{N}$, $X_{I,t} = C_{r,t}$ *if* $C_{r,t} \in \Sigma$ *and* $Y_{I,s} = I_{C_{r,s+1}}$ *if* $C_{r,s+1} \in \mathrm{N}$, $Y_{I,s} = C_{r,s+1}$ *if* $C_{r,s+1} \in \Sigma$;

(b) *for every* k, $0 < k < \pi_r$, *the productions* $I_{D_r} \rightarrow I_r^{(0,\pi_r-1)} X_I$, $I_{D_r} \rightarrow Y_I I_r^{(1,\pi_r)}$ *and* $I_{D_r} \rightarrow I_r^{(0,k)} I_r^{(k,\pi_r)}$ *belong to* $P_I^{(3)}$, *where conditions similar to* (iii)a *hold for symbols* $X_I, Y_I \in (I_G^{(I)} \cup \Sigma)$, *with respect to* C_{r,π_r} *and* $C_{r,1}$.

Note that G_I is a most general bidirectional cover for G, in the sense that any symbol in G can be the triggering element for the productions whose right-hand side it belongs to, and any expansion order is possible for such an element, i.e. G_I does not privilege one particular order.

An island-driven tabular recognizer

This section presents a tabular method that recognizes the set $L(G)$, G being a context-free grammar in a null-production free form. The reader should remember that partial analyses of G's productions are represented by means of the nonterminal symbols of the cover $G_I = (I_G^{(I)}, \Sigma, P_I, I_S)$.

Outline of the algorithm

As for almost all tabular parsing methods, the fundamental step (*expand step*) of the studied algorithm consists of combining two adjacent (partial) analyses. This is done according to the cover G_I. As a general condition, the expand step inserts a symbol $A_I \in I_G^{(I)}$ in $t_{i,j}$ if, for some production $A_I \rightarrow X_I Y_I$ in $P_I^{(2)}$, the disjunction $\bigvee_{k=i}^{j}(X_I \in t_{i,k} \wedge Y_I \in t_{k,j})$ holds true. Productions in $P_I^{(1)}$ are instead used by the *init step* and the *project step*, which respectively try to trigger analyses from the selected islands or from the already found constituents that contain at least one island. Finally, the *left-* and *right-predict steps* use productions in $P_I^{(0)}$ in making top-down predictions for the requested constituents. The execution of the five steps above on an i-item in T can take place in any order. Furthermore, no restriction is made on the order in which i-items in T are processed.

The analysis proliferation problem due to the bidirectional requirement is dealt with by extra bookkeeping, as discussed in the following. Let $I_r^{(s,t)} \in t_{i,j}$ and $I_r^{(s',t')} \in t_{i',j'}$ represent within the input sentence two partial analyses of a production $p_r : D_r \rightarrow C_{r,1} \ldots C_{r,\pi_r}$ in G. We say that $I_r^{(s,t)}$ and $I_r^{(s',t')}$ overlap if $i \leq i' \leq j \leq j'$, $s < s' < t < t'$ and the two corresponding analyses share the same constituents $C_{r,s'+1} \ldots C_{r,t}$ (i.e. share the same analyses for these constituents within the input sentence). A combinatorial argument can show that no analysis proliferation arises if we prevent the formation of overlapping partial analyses.

In Figures 1 and 2 a simple example is presented (i-items in T are represented as edges in a graph).

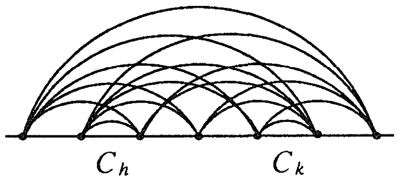

Figure 1: Partial analysis overlapping.

Figure 1 shows all possible partial analyses of some production right-hand side, that can be triggered by two (generic) constituents C_h and C_k. In Figure 2 the same case is shown, in which overlapping analysis formation has been blocked (outward expansion from C_h and C_k has been selected randomly).

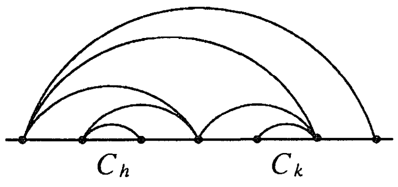

Figure 2: Subsumption blocking.

The algorithm blocks the formation of overlapping analyses by recording analysis expansions (in a matrix Q) and by blocking right expansions of an i-item that has already been expanded to the left (and the other way around). In the following, such a technique will be called *subsumption blocking*.

The problem of redundancy caused by interaction among different strategies, deserves more discussion. First of all, note that subsumption blocking preserves analysis proliferation even for interaction between island-outward and top-down left-to-right (or right-to-left) strategies.

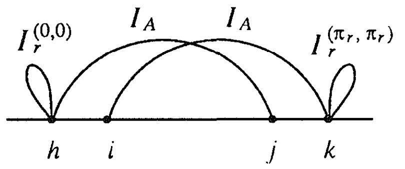

Figure 3: Prediction conflict.

The case of interaction between left-to-right and right-to-left strategies turns out to be more intricate: a simple example will be discussed in order to present the problem. In Figure 3, no island has been selected in the portion $a_{h+1} \ldots a_k$. A top-down right-to-left analysis has revealed a constituent I_A with extension $i - k$, while a top-down left-to-right analysis has revealed a constituent of the same category with extension $h - j$.

Assume that both constituents have been rewritten according to production r of the input grammar. If a top-down left prediction for I_A takes place at position i, the subsumption blocking technique will correctly prevent a left-to-right reanalysis of I_A at $i - k$. But, as a consequence, no left-to-right analysis for a potential constituent I_A at $i - j$ could ever take place. Such a constituent will then be lost, because the symmetric situation is found at position j and because of the absence of any island at $i - j$. The problem is solved by the following trick. Informally speaking, it sufficies to arbitrarily break into two substrings each portion of the input string lying between two successive islands. In the left substring, only left-to-right predictions can be made by the algorithm, while in the right substring only right-to-left predictions can take place. This technique will be called *prediction blocking*. As is shown below, prediction blocking guarantees the correctness of the algorithm and prevents the undesired interactions between left-to-right and right-to-left top-down strategies. Furthermore, such a technique presents the interesting property of filtering head-outward analyses by requesting a sort of "compatibility" with the surrounding islands (see Theorem 1 below).

The algorithm

Let $G_I = (I_G^{(I)}, \Sigma, P_I, I_S)$ be an I-cover for $G = (N, \Sigma, P, S)$. The top-down steps of the algorithm need the following definition.

Definition 3 *Two functions* l-pred *and* r-pred *from* $I_G^{(I)}$ *to* $\mathcal{P}(P_I^{(0)})$ *are defined as follows:*

$$\begin{aligned}
\textit{l-pred}(I_u^{(s,t)}) &= \{I_r^{(\pi_r,\pi_r)} \rightarrow \varepsilon \mid A_I \Rightarrow_{G_I} B_I I_u^{(s,t)}, \\
&\qquad B_I \overset{*}{\Rightarrow}_{G_I} \alpha I_r^{(\pi_r,\pi_r)}, \alpha \in (I_G^{(I)} \cup \Sigma)^*\}; \\
\textit{r-pred}(I_u^{(s,t)}) &= \{I_r^{(0,0)} \rightarrow \varepsilon \mid A_I \Rightarrow_{G_I} I_u^{(s,t)} C_I, \\
&\qquad C_I \overset{*}{\Rightarrow}_{G_I} I_r^{(0,0)} \alpha, \alpha \in (I_G^{(I)} \cup \Sigma)^*\}; \\
\textit{l-pred}(I_{D_r}) &= \textit{r-pred}(I_{D_r}) = \emptyset.
\end{aligned}$$

For notational convenience, the two following functions will also be used. Let $A_I \rightarrow X_I Y_I$ be a production in $P_I^{(2)}$. If $X_I = I_r^{(s,t)}$ then $\textit{mid}(A_I \rightarrow X_I Y_I) = (r,t)$, if $Y_I = I_r^{(s,t)}$ then $\textit{mid}(A_I \rightarrow X_I Y_I) = (r,s)$ (function *mid* is well-defined by the construction of $P_I^{(2)}$). Let $A_I \rightarrow \alpha$ be a production in P_I obtained from a production p_r in P. If $A_I = I_r^{(s,t)}$ then $\textit{index}(A_I \rightarrow \alpha) = (r,s,t)$; by extension, if $A_I = I_{D_r}$ then $\textit{index}(A_I \rightarrow \alpha) = (r, 0, \pi_r)$. The studied algorithm is reported in the following (the algorithm is not specified in its optimal form).

Algorithm 1

Input. An I-cover $G_I = (I_G^{(I)}, \Sigma, P_I, I_S)$ for a context-free grammar G (with no null-production); a string $w = a_1 \ldots a_n$ in Σ^*, $n > 0$; integers $i_k, p_k, q_k, 1 \leq k \leq M$, such that $q_{k-1} = p_k < i_k \leq q_k = p_{k+1}$ ($q_0 = 0, p_{M+1} = n$); functions *l-pred*, *r-pred*, *index* and *mid*.

Program variables. An $(n+1) \times (n+1)$ matrix T whose entries take values in $\mathcal{P}(I_G^{(I)})$; an $(n+1) \times (n+1) \times |I_G^{(I)} \cup \Sigma|$ matrix Q whose entries take values in $|P_I| \times \max_{1 \leq r \leq |P_I|}\{\pi_r\}$.

Output. Accept/reject.

Method.

```
for every $h \in \{1..M\}$ do
    comment: init step
    for every production $A_I \rightarrow a_{i_h}$ do
        $t_{i_h-1,i_h} := t_{i_h-1,i_h} \cup \{A_I\}$
        $(r,s,t) := index(A_I \rightarrow a_{i_h})$
        $Q(i_h - 1, i_h, a_{i_h}) := Q(i_h - 1, i_h, a_{i_h}) \cup \{(r,s)\}$
        $Q(i_h, i_h - 1, a_{i_h}) := Q(i_h, i_h - 1, a_{i_h}) \cup \{(r,t)\}$
    endfor
endfor
for every $A_I$ added to $t_{i,j}, 0 \leq i \leq j \leq n$ do
comment: the following steps to be applied in
comment: whichever order
    comment: left-expand step
    for every production $B_I \rightarrow X_I A_I$ do
        $(r,z) := mid(B_I \rightarrow X_I A_I)$
        $(r,s,t) := index(B_I \rightarrow X_I A_I)$
        if $(r,z) \notin Q(i,j,A_I)$ then
          for every $i' \leq i$ do
              if $(r,z) \notin Q(i,i',X_I)$, $(X_I \in t_{i',i}$
                or $X_I = a_i, i' = i - 1)$ then
                $t_{i',j} := t_{i',j} \cup \{B_I\}$
                $Q(i',i,X_I) := Q(i',i,X_I) \cup \{(r,s)\}$
                $Q(j,i,A_I) := Q(j,i,A_I) \cup \{(r,t)\}$
          endfor
    endfor
    comment: right-expand step
    for every production $B_I \rightarrow A_I X_I$ do
        $(r,z) := mid(B_I \rightarrow A_I X_I)$
        $(r,s,t) := index(B_I \rightarrow A_I X_I)$
        if $(r,z) \notin Q(j,i,A_I)$ then
          for every $j' \geq j$ do
              if $(r,z) \notin Q(j,j',X_I)$, $(X_I \in t_{j,j'}$
                or $X_I = a_{j+1}, j' = j + 1)$ then
                $t_{i,j'} := t_{i,j'} \cup \{B_I\}$
                $Q(i,j,A_I) := Q(i,j,A_I) \cup \{(r,s)\}$
                $Q(j',j,X_I) := Q(j',j,X_I) \cup \{(r,t)\}$
          endfor
    endfor
    comment: project step
    if $i < i_h \leq j$ for some $h$ then
        for every production $B_I \rightarrow A_I$ do
          $t_{i,j} := t_{i,j} \cup \{B_I\}$
          $(r,s,t) := index(B_I \rightarrow A_I)$
          $Q(i,j,A_I) := Q(i,j,A_I) \cup \{(r,s)\}$
          $Q(j,i,A_I) := Q(j,i,A_I) \cup \{(r,t)\}$
        endfor
    comment: left-predict step
    if $p_h < i < i_h$ for some $h$ then
        for every production $B_I \rightarrow \varepsilon$ in $l\text{-}pred(A_I)$ do
          $t_{i,i} := t_{i,i} \cup \{B_I\}$
        endfor
    comment: right-predict step
    if $i_h \leq j < q_h$ for some $h$ then
        for every production $B_I \rightarrow \varepsilon$ in $r\text{-}pred(A_I)$ do
          $t_{j,j} := t_{j,j} \cup \{B_I\}$
        endfor
endfor
if $I_S \in t_{0,n}$ then accept else reject
end. □
```

Formal properties

Formal proofs of the statements in this section have been omitted. To study the correctness of Algorithm 1, we need the following definition.

Definition 4 *The* subsumption relation $\mathcal{S}_I \subseteq I_G^{(I)} \times I_G^{(I)}$ *is defined as follows:*

(i) *for every* $I_r^{(s,t)} \in I_G^{(I)}$, $(I_r^{(s,t)}, I_r^{(s',t)}) \in \mathcal{S}_I$ *whenever* $s' \leq s$, *and* $(I_r^{(s,t)}, I_r^{(s,t')}) \in \mathcal{S}_I$ *whenever* $t' \geq t$;

(ii) *for every* $I_A \in I_G^{(I)}$, $(I_A, I_A) \in \mathcal{S}_I$.

The next result characterizes the set of all i-items inserted in T by Algorithm 1. Note that the correctness of the method obtains as a corollary of such a characterization.

Theorem 1 *Let* i_k, p_k, q_k, $1 \leq k \leq M$, *be integers defined as for the input of Algorithm 1. For every* $A \in I_G^{(I)}$ *and for every* i, j, $0 \leq i \leq j \leq n$, *there exist* $B, (A, B) \in \mathcal{S}_I$ *and integers* $d_1, d_2 \geq 0$, *such that* $B \in t_{i-d_1, j+d_2}$ *if and only if the following conditions hold:*

(i) *there exists a derivation* $A \overset{*}{\Rightarrow}_{G_I} a_{i+1} \ldots a_j$;

(ii) *if there exist* h, i', j', H *such that* $i_h \leq i < i_{h+1}$, $i \leq i' \leq j' \leq \min\{j, q_h\}$ *and such that* (i) *can be broken into* $A \overset{*}{\Rightarrow}_{G_I} a_{i+1} \ldots a_{i'} H a_{j'+1} \ldots a_j$ *and* $H \overset{*}{\Rightarrow}_{G_I} a_{i'+1} \ldots a_{j'}$, *then* $L \overset{*}{\Rightarrow}_{G_I} a_{i''} \ldots a_{i'} H \gamma$ *and* $H \in \{I_r^{(0,t)}, I_{D_r}\}$ *for some* $i'' < i_h$, $L \in I_G^{(I)}$, $\gamma \in (I_G^{(I)} \cup \Sigma)^*$, $1 \leq r \leq |P|$;

(iii) *if there exist* h, i', j', H *such that* $i_{h-1} \leq j < i_h$, $\max\{i, p_h\} \leq i' \leq j' \leq j$ *and such that* (i) *can be broken into* $A \overset{*}{\Rightarrow}_{G_I} a_{i+1} \ldots a_{i'} H a_{j'+1} \ldots a_j$ *and* $H \overset{*}{\Rightarrow}_{G_I} a_{i'+1} \ldots a_{j'}$, *then* $R \overset{*}{\Rightarrow}_{G_I} \gamma H a_{j'+1} \ldots a_{j''}$ *and* $H \in \{I_r^{(s,\pi_r)}, I_{D_r}\}$ *for some* $j'' \geq i_h$, $R \in I_G^{(I)}$, $\gamma \in (I_G^{(I)} \cup \Sigma)^*$, $1 \leq r \leq |P|$;

(iv) *if there exist* h *such that* $i_h \leq i < q_h < j < i_{h+1}$, *then either* $L \overset{*}{\Rightarrow}_{G_I} a_{i'} \ldots a_i A \gamma$, $A \in \{I_r^{(0,t)}, I_{D_r}\}$, *or* $R \overset{*}{\Rightarrow}_{G_I} \gamma' A a_{j+1} \ldots a_{j'}$, $A \in \{I_r^{(s,\pi_r)}, I_{D_r}\}$, *for some* $i' < i_h$, $j' \geq i_{h+1}$, $L, R \in I_G^{(I)}$, $\gamma, \gamma' \in (I_G^{(I)} \cup \Sigma)^*$. □

Condition (i) in Theorem 1 expresses the island-outward bottom-to-top behaviour of the algorithm, while the remaining conditions filter out partial analyses based on of the surrounding islands.

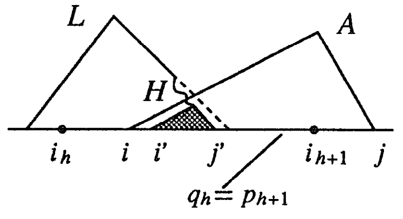

Figure 4: Condition (ii) in Theorem 1.

As an example, Condition (ii) has been depicted in Figure 4, where for simplicity we take constituents A and B to be the same. Positions i_h and i_{h+1} in the figure indicate two selected islands, while the prediction blocking technique has divided the portion $i_h - i_{h+1}$ into a left string $i_h - q_h$ suitable for left-to-right top-down prediction and a right string $p_{h+1} - i_{h+1}$ ($p_{h+1} = q_h$) suitable for right-to-left top-down prediction. Then it holds that no constituent A containing the island $a_{i_{h+1}}$ can be analyzed if its subconstituents in the portion $i_h - q_h$ are not reachable by partial analyses that include (at least) island a_{i_h}. Note that if this is the case, clearly no complete tree can eventually derive the input string. Observe also that the subsumption relation $\mathcal{S}_I$ has been used in Theorem 1 because subsumption blocking does not guarantee that the "if-part" of the statement exactly holds for an i-item A in T, but only for an i-item B that "includes" the partial analysis represented by A.

To conclude the present section, computational complexity issues are addressed. The following result assumes the *Random Access Machine* as the model of computation.

Theorem 2 *Let* $f(G_I, w)$ *be the running time of Algorithm 1. Then* $f(G_I, w) \in O(|I_G^{(I)}|^{\frac{3}{2}} |w|^3)$. □

It is important to observe that $|I_G^{(I)}| = O(|G|^2)$, G being the input grammar from which the cover G_I is derived. Therefore our algorithm runs in time $O(|G|^3 |w|^3)$, while the well-known Earley algorithm [Earley, 1970] for left-to-right recognition of context-free grammar runs in time $O(|G|^2 |w|^3)$. However, this worsening in performance seems a limitation of general bidirectional strategies.

Discussion

As already noted in the natural language parsing literature (see [Woods, 1982]) the main fraility of bidirectional parsing is (partial) analysis redundancy. A solution proposed in [Kay, 1989] for head-driven parsing consists of fixing a privileged order of expansion for each production, which amounts to reducing the size of set $I_G^{(I)}$. The use of a strategy specified by the grammar writer causes obvious problems for parsing applications that need dynamic control strategies, such as those that employ syntactic and acoustic scoring. Furthermore, given that words to be chosen as starting islands can fall more than one in the same constituent in an unpredictable way, it is not clear how to accomplish island-driven strategies through static order expansion. Algorithm 1 allows the use of dynamic strategies, while subsumption blocking prevents (redundant) analysis proliferation.

Although bidirectional, the general approach to parsing found in [Pereira and Warren, 1983] cannot mimic island-driven strategies in a direct way, due to the lack of top-down capabilities. The mixed bottom-to-top top-down strategy adopted by Algorithm 1 should not be confused with the more familiar bottom-up strategies with top-down filtering found in the natural language parsing literature (see [Wiren, 1987]).

In Algorithm 1 in fact, some (not preselected) constituents are entirely analyzed in a top-down fashion, if they do not include any islands. Also this may cause proliferation of analyses: to the best of our knowledge, this problem has not been addressed in the literature. Algorithm 1 proposes a simple solution that does not imply any computational overhead. Other than parsing, the mixed strategy proposed here can be of some relevance for text generation algorithm design, improving techniques derived from bidirectional bottom-to-top parsing methods (see [van Noord, 1990]). Furthermore, due to the (dynamic) tabular technique employed by the algorithm, parallel parsing applications can also be considered to improve known bottom-to-top algorithms (for instance [Nijholt, 1990; de Vreught and Honig, 1989]).

Two final technical notes. It is easy to exhibit an algorithm for the construction of rightmost (or leftmost) parse, under the hypothesis that elements in $I_G^{(I)}$ are stored in T along with a list of pointers to those entries that caused the new entry to appear. The assumption regarding the null-production free form of the input grammar can be dropped. In fact, the most general case can be solved by employing the same technique used in [Graham and Harrison, 1976] to handle null-productions; this simply requires a slight redefinition of the cover G_I.

As a conclusion, the flexibility of the studied algorithm seems very promising for applications in stochastic context-free grammar parsing and automatic speech understanding.

References

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

[Cheatham, 1967] T. E. Cheatham. *The Theory and Construction of Compilers*. Computer Associates, Wakefield, Mass., 1967.

[de Vreught and Honig, 1989] J. P. M. de Vreught and H. J. Honig. A tabular bottom-up recognizer. Report 89-78, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1989.

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

[Giachin and Rullent, 1989] E. P. Giachin and C. Rullent. A parallel parser for spoken natural language. In *Proc. of the 11 th IJCAI*, pages 1537–1542, Detroit, MI, 1989.

[Graham and Harrison, 1976] S. L. Graham and M. A. Harrison. Parsing of general context free languages. In *Advances in Computers*, volume 14, pages 77–185. Academic Press, New York, NY, 1976.

[Kay, 1986] M. Kay. Algorithm schemata and data structures in syntactic processing. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*, pages 35–70. Kaufmann, Los Altos, CA, 1986.

[Kay, 1989] M. Kay. Head-driven parsing. In *Proceedings of the Workshop on Parsing Technologies*, pages 52–62, Pittsburgh, PA, 1989.

[Leermakers, 1989] R. Leermakers. How to cover a grammar. In *Proc. of the 27 th ACL*, pages 135–142, Vancouver, British Columbia, Canada, 1989.

[Nijholt, 1990] A. Nijholt. The CYK-approach to serial and parallel parsing. Memoranda Informatica 90-13, Department of Computer Science, University of Twente, Twente, The Netherlands, 1990.

[Pereira and Warren, 1983] F. C. N. Pereira and D. H. D. Warren. Parsing as deduction. In *Proc. of the 21 th ACL*, pages 137–144, Cambridge, MA, 1983.

[Proudian and Pollard, 1985] D. Proudian and C. Pollard. Parsing head-driven phrase structure grammar. In *Proc. of the 23 th ACL*, pages 167–171, Chicago, IL, 1985.

[Satta and Stock, 1989] G. Satta and O. Stock. Head-driven bidirectional parsing: A tabular method. In *Proceedings of the Workshop on Parsing Technologies*, pages 43–51, Pittsburgh, PA, 1989.

[Stock *et al.*, 1989] O. Stock, R. Falcone, and P. Insinnamo. Bidirectional charts: A potential technique for parsing spoken natural language sentences. *Computer speech and Language*, 3(3):219–237, 1989.

[van Noord, 1990] G. van Noord. An overview of head-driven bottom-up generation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language*, chapter 6, pages 141–165. Accademic Press, New York, 1990.

[Walker, 1978] D. E. Walker, editor. *Understanding Spoken Language*. North-Holland, New York, 1978.

[Wiren, 1987] M. Wiren. A comparison of rule-invocation strategies in parsing. In *Proc. of the 3 rd EACL*, pages 226–233, Copenhagen, Denmark, 1987.

[Woods, 1982] W. A. Woods. Optimal search strategies for speech understanding control. *Artificial Intelligence*, 18:295-326, 1982.

[Woods, 1985] W. A. Woods. Language processing for speech understanding. In F. Fallside and W. A. Woods, editors, *Computer Speech Processing*, chapter 12, pages 305–334. Prentice Hall, Englewood Cliffs, NJ, 1985.