Head-Driven Bidirectional Parsing: A Tabular Method

Giorgio Satta (°)(°°), Oliviero Stock (°°)

(°) Università di Padova, via Belzoni 7, 35131 Padova, Italy

(°°) Istituto per la Ricerca Scientifica e Tecnologica, 38050 Povo, Trento, Italy

1. Introduction

Tabular methods for context-free language analysis [Graham and Harrison, 1976, Graham *et al.*, 1980], and in particular Earley's Algorithm [Earley, 1970], can be considered a major reference for natural language parsing. Even if independently conceived, Earley's Algorithm constitutes the basis for Chart parsing [Kay, 1980, Kaplan, 1973].

One basic aspect of known tabular methods, i.e. that the analysis proceedes monodirectionally, is a relevant limitation, that, although reasonable for artificial languages, seems reductive for natural language. A strong reason for a bidirectional approach within natural language analysis is that modern theories of grammar emphasize the role of a particular element inside each constituent (phrase), called the head; this element carries categorial as well as thematic information about other elements within the constituent. It turns out that the acceptability and the general skeleton of each constituent, crucially depend on such information. More concretely, a number of possible partial interpretations would be pruned out earlier, on the basis of functional information attached to the head, resulting in greater efficiency.

Some recent works in the framework of Chart parsing [Steel and De Roeck, 1987, Stock *et al.*, 1989] have pointed out the importance of bidirectionality for natural language analysis. Another work that deals with some form of bidirectionality [Bossi *et al.*, 1983] can be found in the formal language literature, though the analysis given there presupposes Chomsky normal form grammars.

In this paper we shall introduce a tabular method coinceived for bidirectional context-free parsing, discuss some of its relevant properties and through an example give an idea of how the algorithm works.

2. Definitions

Assume a context-free grammar $G=(N, \Sigma, P, S)$, where N is the finite set of all non-terminal symbols, Σ is the set of terminal symbols, P is a finite set of productions, and $S \in N$ is the start symbol. L(G) represents the language generated by the grammar G. The productions in P are numbered from 1 to $|P|$[1], and are all of form $D_p \rightarrow C_{p,1} \dots C_{p,\pi(p)}$,

[1] The notation $|P|$ here indicates the cardinality of set P.

where π is a function defined over the set $\{1 \ldots |P|\}$ and that takes values in the set Z^+ (the set of positive integers). In the following, the natural number p often will be used instead of the production associated with it. Without loss of generality, here it is assumed that the grammar G is in *e-free* form (see [Aho and Ullman, 1972:147]); a more general formulation of the algorithm does not lead to the loss of the properties shown here.

A function τ is defined over the set $\{1 \ldots |P|\}$ and it takes values in Z^+. This function indicates, for every production p in P, a position in the right–hand side of the production, occupied by a symbol in $N \cup \Sigma$. This position is called the *head position,* and the corresponding symbol is said to be *in the head position* for production p. Every time, during the analysis, a symbol is recognized that is in head position for some production p, the presence of the symbol D_p relative to production p is then locally hypothesized.

DEFINITION 2.1

A *state* is defined to be any quadruple $[p, ldot, rdot, m]$, with $1 \leq p \leq |P|$, $0 \leq ldot < rdot \leq \pi(p)$, $m \in \{ -, lm, rm\}$.

The component p indicates the corresponding production in P; the components *ldot* and *rdot* represent two distinct positions, one after the other, in the right-hand side of production p. The component m is a simple indicator: $m=lm$ indicates that the value of *ldot* cannot be further diminished, even if greater than zero, while $m=rm$ indicates that the value of *rdot* cannot be increased further, even if it is less than $\pi(p)$. Note that, by definition, one limitation excludes the other. The value "-" is used for the indicator m in the absence of both the limitations just described. The use of the index m, as it will be shown, prevents the duplication of "partial analyses" for substrings of w. Every state $s=[p, ldot, rdot, m]$ may be understood to be a partial analysis relative to production p, for which the constituents $C_{p,ldot+1} \ldots C_{p,rdot}$, belonging to the right-hand side, have been recognized. In the following, for convenience, the states will often be referred to in these terms. The symbol I_s denotes the set of all states.

DEFINITION 2.2

The function F is defined as follows:

$F: N \cup \Sigma \rightarrow \mathcal{P}(I_s)$

$F(X)=\{s=[p, ldot, ldot+1, -] \mid X=C_{p,ldot+1}, \tau(p)=ldot+1\}$.

The set F(X) therefore contains all the states indicating partial analyses of productions in which the symbol X occupies the head position.

DEFINITION 2.3

An equivalence relation Q in $I_s \times I_s$ is defined so that for two generic states $s=[p, ldot, rdot, m]$ and $s'=[p', ldot', rdot', m']$, sQs' holds if and only if $p=p'$, $ldot=ldot'$ and $rdot=rdot'$.

3. The Algorithm

A *recognizer* is an algorithm capable of accepting a generic string $w \in L(G)$ for a particular grammar of interest G. In all other cases, the string w is refused. A *parser*, instead, is an algorithm that can solve the problem of whether or not w belongs to L(G) and is also able to indicate the possible derivation trees[2] for every $w \in L(G)$. In this section, a recognizer algorithm for context-free languages is presented. The use of a simple algorithm able to reconstruct the derivation trees by interpreting the recognition matrix T (see for example [Graham et Harrison 1983]) is sufficient to obtain a parser algorithm.

The algorithm uses a matrix T of size $(n+1)\times(n+1)$; each component $t_{i,j}$ of this matrix takes values in the set $\mathcal{P}(I_S)$, and is initialized with as empty set. The presentation of the recognition algorithm is preceded by a schematic illustration of the computation involved.

The algorithm inserts into the recognition matrix T each state s that indicates a partial analysis previously obtained for the generic substring ${}_iw_j$. There is a one to one correspondence between the indicies of the analyzed substring ${}_iw_j$ and the indices of the component $t_{i,j}$, in which state s has been inserted. The algorithm then processes each state, combining it with nearby states in an effort to extend the portions of the string dominated by these states. When the analysis relative to a particular state is completed (for both the right and left sides), if the constituent obtained is in a head position for some production p in P, a new partial analysis for the production p itself is inserted into matrix T. Note that the algorithm straightforwardly separates the problem of the combination of different states from the problem of control. The algorithm in fact does not specify the order in which the different states must be considered, nor in which order every single state must be expanded in the two opposing sides. To that end, the algorithm uses a variable A which takes values in the set $\mathcal{P}(I_S\times N\times N)$.

ALGORITHM 3.1

Given a context-free grammar $G=(N, \Sigma, P, S)$ in *e-free* form, let $w=a_1 \ldots a_n$, $n>0$, be an input string. Develop a recognition matrix T, of size $(n+1)\times(n+1)$, whose components $t_{i,j}$ are coindexed from 0 to n for both sides.

```
     begin
1.   for $i$ in {1 .. $n$} do
2.       for $s$ in $F(a_i)$ do
3.           add triple $e=(s, i-1, i)$ to set A only if $sQs_q$
             does not hold for any triple $e_q=(s_q,$
             $i-1, i)$
4.   while A not empty do
```

[2] A *derivation tree* D associated with a string $w \in L(G)$, is a labeled tree formed by all the productions used in the derivation of$w$, representing the correct hierarchic order.

```
5.          extract any element $e=(s, i, j)$ from the set A and
            insert state $s$ in $t_{i,j}$; apply each of the following
            procedures, in any order, to element $e$:
                  left-expander(e),
                  right-expander(e),
                  left-completer(e),
                  right-completer(e),
                  trigger(e);
6.     if $s=[p, 0, \pi(p), m] \in t_{0,n}$, for some $p \in P$ such that $D_p=S$
7.          then output(true)
8.          else output(error)
       end.
```

The five procedures mentioned above are described in the following.

PROCEDURE 3.1 *Left-expander*

Precondition The procedure is applied only when $e=(s, i, j)$ with $s=[p, ldot, rdot, m]$, $ldot>0$, $m \neq lm$.

Description The following two cases are possible.

Case 1: $C_{p,ldot} \in N$. For every $s'=[p', 0, \pi(p'), m'] \in t_{i',i}$, $i'<i$, such that $D_{p'}=C_{p,ldot}$, the state $s''=[p, ldot\text{-}1, rdot, -]$ is created and the triple $e'=(s'', i', j)$ is inserted in set A, only if $s''Qs_q$ does not hold for any state s_q in $t_{i',j}$ or for any triple $e_q=(s_q, i', j)$ in A. If at least one state s' is found with the above properties, set $m=rm$ in s.

Case 2: $C_{p,ldot} \in \Sigma$. If $C_{p,ldot}=a_i$, the state $s'=[p, ldot\text{-}1, rdot, -]$ is created and the triple $e'=(s', i\text{-}1, j)$ is inserted into set A, only if $s'Qs_q$ does not hold for any state s_q in $t_{i-1,j}$ or for any triple $e_q=(s_q, i\text{-}1, j)$ in A. If $C_{p,ldot}=a_i$, set $m=rm$ in s.

This procedure is applied only if state s can be extended leftward ($ldot>0$) and only if it has not already been extended rightward ($m \neq lm$), that is, if it is not subsumed to the right by a more updated state. There are two cases, depending upon whether the left-hand expansion symbol is a terminal symbol or not. If $C_{p,ldot}$ is a non-terminal symbol, the search proceeds to the left of state s, to any state s' (adjacent), that corresponds to a completed analysis ($ldot'=0$, $rdot'=\pi(p')$) of a constituent usable by state s ($D_{p'}=C_{p,ldot}$). If successful, the analysis is extended in correspondence with state s, including the constituent found nearby; state s then is marked with $m=rm$, since this has been subsumed on the left by a more updated state. If $C_{p,ldot}$ is, instead, a terminal symbol, and if $C_{p,ldot}=a_i$, an extension of the analyses corresponding to state s is made, including the terminal symbol a_i. Still, state s is marked with $m=rm$ for the same reasons as in Case 1. Furthermore, note that Procedure 3.1 never duplicates the triples in A, nor the states belonging to the same component of recognition matrix T.

PROCEDURE 3.2 *Right-expander*

Precondition The procedure is applied only when $e=(s, i, j)$ with $s=[p, ldot, rdot, m]$, $rdot<\pi(p)$, $m\neq rm$.

Description There are the following two cases.

Case 1: $C_{p,rdot+1}\in N$. For every $s'=[p', 0, \pi(p'), m']\in t_{j,j'}$, $j'>j$, such that $D_{p'}=C_{p,rdot+1}$, state $s''=[p, ldot, rdot+1, -]$ is created and triple $e'=(s'', i, j')$ is inserted into set A, only if $s''Qs_q$ does not hold for any state s_q in $t_{i,j'}$ or for any triple $e_q=(s_q, i, j')$ in A. If at least one state s' has been found with the properties described above, set $m=lm$ in s.

Case 2: $C_{p,rdot+1}\in\Sigma$. If $C_{p,rdot+1}=a_{j+1}$, the state $s'=[p, ldot, rdot+1, -]$ is created and the triple $e'=(s', i, j+1)$ is inserted in A, only if $s'Qs_q$ does not hold for any state s_q in $t_{i,j+1}$. If $C_{p,ldot}=a_{j+1}$, set $m=lm$ in s.

This procedure is symmetric to the *left-expander* procedure, so the explanation is omitted.

PROCEDURE 3.3 *Left-completer*

Precondition The procedure is applied only when $e=(s, i, j)$, with $s=[p, 0, \pi(p), m]$.
Description For every $s'=[p', ldot', rdot', m']\in t_{i',i}$, $i'<i$, $rdot'<\pi(p')$, $m'\neq rm$. such that $D_p=C_{p',rdot'+1}$, state $s''=[p, ldot', rdot'+1, -]$ is created and the triple $e'=(s'', i', j)$ is inserted in set A only if $s''Qs_q$ does not hold for any state s_q in $t_{i',j}$ or for any triple $e_q=(s_q, i', j)$ in A. Furthermore, set $m'=lm$ for every s' found.

This procedure is applied whenever the analysis of a constituent D_p has been completed through a state $s=[p, 0, \pi(p), m]$. It proceeds by searching leftward of state s for any adjacent state s' that has not yet been subsumed to the left ($m'\neq rm$) and is able to "expand" state s ($D_p=C_{p',rdot'+1}$). If successful, an extension of the analysis corresponding to s' is carried out, including the constituent D_p. State s' is then marked with $m=lm$, since it has now been subsumed on the right by a more updated state. Again, note that the procedure never duplicates triples in A, nor states belonging to the same component of the recognition matrix T.

PROCEDURE 3.4 *Right-completer*

Precondition The procedure is applied only when $e=(s, i, j)$, with $s=[p, 0, \pi(p), m]$.

Description For every $s'=[p', ldot', rdot', m']\in t_{j,j'}$, $j'>j$, $ldot'>0$, $m'\neq lm$, such that $D_p=C_{p',ldot'}$, state $s''=[p, ldot'-1, rdot', -]$ is created and the triple $e'=(s'', i, j')$ is inserted in set A only if $s''Qs_q$ does not hold true for any state s_q in $t_{i,j'}$ or for any triple $e_q=(s_q, i, j')$ in A. Furthermore, set $m'=rm$ for every s' found.

This procedure is symmetric to the *left-completer* procedure, so the explanation is omitted.

PROCEDURE 3.5 *Trigger*

Precondition The procedure is applied only when $e=(s, i, j)$, with $s=[p, 0, \pi(p), m]$.

Description For every $s \in F(D_p)$, insert the triple $e=(s, i, j)$ in set A only if $s \mathcal{Q} s_q$ does not hold for any state s_q in $t_{i,j}$ or for any triple $e_q=(s_q, i, j)$ in A.

The procedure is applied whenever the analysis of a particular constituent has been completed and this constituent occupies the head position in some production p. In this case a new state corresponding to a partial analysis for production p is created, including the head. Once again, note that the procedure never duplicates triples in A, nor states belonging to the same component of the recognition matrix T.

4. Some Formal Properties of the Algorithm

In this section the most interesting properties of Algorithm 3.1 are stated. For a formal proof of what follows refer to [Satta and Stock, 1989b]. Four major properties have been grouped under Invariant 4.1 below. Note that soundness and completeness for Algorithm 3.1 follow straightforwardly from statements *(i)* and *(ii)* in Invariant 4.1.

INVARIANT 4.1

(i) $s=[p, ldot, rdot, m] \in t_{i,j}$ *only if* $C_{p,ldot+1} \cdots C_{p,rdot} \overset{*}{\Rightarrow} a_{i+1} \cdots a_j$, $i<j$, $ldot+1 \leq \tau(p) \leq rdot$;

(ii) $C_{p,ldot+1} \cdots C_{p,rdot} \overset{*}{\Rightarrow} a_{i+1} \cdots a_j$, $i<j$, $ldot+1 \leq \tau(p) \leq rdot$ *only if* a quadruple $h=[h_1, h_2, h_3, h_4]$, $h_q \geq 0$, $1 \leq q \leq 4$ exists such that $s=[p, ldot-h_1, rdot+h_2, m] \in t_{i-h_3, j+h_4}$;

(iii) $s=[p, ldot, rdot, lm] \in t_{i,j}$ *only if* $s'=[p, ldot, rdot+1, m'] \in t_{i,j'}$, $j'>j$;

(iv) $s=[p, ldot, rdot, rm] \in t_{i,j}$ *only if* $s'=[p, ldot-1, rdot, m'] \in t_{i',j}$, $i'<i$.

Algorithm 3.1 allows the extension of a state to both the left and right sides. This possibility, if not carefully controlled, can lead to the duplication of an analysis, in the following way. If a state s, relative to a partial analysis for a constituent C_s, is independently extended to both sides, it would lead to the introduction of two partially overlapping states, s' and s'', for the same analysis. The completion of s' and s'' then would lead to the duplication of constituent C_s. The algorithm presented here uses the index m, associated with each state, so as to avoid partial overlapping for two (partial) analyses of the same constituent. Formally, we define the partial overlapping relation as follows.

DEFINITION 4.1 *Partial Overlapping Relation*

Two states $s=[p, ldot, rdot, m] \in t_{i,j}$ and $s'=[p, ldot', rdot', m'] \in t_{i',j'}$ are *partially overlapped* ($s \mathcal{D} s'$) iff $i<i'<j<j'$, $ldot<ldot'<rdot<rdot'$, and, furthermore, s subsumes the same constituents $C_{p,ldot'+1} \cdots C_{p,rdot}$ subsumed in s'.

Note that for two states $s=[p, ldot, rdot, m]$ and $s'=[p, ldot', rdot', m']$ such that $s \mathcal{D} s'$, it always holds that $ldot'<\tau(p) \leq rdot$. The following theorem can now be stated.

THEOREM 4.1

Algorithm 3.1 never generates two states s and s' such that $s\mathcal{D}s'$.

The following result regards space and time complexity for Algorithm 3.1. Such a result is intended for a Random Access Machine model of computation.

THEOREM 4.2

Algorithm 3.1 requires an amount of space $O(n^2)$ and an amount of time $O(n^3)$, where n is the length of the input string.

5. A Brief Example

In order to have an insight into Algorithm 3.1, an example regarding a simple computation is given here. Assume an unambiguous context-free grammar $G=(N, \Sigma, P, S)$, where $N=\{S, A, B\}$, $\Sigma=\{a, b, c, d, e\}$, and P is the production set given as follows:

1:	$S \rightarrow A\, a$,	$\tau(1)=2$, $\pi(1)=2$;
2:	$S \rightarrow B\, b$,	$\tau(2)=2$, $\pi(2)=2$;
3:	$A \rightarrow c\, A\, c$,	$\tau(3)=2$, $\pi(3)=3$;
4:	$A \rightarrow d$,	$\tau(4)=1$, $\pi(4)=1$;
5:	$B \rightarrow c\, B\, c$,	$\tau(5)=2$, $\pi(5)=3$;
6:	$B \rightarrow e$,	$\tau(6)=1$, $\pi(6)=1$.

From Definition 2.2 it follows that:

$F(A) = \{[3, 1, 2, -]\}$; $F(B) = \{[5, 1, 2, -]\}$;
$F(a) = \{[1, 1, 2, -]\}$; $F(b) = \{[2, 1, 2, -]\}$;
$F(d) = \{[4, 0, 1, -]\}$; $F(e) = \{[6, 0, 1, -]\}$;
$F(S) = F(c) = \varnothing$.

A run of Algorithm 3.1 on the string $w=cceccb$ is simplified by the following steps (the order of application for the five procedures at line 5 is chosen at random).

1) $s_1=[6, 0, 1, -]$ is inserted in $t_{2,3}$ and $s_2=[2, 1, 2, -]$ is inserted in $t_{5,6}$, by line 3;
2) $s_3=[5, 1, 2, -]$ is inserted in $t_{2,3}$ by the *trigger* procedure;
3) $s_4=[5, 0, 2, -]$ is inserted in $t_{1,3}$ and m is set to rm in state s_3, by Case 2 of the *left-expander* procedure;
4) $s_5=[5, 0, 3, -]$ is inserted in $t_{1,4}$ and m is set to lm in state s_4, by Case 2 of the *right-expander* procedure;
5) $s_6=[5, 1, 2, -]$ is inserted in $t_{1,4}$ by the *trigger* procedure;
6) $s_7=[5, 1, 3, -]$ is inserted in $t_{1,5}$ and m is set to lm in state s_6, by Case 2 of the *right-expander* procedure;
7) $s_8=[5, 0, 3, -]$ is inserted in $t_{0,5}$ and m is set to rm in state s_7, by Case 2 of the *left-expander* procedure;

8) s_9=[5, 1, 2, -] is inserted in $t_{0,5}$ by the *trigger* procedure;

9) s_{10}=[2, 0, 2, -] is inserted in $t_{0,6}$ and m is set to rm in state s_2, by the *right-completer* procedure;

10) the algorithm outputs true and then stops.

Note how the setting of the m components in states s_3 and s_6 prevents the expansion of partial analysis at both sides. Though not shown here, in more complicated cases the setting of the m components permits the *left-completer* procedure to combine a state s with the "leftward largest" partial analyses that are adjacent to the left of s, preventing once more partial analysis duplication (*vice versa* for the *right-completer* procedure).

Finally, note that in the above example Algorithm 3.1 has constructed 10 states, while a run of the classic method of Earley on the same string would have constructed 25 states. Furthermore, by defining $\tau(p)=1$, $1 \leq p \leq |P|$, Algorithm 3.1 mimics the *left-corner* strategy as stated in [Wirén, 1987], resulting in the construction of 17 states for the same analysis.

6. Final Remarks

This paper discusses a parsing algorithm that extends bidirectionally the classic tabular methods for context-free language analysis. The algorithm is given for *e-free* form context-free grammars, but it is not difficult to extend it to the general case, for example by employing the same technique used in [Graham *et al.* 1980] in the treatment of empty categories.

With respect to natural language parsing, the presented tabular method is compatible with the well known "Active Chart Parsing" technique, as pointed out in [Satta and Stock 1989a]. Finally, the extension to Earley's Algorithm proposed in [Shieber 1985] for parsing complex-feature-based formalisms, could be equally applicable to the presented approach.

References

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

[Bossi *et al.*, 1983] Bossi, A., N. Cocco, and L. Colussi. A divide-and-conquer approach to general context-free parsing. *Information Processing Letters*, 16 - pp. 203-208, 1983.

[Earley, 1970] Earley, J. An Efficient Context-Free Parsing Algorithm. *Communications of the ACM*, 13(2), pp. 94-102, 1970.

[Graham and Harrison, 1976] Graham, S. L., and M. A. Harrison. Parsing of General Context Free Languages. *Advances in Computers*, pp. 77-185 , Academic Press, New York, 1976.

[Graham *et al.*, 1980] Graham, S. L., M. A. Harrison, and W. L. Ruzzo. An Improved Context-Free Recognizer. *ACM Toplas*, 2(3), pp. 415-462, 1980.

[Kaplan, 1973] Kaplan, R. M. A General Syntactic Processor. In: (R. Rustin, ed) *Natural Language Processing*, pp. 193-241, Algorithmics Press, New York, New York, 1973.

[Kay, 1980] Kay, M. Algorithm Schemata and Data Structures in Syntactic Processing. *Tecnical Report CSL-80* Xerox-PARC, Palo Alto, California, 1980.

[Satta and Stock, 1989a] Satta, G., and O. Stock. Formal Properties and Implementation of Bidirectional Charts. *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989.

[Satta and Stock, 1989b] Satta, G., and O. Stock. Bidirectional Context-Free Language Parsing within a tabular approach, submitted for publication.

[Shieber, 1985] Shieber, S. M. Using Restriction to Extend Parsing Algorithms for Complex-Feature-Based Formalisms. *Proceedings of the 23rd Conference of the Association for Computational Linguistics,* Chicago, Illinois, 1985.

[Steel and De Roeck, 1987] Steel, S. and A. De Roeck. Bidirectional Chart Parsing. *Proceedings of AISB-87,* Edinburgh, Scotland, 1987.

[Stock *et al*., 1989] Stock, O., R. Falcone, and P. Insinnamo. Bidirectional Charts: a Potential Technique for Parsing Spoken Natural Language. Computer Speech and Language, 3, 1989.

[Wirén, 1987] Wirén, M. A Comparison of Rule-Invocation Strategies in Context-Free Parsing. *Proceedings of the 3rd Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark, 1987.