USING BIDIRECTIONAL CHART PARSING FOR CORPUS ANALYSIS

A. Ageno H. Rodriguez [1]

Universitat Politècnica de Catalunya Dpt. LSI
email: ageno@goliat.upc.es, horacio@lsi.upc.es

Abstract

Several experiments have been developed around a bidirectional island-driven chart parser. The system follows basically the approach of Stock, Satta and Corazza, and the experiments have been designed and performed with the purpose of examining several ways of improvement: basic strategyt of the algorithm (pure island-driven vs mixed island-driven/bottom-up approaches), strategies for extending the islands, strategies for selecting the initial islands, ways of scoring the possible extensions, etc. Both the system and the results obtained up to date are presented in this paper.

1. Introduction.

Recent interest in parsing great amounts of non-restricted corpus has led to the use of robust parsing technologies in which any text can be parsed, frequently not completely but producing a parse forest of eventually partially attached constituents. Using a chart-based method and introducing in it some features (bidirectionality, probabilities) to improve its flexibility seems a reasonable approach.

In this paper we present a preliminary attempt to use an island-driven probabilistic chart-based approach to parse unrestricted text. The paper discusses the basic guidelines of such approach: selection of islands, parsing algorithm, probability estimation for different island growing possibilities, etc. An environment for testing such options along these dimensions has been built and several experiments have been carried out with encouraging results.

[1] This research has been partially funded by the Spanish Research Department (CICYT) and inscribed as TIC92-0671.

The organisation of the paper is as follows: After this introduction, section 2 provides a background on the basic techniques used in our approach, starting with chart parsing and focussing in island driven chart parsing. Section 3 is devoted to a detailed presentation of our proposal. Section 4 accounts for the initial experimental evaluation. Section 5, finally, states some conclusions and further lines of research.

2. Background.

2.1. Chart Parsing.

As it is well known, Chart Parsing was proposed by Martin Kay (see [Kay, 85]) as an extension of the Well Formed Substring Tables (WFST) that, in turn, were proposed as a way of avoiding the inadequacies of previous parsing mechanisms, specially the ATNs. A Chart is a directed graph where nodes correspond to the separations between the words of the sentence to be parsed, so that, if the length of the sentence is *n*, then the chart will own *n+1* nodes. Edges correspond to constituents partially or totally built. If an edge linking nodes *i* and *j* is labelled with a category *C*, this means that the portion of the sentence between nodes *i* and *j* is a derivation of *C* in case *C* is totally built or a prefix of *C* in the other case.

Chart Parsing consists, basically, of a dynamic incremental construction of the Chart structure corresponding to the sentence to be parsed. There are two kinds of edges, active and inactive edges. Active edges correspond to still not completely built constituents (i.e. to the partial application of a rule) while inactive edges correspond to totally built constituents (i.e. to the completion of a rule). Active edges are, thus, expectations or predictions while inactive edges are stated facts.

Chart Parsing was at first proposed for dealing with CFG although the formalism was further extended to cover unification grammars. The labelling of the edges is indicated by means of dotted rules. A dotted rule is simply a CF rule where a dot appears in the right part of the rule separating those constituents already derived from the rest of daughters. An edge can be implemented, thus, as a triple consisting of the origin node, the end node and a dotted rule. For instance, assuming that a production *A --> BCD* belongs to the grammar, the following edge: *<i, j, A --> BC.D>* links nodes *i* and *j* and its label indicates that on trying to derive from the nonterminal *A* applying the rule *A --> BCD*, the first two components have already been derived (of course, when dealing with unification grammars the symbols appearing in the dotted rules are not atomic but complex ones, frequently feature structures).

One of the known advantages of Chart Parsing is the flexibility of the formalism for allowing the implementation of different parsing strategies. There is a basic combination rule that is in

charge of extending the active edges when a valid continuation (or even completion) is enabled by an existing inactive edge. But we can add other rules in order to implement the top-down, the bottom-up or other less conventional strategies.

The combination rule can be stated as follows: "If the chart contains two contiguous edges, the first one active and the other one inactive, being the latter a valid continuation of the former, i.e. being labelled the latter with the category appearing just after the dot in the label of the first edge, then a new edge can be built and attached to the chart, with origin in the origin of the active edge and end in the end of the second edge. The label of this edge will be the same as the label of the active edge with the dot shifted one position to the right. The new edge can be considered as active or inactive depending on the position of the dot":

<i, j, A --> u.Bw> (active)
<j, k, B --> v.> (inactive)

will produce, if the combination rule is applied,

<i, k, A --> uB.w>

The bottom-up strategy can be implemented by the following rule: " Each time an inactive edge is added to the chart, *<i, j, A --> u.>*, a new active edge *<i, i, B --> .Aw>* must be added to the chart for every rule B --> Aw existing in the grammar".

The top-down strategy can be implemented by the following rule: " Each time an active edge is added to the chart, *<i, j, A --> u.Bv>* , a new active edge *<j, j, B --> .w>* must be added to the chart for every rule *B --> w* existing in the grammar".

It is easy to see that using the combination rule and one of the top-down or bottom-up rules (or a clever combination of both) will result in a valid parsing strategy.

When dealing with unification grammars, the Chart Parsing Algorithm remains basically the same although some modifications must be performed to the rules.

The combination rule is changed in the following way: "If the chart contains the edges *<i, j, A --> u . B_1 w>* and *<j, k, B_2 --> v. >* and assuming that the unification of B_1 and B_2 produces B_3, then the following edge must be added: *<i, k, A --> u B_3 . w>*.

In the same way the bottom-up rule becomes: "If an inactive edge *<i, j, B_1 --> w . >* is added to the chart, then for every rule in the grammar *A --> B_2 u*, so that B_1 and B_2 are unifiable with unifier B_3, then the edge *<i, i, A --> . B_3 u>* must be added to the chart".

On the other side, the top-down rule will be changed into: "If an active edge <*i, j, B --> w . A1 u*> is added to the chart, then for every rule in the grammar *A2 --> v*, so that *A1* and *A2* are unifiable with unifier *A3*, then the edge <*j, j, A3 --> . v*> must be added to the chart"

Although the usual way of applying chart parsing is a unidirectional approach, i.e. the input string is analysed from left to right and the application of the rules is made by consuming the right part of the rule from left to right too, alternative approaches have been proposed that take profit of the flexibility of the formalism we have noted before. Two different alternatives that have been heavily investigated recently are the head-driven chart parsing and the island-driven chart parsing. Both methods can be defined as bidirectional chart parsing formalisms.

We will next address the first of these extensions and leave the island-driven approach, closer to our proposal, for next section. There are, however, some unified approaches than address the bidirectional chart parsing methods as a whole. For instance, in [Satta & Stock 1992] a formal framework for dealing with bidirectional tabular parsing of nondeterministic context-free languages is proposed. The method, which is based on covering grammars ("easy to process" grammars that cover the general form input grammars), is applied in two general algorithms for head-driven parsing and island-driven parsing.

The concept of head-driven parsing is quite intuitive: when parsing, you should start by trying to find the main word, as this will tell you most about what else to look for (and not only at the initial stage). This idea has been linguistically realised in several linguistic theories starting with the Head Grammars (HG) and ending, for the moment, in the Head Driven Phrase Structure Grammars (HPSG). Some interesting comments motivating head-driven approaches can be found in [Kay, 89].

Heads are symbols within the righthand side of the rewriting rules that bear the most relevant information in that rule (that is, the information attached to the lexical entry of the head constrains other elements that co-occur in the righthand side of the rule). It is important for carrying out the parsing process that each rule has defined its head. The task of assigning heads is not easy whenever the original grammar is not based on a Head approach; in most cases CFG or conventional Unification-based grammars, as DCG, have been created without head information and have to be enriched, in order to follow this approach. The process of attaching head information is frequently undertaken manually, sometimes using rather intuitive, not linguistically based, information; sometimes, however, the head information is not added manually but is automatically acquired.

In [Satta & Stock 1992], both the head condition and the bidirectional requirement can be accomplished by defining an appropriate cover for the input grammar (which will be the actual

input for the parser). The parser implements a generalisation of the left-corner nondeterministic parsing strategy, where partial analyses are obtained starting from every head element which is found within the input sentence (purely bottom-up), as specified by the cover; the process then proceeds by outward expansion.

Worst case performance of this algorithm is proven to be $O(|G|^2 n^3)$. However, redundant analysis (due to the expansion of partial analyses on both sides) are eliminated in the implementation by means of the so called *subsumption blocking* technique, thus accomplishing better average case performance.

This proposal follows a generalisation of the left-corner fixed strategy. Other possibilities can arise when trying to generalise other left to right strategies. [Nederhof, Satta, 1994] discusses several approaches for adapting CFG parsing strategies to head-driven parsing. The approaches presented are the following:

TD (Top down parsing)	==> HTD (Head driven top down)
LC (Left corner)	==> HC (Head Corner)
PLR (Predictive LR)	==> PHI (Predictive Head Inward)
ELR (Extended LR)	==> EHI (Extended Head Inward)
LR	==> HI

The proposed algorithms are increasingly deterministic, i.e, the search trees have a decreasing size, and therefore simple realisations, such as backtracking, are increasingly efficient. Nederhof and Satta propose a uniform approach based on different kinds of items that must provide enough information for controlling the process.

A new formalism, the "Extended Head Grammar", is proposed and one of the algorithms discussed so far is adapted to this new formalism. The Extended Head Grammar is a formalism in which for each rule not only the head is marked but a complete head tree (a binary tree) is attached to the right part of the rule. The last algorithm (HI) is adapted to this extension (leading to GHI). The authors claim for easy adaptation of the other algorithms.

[Sikkel, den Akker, 1993], present as well some improvements over the basic technique proposed by Satta and Stock. Their parser implements a tabular bottom-up approach, very similar to the basic one but enhanced with top-down prediction.

As the head-driven approach is regarded as a heuristic, rather than a fail-safe principle, worst-case behaviour is specially taken into account (as prediction usually speeds up a parser but may slow it down in pathological cases). Without prediction the HC parser has a time complexity of

$O(|G| r n^3)$, r being the size of the longest production. The use of prediction in the worst case adds a factor |N|, N being the number of nonterminal symbols, yielding a time complexity of $O(|N||G|r n^3)$. The obtained worst-case complexity is claimed to be optimal.

A final line for extending such approaches is the possible application to Unification-based grammars. As we said before, an important issue regarding the head-driven approach is the need of identification of the head of every rule (of course in the case of lack of the relevant linguistic knowledge). If the constituents are not reduced to simple atoms, i.e. if they own complex information, this information could be used in order to restrict the combination possibilities with other constituents. For instance, if the verb owns lexical information as subcategorisation patterns, selectional restrictions, etc. which could be used for constraining its arguments or complements, it is clear that whatever its superficial position is, it should be treated before the other constituents. [Verlinden,93] proposes an interesting system following this line as well as several mechanisms for improving the efficiency. The most interesting of these mechanisms is the use of feature structures as top-down filter between the left part of the rule and its head.

2.2. Island-Driven Parsing.

This strategy consists of starting the analysis from several dynamically determined positions within it (the islands, symbols that have been associated the highest degree of likelihood), and then proceed outward from them in both directions: island-driven flexibility allows the employment of optimal heuristics that, when used with monodirectional strategies, would not guarantee admissibility.

To a certain extent, this strategy can be considered as a generalisation of the head-driven strategy, in which the head may be any element in the rule (depending on the chosen islands). But as opposed to head-driven parsing, one must also deal with cases in which no islands have been selected within the portion of the input where a certain constituent is required by the surrounding analyses. Therefore, the parser must obligatorily employ top-down prediction to be sure that no constituent is lost.

As in the case of head-driven parsing, both the bidirectional and the top-down monodirectional requirements can be accomplished by defining an appropriate CFG that covers the input grammar, as stated in [Satta & Stock 1992].

In the general case, island-driven parsing requires substantially more computational effort than head-driven one, as its worst case performance is proven to be $O(|G|^3 n^3)$. However, as

explained in [Satta & Stock 1992], analysis proliferation (this time due not only to the expansion of partial analyses on both sides, but also to collision of partial analyses when more than one island is selected within the same constituent in the input sentence, and to interaction between island-driven analyses and top-down analyses) is again avoided by applying *subsumption blocking* as well as imposing certain restrictions to the recognition process. Therefore, average case performance improves substantially.

3. Overall description of our approach.

Our work is based on the above stated idea introduced by [Satta & Stock '92]: the use of a chart-based formalism to perform bidirectional parsing of context-free languages, so that bottom-up parsing is carried out starting from different dynamically chosen position(s) in the sentence: the islands. A probabilistic version of such island-driven strategy is also dealt with, our approach being based on an incremental evaluation of the probabilities associated to the left or right extension of the islands.

Three main lines are being followed. Firstly, different strategies for the selection of those elements of the sentences to be regarded as islands are being considered. The strategy to be followed by the algorithm is considered as another interesting point. Finally, a third line of work is the estimation of the necessary probabilities. Obviously, these three lines are closely interrelated though described separately in the next three sections.

3.1. Selection of islands.

The strategies range from the selection of all those non-ambiguous words (when dealing with categorised but non tagged corpus), to the consideration of parsed nominal groups [Voutilainen '94], and include also the treatment of proper names [Coates-Stephens '92], punctuation signs [Briscoe '94] or even collocation analysis when dealing with the special case of parsing of dictionary definitions. Apart from these strictly syntactic criteria, other ones could be considered, such as semantic criteria (it seems intuitive that if islands were those items presenting certain significant semantic features this could be definite in order to prune the analysis process), or the obvious case when dealing with speech processing, in which the islands would be those fragments of speech which have been completely understood.

Once the islands selected, the effects of other factors in performance are being studied. These factors include:

a) the number of islands: out of the set of islands constructed according to the chosen criterion, we have observed that there is an optimum number of islands, so that

increasing this number does not improve efficiency of the parsing process; whether this depends on the particular strategy being employed or not is still a point to be more deeply studied.

b) the order in which the islands are dealt with. In principle, probabilities will help to discriminate (see section 3.3 below), though hybrid methods are not discarded.

3.2. Basic algorithm.

The main idea is to implement a flexible approach allowing the customisation of certain features so that the performance of the algorithm for different kinds of corpora and/or grammars can be easily evaluated for the different strategies. As a matter of fact, the algorithm is implemented in CLOS, and the object oriented features of the language have been used to structure the general strategies described in section 2 as a chart parsers hierarchy in which the purely bottom-up and top-down charts represent the first stage, and the head-driven and the island-driven a second one, inheriting from both the ascendant and descendent strategies to implement respectively the conventional analysis and the prediction.

Currently only a single strategy (explained below) is being tested for the island-driven parser, starting a third stage of the hierarchy which will be composed by the different possible strategies inside the island-driven class (although these strategies might imply the adoption of a more general strategy from a certain point of the analysis...). As selection of the islands is not completely systematic, part of the sentence may remain uncovered (leaving untreated gaps) if a purely bottom-up approach were followed. Therefore, a phase of top-down prediction has to be introduced. In principle we are trying to delay prediction as much as possible, and to restrict it to those positions immediately adjacent to islands (though as already mentioned, several strategies are being considered as regards the relationship between bottom-up and top-down phases).

It is important to notice that our goal is to find just the first analysis (in case there are several ones), even though it doesn't have to coincide with the most probable.

An important issue to be considered is the relationship between: 1) the first analysis obtained from our approach; 2) the most probable one, according to the probability of the derivation tree, computed from the probabilities of the rules of the grammar, and 3) the most likely one, according to linguistic considerations.

The main characteristics of the basic algorithm will be stated next. It must be noticed first that there is a slight variation as regards to the charts notation used in monodirectional parsing. In

left to right parsing, edges where denoted by the rule being employed, in whose righthand side a dot separated the part already covered from the one still not derived (so that in inactive edges the dot appeared after the last symbol of the rhs of the rule). In bidirectional parsing, though two dots are necessary to denote the part of the rule already covered, as we can start parsing from any point of it. For example, an active edge *A --> B . C . D* has still to be extended to the right in search of a category *D*, as well as to the left in search of a category *B*. In case such extensions could be performed, we would have an inactive edge, whose dots would appear respectively at the beginning and the end of the rhs of the rule (*A --> . B C D .* in our example).

Currently the algorithm works by following an agenda-based approach. A priority queue, implemented as a heap, is used to deal with the idea of choosing the most probable island, to be extended in the most probable side. In fact, the sorting criterion for the heap will always be a real number representing a probability attached in a way or another to each chart edge. Two different instances of heap are currently used by the algorithm, though with identical type of contents: the "extension heap" and the "prediction heap". An element of any of both heaps consists of a bidirectional chart edge (either active or inactive for the extension heap, always active for the prediction one), a direction field indicating whether the edge must be extended/used for prediction to the left or to the right, and a probability field stating the probability of extension/fruitful prediction of the edge in question to the indicated direction. The current algorithm consists of a loop composed by two stages:

1) A purely bottom-up phase which operates with the "extension heap" as agenda. It extends the bidirectional chart edges contained in the heap and in turn may add new elements to it, always according to the attached probability. In the first step of this phase, only those inactive edges representing islands are taken into account. The order in which the extension of the existing islands to the possible sides will be carried out is therefore determined by the computed probabilities (though once the process started up, new elements may be added with higher probabilities so that the extension of certain islands may be delayed).

2) Whenever the first phase does not lead to a complete analysis, a prediction phase is started. It uses a "prediction heap" which will be updated at the beginning of every step of this type with only those active edges adjacent to a gap (and not already used in a previous prediction phase), always according to a computed probability for each edge and direction. Therefore, a "coverage" structure must be maintained storing which elements of the sentence form a part of an island. This second phase lasts until coverage is incremented (i.e. one of the islands grows a word to one side), which is when we will go back to the first stage, presumably with a non-empty extension heap.

The prediction phase may add new elements to the extension heap, never to the prediction one,

as once an element chosen from which to launch a prediction, the process goes on creating (and directly treating) new prediction edges originated from this one until either a combination with an existing edge is accomplished or no new prediction edges can be built. It is in the latter case that a new element must be taken from the heap. In the former, the new edge(s) got as a result of the combination is not expanded bottom-up as in a conventional upward chart operation. Instead, it is just combined with those active edges which originated the prediction itself. This way, we take advantage of the info got in the downward process of prediction not to create spurious edges. The ascending process goes on, combining resultant edges with the prediction ones as soon as these new formed edges keep on being inactive ones.

In order to illustrate the operation of the two phases of this algorithm, the contents of the two heaps during the parsing of the extremely simple sentence "the cat eats fish" is shown next. The sentence is parsed using the small grammar G1 (see Appendix 1), and two versions of parsing are presented. In the first one, no prediction is necessary, while in the second one, two steps of prediction have to be performed. For the sake of simplicity, the words in the sentence have been assigned a single part of speech (shown in brackets for each word):

"the(*Det*) cat(*N@*) eats(*Vt*) fish(*N@*)"

Whose parse tree for G1 would be:

S -->	NP -->	D	-->	DET	-->	THE
		N	-->	N@	-->	CAT
	VP -->	V	-->	VT	-->	EATS
		NP	-->	N@	-->	FISH

As mentioned before, each element of the heap is composed by:

- The score assigned to this extension/prediction (not exactly equivalent to the probability)
- The direction of extension/prediction.
- The type of edge.
- The edge itself.
- The position of the sentence where the edge starts.

1) First version: the chosen islands are all the elements in the sentence.

Only an extension step is necessary to complete the analysis, in which the following edges are added to the extension heap (presented in the order in which they are processed):

(1.3 (L INACTIVE) N@ --> CAT . 1)
(1.3 (R INACTIVE) N@ --> CAT . 1)
(1.3 (L INACTIVE) N@ --> FISH . 3)
(1.3 (R INACTIVE) N@ --> FISH . 3)
(1.3 (L INACTIVE) N --> . N@ . 1)
(1.3 (L INACTIVE) N --> . N@ . 3)
(1.0 (R INACTIVE) NP --> . N@ . 1)
(1.0 (L ACTIVE) NP --> D . N . 1)
(1.0 (R INACTIVE) NP --> . N@ . 3)
(1.0 (R INACTIVE) VT --> EATS . 2)
(1.0 (R INACTIVE) DET --> THE . 0)
(1.0 (L INACTIVE) VT --> EATS . 2)
(1.0 (L ACTIVE) VP --> V . NP . 3)
(1.0 (L INACTIVE) VP --> . V NP . 2)
(0.6 (L ACTIVE) S --> NP . VP . 2)
(0.4 (R INACTIVE) V --> . VT . 2)
(0.6 (R ACTIVE) VP --> . V . NP 2)
(0.4 (R ACTIVE) S --> . NP . VP 1)
(0.4 (L INACTIVE) NP --> . N@ . 3)
(0.4 (L INACTIVE) NP --> . N@ . 1)
(0.3 (R INACTIVE) D --> . DET . 0)
(1.0 (R INACTIVE) NP --> . D N . 0)

2) Second version: the chosen islands are first (“the”) and third (“eats”) elements in the sentence.

In a fist step, the trace of the extension heap would be the following:

(1.0 (R INACTIVE) DET --> THE . 0)
(1.0 (L INACTIVE) VT --> EATS . 2)
(1.0 (R INACTIVE) VT --> EATS . 2)
(0.4 (R INACTIVE) V --> . VT . 2)
(0.6 (R ACTIVE) VP --> . V . NP 2)
(0.3 (R INACTIVE) D --> . DET . 0)
(1.0 (R ACTIVE) NP --> . D . N 0)

As it does not lead to any complete analysis, a prediction phase is necessary. The most probable prediction element is:

(1.0 (R NIL) NP --> . D . N 0)

from which a new inactive edge is constructed and added to the extension heap, starting up a second step, in which the following items are processed from the extension heap:

(1.0 (R INACTIVE) NP --> . D N . 0)
(0.4 (R ACTIVE) S --> . NP . VP 0)

A new prediction phase is necessary. This time, the most probable item from the prediction heap, namely,

(0.6 (R NIL) VP --> . V . NP 2)

gives rise to the first (and in this case the only) edge of category *S* spanning the whole sentence.

3.3. Probabilities.

A central point regarding island driven parsers is the way the different islands compete for extending themselves. This issue is important in the case of tabular implementations of chart parsers (e.g. Earley-like parsers) but is crucial in the case of nondeterministic parsing as island-driven parsing is. The three Knowledge Sources for guiding this process are:

- The grammar, i.e. the CF rules together with the probability attached to each one.
- The likelihood of the different islands currently built.
- The extension and contents of the different gaps (if any) that appear between the islands.

Two issues must be addressed regarding the process of growth of the islands:

- How to determine the probabilities that must be attached to the rules of the grammar.
- How to decide the most likely action for extending the islands, that is, which island must be extended first?, in what direction must it be extended?, what rule must be applied for carrying out such extension?

We will now consider in turn such two issues.

To begin with, probabilities attached to the grammar rules are determined by means of conventional methods "inside/outside-based" [Lari & Young '90]. Basically, the Inside-Outside algorithm ([Baker,79]) tries to optimise the parameters (the probabilities) of the rules in order to locally (not necessarily globally) maximise the likelihood of a training corpus.

The computation of probabilities associated to the extension of islands either to the left or to the right represents a problem less more treated in the literature. Several alternatives are being regarded, depending on the scope of the information to be considered, ranging from a local approach which only takes into account the immediate left/right context of each island, to a global one in which all the islands and gaps are taken into account together with the possible top-down predictions (at least the global expectation that the initial symbol of the grammar must appear as the root of a parse tree covering completely the input string).

As mentioned in section 3.2, two kinds of probabilities must be computed:

1) For the first phase, the probability of bottom-up extension of the edges, both to the left and to the right. The computation of these probabilities will obviously differ from active to inactive edges.

2) In the prediction phase, probabilities are also calculated for the eventual left/right prediction attached to each island: the probability of extension top-down of the relevant active edges, to the relevant directions.

Whenever the computed probability for a certain potential element of any of the heaps results to be zero, the element in question is not added to the heap and is therefore not considered anymore.

Three approaches will be considered:

- Local approach.
- Neighbouring approach.
- Global approach.

It is not clear that the most informative approach, i.e. the global one, leads to the best results (in terms of efficiency of the parsing process, that can be measured by means of the number of edges of the resulting chart when the first parse is obtained, according to the resources needed for such approach). Therefore, we plan to perform several experiments covering the three approaches.

3.3.1. The local approach.

It can also be regarded as a static approach: all the computations can be performed in advance with only the data contained in the grammar, and are stored in tables which are easily accessed

whenever a particular case is needed.

The basic probabilities needed are left-corner and right-corner probabilities, *left-corner(nt1, t1)* being the probability that a certain nonterminal *nt1* has terminal *t1* as a left corner. If *t1* is the word (or part-of-speech of the word) in position *i* of the input sentence, according to [Jelinek & Lafferty-91]'s notation we have the following equivalence:

left-corner(nt1, t1) = P(nt1<<i,i).

That is, the sum of probabilities of all trees with root node *nt1* resulting in word strings whose initial substring is *t1*. In fact in the algorithm the expression indicates the sum of the *left-corner(nt1, t)* for each terminal *t* in a list of terminals, as it is used for sentences in which any word may be ambiguous as regards its part of speech. Right-corner probability is symmetrically defined. These expressions are applied to the computation of the two necessary probabilities described above, so that:

1) For the first phase (extension), in order to compute the probability of extension to the left side of an inactive edge representing a nonterminal category *nt1*, the probability that we call *prob-rc(nt1)* is computed. This quantity is simply the sum of probabilities of those rules in which *nt1* appears as the rightmost element of the righthand side of the rule. Thus, only the grammar (not the concrete situation of the edge in the particular sentence) is considered. The probability of extension to the right of an inactive edge is symmetrically computed by means of the expression *prob-lc*.

2) For the first phase, the probability of extension of an active edge is calculated by means of the left/right-corner probability. Thus, the probability of extension to the left of an edge representing a rule whose partially covered righthand side has the form *nt -> nt1 . nt2 .* will be *right-corner(nt1,lt)*, lt being the list of terminal categories attached to the first word adjacent to the left limit of the island represented by the edge. The probability of extension of any edge to the right is computed in the same way by means of the *left-corner* expression.

3) For the second phase (prediction), we must compute the probability that the predicted nonterminal appears at that point, that is, the probability that it has the word at that point as a left corner in case we are predicting to the right or as a right corner in case we are predicting to the left. Therefore, *left-corner* and *rightt-orner* expressions are again the ones to be utilised.

In order to calculate the left and right-corner probabilities, two kinds of probabilities tables are computed:

a) *Lreachability* table: probability that a certain nonterminal *nt1* of the grammar has a certain nonterminal *nt2* as left corner. In fact, *Lreachability(nt1, nt2)* gives the value of *Ql(nt1=> nt2)* (again following [Jelinek & Lafferty-91]'s notation).

b) *Rreachability* table: probability that a certain nonterminal *nt1* of the grammar has a certain nonterminal *nt2* as right corner.

In a first approach, the computation of these tables was performed by following [Horowitz & Sahni '78] approach. That is, for the left corner relationship, nonterminals were firstly topologically sorted according to the *precedence* relation (where a nonterminal *A* is said to *precede* another nonterminal *B* whenever it has it as a left corner). Secondly, the tables were computed by calculating the transitive closure (assuming the previously computed order), by incrementally doing:

Lreachability (i,j) <- Lreachability (i,j) + Lreachability (i,k) Lreachability (k,j)*

(for loops for *k*, *i* and *j* going from 1 to the number of nonterminals in the grammar). The symmetrical process was performed in order to calculate *Rreachability* table.

The problem with this method was that in order to be able to perform both topological sorts, the one for the left-corner and the one for the right-corner relation, the grammar cannot be left-recursive nor right-recursive respectively.

Therefore, a different approach had to be adopted. [Jelinek & Lafferty-91] propose an elegant method (see, above, section 3.3.3) for calculating the *Lreachability* table. We have discarded it, however, because it involves the calculation of the inverse of a matrix with dimension equal to the number of nonterminals of the grammar and this number can be a huge figure (taking into account the size of the grammar and the need of expressing it in CNF). We prefer another approach that can take advantage of the extremely sparse nature of the involved matrices for real size grammars.

The approach we have followed is based on the solution proposed by [Ng & Tomita '91], to avoid the problems derived by massively left-recursive grammars in the computation of LR-tables, but this time applied to the computation of left and right-reachability tables and generalised not only to the massive left-recursivity but also to the right one. In this approach, the probabilities are computed by solving a linear system of equations (which represents the interdependencies among nonterminals), using the Gaussian Elimination method.

Let rAX denote a rule whose lefthand side is A and whose righthand side's leftmost element is

X, and p_{rAX} the probability of such a rule. Then the equation for the probability of left reachability of a nonterminal A to a nonterminal B, namely P_{AB}, would be:

$$P_{AB} = \sum_{X} (\sum_{rAX} p_{rAX}) * P_{XB}$$

unless A is equal to B (as the probability of reachability of an element to itself is always considered to be one).

As the coefficients matrix is quite sparse, the data structure used to store the information is a vector of dimension *number-of-nonterminals* * *number-of-nonterminals* each of whose positions contains a list of tuples *(column coefficient)*. The solution to the system of linear equations is calculated by means of the mathematic library Mapple (as mentioned using the Gaussian Elimination method, and taking advantage of the sparseness of the coefficients matrix). Probabilities are truncated to a maximum of 3 digits after the decimal point (both the input coefficients to the Mapple program and the output results from it).

A problem encountered has been that the dimension of the system is impractical, once non toy grammars are used. These grammars do not even need to be real-size ones to give problems, for example we first got into trouble when testing the system with a grammar composed by 96 context-free rules, containing 58 nonterminal symbols, with which a coefficients matrix of dimension 3364 * 3364 resulted.

In order to avoid the resolution of such huge systems, it was decided to consider the nonterminal interdependencies, as regards both relationships, as graphs. Therefore, we could calculate the strongly connected components, separately solve the systems for each of these components, and adequately combine the results got to obtain the solution to the whole system. These strongly connected components are composed by those elements which are completely interrelated (the contents of the reachability matrix for any pair of indexes corresponding to members of the component is 1[2]).

The method followed to compute the strongly connected components of a graph (as mentioned, in our case these graphs will correspond to the left and right reachability matrices) is composed by three steps, firstly computing the reflexive transitive closure R according to the direct reachability relation, secondly transposing this matrix (obtaining *Rt*), and finally getting the strongly connected components in the matrix result of intersecting R and Rt (the algorithm is $0(n^2)$, for a matrix of dimension n * n).

[2] Be aware that at this point we are talking about a boolean reachability matrix whose contents are either 1 or 0, and not about the matrices containing probabilities of reachability.

Once the systems of equations solved for each component, their results must be combined in order to obtain the final result. This is done in two steps, which will be described next.

First, components are topologically sorted according to the above mentioned "precedence" partial relationship. When dealing with components (and not individual symbols), we establish that a component A is said to precede a component B if any element of B (any, as whatever two elements belonging to the same strongly connected component are reachable between them) appears as a left/right-corner of any element in A (reachability-matrix[component-of-A, component-of-B] is 1). In fact, in this case sorting is performed according to the inverse of the "precedence" relationship. Notice that as A and B are different strongly connected components, it can never happen any cycle in the precedence relation (and therefore, the problem of recursivity found in the first method is completely avoided).

Secondly, the combination of results is an incremental process in which each component is dealt with following the previously calculated order, starting from the probabilities of reachability among the elements of each connected component, already known. Each step of the loop is in turn composed by four steps which go completing the probabilities (from now on, when we talk about “reachabilities” from a component to another we will mean probabilities of reachability from the elements of a component to the elements of the other, and “reachabilities” from a single element to another will be probabilities of reachability from the element to the other) :

1) In a first loop, what we call the direct reachabilities (for existing rules with an element of the current component in the lefthand side and an element of precedent components in the left/right corner) are calculated.

2) In a second loop, the elements of a first temporal matrix are calculated: the reachabilities from the current component to all the previous ones (also dealt with in the corresponding order, so there will also be a “current preceding component” each time), but passing through elements inside the current strongly connected component.

3) A third loop must be computed (and its results stored in another temporal matrix) for considering those reachabilities going through elements of components preceding the current preceding component (but taking into account that the used reachabilities from the origin to the previous preceding components are the ones in the temporal matrix).

4) A fourth and last loop considers the reachabilities passing through elements inside the current preceding component, using the ones computed in the previous loops (which have been stored in the mean time in the temporal matrices).

To illustrate the algorithm, let's see an example for the *Lreachability* computation given a very simple grammar in which nonterminals A, B, C, D, E and F form the strongly connected components shown in Figure 1 (the arrows indicate that there is at least a rule in the grammar in which a member of the origin component appears as lefthand side and an element of the arrival component is the first righthand side element).

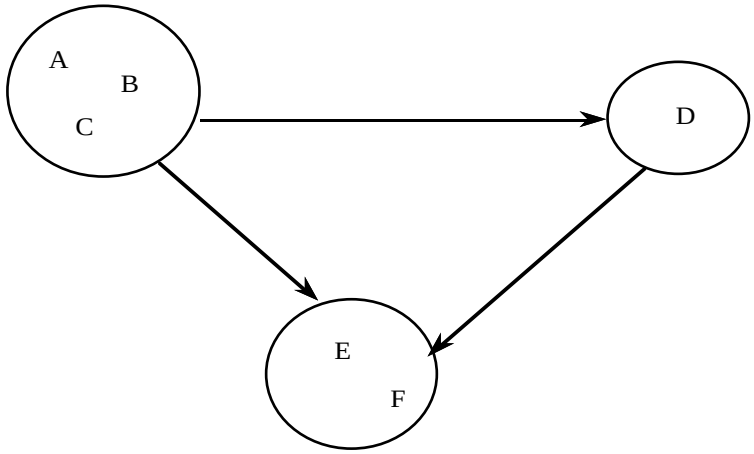

Figure 1. Single grammar reachabilities

The order of components would be: (E, F) (D) (A, B, C).

Let's suppose we are calculating the reachabilities from the (A, B, C) component ("current component") to the (E, F) one ("current preceding component"). The four steps are described next.

1) Taking into account that both the reachabilities P_{AA}, P_{AB}, P_{AC}, ... and P_{EE}, P_{EF},... are known, the definitive matrix is filled in those positions for which a rule exist (for example, if the direct relation between both components is due to a rule A -> E X of probability p_r, matrix[A,E] is assigned value pr).

2) The first temporal matrix is filled with those reachabilities from (A, B, C) to (E, F) which are not direct but have other elements from (A, B, C) as intermediate nodes (for example, temporal-matrix[A,E] is assigned matrix[A,A]*matrix[A,E] + matrix[A,B]*matrix [B,E] + matrix[A,C]*matrix [C,E]).

3) Reachabilities from component (A, B, C) to (D), which have already been computed

(as (A, B, C) also precedes (D) but (E,F) doesn't), are now taken into account to fill a second temporal matrix (for instance, temporal-matrix'[A,E] is assigned temporal-matrix[A,D]*matrix [D,E]).

4) Finally, the resultant matrix is filled by considering also the reachabilities among the elements of the component (E, F) (for example, matrix[A,E] is assigned temporal-matrix'[A,E] + temporal-matrix[A,E]*matrix [E,E] + temporal-matrix[A,F]*matrix [F,E]).

Once the *Lreachability* table computed, *left-corner* expression is computed by doing:

$$left\text{-}corner\ (nt1, lt) = \sum_{nt} Lreachability\ (nt1, nt) * unitprob\ (nt, lt)$$

(for all nonterminal symbols *nt*, where *unitprob (nt, lt)* represents the sum of probabilities that nonterminal *nt* directly derives the terminals included in *lt* list).

As usual, *right-corner* is similarly computed from the *Rreachability* table.

3.3.2. Neighbouring approach

In this approach, for extending an island we will take into account only the information provided by the neighbours, i.e. the islands and gaps immediately surrounding such island. We follow the method presented in [Corazza et al. 91].

Corazza and her colleagues present a theoretical framework for scoring partial theories (or interpretations of a sentence) in the island-driven approach. The problem is to compute the probability that a SCFG generates sequences of words (islands) separated by gaps (or portions of the input still uninterpreted). Then, the best theories already obtained can drive the analysis of the remaining gaps, by predicting words adjacent to an already recognised string and computing the probability that the first (last) word of a gap is a certain symbol (or preterminal), therefore restricting the class of possible preterminals in order to maximise the probabilities of the new extended theories.

Expressions for the computations of these quantities are presented, where the computation is based on inside/outside probability (of a nonterminal deriving a concrete string), gap probability and prefixstring probability (of a nonterminal deriving a determined string followed of a gap). The complexity evaluation for the algorithms which can be straightforwardly derived from these expressions is also stated, resulting to be cubic (in terms of the number of productions of the grammar and the length of the strings involved).

Although the framework presented by Corazza et al. is quite general, they point out that only for known-sized gaps the method is of practical application (i.e. remains in polynomic time). Fortunately this is the case of our application (we know the length of the gaps). An important limitation of Corazza´s method is the need for expressing the grammar in Chomsky Normal Form (CNF).

Corazza et al. apply their method to several cases, the most complex of which is the sequence of two islands surrounded by the corresponding gaps (three gaps in total). In our approach we will use only some of their results, namely those corresponding to the following situations:

> **gap-island.** Corresponding to the possible left extension of the first island appearing in the chart. The gap corresponds to the left unknown part of the input string.
>
> **island-gap-island.** Corresponding to the possible right extension of every island (but the last one) appearing in the chart and the possible left extension of the following island. The gap corresponds to the unknown part of the input string between the two islands.
>
> **island-gap.** Corresponding to the possible right extension of the last island appearing in the chart. The gap corresponds to the right (and last) unknown part of the input string.

In [Corazza et al. 91] detailed information about the algorithms corresponding to these cases can be found.

3.3.3. Global approach

There is, to our knowledge, no specific treatment of this case in the literature. Therefore we will try to extend some of the results reported for the case of left to right parsing, which are described next.

In [Jelinek,Lafferty, 91] the authors stated that four problems must be addressed regarding SCFG.

> 1) What is the probability $P(S \dashrightarrow w_1 w_2 \ldots w_n)$ that a grammar, with the initial symbol S, generates a given word string (sentence) $w_1 w_2 \ldots w_n$?
>
> 2) What is the most probable parse (derivation) for a given word string $w_1 w_2 \ldots w_n$?

3) What is the probability $P(S \dashrightarrow w_1 w_2 \ldots w_n \ldots)$ that a grammar, with the initial symbol S, generates a word string (sentence) whose initial substring is $w_1 w_2 \ldots w_n$?

4) Given a CFG, how should the probabilities attached to each rule be determined?

It is well known that problem 1 can be solved by means of the Inside Algorithm ([Baker,79]). Problem 4 can be solved by applying the Inside-Outside Algorithm, as we commented at the beginning of section 3.3. The Viterbi Algorithm can be used for dealing with problem 2. These three approaches involve the application (and extension) of the well known CKY Chart parser. The core of [Jelinek,Lafferty, 91] consists of solving problem 3, just in the case the grammar is expressed in CNF.

The solution proposed involves the calculation of two square matrices: P_L and Q_L with dimension equal to the number of nonterminals of the grammar.

$P_L\ (H,G)$ contains the sum of the probabilities of all the rules $H \dashrightarrow GG_i$ for all G_i . $Q_L\ (H,G)$ contains the sum of the probabilities of all the trees with root node H that produce G as the leftmost (first) nonterminal, which is what we have named the *Lreachability* matrix. P_L can be computed quite straightforwardly from the grammar, while Q_L can be computed by means of the following formula: $Q_L = P_L\ (I - P_L)^{-1}$ (or, of course, following the method described in section 3.3.1). An important fact is that these two matrices can be computed directly from the grammar and that, in spite of their rather big size, they are quite sparse and can be efficiently stored. Once these matrices computed, the calculation of the desired prefix probabilities can be carried out in an iterative, incremental way. See [Jelinek,Lafferty, 91] for details of the algorithm.

[Stolcke,94] presents a unified way of approaching the same problems claiming for better results than the previous approach. Stolcke´s proposal, not reduced to CNF as in Jelinek´s, depends heavily on the use of the Earley Algorithm for parsing. In fact the core of his system is the concept of *Earley-path*, i.e. a sequence of *Earley-states* together with the operations for the transitions between these states according to the Earley Algorithm: prediction, scanning and completion. Stolcke proposes a formula for computing the probability of an *Earley-path* and reduces the different problems previously stated to computations involving these probabilities (in fact different *Earley-paths* are considered: unconstrained, constrained, complete, starting_with_X, etc. In the end all the algorithm is reduced to the incremental construction (carried out following the parser performance) of two matrices implementing two probabilities: Forward and Inner, corresponding roughly, to the prefix and the string (inside) portions of the input at any moment of the process.

Another interesting approach is presented in [Kochman,Kupin,91]. In this case the proposal is quite different from the preceding ones. The system is applied to shift-reduce (LR) parsers allowing general (and perhaps ambiguous) CFGs (in the line of Tomita´s parser). The grammar must also be expressed in this case in CNF. The aim of the proposal is evaluating the likelihood that a collection of subtrees, that has been built so far, could be completed into a full parse tree by means of the steps that the parser is constrained to follow taking into account the possibilities for the unscanned part of the input.

The algorithm distinguishes four cases depending on whether the input string has been consumed or not, and on the existence of one or more than one subtrees already built. As in the cases of Jelinek´s and Stolcke´s, the system involves the manipulation of several matrices by an iterative incremental algorithm.

4. Experimental evaluation.

4.1. The resources.

As mentioned in the first section, this parsing system is basically oriented to the corpus analysis. Therefore several corpus, both in English and Spanish, have been compiled in order to test the system. In English, SEC and Susanne corpus are intended to be used, while in Spanish the subset of noun definitions from VOX dictionary (160.000 dictionary senses, around 1Mw) and the corpus described in [Cervell et al. 95], by now consisting of about 0.8 Mw, will be tested.

As regards grammars, a grammar for the tag sequences of CLAWS2 and SEC/Susanne provided by Ted Briscoe has been processed in order to be used for the English corpus. The grammar, originally written in the ANLT formalism, has 374 rules. It was loaded into the GDE (the ANLT environment), obtaining the expanded grammar, from which a context-free backbone (see [Briscoe & Carrol, '91]) was extracted. Thus, we obtained the resultant context-free grammar, composed by 717 context-free rules, 50 nonterminal categories and 31 terminal categories (which have been translated to the tag sequences of the corpus mentioned). As to grammars for the Spanish corpora, a derived version of the grammar described in [Castellón et al. ‘89] will be used. The grammar is written in the LSP (Sager) formalism, and is composed by 60 rules (but expressed by means of an extended BNF), with 60 nonterminal and 30 terminal symbols. This grammar has been expanded to DCG formalism, resulting in a grammar of 94 rules.

Anyway, we also intend to test our approach with other types of non-conventional grammars,

including wide coverage ones, and other ones specifically designed to test pathological cases (extremely recursive grammars, extremely ambiguous ones, ...).

4.2. The experiments.

So far two simple context-free grammars have been used to test individual sentences extracted from the [Cervell et al. 95] corpus. These grammars, G1 and G2, are showed in Appendix 1 and Appendix 2 respectively. Rule probabilities have been manually added to G1. As to G2, this grammar is the result of converting to CNF a previous grammar to which probabilities had been manually added. Our conversion algorithm takes also care of computing the probabilities of the new rules from the previous ones.

The results have been evaluated in terms of spatial cost (i.e. number of edges incorporated to the chart, either useful or spurious). The different strategies compared are the implementations of the charts in the hierarchy described in section 3.2. Specifically, purely bottom-up and top-down strategies' performance has been compared to the island-driven one. For the latter strategy, some decisions have been taken, namely:

- The initial islands have been chosen to be all those words with non-ambiguous part of speech tag.
- The probabilities have been calculated by following the local approach described in section 3.3.1.
- The strategy to be tested has been the one described in section 3.2: a loop in which expansion and prediction phases alternate until either a first analysis is found or the parser fails. Other strategies will be taken into account in further experiments: a first island-driven phase followed by a purely bottom-up phase from a certain point, the use of a certain threshold in order to control the application of the expansion-prediction loop (so that from the moment the prediction probability goes down this threshold, the parsing process begins to be bottom-up), ...

A small corpus of sentences of 10 words of average length from the [Cervell et al. 95] corpus has been parsed using:

- the bottom-up strategy (an algorithm inspired in the ANLT chart parser described by [Carrol '93]).
- the top-down strategy.
- the island-driven strategy, testing in turn the performance for different sets of islands for each sentence, namely, taking just one island from the set of non-ambiguous words, taking the whole set, or randomly choosing a set of islands (which obviously implies

the existence of ambiguous islands).

Average results for 3 different sentence lengths are shown in Table 1. Each position in the table indicates the average number of edges built for each sentence length (the sum in brackets means *number-of-inactive-edges+number-of-active-edges*) using each parsing strategy.

sentence-length	bottom-up	top-down	island-driven (one island)	island-driven (all islands)	island-driven (random islands)
5 words	184 (64+120)	163 (27+136)	126 (30+96)	171 (38+133)	223 (48+175)
10 words	458 (182+276)	329 (86+243)	293 (119+174)	318 (104+214)	492 (123+269)
15 words	569 (236+333)	477 (102+375)	329 (123+269)	360 (115+245)	404 (146+258)

Table 1.

Some immediate conclusions can be drawn from these data:

1) It is quite logical the better performance of the top-down method with respect to the bottom-up one, as the ambiguity in order to test islands performance has been introduced from the lexicon, and it is observed that the more ambiguities in it, the more the performance of the bottom-up method decreases (while in toy grammars it has been detected that when ambiguity is introduced through the grammar rules, it is the performance of the top-down method the one which worsens).

2) Islands performance is always better than bottom-up and top-down ones (obviously whenever the selection of the islands is not random), but the difference between them grows as the length of the sentence increases (which is a good signal taking into account that the method is corpus-oriented).

3) The increase of the number of islands, which causes more “bureaucracy”, worsens performance. We have just tested the extremes (one or all islands), a more accurate

study of intermediate cases has still to be carried out, as well as an analysis of which particular islands from the set (in case of not taking all) should be taken (according to position in the sentence, sort of part of speech, etc.) so as to increase performance.

5. Conclusions and future work.

A first insight into the possibility of using island-driven chart parsing for syntactic corpus analysis has been carried out, and its first results have been presented. Several items appearing in textual corpora can be considered promissory initial islands for further island-driven approaches: punctuation signs, proper names, unambiguous words, nominal phrases, etc. (in general textual corpora), identified words (in speech corpora), collocational signs (in dictionary definitions), etc.

An environment for carrying out different experiments using different data sources (corpora, grammars), different parsing strategies and different ways of computing the probabilities involved has been developed. Some experiments have been performed using the number of created edges to compare the efficiency of the various strategies. First results seem to be encouraging, moreover taking into account that the difference in performance in favour of the island-driven strategy increases as the length of the sentence grows.

Future lines of work include, among other points:

1) Confirming the initial results applying the proposed techniques to real sized grammar and big corpora, both in English and Spanish.

2) Testing different hybrid strategies including selective activation of the extension step, shifting from the island-driven to the pure bottom-up approach (depending on the expectation of growth of the islands), ways of performing the prediction step, number of islands, criteria for island selection, etc.

3) Testing the different approaches to the computation of the probabilities involved (described above).

4) Using additional methods to evaluate performance, adding efficiency measures (such as temporal complexity), and introducing accuracy considerations.

5) Testing several approaches to the identification of initial islands.

6) Proposing a probability estimation method for the rules in the grammar which is island-driven (that is, that takes into account the way the parsing process will be carried out).

7) Trying to relax, for methods 2 and 3 of probability computation, the CNF constraint.

8) Empirical evaluation of the likelyhood of the obtained parse tree compared with the most likely one (according to linguistic criteria) and the most probable one.

References

[Baker,79] Baker, J.K. *Trainable grammars for speech recognition* . Proceedings of Spring Conference of the Acoustical Society of America. Boston MA, 1979. Pags. 547-550.

[Briscoe & Carrol '91] Briscoe, T. & J.A. Carrol. *Generalised Probabilistic LR Parsing of Natural Language (corpora) using unification-based grammars*. Cambridge University, Computer Laboratory, Technical Report No. 224.

[Briscoe '94] Briscoe, T. *Prospects for Practical Parsing of Unrestricted Text: Robust Parsing Techniques*. Acquilex II Working Paper 26. March 1994.

[Carrol '93] Carrol, J.A. *Practical Unification-based Parsing of Natural Language*. PhD Thesis. Computer Lab. University of Cambridge. September 1993.

[Castellón et al. '89] Castellón, I., M.A. Martí, M. Taulé. *Diseño de una gramática para sistemas de diálogo hombre máquina* . Report LSI-89-7, 1989.

[Cervell et al. '95] Cervell, S., S. Climent, R. Placer. *Using MACO and MDS to tag balanced corpus of Spanish*.. AcquilexII Working Paper, July 1995.

[Coates-Stephens '92] Coates-Stephens, S. *The Analysis and Acquisition of Proper Names for Robust Text Understanding*. PhD. Thesis. City University, London. October 1992.

[Corazza et al. 1991] Corazza A., De Mori, R., Gretter R., Satta, G. *Stochastic Context-Free Grammars for Island-Driven Probabilistic Parsing* International Workshop on Parsing Technologies 1991.

[Horowitz & Sahni '78] Horowitz, E., S. Sahni. *Fundamentals of Computer Algorithms* . Computer Science Press, 1978.

[Jelinek,Lafferty,91] Jelinek, F; Lafferty, J.D *Computation of the Probability of Initial Substring Generation by Stochastic Context-Free Grammars* . Computational Linguistics Vol. 17, Num. 3, 1991, pags 315-323.

[Kay, 85] Kay, M. (1985) *Parsing in functional unification grammar* in *Natural Language Parsing* . D.Dowty, L. Karttunen y A.M. Zwicky eds. pages .251-278. Cambridge University Press. Cambridge.

[Kay, 89] Kay M. *Head-Driven Parsing* *International Workshop on Parsing Technologies '89*.

[Kochman,Kupin,91] Fred Kochman, Joseph Kupin *Calculating the probability of a partial parse of a sentence* . DARPA-91.

[Lari,Young,90] K. Lari, S.J. Young *The estimation of stochastic context-free grammars using the inside-outside algorithm* . Computer Speech and Language Vol. 4 page 35- 56.

[Nederhof,Satta, 94] M.J. Nederhof, G. Satta *An extended theory of head-driven parsing* ACL '94.

[Ng & Tomita '91] Ng, S. & M. Tomita. *Probabilistic LR Parsing for General Context-Free Grammars*. 2nd International Workshop on Parsing Technologies, 1991.

[Satta & Stock 92] Satta, G. & Stock, O. *Bi-Directional Context-Free Grammar Parsing for Natural Language Processing* *IRST- Technical Report #9204-11*.

[Sikkel, den Akker, 93] Sikkel, K., op den Akker, R. *Predictive Head-Corner Chart Parsing* Proceedings of IWPT '93.

[Stolcke,94] Stolcke, A. *An efficient Probabilistic Context-Free Parsing Algorithm that computes prefix probabilities* . International Computer Science Institute, TR-93-065.

[Verlinden,93] M. Verlinden *Head-corner parsing of unification grammars: a case study* Proceedings of sixth Twente Workshop on Language Technology (1993).

[Voutilainen, 94] A. Voutilainen *Designing a Parsing Grammar* Report Num. 22, 1994 Department of General Linguistics. University of Helsinki.

APPENDIX 1: Grammar G1.

S --> NP VP	1.0
VP --> V NP	0.4
VP --> Vi	0.6
V --> Vt	1.0
NP --> ProNP	0.2
NP --> D N	0.3
NP --> N PP	0.2
NP --> N@	0.3
D --> Det	1.0
N --> N@	1.0
PP --> P1 N	1.0
P1 --> P	1.0

APPENDIX 2: Grammar G2.

S --> GN EN-NOSUBJ	0.4
EN --> GN EN-NOSUBJ	0.4
EN --> VERB GN	0.18
c-43 --> GN COMPS	1.0
EN --> VERB c-43	0.12
EN --> GN VERB	0.18
c-42 --> VERB COMPS	1.0
EN --> GN c-42	0.12
c-41 --> GN COMPS	1.0
EN-NOSUBJ --> VERB c-41	0.4
EN-NOSUBJ --> VERB GN	0.6
c-40 --> VERB COMPS	1.0
EN-NOOD --> GN c-40	0.4
EN-NOOD --> GN VERB	0.6
GN --> GN2 GP	0.18
GN --> GN2 ADJS	0.18
GN --> GN1 REL	0.1
GN1 --> PRONOM	0.18
GN1 --> NOM	0.12
c-39 --> NOM	1.0
GN1 --> DET c-39	0.3
GN1 --> GN2 ADJS	0.2
GN1 --> GN2 GP	0.2
c-38 --> NOM	1.0
GN2 --> DET c-38	0.5
GN2 --> NOM	0.2
GN2 --> PRONOM	0.3
DET --> ART	0.6
c-37 --> POS	1.0
c-36 --> ART	1.0
DET --> c-36 c-37	0.1
DET --> DEM	0.1
DET --> QUANT	0.1
DET --> INDEF	0.1
c-35 --> QUE	1.0
REL --> c-35 EN-NOSUBJ	0.5
c-34 --> QUE	1.0

REL --> c-34 EN-NOOD	0.5
c-33 --> ADJ	1.0
ADJS --> c-33 ADJS	0.2
ADJS --> ADJ	0.8
c-32 --> INF	1.0
c-31 --> AUX	1.0
VERB --> c-31 c-32	0.08
c-30 --> PART	1.0
c-29 --> VHAVER	1.0
VERB --> c-29 c-30	0.08
VERB --> TV	0.56
c-28 --> NEG	1.0
VERB --> c-28 VERB1	0.2
VERB1 --> TV	0.7
c-27 --> GER	1.0
c-26 --> AUX	1.0
VERB1 --> c-26 c-27	0.06
c-25 --> INF	1.0
c-24 --> PREP	1.0
c-22 --> c-24 c-25	1.0
c-23 --> AUX	1.0
VERB1 --> c-23 c-22	0.04
c-21 --> PART	1.0
c-20 --> VHAVER	1.0
VERB1 --> c-20 c-21	0.1
c-19 --> INF	1.0
c-18 --> AUX	1.0
VERB1 --> c-18 c-19	0.1
c-17 --> INF	1.0
c-16 --> PREP	1.0
c-14 --> c-16 c-17	1.0
c-15 --> AUX	1.0
PERIF --> c-15 c-14	0.4
c-13 --> GER	1.0
c-12 --> AUX	1.0
PERIF --> c-12 c-13	0.6
c-11 --> PREP	1.0
COMPS --> c-11 GN	0.7
COMPS --> GP COMPS	0.3
c-10 --> PREP	1.0

GP --> c-10 GN	1.0
S --> GN VERB	0.18
c-9 --> VERB COMPS	1.0
S --> GN c-9	0.12
S --> VERB GN	0.18
c-8 --> GN COMPS	1.0
S --> VERB c-8	0.12
GN --> PRONOM	0.162
GN --> NOM	0.108
c-7 --> NOM	1.0
GN --> DET c-7	0.27
c-6 --> GER	1.0
c-5 --> AUX	1.0
VERB --> c-5 c-6	0.048
c-4 --> INF	1.0
c-3 --> PREP	1.0
c-1 --> c-3 c-4	1.0
c-2 --> AUX	1.0
VERB --> c-2 c-1	0.032