# Efficient Parsing for Information Extraction

**R. Basili, M.T. Pazienza, F.M. Zanzotto**
**Dipartimento di Informatica, Sistemi e Produzione,**
**Universita' di Roma Tor Vergata (ITALY)**
{basili,pazienza,zanzotto}@info.uniroma2.it

**Abstract.** Several (and successfull) Information Extraction systems have recently replaced the core parsing components with shallow but more efficient recognizers. In this paper we argue that the absence of an underlying grammatical recognizer, given the complex nature of several (non-english) languages, is a strong limitation for text processing functionalities, like those an IE system needs. We propose a robust and efficient syntactic recognizer mainly aimed to capture grammatical information crucial for several linguistic and non linguistic inferences. The proposed system is based on a novel architecture exploiting two major principles: *lexicalization* and *stratification* of the parsing process. As several linguistic theories (e.g. HPSG) and parsing frameworks (e.g. LTAG, SLTAG, lexicalized probabilistic parsing) suggest, lexicon-driven systems ensure the suitable forms of grammatical control for many complex phenomena. In our system an analysis guided by information on typical verb projections (e.g. verb subcategorization structures) is coupled with extended locality constraints (i.e. recognition of clause boundaries). Furthermore, *stratification* is also employed. A cascade of processing steps starts from chunk recognition and proceeds through clause analysis to dependency detection. Recognition of chunks allows to minimize the input ambiguity to the remaining phases. The resulting system is thus robust against ungrammatical phenomena (e.g. complex clause embedding, misspellings, unknown words). Efficiency is also retained, although ambiguous phenomena (multiple PP attachments) are recognized.

## 1 Introduction

Several (and successful) IE systems have recently replaced the core parsing components with shallow but more efficient recognizers [1, 8]. However, the absence of a grammatical recognizer, given the complex nature of several (non-english) languages, is a strong limitation for text processing functionalities, like those an IE system needs. Let us provide a sentence, extracted and translated from a financial corpus in Italian: [1]
*Assuming to have at disposal a certain budget level for an environmental recovery action, ACE s.p.a. intends to prepare the necessary plan to coordinate the following work activities, which will end in the completion of the operational implementation project.*
that exhibits a complex but very common structure in Italian texts. Typical information to be extracted from the above sentence is the named organization (i.e., *Ace*), the type of intended activity (i.e., *environmental recovery*) and a variety of specifications and participants to the core event. For example, understanding of the intended action of the *Ace* implies the recognition of the causative/agent role of the *Ace* itself in the subordinate clause
*... will end in the completion of the operational implementation project.*
None of the proposed finite state (or regular) automata, widely employed in IE [1], would be easily adapted to tasks related to such complex phenomena. Main problems posed by the complex structures of the example are strict requirements for effective IE systems:

- Several grammatical dependencies between content words are to be precisely determined and annotated;
- Dependencies among clauses are relevant to discover significant relationships between participants to the target event;
- Resolution of complex anaphoric references depends often on selectional constraints requiring many dependencies among sentence fragments to be available; in the example sentence, the *Ace* as proposer of the implementation project requires more subtle inferences on a semantic basis.

Several research works suggest that efficient and robust syntactic processing is viable through processes of decomposition of the grammatical knowledge and lexicalization [4, 7, 13]. We propose a robust and efficient syntactic recognizer, mainly aimed to capture grammatical information crucial for several linguistic and non linguistic inferences required by an application system. The approach is based on a novel architecture exploiting two major principles: *lexicalization* and *stratification* of the parsing process. In particular, the *stratification* is realized by a cascade of processing steps, and will be described in section 2.1. The adopted verb driven analysis employs a strong notion of *lexicalization*, mainly based on verb subcategorization information (Section 2.2). The system has been fully implemented in Prolog and tested over different corpora of texts.

Within an IE framework, portability and robustness of the parser are extremely important features [8]. Portability is ensured by minimizing requirements at the level of grammatical competence: no monolithic notion of grammar is adopted and tests carried out on different sublanguages did not required any specific tuning of the grammar rules. Furthermore, techniques to adapt subcategorization lexicons used by the system have been designed. Analysis of results obtained by domain specific and automatically acquired patterns of subcategorization demonstrated similar (when not improved) results when contrasted with those relying on general and manually coded lexicons (Section 3).

[1] see the Appendix for the Italian version.


ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

## 2 CHAOS: a Chunk-Oriented Syntactic Analyzer

The proposed method starts from the basic assumption that it is possible to define an interesting intermediate level between words and sentences. Usually a sentence is thought as a sequence of words, even if, in its written version, it is a sequence of characters. This because subsequences of characters are considered unambiguously as units, even when showing different behaviors in different sentences. Using the knowledge collected in a dictionary, recognizing words in stream of characters (tokenization) is usually carried out without any complete syntactical analysis of the sentence. The computational cost of gathering characters in words is linear. The idea is to build up a machinery, computationally simple as a tokenizer, able to group words of a sentence in functionally justified larger entities. For example, given the fragment in the early sentence

*... intends to prepare the necessary plan to coordinate* ,

any potential syntactic interpretation will detect the following groups of words:

```
[intends] [to prepare] [the necessary plan]
                              [to coordinate]
```

The grouping of words has been introduced as a level of sentence interpretation in [13], where the groups are called *chunks*.

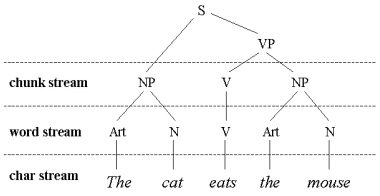



Figure 1. Interpretation levels of a simple derivation tree

The chunk level of interpretation is thus the intermediate level between words and sentences, as shown in the simpler example of fig. 1. Note that the stratification of the parsing tree induces a stratification of the underlying grammar.

The idea of chunking is successfully applied to Italian in **Chaos**, a Chunk-Oriented Analysis System for the syntactic analysis of Italian, described in [5]. A similar approach has been adopted In [14].

For the detection of inter-chunk dependencies (*icd*s), a further stratification of grammar rules is proposed in Chaos. This is mainly inspired by the crucial role played by verbs in the recognition of clauses as well as syntactic relations established between content words in a sentence. The verb is widely conceived as the grammatical head of the sentence [2]. Ambiguity is controlled via grammatical rules related to the verb functional class and specific to each verb [2, 3, 15]. Main grammatical rules for verbs used in Chaos are the verbal subcategorization frames. The constraints they impose are integrated in the recognizer of clauses and guide the determination of clause maximal and minimal boundaries (see section 2.2).

Finally, the syntactic representation of the analyzed text (a graph whose node are *chunks* and whose links are *icd*s)

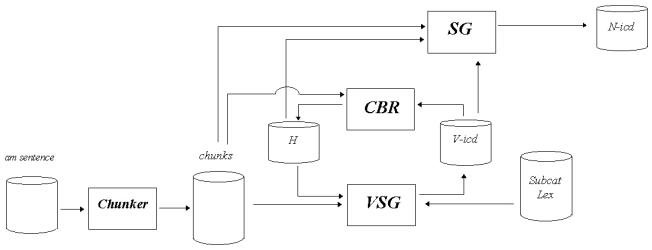



Figure 2. CHAOS: the functional architecture

is completed by recognition of another set of dependencies between chunks (e.g. post nominal prepositional phrases) by a technique already proposed in the SSA parser [11, 9]: in this phase dependencies between chunks within the discovered clauses (i.e. infra-clausal dependencies) are extracted from the source text.

In Fig. 2 the overall architecture of the system Chaos is depicted. *am sentences* are tokenized and morphologically annotated sentences, given as input to the Chunker. Chunks are used as input to the *Clause Boundary Recognition* (CBR) aiming to recognize clauses and structure them in a hierarchy (see H is Fig. 2). The recognition of clauses is integrated with a special purpose parser (Verb Shallow Recognizer, VSG) aiming to detect relations between a verb and members of its subcategorization pattern (i.e. its arguments). The interaction between the CBR and VSG provides a combined recognition of the clause hierarchy and the set of argumental dependencies of verbs. The Shallow recognizer (SG) is finally triggered by Chunks, the hierarchy H and the already extracted argumental relations (*V-icd*).

### 2.1 The Chunker

As Fig. 2 outlines, the first processing step is the recognition in the input word stream of bigger functional units, i.e. chunks. The corresponding module, called *chunker*, should be not computationally more expensive than a finite state automaton and aims to disburden later phases of bottom-up parsing.

Given a grammar and a bottom-up parsing strategy, the chunker is based on the notion of *island of non ambiguity* for the grammar. It fully characterizes the nature of those unambiguous fragments of sentences that can in fact appear in a chunk. Searching *island of non ambiguity* in a sentence can be better explained throughout an example. Consider the fragment *prepare the necessary plan to coordinate* that is ambiguous and generates the two interpretation trees for the sentence:

```
(to prepare (the (necessary plan) (to coordinate)))
(to prepare (the (necessary plan)) (to coordinate))
```

Three equivalent subgraphs are:

```
(to prepare)
(the (necessary plan))
(to coordinate)
```

These subgraphs are those islands of non ambiguity claimed to be the focus of chunk analysis. In order to find out equivalent subgraphs, a strategy less expensive than building up all

the interpretation trees has to be devised. Words belonging to equivalent subgraphs are characterized by specific sequences of morphological classes. The definition of a chunk prototype for each specific sequence characterizes all and only those fragments to be collapsed, as unambiguous. A sequence of words can thus be considered a chunk if and only if it is an instance of a chunk prototype. Note that the unambiguous islands can be entirely defined once a general grammar for the underlying language is available. The formal derivation of chunk prototypes from a grammar is proposed in [13] and developped in [5]. As subtrees are given a precise grammatical function, each chunk prototype preserves the indication of that function. This indication is called *chunk class*. Furthermore, not all the words of a chunk has the same importance. In general, there is a word that yields the meaning: it is *the potential governor*. A word is the handle by which the chunk relates to other chunks: it will be referred as *chunk handle*.

So the chunks of the example are represented by the following notation:

```
([the necessary plan / Nom / 2], plan, plan)
([to coordinate / VerInf /3 ], to, coordinate)
```

where the *type* (e.g. Nom, VerInf) are chunk annotations, while the *handle* (e.g preposition *to* in second position) and the *head* (e.g. *coordinate*, in third position) can be different.

## 2.2 Clause Boundary recognition via verb-driven analysis

In order to complete the information on sentences given by the chunking phase, inter-chunk dependencies (*icd*) must be detected by the later processing steps. Since verbs play a crucial role in determining syntactic relations between words, and thus chunks, our strategy is to look first for verbal *icd*s (i.e. those including at least a verbal chunk). A critical choice is thus to select first the more significant among these verbal *icd*s. We propose here to use verb subcategorization frames.

As sentences have more than one verb and verbs define the different sentence clauses, the recognition of argumental *icd* influences also the identification of clause boundaries. Thus, problems such as coordination and subordination between clauses are solved on the basis of verb arguments recognition.

This lexicalized approach strictly depends on the availability of accurate information on verb subcategorization frames. Two sources have been used for this information:

- a computational lexicon, LIFUV [12], manually compiled for the 5,000 most frequent Italian verbs
- a lexicon of subcategorization patterns automatically acquired from the target corpora, via a learning method based on Galois lattice theory [10].

Both these approaches have been tested and the corresponding performance is discussed in Section 3.

The recognition of the complete hierarchy of the sentence clauses is refined incrementally along with the discovery of verb argumental *icd*s for the different verbs.

Given (i) a sentence representation $S = (C, L)$ where $C$ is the sequence of its chunks and a set $L$ of dependencies (i.e. *icd*s) between chunks of $S$ and (ii) a verb $v$ of $S$, then the clause $C(v)$ of $v$ is determined. It is a sentence fragment included between a lower bound $Inf(C(v), L)$ and an upper bound $Sup(C(v), L)$, i.e.:

$$Inf(C(v), L) \subseteq C(v) \subseteq Sup(C(v), L).$$

The approximations, obtained by using the planarity constrain, are defined as:

- $Inf(C(v), L)$ is the longest subsequence of the sentence of which verb $v$ *is actually* the head, given the set $L$ of dependencies;
- $Sup(C(v), L)$ is the longest subsequence of the sentence of which verb $v$ *can be* the head.

Note that while $L$ changes (i.e. new *idc*s are captured as arguments of verb $v$), $Inf(C(v), L)$ widens. Moreover, $icd \in L$ captured for verbs other than $v$ constraint $C(v)$, so that $Sup(C(v), L)$ may be reduced accordingly.

Since the clause boundary approximation depends on the set $L$, without any other information, at the beginning ($L = \emptyset$) and for each $v$ in $S$ the only trivial information is $Inf(C(v), L) = v$ and $Sup(C(v), L) = S$. On the other hand, (in Italian sentences) finite verbs are introduced by a specific set of function words (e.g. relative pronouns, coordinations). These latter are recognized as starters of clauses and a not trivial first approximation is the main clause, a specific starter of the sentence is required (a null word in position 0). So the first approximation of the lower boundary of each clause is $Inf(C(v), L) = (s, ...., v)$, where $s$ is the starter.

A resulting hierarchy of clauses is derived from the embedding of the approximations:

$$Inf(C(v), L) \subseteq Inf(C(v'), L) \Rightarrow C(v) \subseteq C(v').$$

(The viceversa is not valid).

The derived hierarchy, $H(L)$, is then used to determine the upper bound of each clause $Sup(C(v), L)$, i.e. the maximal extension of the embedded parenthesis that does not violate the planarity.

Note that there is always a direct link from the starter to the finite verb. This implies that planarity constraints reflects on the derived approximations. Therefore, assigning open brackets to starters of the clauses and closed brackets to the finite verbs, $Inf(C(v), L)$ for all the verbs $v$ in $S$ is determined via bracket balancing. Unbalanced approximations are refused.

For example, given the fragment of the Italian sentence:
*che permetteranno, infine, di elaborare un progetto*
and using curly braces to express minimal clause boundaries ($Inf$) the structure determined at the beginning of the clause recognition phase is depicted in Fig. 3 (1).

The algorithm to collect the link established by verbs of a sentence $S$ according to their subcategorization frames is:

$L := \emptyset$
Guess initial $H(L)$.
While there are verbs not analyzed in $S$:
  Let $v$ be the depth and right-most verb not analyzed;
  Let $Sons = \bigcup_{v' \in Subclauses(v)} Inf(C(v'), L)$
  Let $Sib = \bigcup_{v' \in Siblings(v)} Inf(C(v'), L)$
  Let $L'$ be the set of new argumental *icd*s of $v$ in the segment $Sup(C(v), L) - Sons - Sib$
  $L := L \cup L'$

For example, the above algorithm will analyze first the verb *elaborare* (*define*) that will produce the argumental *icd* (i.e. direct object) *elaborare un progetto* and enlarge the *elaborare*-minimal-clause-approximation (Fig. 3 (2)).

The analysis of the verb chunk *permetteranno* (*will allow*) will consume the expected infinitival clause (i.e. (*allow to define*)), resulting in the new structure (Fig. 3 (3)).

In appendix, the full clause hierarchy for the early example sentence is reported, as well as the set of verb argumental *icd*
.

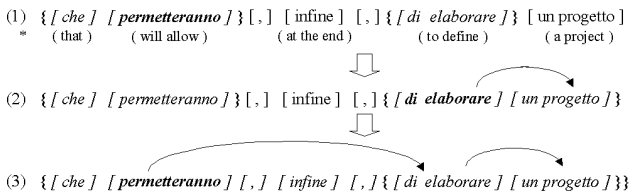



Figure 3. Clause boundary approximation example

## 2.3 Recognition of Inter chunks dependencies

Although a clause hierarchy is recognized by the previous phase further grammatical information is to be extracted from the sentence. For example information concerning non argumental verb modifiers (e.g temporal and spatial expressions) or typical noun modifiers (e.g. prepositional phrases or adjectival specifiers) has not been extracted in the previous phase. A special purpose parser is here adopted, following the approach in [11, 9]. A discontinuous grammar is applied here to the fragments belonging to the different recognized clauses. Such an infra-clausal analysis allows specific rules being defined to capture binary relations between chunks (e.g. a nominal chunk, type `Nom`, and a prepositional chunk, type: `Prep`). These relations are somewhere else called *elementary syntactic links* [9] and correspond here to possibly ambiguous *icd*. Ambiguity is controlled via a *plausibility* score [9] that is inversely proportional to the number of conflicting syntactic interpretations. Unambiguous links are thus characterized by a score equal to 1. Note that the infra-clausal analysis of the shallow grammar is consistent with the planarity constraint [4], so that the level of ambiguity is kept limited. The results of the derivation of this extended set of *icd* is reported in Appendix 1.

# 3 Performance Evaluation

The evaluation of parsing results is usually a critical task as most systems are crucially tied to constraints and features directly related with the underlying linguistic theories. In this specific perspective, the focus is on the beneficial effects that efficient parsing has on NLP applications, like IE [8]. In order to measure the performance of the CHAOS system some specific problems have been identified:

- Extensive controlled data set are not available for samples of Italian language.
- The variety of information extracted by CHAOS is hard to be compared on the basis of any existing bracket oriented metrics [6]: in fact, no exact notion of constituent (similar to those adopted in tree banks) is derived.
- Our specific interest is on specialized sublanguages, so that portability and robustness over different knowledge domains are crucial features. The suitability of the reference samples is a critical problem, even more than dimension of test sets.
- System requirements in terms of complexity of the source information (i.e. lexicons and grammars) are also relevant to evaluate portability and robustness

For these reasons we tested our system on different samples, extracted from corpora related to different domains, with differences in style and grammar. A contrastive analysis with SSA [11] has thus been applied. A specific test has been carried out to estimate differences in using hand-coded (i.e. LIFUV) and automatically derived lexicons (the induced subcategorization lexicon, [10]).

Specific evaluation metrics have been adopted. In fact, no grammatically-annotated treebank was available for the target sublanguages. For these reasons, traditional *recall* and *precision* have been estimated not over any constituent based structure, but over the set of *icd*s extracted by the system. Manually compiled test sets of *icd* have been extracted from sample sentences of the different corpora, and used as reference set (i.e. *correct_icd*). The comparison with automatically derived *icd*s resulted in the following figures:

$$precision = \frac{\#correct\_derived\_icd}{\#derived\_icd}$$
$$recall = \frac{\#correct\_derived\_icd}{\#correct\_icd}$$

We applied the test over three corpora. A collection of financial news (referred hereafter as $Sole24Ore$), a collection of technical and scientific papers on the environment ($ENEA$) and excerpts of legal documents on V.A.T. laws ($Legal$) whose features and processing times are described in Table 1. [2]

Table 1. Features figures of the three corpus

| | ENEA | Sole24Ore | Legal |
|---|---|---|---|
| #words | 1,149 | 494 | 1460 |
| #sentences | 56 | 22 | 80 |
| average #verbs per sentence | 2.14 | 3.1 | 2.2 |
| average chunk length | 1.53 | 1.44 | 1.54 |

Results obtained over the $ENEA$ and $Sole24Ore$ corpus are reported in Table 2. Data suggest that chunk analysis provides an effective grouping of words: at least two words over three appear in a non singleton chunk. The argumental *icd* are recognized with high precision although recall is low. However, this specific figure does not distinguish between argumental and other non-argumental verbal *icd*s. A lower recall is related to the higher frequency of non-argumental vs. argumental verb modifiers. Some of the latter are recognized by the SG module, as reported in Appendix. The system precision and recall are satisfactory ($> 70\%$) over the different *icd*s types, and, as the argumental *icd* catching phase is more productive, the precision of the system improves compared to pure SSA [11]. Processing speeds, measured in terms of number of words per second for the overall parsing process, is not considerably distant from a pure SSA system performance developped and run on the same platform [11].

Table 2. Performance figures on the $ENEA$ and $Sole24Ore$ corpus

| | ENEA | | Sole24Ore | |
|---|---|---|---|---|
| *icd* | Recall | Precision | Recall | Precision |
| Argumental | 30.2 % | 96.7 % | 43.6 % | 97.2 % |
| Unambiguous | 58.9 % | 88.6 % | 63.6 % | 88 % |
| All | 75.2 % | 72.1 % | 69.9 % | 72.5 % |
| Pure SSA | 49.9 % | 78.8 % | 32 % | 69.2 % |
| Processing Speed | | | | |
| *Chaos* | 99.48 w/s | | 184 w/s | |
| Pure SSA | 105.32 w/s | | 170 w/s | |

The results obtained over the *Legal* corpus are reported in Table 3. Recall and precision over this corpus have been measured against two sources lexical information. The data set

[2] *Legal*, *ENEA* and *Sole24Ore* have a size of about 320,000, 350,000 and 1,300,000 words, respectively. This is relevant for the corresponding quality and coverage of the subcategorization frame acquisition.

from the *Legal* corpus have been processed in two different experiments. In the first run (see Prec1 and Rec1 in Table 3), the VSG module has been fed with subcategorization information derived from the (hand-coded) LIFUV lexicon. Prec2 and Rec2 are obtained from a run where patterns of subcategorization automatically derived from the corpus (see [10]) have been used.

Table 3. Performance on the *Legal* corpus using two lexicons

| *Icd* | Rec1 (LIFUV) | Prec1 (LIFUV) | Rec2 (Galois) | Prec2 (Galois) |
|---|---|---|---|---|
| Argumental | 28.7 % | 88.5 % | 29.9 % | 89.1 % |
| Unambiguous | 53.6 % | 85.8 % | 54.4 % | 86.3 % |
| All | 67.4 % | 71.6 % | 68.1 % | 72.1 % |

The similar results show that using automatically acquired lexical information does not affect the system performances. Efficiency is still very high and does not show relevant changes over the three samples.

## 4 Conclusions

A new system, CHAOS, based on stratification and lexicalization of the parsing process has been described. The results of the proposed syntactic analysis are: (*i*) a set of unambiguous word chunks; (*ii*) the hierarchy of clauses recognized in the source sentence; (*iii*) a set of inter-chunk dependencies (*icd*) describing major grammatical relations between the recognized structures. Experimental results demonstrate significant improvements with respect to a simpler parsing technique (SSA [9]) over portions of real corpora in Italian. Recall and precision metrics against extensive test data improve in general, while the resulting information is richer (e.g. clause hierarchy is also built). The processing speed shows that the system can effectively be integrated in a complex NLP system. Another positive aspect is the system portability. Specific lexical information (i.e. the verb subcategorization lexicon) is a core component. Measures against different sources suggest that the automatically acquired lexicons even improve the overall system performance.

## REFERENCES


[1] 'Proceedings of the sixth message understanding conference (muc-6)', in *Columbia, MD*. Morgan Kaufmann, (1995).

[2] C.Pollard, I.A.Sag , *Information Based Syntax and Semantics, Vol. 1*, Chicago CSLI, Stanford, 1987.

[3] C.Pollard, I.A.Sag, *Head-driven Phrase Structured Grammar*, Chicago CSLI, Stanford, 1994.

[4] D.Grinberg, J.Lafferty, D.Sleator, 'A robust parsing algorithm for link grammar', in *4th International workshop on parsing tecnologies*, Prague, (1996).

[5] F.M.Zanzotto, *Una Metodologia Stratificata per la Analisi del Linguaggio Naturale: il sistema CHAOS*, Thesis dissertation, Engineering Fac., Univ. of Rome "Tor Vergata", 1997.

[6] J.Goodman, 'Parsing algorithms and metrics', in *34th Annual Meeting of ACL*, Santa Cruz, CA, (1996).

[7] Ted Briscoe John Carrol, 'Robust parsing - a brief overview', in *Workshop on robust parsing ESSLLI*, Prague, (1996).

[8] M.T.Pazienza, *Information Extraction. A Multidisciplinary Approach to an Emerging Information Technology*, number 1299 in LNAI, Springer-Verlag, Heidelberg, Germany, 1997.

[9] R.Basili, A.Marziali, M.T.Pazienza, 'Modelling syntactic uncertainty in lexical acquisition from texts', *Journal of Quantitative Linguistics*, 1, (1994).

[10] R.Basili, M.T. Pazienza, M.Vindigni, 'Corpus-driven unsupervised learning of verb subcategorization frames', number 1321 in LNAI, Heidelberg, Germany, (1997). Springer-Verlag.

[11] R.Basili, M.T.Pazienza, P.Velardi, 'A shallow syntactic analyser to extract word association from corpora', *Literary and linguistic computing*, 7, 114–124, (1992).

[12] R.Delmonte, *Linguistic and Referential Processes in Text Analaysis by computers*, UNIPRESS, Venezia, 1992.

[13] S.Abney, 'Part-of-speech tagging and partial parsing', in *Corpus-based methods in language and speech*, ed., G.Bloothooft K.Church, S.Young, Kluwer academic publishers, Dordrecht, (1996).

[14] S.Federici, S.Montemagni, V.Pirrelli, 'Shallow parsing and text parsing: a view in underspecification in syntax', in *Workshop on robust parsing ESSLLI*, Prague, (1996).

[15] Y.Schabes, 'Stocastic lexicalized tree adjoining grammars', in *COLING 92'*, Nantes, (1996).


## 5 Appendix 1 : Parsing Results for a Complex Sentence

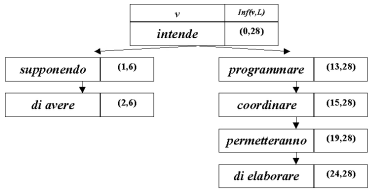
**Sentence**:

*Supponendo di avere a disposizione un certo budget economico relativo ad un intervento di risanamento ambientale, ACE s.p.a intende programmare e coordinare le fasi successive di lavoro che permetteranno, infine, di elaborare un progetto esecutivo d'intervento.*

**Chunks**:

```
[supponendo /VerGer/1][di avere /VerInf/2] [a disposizione /Prep/3][uncerto budget /Nom/4]
[economico /Agg/5][relativo /Agg/6][ad un intervento/Prep/7][di risanamento /Prep/8]
[ambientale /Agg/9][, /CongCo/10][ACE s.p.a. Nom/11][intende /VerFin/12]
[programmare /VerInf/13][e /CongCo/14][coordinare /VerInf/15][le fasi /Nom/16]
[successive /Agg/17][di lavoro /Prep/18][che /NomRel/19][permetteranno /VerFin/20]
[, /CongCo/21][infine /Avv/22][, /CongCo/23][di elaborare /VerInf/24][un progetto /Nom/25]
[esecutivo /Agg/26][d' intervento /Prep/27] [. /CongCo/28]
```

**Clause hierarchy**:




**Verbal ICDs**:

```
link(0, 12, MainClause).
link(2, 4, G_V_Ogg).
link(12, 11, G_V_Sogg).
link(12, 13, G_V_Fras).
link(15, 16, G_V_Ogg).
link(19, 20, Clausole).
link(20, 24, G_V_Fras).
link(24, 25, G_V_Ogg).
```

**Unambiguous ICDs**:

```
link(4, 5, G_Nom_Agg, plaus('1.000')).
link(4, 6, G_Nom_Agg, plaus('1.000')).
link(16, 17, G_Nom_Agg, plaus('1.000')).
link(25, 26, G_Nom_Agg, plaus('1.000')).
```

**Ambiguous ICDs**:

```
link(6, 7, G_Agg_Prep, plaus('0.500')).
link(4, 7, G_Nom_Prep, plaus('0.500')).
link(17, 18, G_Agg_Prep, plaus('0.500')).
link(16, 18, G_Nom_Prep, plaus('0.500')).
link(20, 22, G_Ver_Avv, plaus('0.500')).
link(24, 22, G_Ver_Avv, plaus('0.500')).
link(25, 27, G_Nom_Prep, plaus('0.500')).
link(26, 27, G_Agg_Prep, plaus('0.500')).
```