On Distributing the Analysis Process of a Broad–coverage Unification–based Grammar of Spanish

Montserrat Marimon Felipe

A dissertation submitted to the Institut de Ciències de l'Educació
Universitat Politècnica de Catalunya

Thesis supervisors:
Dra. Núria Bel Rafecas
Dr. Axel Theofilidis

November 2002

A en Xevi.

Acknowlegments

This thesis would not have been possible without the help, support and encouragement of many people whom I wish to give my thanks.

First of all, I'd like to thank my supervisors Núria Bel and Axel Theofilidis, who prompted me to write this thesis, for their valuable comments and guidance on this thesis.

I am thankful to all my former and current colleagues at *gilc*UB, specially, Ramon Cerdà, Núria Bel, Marta Villegas, Maite Melero and Susana García, for providing the working environment which made it possible to write this thesis. A special mention is deserved by Jordi Porta, who has always been available to help and encourage me.

I would like to express thanks to the Real Academia Española and its Departamento de Lingüística Computacional for allowing me to use `Latch`, and to Fernando Sánchez for discussions and Adelaina Fernández for reading drafts. My frequent stays in Madrid were possible thanks to Flora Ramírez, who always offered my accommodation.

From February 1998 till May 1999, I worked at the *IAI –Institut der Gesellschaft zur Förderung der Angewandten Informationsforschung e.V. an der Universität des Saarlandes*, I would like to thank all my colleagues at the institute for the friendly atmosphere I enjoyed during my stay. Also thanks to all my friend in Saarbrücken —Thierry Declerck, Sabine Lehmann, Andrew Bredenkamp, Rita Nübel and Paul Buitelaar— for making my stay so enjoyable. Special thanks to my friends at SemperStrasse —Axel, Jürgen, Lurch, Peggy and Joana— for entertaining breakfasts, lunches and dinners, and funny uno, skippo and dart evenings.

Since May 1999, I've been traveling on and off to Saarbrücken to work on my thesis. This would not have been possible if IAI had not offered me an office and a computer, and if Rita Nübel and Christoph Horschmann had not offered me accommodation.

I would also thank all the partners participating in the projects I've been involved in, Detlef Prescher for fruitful discussions and encouragement, and Lluís Padró for his comments on drafts of this thesis.

Finally, thanks to my family and Xevi for continuous support.

Contents

Chapter 1

Introduction

1.1 Goals of the Research

This thesis describes research into the development and deployment of engineered large–scale unification–based grammar to provide more robust and efficient deep grammatical analysis of linguistic expressions in real–world applications, while maintaining the accuracy of the grammar and keeping its precision up to a reasonable level.

1.1.1 Problems and Primary Proposals for Practical Broad–coverage Unification–based Grammars

Deep linguistic processing produces a complete syntactic and semantic analysis of the sentences it processes, as needed for accurate Natural Language Processing (NLP), however, it fails in producing a result when the linguistic structure being processed and/or words in the input sentences fall beyond the coverage of the grammatical resources. NLP systems with monolithic grammars, where all dimensions of linguistic information (morphological, syntactic, semantic, etc.) are interleaved, in addition, have to deal with huge search space due to several sources of non–determinism (i.e. ambiguity). This is particularly true of broad–coverage unification–based grammars.

Nevertheless, deep natural language processing techniques are a very active area of work linked to real–world applications where precise interpretation of linguistic expression is important, for instance Natural Language Interfaces (NLIs)

[Marimon *et al.* 99] and speech–to–speech Machine Translation (MT) systems [Müller & Kasper 00]. More recently, deep processing has been used in the area of industrial applications such as Information Extraction (IE) and language checking applications [Crysmann *et al.* 02], in which shallow processing has so far played a major role. Not surprisingly, a growing effort has been devoted in the NLP research community, addressing more suitable parsing and generation algorithms [Earley 70, Shieber 85, Kay 89, Noord 97, Tomita 87], unification algorithms for unification–based systems [Wroblewski 87, Tomabechi 91, Pereira 85, Karttunen & Kay 85, Kogure 90, Godden 90, Emele 91] and feature structure and type hierarchy compilation techniques [Alshawi 91, Erbach 95, Krieger 95, Simpkins *et al.* 93, Wintner & Francez 99, Miyao *et al.* 00, Porta 00]. Besides, several engineering solutions have been proposed for efficient and robust deep linguistic processing. These engineering proposals report on the deployment of filtering techniques to avoid failing unification [Kiefer *et al.* 99, Malouf *et al.* 00], the idea of using CFG filters within high–level grammars [Kiefer & Krieger 00, Kiefer *et al.* 00, Torisawa *et al.* 00] and the development of hybrid processing methods [Srinivas *et al.* 97, Ciravegna & Lavelli 97, Yoon *et al.* 99, Venkova 00, Prins & Noord 01, Grover & Lascarides 01, Crysmann *et al.* 02].

1.1.2 Primary Approaches used in this Research

This thesis describes research into the development and deployment of engineered large–scale unification–based grammar to provide more robust and efficient deep grammatical analysis of linguistic expressions in real–world applications, while maintaining the accuracy of the grammar (i.e. percentage of input sentences that receive the correct analysis) and keeping its precision up to a reasonable level (i.e. percentage of input sentences that received no superfluous analysis).[1]

In tacking the efficiency problem, our approach has been to prune the search space of the parser by integrating shallow and deep processing. We propose and implement a NLP system which integrates a Part–of–Speech (PoS) tagger and chunker as a pre–processing module of broad–coverage unification–based grammar of Spanish. This allows us to release the parser from certain tasks that may be efficiently and reliably dealt with by these computationally less expensive processing techniques. On the one hand, by integrating the morpho–syntactic information delivered by the PoS tagger, we reduce the number of morpho–syntactic

[1]First results of the system we describe may be found in [Marimon *et al.* 01, Marimon 02].

ambiguities of the linguistic expression to be analyzed. On the other hand, by integrating chunk mark–ups delivered by the partial parser, we do not only avoid generating irrelevant constituents which are not to contribute to the final parse tree, but we also provide part of the structure that the analysis component has to compute, thus, avoiding a duplication of efforts.

In addition, we want our system to be able to maintain the accuracy of the high–level grammar. In the integrated architecture we propose, we keep the ambiguities which can not be reliably solved by the PoS tagger to be dealt with by the linguistic components of the grammar performing deep analysis.

Besides improving the efficiency of the overall analysis process and maintaining the accuracy of the grammar, our system provides both structural and lexical robustness to the high–level processing. Structural robustness is obtained by integrating into the linguistic components of the high–level grammar the structures which have already been parsed by the chunker such that they do not need to be re–built by phrase structure rules. This allows us to extend the coverage of the grammar to deal with very low frequent constructions whose treatment would increase drastically the parsing search space and would create spurious ambiguity. To provide lexical robustness to the system, we have implemented default lexical entries. Default lexical entries are lexical entry templates that are activated when the system can not find a particular lexical entry to apply. Here, the integration of the tagger, which supplies the PoS information to the linguistic processing modules of our system, allows us to increase robustness while avoiding increase in morphological ambiguity. Better precision is achieved by extending the PoS tags of our external lexicon so that they include syntactic information, for instance subcategorization information.

1.2 History and Background to the Research

The development of the grammar that served as the basis of our research work was mainly done in the framework of the Advanced Language Engineering Platform (ALEP) [Simpkins *et al.* 93] during two European projects: Large–Scale GRAMmars for EC languages (LS–GRAM – LRE 61029) [Schmidt *et al.* 96] and MEthods and tools for natural Language Interfacing to Standard Software Applications (MELISSA – ESPRIT 22252) [Groenendijk & Schmidt 97, Bredenkamp *et al.* 98]. The goal of the LS– GRAM project was to create language resources with high–level properties which could be used for both research and application projects.

Sufficient coverage for “real life text” phenomena and efficient processing were main requirements in developing the grammatical resources. In MELISSA, the grammar was being used for the first time in an industrial context, therefore, a major requirement in it was efficiency.

Different aspects must be taken into account when a PoS tagger is to be developed: accuracy, robustness, human–labour cost, portability to other languages, amount and type of linguistic information required. Since our research work aims at integrating shallow and deep linguistic processing, results of the former will seriously influence the performance of the latter. Therefore, we require a linguistic tagger (as opposed to data–driven tagger) that leaves ambiguities to be solved by following modules rather than making risky predictions. This linguistic approach, however, requires a high effort for writing an exhaustive grammar to deal with ambiguities which quite often go beyond the definition of morpho–syntax. In addition, a lineal generalization does not always provide enough information for effective disambiguation. Phenomena like non–local agreement and discontinuity are difficult to express without information about constituency structure and head information. The system we have developed can also be used to do partial parsing and to use that information for PoS disambiguation.

1.3 Overview of the Thesis

Chapter 2 – A Unification–based Grammar of Spanish. This chapter addresses deep linguistic processing and it presents the broad–coverage unification–based grammar of Spanish. We present both the theoretical and the engineering aspects of the grammar. On the one hand, we will see how linguistic knowledge is represented in the grammar components. On the other hand, we will see how we have matched the requirements of the theoretical framework on which the grammar is based to the capabilities of a so–called “lean” formalism. Furthermore, we will describe how we have deployed several devices provided by the ALEP platform to the effect of developing well–engineered efficient broad–coverage grammars. The engineering solutions we propose constitute a first step towards distributing the analysis process for efficient processing.

Chapter 3 – A Linguistic Tagger and Chunker of Spanish. This chapter addresses shallow linguistic processing and it presents the linguistic tagger and chunker of Spanish. We describe the work done for adapting and extending existing tools and resources for tokenizing input text, morphological analysis and handling unknown words, and we present the tool we have used for disambiguation and chunking. We give an outline of the formalism and we present our disambiguation and chunking grammars. In presenting the grammars, we will show how they have been designed as a set of modular components, such that they can be updated efficiently on the basis of new corpus evidence and they can be adapted easily to deal with different types of input texts. We will show how the interaction of PoS disambiguation and partial parsing reduces the effort needed for writing rules considerably.

Chapter 4 – Integrating Shallow and Deep Linguistic Processing. This chapter presents the system we have developed which integrates shallow and deep processing. We show that the strategy we propose ensures that the accuracy of the grammar remains the same. Furthermore, we show how our approach deploys the ALEP components such that the information delivered by the shallow processing tool is fully integrated without the need of extending the machinery provided by the ALEP system nor its linguistic processing components. Reported results will demonstrate that the efficiency of the overall analysis process improves significantly and that our system also provides robustness to the linguistic processing.

Chapter 5 – Conclusion. This chapter presents the conclusions.

The appendices show a classification of the ambiguities found in the lexicon we have used for morpho–syntactic annotation (Appendix A), some examples of the output of the tokenizer, the tagger and chunker, and the ALEP grammar (Appendix B), and the test suites we have used to test and evaluate our integrated system (Appendix C).

Chapter 2

A Unification–Based Grammar of Spanish

This chapter addresses deep linguistic processing and it presents the broad–coverage unification–based grammar of Spanish implemented in the framework of the ALEP platform. The grammar we present here is the one we have used to integrate shallow and deep linguistic processing.

The first section briefly presents the LS–GRAM and the MELISSA projects. In that, our goal is to supply a background to the grammatical resources we will discuss. Section two presents the coverage of the grammar. Section three and four present the grammar architecture and the ALEP linguistic formalism. In section four we describe the pre–processing and lifting components. Finally, we describe the linguistic processing components of the grammar. Here, we discuss both the theoretical and the engineering aspects of the grammar.

2.1 Introduction

The grammar we present was mainly developed during two projects: Large–Scale GRAMmars for EC languages (LS–GRAM) and MEthods and tools for natural Language Interfacing to Standard Software Applications (MELISSA).[1]

- LS–GRAM was funded by the Commission of the European Union under LRE 61029.[2] The goal of the project was "to create resources with high–level properties which can be used for training, research and application projects" (cf. [LSGRAM 93], pp. 3) for nine European languages —English, German, Spanish, Danish, Dutch, French, Greek, Italian and Portuguese.
- MELISSA was funded by the Commission of the European Union (DG III–6) under ESPRIT 22252.[3] The goal of the project was "to develop the technology and provide tools which will enable end users to interface, using natural–language, to computer application systems, and to apply this technology successfully to obtain a pre–competitive product validated in selected end–user applications" (cf. [MELISSA 95], pp. 3).

Grammar development in both projects was done in the framework of the Advanced Language Engineering Platform (ALEP), "... an initiative of the Commission of the European Communities (CEC) to provide the natural language research and engineering community in Europe with a versatile and flexible general purpose research and development environment." (cf. [Simpkins *et al.* 93], pp. 11). ALEP is supplied with formalisms (for modeling morphological, grammatical and translation knowledge), parsing algorithms (for both analysis and synthesis), application tools (for text handling, lingware development, testing and debugging), environment tools (to support the use of the system itself) and a full–fledged working environment consisting of an Emacs–based editor and a MOTIF–based graphical interface.

[1]The grammar is currently being used in the European project Interfacing Mobile Applications with Voice Natural Language Interactivity (IMAGINE – IST–2000-29490). The main goal of the IMAGINE project is to develop software technology that allows the interaction with e–business applications by using a multi–lingual NLI from mobile devices and other appliances. See `http://www.rtd.softwareag.es/imagine`.

[2]See `http://www.ub.es/gilcub/ingles/projects/european/lsgram.html`. See also [Schmidt *et al.* 96].

[3]See `http://www.ub.es/gilcub/ingles/projects/european/melissa.html`. See also [Groenendijk & Schmidt 97, Bredenkamp *et al.* 98].

Grammar design and implementation in LS–GRAM were based on corpus investigations in order to extend grammars in the most realistic way.[4] A core grammar, covering basic linguistic phenomena, was implemented on the basis of the grammar developers' linguistic knowledge and previous experiences implementing grammars. Then, a priority list for extending it was defined on the basis of the frequency of the phenomena occurring in the corpus.

In MELISSA, the English, German and Spanish ALEP grammars were being used for the first time in an industrial context, therefore, a major requirement in them was efficiency. Acceptance of a NLI, in addition, required that the NLP capabilities coped with a large range of natural ways of accessing application functionalities. In this respect, corpus study was also relevant in the context of this project to define and refine the coverage of the existing grammars. Nevertheless, this did not imply a move towards particular corpus–driven grammar designs. Generality of the grammar components was maintained on the basis of a modular design, so that the core components of the grammars could be re–used across different applications by supplementing them with add–on modules according to their specific requirements.

During LS–GRAM, grammar development was a collaborative effort of Maite Melero and the author of this thesis. In MELISSA, the author was fully in charge of the revision, re–design and extension of the grammar. Beyond the two projects, and in the framework of our research work, the grammar was further revised and extended by the author. The state of the grammar we are describing here is the result of this work. Henceforth, we are referring to this grammar as "our grammar".

2.2 Coverage of the Grammar

This section presents the coverage of the grammar. Our goal here is to show which linguistic phenomena our grammar of Spanish deals with, and, in that, to demonstrate that it is, indeed, a large–scale grammar.[5]

The basic linguistic phenomena constituting the core grammar are:

[4]The Spanish corpus consisted of a collection of articles from the newspaper "El Diario Vasco" (September and October 1991). All of them deal with economic topics [Melero 95]. Corpus investigations are available at `http://www.iai.uni-sb.de/LS-GRAM/papers.html`.

[5]See [Bosque & Demonte 99] for a detailed presentation of the phenomena.

- Main clauses with basic canonic word order, i.e. SVO (e.g. *Argentina colocó bonos en Europa* (Argentina placed bonds in Europe)).

- All types of verbal, nominal, prepositional, adjectival and adverbial subcategorization structures: attributive complements (e.g. *es imposible* (it is impossible)), direct objects (e.g. *compré acciones* (I bought stock shares)), indirect objects (e.g. *informé a mis colegas* (I informed my colleagues)), prepositional complements (e.g. *invierto en bolsa* (I invest on the stock market)).

- Determination (simple and complex): articles, demonstratives, possessives, indefinites, quantifiers, cardinals (e.g. *una/la/su/esta/otra propuesta* (a/the /his/this/another proposal), *todas las propuestas* (all the proposals)).

- A full coverage of agreement: subject–verb agreement, subject–attribute agreement and agreement within the NP.

- Null–subjects: pro–drop and impersonal sentences with the particle '*se*' (e.g. *se espera que la situación mejore* (it is expected that the situation improves)).

- Compound tenses (e.g. *ha mejorado* (has improved), *está mejorando* (is improving)) and periphrastic forms (e.g. *sigue aumentando* (keeps increasing), *acaba de aumentar* (has just increased)).

Modules extending the core grammar cover the following phenomena:

- Clausal complements:
  - completive clauses (e.g. *dije que iría* (I said that I would go)).
  - indirect questions: finite (e.g. *no sé cuándo ocurrió* (I don't know when it happened)) and non–finite (e.g. *no sé qué hacer* (I don't know what to do)).

- Control and raising structures:
  - control verbs: subject–control verbs (e.g. *Telefónica decidió abandonar* (Telefónica decided to quit)), object–control verbs (e.g. *obliga a Israel a cumplir* (obliges Israel to fulfil)), subject–to–subject raising verbs (e.g. *puede presentar su obra* (he/she can present his/her work)).

  - control adjectives: subject–control adjectives (e.g. *no estoy seguro de poder ayudarte* (I'm not sure to be able to help you)), object–control adjectives (e.g. *está pendiente de cumplimentar* (it is not filled in)).
  - nouns taking infinitival complements (e.g. *permiso para asistir* (permission to attend)).

- Support verb constructions (e.g. *tener interés en* (to have interest in)).

- Passive constructions:

  - passive constructions with the copula, with or without the by–agent complement (e.g. *será juzgado por un jurado popular* (he will be judged by a popular panel)).
  - reflexive passive construction with the clitic '*se*' (e.g. *se celebrará un almuerzo* (a breakfast will be celebrated)).

- Modification:

  - modifiers of verbs: adverbs (e.g. *lo rechazamos enérgicamente* (we refuse it drastically)), PPs (e.g. *lo adelantaron al momento actual* (they advanced it to the current moment)), and temporal NPs (e.g. *vine el pasado martes* (I came last Tuesday)).
  - modifiers of nouns: adjectives (e.g. *empresas nacionales* (national companies)), participles (e.g. *propuestas presentadas a la comisión* (proposals presented to the commission)), appositional NPs (e.g. *el empresario malagueño Manuel Olivares* (the manager of Malaga Manuel Olivares)) and PPs (e.g. *el presidente de la compañía* (the president of the company)).
  - modifiers of adjectives (e.g. *muy interesantes* (very interesting)).
  - modifiers of adverbs (e.g. *muy bien* (very well)).

- Negation: adverb '*no*' (e.g. *no es apropiado* (it is not appropriate)), negative pronouns, quantifiers and adverbials like '*nada*' (e.g. *no significa nada* (it means nothing)), '*ninguna*' (e.g. *ninguna posibilidad* (no opportunity)), and '*nunca*' (e.g. *nunca estamos preparados* (we are never ready)).

- Sentential adjuncts: subordinate clauses modifying the main clause (e.g. *un poeta ha muerto porque no tenía ganas de vivir* (a poet has died because he didn't feel like living)).
- Topicalization (e.g. *su actitud parece que está motivada por un deseo* (his/her attitude seems to be motivated by a desire)).
- Relative clauses:
  - clauses where the relativizer is an argument of the subordinated verb: subject (e.g. *La CE, cuya unión política corre peligro* (The EC, whose political union is in danger)), direct object (e.g. *las obras que están realizando* (the works that they are doing)), indirect object (e.g. *las personas a quienes agradezco su ayuda* (the people whom I thank their help)), adverbial/prepositional complement (e.g. *el lugar donde apareció* (the place where it appeared)) or any bound argument (e.g. *la empresa en la que invierto mi dinero* (the company on which I invest my money)).
  - clauses where the relativizer is modifying the subordinated verb (e.g. *la terraza donde lo vieron* (the terrace where they saw it)).
- Surface word order variation: subject–predicate inversion (e.g. *Los bonos argentinos tienen un plazo de dos años, anunció Fernández* (The Argentinian bonds have a term of two years, said Fernández)).
- Coordination:
  - binary coordination of main clauses (e.g. *ha aumentado su producción y espera poder exportar sus productos* (has increased his/her production and hopes to be able to export his/her products)) and subordinate clauses (e.g. *los trabajadores que están emergiendo y que sucederán* (the workers who are emerging and who will succeed)).
  - binary coordination of verbal phrases (e.g. *ha aumentado su producción y espera poder exportar sus productos* (has increased his/her production and hopes to be able to export his/her products)), nominal phrases (e.g. *celebraron diez conferencias y una mesa redonda* (they celebrated ten conferences and a round–table conference)), adjectival phrases (e.g. *las pequeñas y medianas empresas* (the small and medium

companies)), adverbial phrases (e.g. *aquí y allí* (here and there)) and prepositional phrases (e.g. *en el ámbito central y en el autonómico* (in the national and autonomous field)).

  - multiple coordination (or enumeration) of all major category projections (e.g. *autoridades militares, civiles y municipales* (militar, civil and town authorities)).
  - coordination of unlike categories: adjectives and PPs (e.g. *personal auxiliar y de administración* (assistant and management staff)), adjectives and adverbs ending in '*–mente*' (e.g. *única y exclusivamente* (unique and exclusively)).

- Clitics:[6]
  - cliticization (or clitic–NP alternation), including multiple cliticization, (e.g. *la he visto* (I have seen her), *me dió algo* (gave me something), *me lo dió* (gave it to me)).
  - clitic doubling (or clitic–NP co–occurrence) where the clitic appears together with the full complement (e.g. *a España se le va a exigir* (to Spain they will demand)).
  - clitic climbing, occurring with a few control verbs and periphrastic constructions, where the clitic moves from the infinitival clause to the control verb (e.g. *te puede conmover* (it may affect you)).
  - enclitics (e.g. *tomárselo* (to take it)).
- Ellipsis: NPs with no noun–head (e.g. *en el ámbito central y en el autonómico* (in the national and autonomous field)).
- Non–sentential input strings: infinitival phrases (e.g. *consultar propuestas* (to consult proposals)), NPs (e.g. *lista de propuestas* (list of proposals)).
- Special constructions: numbers (e.g. *7, –14, 3,7*), percentages (e.g. *40%*), measures (e.g. *50 toneladas* (50 tones)), currencies (e.g. *300 millones de*

[6]Including the '*leísmo*' (the use of *le* instead of *lo* (or *la*) as clitic to substitute the direct object), '*laísmo*' (the use of *la* instead of *le* as dative clitic with feminine antecedent) and '*loísmo*' (the use of *lo* instead of *le* as dative clitic with masculine antecedent). See [Fernández-Ordóñez 99] for a detailed presentation of the phenomena.

dólares (300 million dollars)), dates (e.g. *14 de julio* (14th of July)), periods (e.g. *del 23 de junio al 16 de septiembre* (from 23rd June till 16th September)), multi–word units (e.g. *a través de* (across), *por parte de* (on the part of)).

The following table indicates the number of phrase structure rules by which we accomplish the coverage we have just described. Diverging figures for source rules and compiled rules are due to disjunctive statements being compiled out.

	source rules	compiled rules
core grammar	21	33
extensions	208	329

Table 2.1 Phrase Structure Rules

Besides by phrase structure rules, many of the grammatical phenomena we cover are modeled at the level of lexical entries, as we will see in section 2.6.2. Here, we will just mention that our grammar contains 1962 lexical entries. In the following table we indicate the number of entries per category. Again, diverging figures for source terms and compiled terms are due to disjunctive statements being compiled out.

	source terms	compiled terms
verbs	355	752
nouns	181	240
adjectives	232	453
adverbs	35	35
closed class items	412	482

Table 2.2 Lexical Entries

With the coverage outlined, our grammar copes with input ranging from short instructive statements or queries to complex sentential structures as are found e.g. in newspaper articles, business letters, etc.

The following are typical examples of queries that appeared in the corpus of MELISSA:

(1) a. *Propuestas aprobadas.* (Approved proposals.)

b. *Consultar propuestas aprobadas.* (To consult approved proposals.)

c. *¿Qué propuestas fueron aprobadas?* (Which proposals were approved?)

d. *Quiero saber qué propuestas fueron aprobadas.* (I want to know which proposal were approved.)

Examples (2) and (3) are taken from the corpus that served as the basis of the experiments presented in chapter 4 (cf. Appendix C for the full corpus). Both examples show a high level of complexity and interaction of phenomena, as we will show.

(2) *El funcionario indicó que los títulos argentinos comenzaron a negociarse el martes en los mercados londinenses, donde se cotizaron al 100,75 por ciento de su valor nominal, debido a la fuerte demanda de los inversores.*

This sentence has a verb which takes a completive clause (*indicó que*). The completive clause has a periphrastic control verb (*comenzaron a*). The controlled verb is a reflexive passive (*negociarse*). It is modified by a temporal NP (*el martes*) and a locative PP (*en los mercados londinenses*). The locative PP includes a complex NP. The head of the NP is modified by an adjective (*londinenses*) and a relative clause where the relative adverb (*donde*) is the complement of the subordinated verb (*cotizaron*). In the relative clause we find again a reflexive passive (*se cotizaron*) and two modifiers: the PP introduced by the contracted preposition (*al*), and the PP introduced by the complex preposition (*debido a*) which takes an argumental noun (*demanda*). Finally, this sentence shows a sample of an special construction (the percentage *100,75 por ciento*).

(3) *Los bonos argentinos tienen un plazo de dos años y un interés anual del 11,2 por ciento, añadió Fernández.*

This example presents a special word order pattern, though very common in the journalistic style in Spanish: the completive clause (without the conjunction *que*) is extraposed and it is followed by a comma and an inverted subject. The verb in the completive clause takes a coordinated NP complement. Both coordinated nouns are modified by a PP (*de dos años*, *del 11,2 por ciento*). The last PP shows another instance of a percentage (*11,2 por ciento*).

2.3 Grammar Architecture

ALEP distinguishes pre–processing operations and linguistic processing operations. The former account for surface properties of input text (document formatting, delimitation of textual structural elements, orthographemic aspects of morphology), while the latter deal with its non–surface properties (morpho–syntactic analysis, constituent structure, semantic representation). Certain operations can be left out in the processing chain, provided that the output data structure produced by the previous processing operation suites the input data requirements of the successive one.

In particular, the pre–processing operations are:

- `Text Handling`. It is performed by the Text Handling system. It deals with document formatting and textual structural information.
- `Morpho-graphemic analysis` (optional). It is performed in the paradigm of the two–level morphology. It establishes a correspondence between surface word strings and lexical morpheme strings, and it handles phenomena of morpho–graphemic variation.

The linguistic processing operations (for analysis) are:

- `Parsing`. It is performed on the basis of a set of linguistic structure rules and a parsing lexicon, both being based on a type system declaration. It is in charge of deep linguistic analysis building up a parsed tree.
- `Refinement` (optional). It is performed on the basis of a set of linguistic structure rules and a refinement lexicon, both being based on a type system declaration. It is in charge of enriching parsing output structures, typically with semantic information.

A special rule–based operation interfaces the output of the pre–processing operation with the parsing operation.

- `Lifting`. It is performed on the basis of a set of so–called text structure to linguistic structure rules.

The overall architecture is outlined in Figure 2.1.

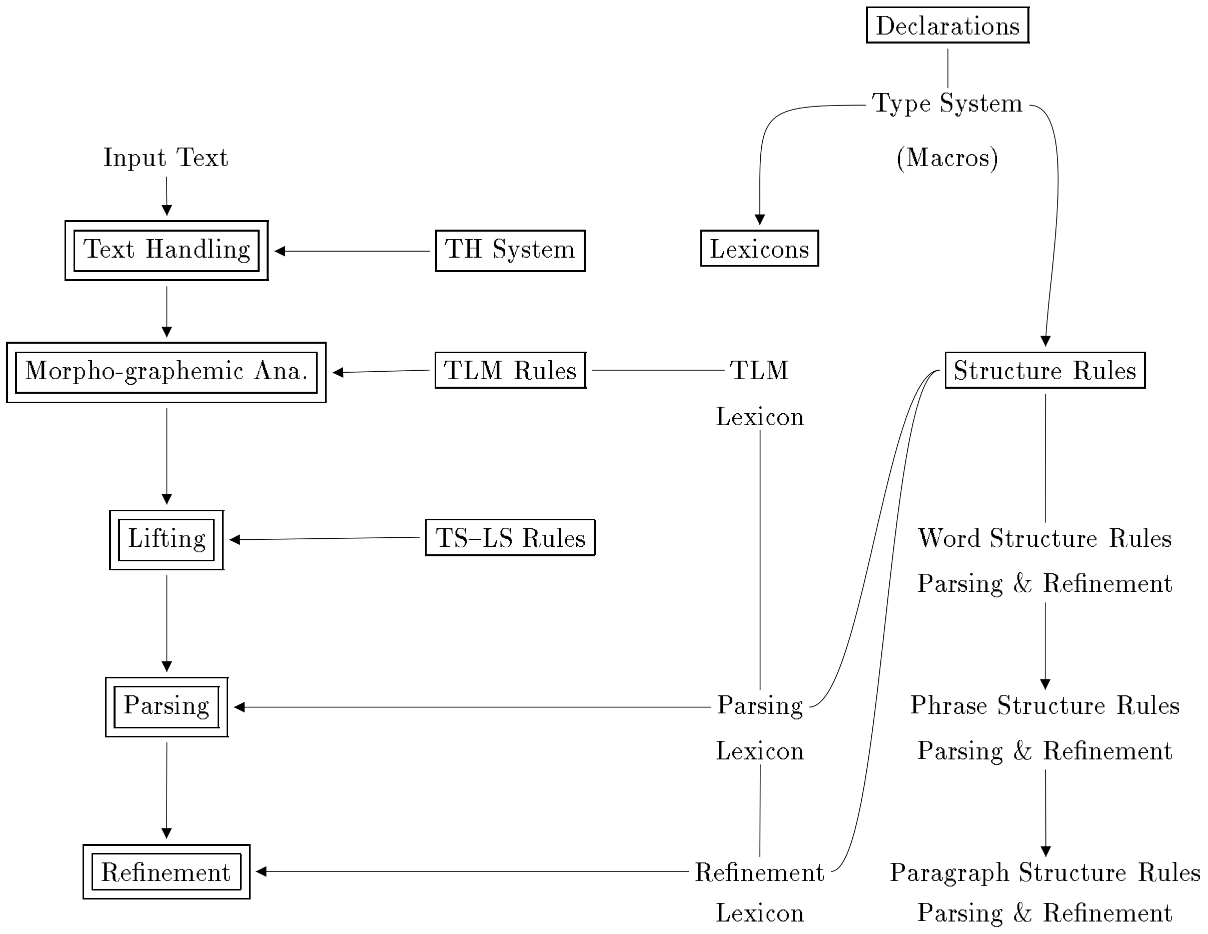


Figure 2.1 Grammar Architecture

2.4 The Linguistic Formalism

The linguistic formalism proposed by the ALEP system has been developed on the basis of the specifications resulting from the ET–6 design study [Alshawi *et al.* 91]. It is a so–called "lean" formalism compilable into first–order (Prolog) terms and thus avoiding computationally expensive formal devices.

Three basic data structures are assumed in the ALEP linguistic formalism:

- `linguistic descriptions`, collecting constraints on a type system, and, in that, modeling lexical entries and structure nodes.
- `linguistic structures`, expressing immediate dominance relations between linguistic descriptions.
- `partial linguistic structures`, expressing weak (or non–immediate) dominance relations between linguistic descriptions.

These data structures are based on a simple data structure:

- `type declarations`, constituting the type system.

In the following we present these data structures in more detail and give partial illustrations from our grammar. More detailed examples will follow in sections 2.5 and 2.6.[7]

2.4.1 Type Declaration

Type declaration in ALEP defines for a type a set of feature structures (attribute–value pairs). The syntax of the type declaration is of the form (cf. [Simpkins *et al.* 93]):

```
type(<type_name>:{ <feature_decl> }, <doc_atom>).

where:

<type_name>     is an atomic unique type name.
<feature_decl>  is an optional sequence of feature declarations.
<doc_atom>      is a possibly empty atomic documenting string.
```

[7]See also [Badia *et al.* 95].

Each feature declaration takes the form:

```
(<feature_name> => <feature_type>,<doc_atom>)
```

The types of the features are restricted to take one of the following values:

- `any`. The value that the feature takes is unrestricted. This value type has not been used in our grammar, since, in fact, for large–scale grammar development more specific types are more appropriate.

- `atom` (i.e. strings) where the range of values may be stated (4.a) or not (4.b).

  (4) a. vtype => atom(pron,nopron,pronom).

  b. lemma => atom.

- `list`, where the type of elements within the list may optionally be specified. A typical use of this type of feature is to specify the subcategorized for elements (5.a). One expresses a saturated COMPL–list in linguistic descriptions with the empty list (5.b).

  (5) a. compl => list(type({t_synsem:{}})).

  b. compl => [ ].

- `tuple`. This type of feature has not been used in our grammar. An example taken from the ALEP User Guide [Simpkins *et al.* 93] is (6.a), which in linguistic descriptions would allow values like (6.b).

  (6) a. gaps => tuple.

  b. gaps => (gaps_in:{. . .},gaps_out:{. . .}).

- `functor`. Examples of this type in our grammar are found in the compositional meaning representation (cf. section 2.6.4). In linguistic descriptions it allows values like (7.b).

  (7) a. index => functor.

  b. index => sf(index(event,I)).

- `boolean expression`, (true/false) using atomic constants and the three operators: disjunction (';'), conjunction ('&') and negation ('~'). The typical use of the boolean type is to express the agreement features where we have three sets in the list of possible values.

  (8) agr => boolean([{p1,p2,p3}{sing,plur}{masc,fem,neut}]).

  ALEP also allows the use of values which are outside the defined set (or sets) of values, by adding the marker '*' to the set. Note that values outside the explicit range can not be combined with those ones which are given, so that the constraint expressed in (9.c) is illegal.

  (9) a. bar => boolean({zero,one,max,*}).

  b. bar => minus.

  c. bar => (minus;zero).

- `user-defined type`. The value of some attribute may be another feature structure with its own internal structure.

  (10) morph => type({t_morph:{}}).

No sets, and therefore, no set–operations are supported in ALEP, which have to be simulated by list and list–operations.[8]

The following is an example of a type declaration from our grammar.[9]

(11)

```
type(
t_local:{
        (cat => type({t_cat:{}} ),
                'Syntactic information.'),
        (sem => type({t_sem:{}} ),
                'Semantic information.')
        }, 'Local type' ).
```

[8]The substitution of set–operations by list–operations, however, suffers from several problems in efficiency and it may not guarantee correct results (cf. [Erbach *et al.* 95]). An alternative solution is to implement set–descriptions and set–operations (subset, intersection, union) by means of external constraint solvers [Erbach *et al.* 95].

[9]Types in our grammar take the prefix 't_'.

ALEP provides a simple monotonic inheritance mechanism for types. Monotonic in that each subtype inherits all feature declarations (attribute–value pairs) from its supertype, as opposed to default inheritance, which may overwrite values which have been inherited from the type hierarchy to account for irregularities or exceptions in the lexicon, e.g. [Krieger & Nerbonne 93] for derivational morphology.[10] On the other hand, simple inheritance means that it only allows one supertype for each given type from which it inherits the feature declarations, as opposed to multiple inheritance, where information can be inherited from more than one supertype.[11]

Type hierarchies are declared in the form of (cf. [Simpkins *et al.* 93]):

```
<supertype>   >   <subtypes>
```

(12) shows how substantive and functional categories are diversified in terms of a type hierarchy declaration in our grammar.

(12)

```
t_cat > {
      t_subst,
      t_funct }.
```

The use of feature structures —variously known as f–structures, functional structures, terms, dags or categories— to model linguistic information corresponds to the one adopted by contemporary theoretical and computational linguistics, e.g. Lexical Functional Grammar [Bresnan 82], Generalized Phrase Structure Grammar [Gazdar *et al.* 85], Head–Driven Phrase Structure Grammar [Pollard & Sag 87, Pollard & Sag 94], Functional Unification Grammar [Kay 85], Unification Categorial Grammar [Zeevat *et al.* 87] and PATR–II [Shieber *et al.* 83].

The use of feature structures is also reflected in other implemented systems like Attribute Logic Engine (ALE) [Carpenter & Penn 94], Core Language Engine (CLE) [Alshawi 91], ConTroll [Götz *et al.* 97], Comprehensive Unification Formalism (CUF) [Dörre *et al.* 94], Lexical Knowledge Base (LKB) [Copestake 99],

[10]Default inheritance should not be confused with the ALEP facility for specifying default values in the type system declaration (and used at compiled time). Default values in ALEP do not overwrite values, but they are inserted whenever no value has been specified for the attribute, e.g. t_noun:{ agr => boolean([{p1,p2,p3}{sing,plur}{masc,fem,neut}]) => p3.}.

[11][Erbach 94] —see also [Erbach *et al.* 94a]— presents a system with multidimensional inheritance which can be encoded in ordinary Prolog terms, thus making the system compatible with ALEP.

Prolog with Features, Inheritance and Templates (ProFIT) [Erbach 95], CL-ONE [Manandhar 94], Type Description Language (TDL) [Krieger 95] and Typed Feature Structure (TFS) [Emele 93].[12]

The popularity of feature structures in linguistics comes from their capability to bear complex information, which results from 'recursive embedding' (i.e. the value of some attribute may itself be another feature structure) and 'structure–sharing' (i.e. distinct attributes in a larger structure may be specified as having the same value, even when that value is unknown). These structures, in addition, are partially ordered on the basis of their information content by the 'subsumption principle', according to which a given feature structure F subsumes another feature structure F' if F is less informative. But the crucial operation on feature structures is 'unification'. Unification operates upon a set of two or more feature structures giving as result a feature structure which contains all the information present in the members of the set and nothing else, unless these structures contain incompatible or conflicting information.[13]

2.4.2 Linguistic Descriptions

A Linguistic Description (LD) in ALEP is "a collection of constraints represented as a composite data structure which is given a name, its type, and has a strictly defined content; constraint names with (recursively) a well defined and range of values." (cf. [Simpkins *et al.* 93]).

Lexical entries in ALEP are based on the data type LD. The syntax of a lexical entry is (cf. [Simpkins *et al.* 93]):

```
<tag> ~ <LD>

where:

<tag>          is an atomic identifier.
<LD>           is a linguistic description.
```

[12]See [Backofen *et al.* 96] for a survey and comparison of grammar formalisms in the 1990s and closely related issues e.g. grammar theories, implemented grammars and development environments. See also [Bolc *et al.* 96] for an overview and detailed comparison of the systems TFS, CUF, ALE, ALEP, PAGE, ProFIT and ConTroll.

[13]Standard references are [Shieber 86] for an introduction for the use of feature structure in linguistics, and [Carpenter 92] for a formal account of typed feature structures.

The following is an example of a partial lexical entry taken from our grammar.

(13)

```
aumentar ~
ld:{
       synsem => t_synsem:{
              local => t_local:{
                    morph => t_morph:{
                          lemma => aumentar },
                    cat => t_verb:{} } } }.
```

2.4.3 Linguistic Structures

A Linguistic Structure (LS) in ALEP is "a tree composed of LDs in an immediate dominance relation." (cf. [Simpkins *et al.* 93]).

Phrase structure rules in ALEP are based on the data type LS. The syntax of a phrase structure rule is (cf. [Simpkins *et al.* 93]):

```
<LD>   <   <daughters>

where:

<LD>          is a linguistic description.
<daughters>   are the dominated daughters, each represented
              as a LD.
```

Dominated daughters may be represented both as ordered lists of LDs, by the use of the square brackets, and as unordered multi–sets of LDs, in which case we use the curly brackets instead. Optionality of a daughter is marked by '?'. Finally, disjunction is expressed by parenthesis. Unordering, optionality and disjunction are compiled out.[14]

[14]The project *Reusability of Grammatical Resources* (LRE–61–061) has, in addition, implemented an extended syntax for phrase structure rules in ALEP for arbitrary repetition of sequences of daughter nodes, expressed by '*', which is also compiled out [Erbach *et al.* 95].

The following is an example of a partial phrase structure rule taken from our grammar which cancels the complements following the head–dtr.

(14)

```
ld:{
     synsem => t_synsem:{
              local => t_local:{
                     cat => t_subst:{
                            head => HEAD,
                            subj => SUBJ,
                            compls => [REST] } } } },
< [
ld:{
     synsem => t_synsem:{
              local => t_local:{
                     cat => t_subst:{
                            head => HEAD,
                            subj => SUBJ,
                            compls => [COMPL|REST] } } } },
ld:{
     synsem => COMPL } ].
```

2.4.4 Partial Linguistic Structures

A Partial Linguistic Structure (PLS) in ALEP is "a LS where dominance may be non–immediate." (cf. [Simpkins *et al.* 93]).

The output of the lifting process is a PLS. The syntax of a PLS is (cf. [Simpkins *et al.* 93]):[15]

```
<LD>   <<   <daughters>

where:

<LD>           is a linguistic description.
<daughters>    are the dominated daughters, each represented
               as a LD.
```

[15]An example is given in section 2.5.3.

2.5 Pre–processing and Lifting Components

2.5.1 Text Handling

The ALEP platform provides a Text Handling (TH) system for processing texts before and after linguistic processing [Cruickshank 95].

In the analysis direction, the default setup of the system defines the following processing chain: conversion of (nroff, latex and ascii) document formats to an Eurotra Document Interchange Format (EDIF) based upon the Standard Generalized Markup Language (SGML) standard; then, a number of recognition processes are applied sequentially from paragraph to words: (i) paragraph recognition (where paragraph boundaries are indicated by one or more blank lines), (ii) sentence recognition (where full–stops or question marks constitute the end of a sentence),[16] and (iii) word recognition.

The output of these processes consists of an EDIF mark–up —<P>...</P>, <S ID=...>...</S> and <W TYPE=...>...</W>[17]— and normalization of the recognized structural elements.[18] In a final stage, EDIF marked–up documents are converted into so–called text structure chunks (`ts-chunks`).

A simple input text like "*acciones*", which has both a nominal reading (stock shares) and a verbal reading ((you) activate), after going through the TH component looks as follows:

(15)

```
tsn:{
      attr => 'S',
      feature_list => [ ft('ID', '1') ],
      children => [
                     tsl:{
                           attr => 'W',
                           feature_list => [ ft('TYPE','WORD') ],
                           data_content => acciones } ] }.
```

[16] A user defined list of abbreviations is checked to avoid errors when sentences are split.

[17] A further tag '<PT TYPE=...>...</PT>' is used for punctuation marks. As can be observed, when tag features are required they are part of the start tag.

[18] Text normalization is required to allow the processing stages in analysis (before linguistic processing) and generation (after linguistic processing) to be symmetrical.

2.5.2 Morpho–graphemic Analysis Component

After TH analysis, the morpho–graphemic analysis component operates on words within a `ts-chunk`. Here, the orthographic aspects of morphology are processed.

Morpho–graphemic analysis is performed in the paradigm of Two–Level Morphology (TLM) [Koskenniemi 83] using the ALEP TLM formalism and algorithm [Cruickshank 94]. The basic functionality of the morpho–graphemic analysis component is that of establishing a correspondence between surface word strings and lexical morpheme strings to the effect of segmenting word–forms. In addition, the morpho–graphemic analysis component deals with orthographic variation phenomena. The orthographic variation phenomena which are covered by our grammar are: graphemic alternations (z/c, c/g, j/g, qu/c, gu/g), diphthongs (ue/o, ie/i, ie/e,üe/o), and closing (a/u, a/i, a/e, u/o, e/i, e/í).

By executing the morpho–graphemic analysis operation, a level marked–up by the tag 'M' (for morphemes) is added below tag 'W' nodes in the `ts-chunk`:

(16) [19]

```
    tsn:{
          attr => 'S',
          feature_list =>  [ ft('ID', '1') ],
          children => [
                 tsn:{
                       attr => 'W',
                       feature_list => [ ft('TYPE','WORD') ],
                       children =>  [
                          tsl:{
                                attr => 'M',
                                feature_list => [ ft('CAT',_) ],
                                data_content => accion },
                          tsl:{
                                attr => 'M',
                                feature_list => [ ft('CAT', _) ],
                                data_content => es } ] } ] }.
```

Morpho–graphemic analysis in ALEP is non–deterministic. It produces as many outputs as permitted by the TLM rules. ALEP provides two versions of the TLM algorithm: the 'basic' and the 'record'. By executing the 'basic' version, alternative analyses are presented in alternative versions of the `ts-chunk`. Overgeneration leads to a combinational exploitation in the number of `ts-chunks`.

[19]Note that here we only show the analysis for the verbal reading.

The 'record' version displays alternative analysis in a single `ts-chunk` as alternative *tsl*, as we show in (17), where the *ts_1* corresponds to the verbal reading and the *ts_2* corresponds to the nominal reading.

(17)

```
...
tsn:{ attr => 'W',
      feature_list => [ ft('TYPE','WORD') ],
      children => [
           ts_alternate:{
                ts_1 => [ tsl:{ attr => 'M',
                                feature_list => [ ft('CAT',_) ],
                                data_content => accion },
                          tsl:{ attr => 'M',
                                feature_list => [ ft('CAT',_)],
                                data_content => es } ],
                ts_2 => [ tsl:{ attr => 'M',
                                feature_list => [ ft('CAT',_) ],
                                data_content => acción },
                          tsl:{ attr => 'M',
                                feature_list => [ ft('CAT',_)],
                                data_content => s } ] } ] }
...
```

2.5.3 Text Structure to Linguistic Structure Rules

The output of the morpho–graphemic analysis component (the `ts-chunk` structure) is converted into LDs, which are the data structures needed for deep linguistic analysis.

Format conversion is performed by the 'lifting' operation based on a particular set of rules, so called Text Structure to Linguistic Structure (TS–LS) rules.

TS–LS rules are prolog terms taking 3 to 4 arguments (cf. [Simpkins *et al.* 93]).

```
ts_ls_rule( <LD>, <tag_name>, <features>, <tag_content>).
```

The first argument specifies the target LD, the second one the name of the structural tag, and the third one the tag–feature list. The last argument is optional, and it is for the atomic value of the string within the tag, which in (17) appears as value of the attribute 'data_content'.

Different rules must be provided for the different tags —'M', 'W' and 'S'— [20] where the resulting LD data structures must be unambiguously identified as representing morphemes, words (either a non–inflecting lexical sign or the mother node dominating the morphological structure, in case morpho–graphemic analysis has been performed) and whatever node which defines the kind of LD that is to be sought as the parsing root, establishing the axiom of the grammar.[21]

The following rules are the three TS–LS rules accounting for the three structural tags ('S', 'W' and 'M') we implemented in our grammar, where LDs are unambiguously identified by the BAR feature: 'bar=root' characterizing the top node, 'bar=zero' characterizing words, and 'bar=minus' characterizing morphemes (cf. section 2.6.3.2).

(18)

```
ts_ls_rule(
ld:{
    synsem|local|bar => root },
'S', [ ] ).

ts_ls_rule(
ld:{
    synsem|local|bar => zero },
'W', [ ] ).

ts_ls_rule(
ld:{
    synsem => t_synsem:{
           local => t_local:{,
                  bar => minus,
                  morph => X => t_morph:{
                         morpheme =>  MORPHEME } } } },
'M', [ CAT = X ], MORPHEME ).
```

The last rule shows how by variable sharing between tag–features and LD feature structures we can transfer information from the former to the latter. Variable sharing between 'tag_content' and the MORPHEME attribute of the LD enforces unification with a particular lexical item.

[20]And for 'P' when paragraphs are processed by the linguistic processing components.

[21]Normally, this will be the sentence node, though it can also be any phrasal node when partial input strings are to be processed.

The output of the lifting process is a PLS data structure where the hierarchical relations between the different LDs is expressed in terms of week dominance relations (indicated by the symbol '<<'):

(19)

```
ld:{
     synsem => t_synsem:{
             local => t_local:{
                    bar => root } } },
<<
ld:{
     synsem => t_synsem:{
             local => t_local:{
                    bar => zero } } },
<< [
ld:{
     synsem => t_synsem:{
             local => t_local:{
                    bar => minus,
                    morph => t_morph:{
                           morpheme => accion } } } },
ld:{
     synsem => t_synsem:{
             local => t_local:{
                    bar => minus,
                    morph => t_morph:{
                           morpheme => es } } } } ]
```

2.6 Linguistic Processing Components

The process of designing and implementing a grammar in ALEP is split into two separate steps: the first one corresponds to the type system declaration and the second one to the specification of the body of the linguistic knowledge (lexical entries and grammar rules) based on the type system that must have been declared previously. In our grammar, the adopted approach is based on one of the currently most prominent unification–based grammatical theories, so–called Head–Driven Phrase Structure Grammar (henceforth HPSG).

In what follows, we will represent both type declarations and LDs using Attribute Value Matrices (AVMs) diagrams. The hierarchical structure as well as immediate dominance relations will be represented using graphs. Attributes and values will be annotated in SMALL CAPITALS and *italics* respectively. To reduce the complexity of AVMs, occasionally we will employ path descriptions.

2.6.1 The Type System

In the following, we give a partial view on the type system underlying our grammar. Specific feature paths which we do not elaborate on in this section, but which will become relevant later on, will be described in the relevant sections.[22]

As the top–most of the type hierarchy, we define the type *ld* (abbreviating linguistic description). Here, morphological, syntactic and semantic information fall under the attribute SYNSEM, taking as value a *t_synsem* type. This type structures both the local information (morphological, categorial and semantic) and the information related to the treatment of long distance dependencies (topicalization, relative clauses and interrogative clauses) by means of the attributes LOCAL (taking as value a type *t_local*) and NLOCAL (taking as value a type *t_nlocal*).

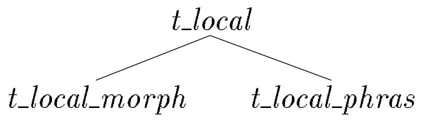
$$\begin{bmatrix} \textit{ld} & \\ \text{SYNSEM} & \begin{bmatrix} \textit{t\_synsem} & \\ \text{LOCAL} & \textit{t\_local} \\ \text{NLOCAL} & \textit{t\_nlocal} \end{bmatrix} \end{bmatrix}$$

t_local encodes the morphological, categorial and semantic information by means of two attributes: CAT and SEM, which take as value a type *t_cat* and a type *t_sem*, respectively.

$$\begin{bmatrix} \textit{t\_local} & \\ \text{CAT} & \textit{t\_cat} \\ \text{SEM} & \textit{t\_sem} \end{bmatrix}$$

The two attributes CAT and SEM are inherited by the two local subtypes: *t_local_morph* and *t_local_phras*, which we have declared for data tokens participating in the domain of morpho–graphemics and morphological constituency (i.e. word structure), and phrasal constituency (i.e. phrase structure), respectively.

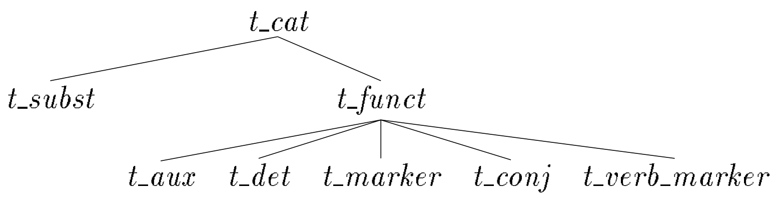
[22]See also [Badia *et al.* 96].



Only *t_local_morph* introduces an attribute of its own (MORPH) to encode morphologically relevant information. MORPH takes as value a *t_morph* type, which introduces the attributes LEMMA, encoding the canonic form, and MORPHEME, encoding the morpheme. Both attributes take an atomic value the range of which is not stated.

$$\begin{bmatrix} \textit{t\_local\_morph} & \\ \text{MORPH} & \begin{bmatrix} \textit{t\_morph} & \\ \text{LEMMA} & \textit{atom} \\ \text{MORPHEME} & \textit{atom} \end{bmatrix} \end{bmatrix}$$

The *t_cat* type has no attributes itself, but it is highly subtyped. First, it distinguishes major (substantive) and minor (functional) categories. Then, minor categories are subtyped along categorial dimensions.



In [Pollard & Sag 94], the distinction between substantive and functional categories is encoded in the feature 'SYNSEM|CAT|HEAD' as subtypes of the type *head*. PoS distinctions are then realized as subtypes of the head subtypes *subst* and *funct*. Our option, however, allows us to reduce the size of features structures for efficient processing.[23]

[23]A similar solution to reduce the size of feature structures is described by [Flickinger 00]. [Flickinger 00] proposes to enrich the type hierarchy in order to reduce the size of feature

Local categorial properties for major categories are divided into head (or part–of–speech) information, marking information (i.e. marking features that produce a marking effect of some constituent on some other constituent), modification information and subcategorization information. Subcategorization information is split into three valence features: SUBJ, COMPL and SPECIFIER.[24] Supplementary to this, the BYAGENT attribute is used to encode the by–agent complement of passive verbal forms. Following [Theofilidis & Reuther 95], by–agent complements are dealt with by a separate list to exclude them from completeness checks being performed on COMPL lists.

$$\begin{bmatrix} \textit{t\_subst} & \\ \text{HEAD} & \textit{t\_head} \\ \text{MARKING} & \textit{t\_marking} \\ \text{MODIFIES} & \textit{list(t\_modifies)} \\ \text{SUBJ} & \textit{list(t\_synsem)} \\ \text{COMPL} & \textit{list(t\_synsem)} \\ \text{SPECIFIER} & \textit{list(t\_synsem)} \\ \text{BYAGENT} & \textit{list(t\_synsem)} \end{bmatrix}$$

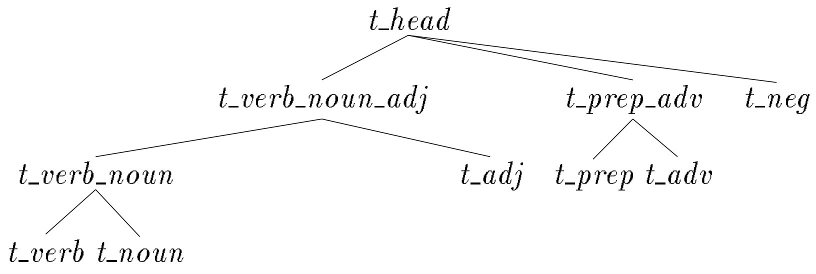
Substantive categories are distinguished by the attribute HEAD encoding the head information. This attribute takes a type *t_head*, which we have subtyped along categorial dimensions.

The introduction of intermediate types in the *t_head* hierarchy allows us to reduce lexical disjunction dealing with frame variation according to the categorial realization of the syntactic functions, e.g. direct objects being realized by verbal and nominal projections.[25]

structures and reports on the introduction of an abstract type *min_head* taking no feature as a supertype of *head*, such that the attributes of the latter are only introduced when a unification occurs. Further strategies for reducing the size of features are to apply a filtering process to remove non–informative features [Götz 93], or to reduce the number of named disjunctions [Gerdemann & King 94].

[24]Following chapter 9 (Reflections and Revisions) of [Pollard & Sag 94]. The attribute SPECIFIER in our grammar, however, records information about the determiners (or quantifiers) which have been attached to nouns; this means that, unlike HPSG, only specifiers select heads.

[25]Eliminating disjunction in feature values by enriching the type hierarchy is also described in [Flickinger 00].



Each subtype in the *t_head* hierarchy introduces attributes of its own: *t_verb _noun_adj* has the attribute AGR encoding agreement information, *t_verb* has the attribute VTYPE to encode the verb type (pronominal, nonpronominal, pronominalizable), *t_noun* has the attributes NTYPE and CASE to encode the noun type (countable, mass, uncountable/concrete, abstract/human, animate, inanimate/...) and the case, respectively, *t_adj* has the attributes ATYPE, DEGREE and COPULA to encode the adjective type (mod, quant), the degree of the adjective (positive, comparative and superlative) and the copula verb (*ser, estar*) it takes, finally, *t_prep_adv* takes the attribute PATYPE to encode the preposition/adverb type.

In minor categories, we distinguish part of speech information, encoded in the attribute CAT_MIN, and information about the construction they specify, encoded in the attribute SPECIFIES, which takes as value a list of *t_synsem*.

$$\begin{bmatrix} \textit{t\_funct} & \\ \text{CAT\_MIN} & \textit{boolean(aux,det,marker,conj,verb\_marker)} \\ \text{SPECIFIES} & \textit{list(t\_synsem)} \end{bmatrix}$$

Finally, the semantic local features encoded in the type *t_sem* cover the following major domains of semantic description: predicate–argument structure, modifier relation and functional semantic information, distributed over the attributes PREDARG, MODS and INDX.

$$\begin{bmatrix} \textit{t\_sem} & \\ \text{INDX} & \textit{t\_indx} \\ \text{PREDARG} & \textit{t\_predarg} \\ \text{MODS} & \textit{list(t\_sem\_mod)} \end{bmatrix}$$

Borrowing from [Badia & Colominas 98], a rather small set of argument relation labels is assumed, with external arguments and least oblique arguments being assigned *arg1* and *arg2* respectively. A number of semantically interpreted argument relation labels, such as *arg_place* or *arg_goal*, is reserved for arguments the meaning of which resembles that of adjuncts. For adjuncts, a standard set of modifier relations is used including e.g. *place*, *goal*, *origin*, *time*, *beneficiary* or *instrument*. Finally, functional semantics covers those aspects relating to negation, determination and tense and aspect, we will discuss in section 2.6.4.

2.6.2 Lexical Entries

The richness of information reflected in the type system of our grammar favours the development of a highly lexicalized grammar, following the proposals of contemporary theoretical and computational linguistics.

The lexical component of the grammar, therefore, plays a crucial role in the grammatical description needed for processing. Linguistic phenomena, such as subject–verb agreement, subcategorization, modification, control relations, etc., traditionally dealt with by means of specialized phrase structure rules, are in our grammar treated in the lexicon.

So, for example, subject–verb agreement (in simple forms) is modeled with the structure sharing mechanism between the agreement feature of the SUBJ–list and that of the verb, as shown in the sample lexical entry we present in (20) for the verb *aumentar* (to increase). Subcategorization is implemented by using a few binary rules, which apply recursively, fed by rather complex lexical entries which bear information about the subcategorization restrictions they impose on their subjects and complements. This is also illustrated in (20), where the attributes SUBJ and COMPL specify the restrictions on the nominal subject and complement this particular verb takes. (20) also shows how the predicate–argument structure is built up in the lexical entries by sharing of the semantic features of external argument and the least oblique argument with the semantic features of the subject and the complement, respectively.

(20) [26] aumentar

[26]Note that we only show the most relevant features.

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{ld}} \\
\text{SYNSEM}\,|\,\text{LOCAL}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_local\_morph}} \\
\text{MORPH}: & \left[\begin{array}{l}
\textit{t\_vstem} \\
\text{LEMMA}: \ \textit{aumentar} \\
\text{MORPHEME}: \ \textit{aument}
\end{array}\right] \\
\text{CAT}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_subst}} \\
\text{HEAD}: & \left[\begin{array}{l}
\textit{t\_verb} \\
\text{VTYPE}: \ \textit{nopron} \\
\text{AGR}: \ \boxed{1}
\end{array}\right] \\
\text{MARKING}: & \left[\begin{array}{l}
\textit{t\_marking} \\
\text{MARK}: \ \textit{none}
\end{array}\right] \\
\text{SUBJ}: & \left\langle \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_synsem}} \\
\text{LOCAL}: & \left[\begin{array}{l}
\textit{t\_local} \\
\text{CAT}\,|\,\text{HEAD}\,|\,\text{NTYPE}: \ \textit{nilts} \\
\text{CAT}\,|\,\text{HEAD}\,|\,\text{AGR}: \ \boxed{1} \\
\text{CAT}\,|\,\text{MARKING}\,|\,\text{MARK}: \ \textit{none} \\
\text{SEM}: \ \boxed{2}
\end{array}\right]
\end{array}\right] \right\rangle \\
\text{COMPL}: & \left\langle \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_synsem}} \\
\text{LOCAL}: & \left[\begin{array}{l}
\textit{t\_local} \\
\text{CAT}\,|\,\text{HEAD}\,|\,\text{NTYPE}: \ \textit{inanim\&nilts} \\
\text{CAT}\,|\,\text{MARKING}\,|\,\text{MARK}: \ \textit{none} \\
\text{SEM}: \ \boxed{3}
\end{array}\right]
\end{array}\right] \right\rangle \\
\multicolumn{2}{l}{\text{BYAGENT}: \ \langle\rangle} \\
\multicolumn{2}{l}{\text{SPECIFIER}: \ \langle\rangle} \\
\multicolumn{2}{l}{\text{MODIFIES}: \ \langle\rangle}
\end{array}\right] \\
\text{SEM}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_sem}} \\
\multicolumn{2}{l}{\text{INDX}: \ \textit{t\_indx}} \\
\text{PREDARG}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_predarg}} \\
\text{ARGS}: & \left[\begin{array}{l}
\textit{t\_args} \\
\text{ARG1}: \ \boxed{2} \\
\text{ARG2}: \ \boxed{3}
\end{array}\right]
\end{array}\right] \\
\multicolumn{2}{l}{\text{MODS}: \ \langle\rangle}
\end{array}\right]
\end{array}\right] \\
\multicolumn{2}{l}{\text{SYNSEM}\,|\,\text{NLOCAL}: \ \textit{t\_nlocal}}
\end{array}\right]
$$

The lexicalist treatment of linguistic phenomena, however, has as a direct result an increase in the complexity and redundancy of lexical entries. The information specified in a lexical entry is shared by different entries related to one another in different ways.

On the one hand, the verb *aumentar*, for instance, shares with all verbal entries the empty list values for the attributes BYAGENT (only passive forms take by–agent complements), MODIFIES (only passive forms and relative clauses can modify) and SPECIFIER. In addition, it shares with other transitive verbs its subcategorization schema (though not all lexical restrictions it imposes on its subject and complement). On the other hand, (20) is closely related to the entries encoding the decausative and passive readings, as we show in (21).

(21) a. *Seat aumentó las ventas.* (Seat increased the sales.)

b. *Las ventas aumentaron.* (The sales increased.)

c. *Las ventas fueron aumentadas.* (The sales were increased.)

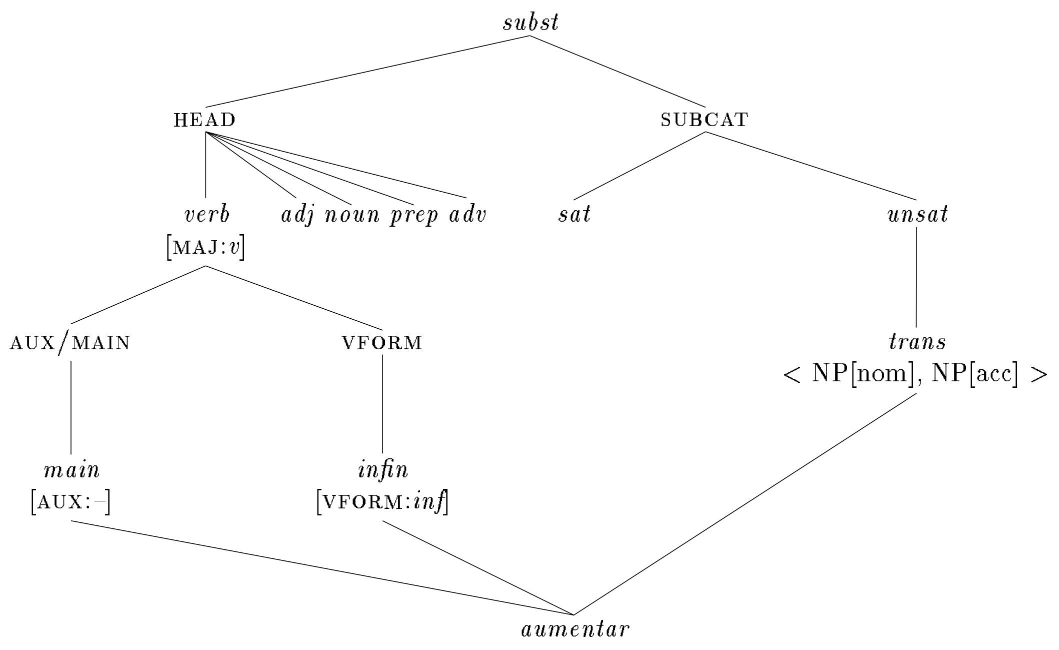
Not surprisingly, the design and development of lexicons has become the focus of attention in contemporary theoretical and computational linguistics and NLP research community with the aim of providing both theoretical and practical solutions, e.g. providing lexicons with some structure capable of capturing linguistic generalizations to avoid unnecessary and undesirable redundancy, and ensuring lexical entries to be efficiently encoded.[27]

As described in [Pollard & Sag 87, Flickinger 87], in HPSG properties of lexical entries and relationships among them are captured in a concise and principled fashion in terms of classifications by a multiple inheritance hierarchy of types and lexical rules, respectively.

On the one hand, lexical information is organized as a multiple inheritance hierarchy of types, such that lexical types can inherit features from more than one supertype, as we show in (22). With this, the amount of information to be explicitly encoded in lexical entries is drastically reduced. This is shown in (23) where only very specific information needs to be represented.

[27]See [Meurers 97] and works published in [Briscoe *et al.* 93]. Another issue is efficient processing of lexical information and lexical rules, for which distinct techniques have been described in the literature. See, for instance, [Bouma 97, Theofilidis 95a, Theofilidis 95b, Bouma & Noord 94, Johnson & Dörre 95, Meurers & Minnen 95, Briscoe & Copestake 96].

(22) [28]



(23)

aumentar: *main&infin&trans*

$$\left[\text{SYNSEM}\,|\,\text{LOCAL}: \left[\begin{array}{l} \text{PHON: } \textit{aumentar} \\ \text{SUBCAT: } \left\langle \text{NP}[\,]_{\boxed{1}}, \text{NP}[\,]_{\boxed{2}} \right\rangle \\ \text{SEM}\,|\,\text{CONT: } \left[\begin{array}{l} \text{ARG1: } \boxed{1} \\ \text{ARG2: } \boxed{2} \end{array}\right] \end{array}\right]\right]$$

[28]We take the type and feature system of ([Pollard & Sag 87], pp. 207). The same holds for the next example.

On the other hand, redundancy in the lexicon may be further eliminated by the use of lexical rules. While the hierarchical organization of the lexical types accounts for the 'vertical' relations within a lexicon, lexical rules account for the 'horizontal' relations lexical signs have with one another (cf. [Pollard & Sag 87]). Lexical rules describe processes (inflectional, derivational, valence–alternation) operating on feature structures, and, in that, they reduce the number of entries to be manually encoded in a dictionary.

(24) shows the 'Passive Lexical Rule' (cf. [Pollard & Sag 87], pp. 215), which takes a lexical type '*base & trans*' and produces a lexical type '*passive*'.

(24)

$$\left[\begin{array}{l} \textit{base\&trans} \\ \text{PHON: } \boxed{1} \\ \text{PAST–PART: } \boxed{2} \\ \text{SYN} \mid \text{LOC} \mid \text{SUBCAT: } \left\langle \ldots, \boxed{3}, \boxed{4} \right\rangle \\ \text{SEM} \mid \text{CONT: } \boxed{5} \end{array}\right] \rightarrow \left[\begin{array}{l} \textit{passive} \\ \text{PHON: } \left(\boxed{1}, \boxed{2} \right) \\ \text{SYN} \mid \text{LOC} \mid \text{SUBCAT: } \left\langle \left(\text{PP}\left[\text{BY}\right]\boxed{4} \right), \boxed{3} \right\rangle \\ \text{SEM} \mid \text{CONT: } \boxed{5} \end{array}\right]$$

Since the ALEP formalism does not support multiple inheritance, neither hierarchical lexical types nor lexical rules can be implemented in the lines proposed within HPSG. Nevertheless, we will now see some properties of the ALEP formalism which provide an efficient solution, in terms of development resources, to deal with the lexical component of the grammar.

2.6.2.1 Lexical Macros

Even though lexical information in ALEP can not be organized on the basis of a set of defined types, as proposed within the HPSG framework, a rich set of macros (or template definitions) can be defined for an efficient (i.e. less time consuming) encoding of lexical entries.

Lexical information that is common to a subset of lexical entries can be moved from lexical entries to a set of macros.[29] So, just as in HPSG lexical information comes from the idiosyncratic information encoded in the lexical entries and the

[29]Lexical macros are defined in the type system and apply over all linguistic material compiled under that type system. All macros are expanded at compile time before any defaults are applied.

information inherited from the types that lexical entry belongs to, the lexical component of an ALEP grammar may consist of a number of macro definitions, where general information is encoded, and the lexical entries, encoding only the information particular to them.

To illustrate this, see the entry for *aumentar* we have presented in (20) as it is encoded by means of a macro. The parameters are for lemma, verb type, semantic properties of the subject and the complement, and role of the second argument:

(25)

```
aumentar ~
m_LEX_ANA_V_SnpOnp[ aumentar, nopron, nilts, inanim&nilts, arg2 ].
```

Lexical macros in our grammar are defined by several macros, each one contributing part of their definition.[30] This is illustrated in (26), which shows the macro 'm_LEX_V_SnpOnp/5' as it appears in our grammar, where macros with open parameters are filled by other macros.[31]

(26)

```
macro(m_LEX_V_SnpOnp[LEMMA,VT,NT_S,NT_C,ROLE2],
m_LEX[
     m_LOCAL[                                          /* LOCAL */
          m_VSTEM[LEMMA,AGR],                          /* MORPH */
          m_CAT_VERB[                                    /* CAT */
               m_HEAD_V[VT,AGR],                        /* HEAD */
               m_MARKING[none],                      /* MARKING */
               m_SUBJ_NP[AGR,NT_S,ARG1],                /* SUBJ */
               m_V_FR_Onp[NT_C,ARG2]],                 /* COMPL */
          m_SEM_Varg12[LEMMA,ARG1,ARG2,arg1,ROLE2]]      /* SEM */
      m_NLOCAL[]] ).                                  /* NLOCAL */
```

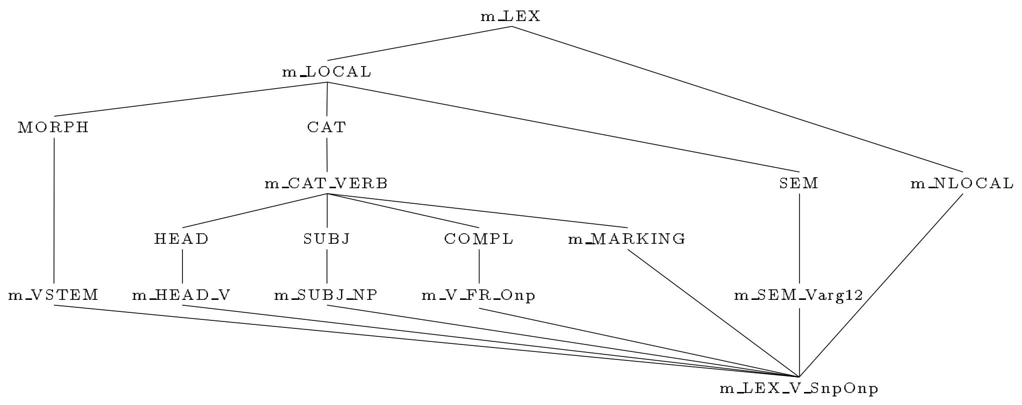
Our lexical macros result in a highly structured collection of simple macros where, as we show in (26), the linguistic data is partitioned according to the type system specification.

[30]One may also write very specific and exhaustive macros so that they account for a whole class of lexical entries. This strategy, however, does not seem very appropriate, since the redundancy we want to avoid in the lexical entries is present in the macros.

[31]Note how parameters can be used to co–index two or more values. This is a very useful facility that allows to establish structure sharing between different features in a lexical entry.

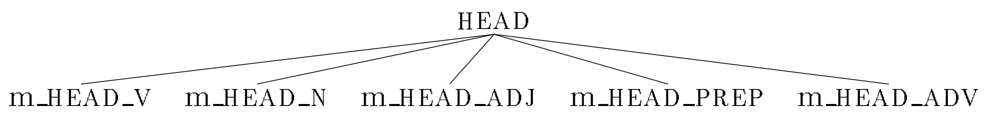
The macro we present in (26) may also be represented with a diagram which suggests a type inheritance hierarchy (27), thus showing the powerful use of our macro definitions. The crucial difference, recall, is that while type inheritance hierarchies impose well–formedness constraints on objects, macros are pure abbreviations for feature structure descriptions.

(27)



Simple macros (i.e. macros filling the different parameters) have been defined on the basis of the properties different lexical entries have in common. So, for example, different macros have been defined to fill the parameter for head information of substantive categories (28), like the one we show in (29) and (30), expanding verbal and nominal head properties, respectively.

(28)



(29)

$$\texttt{macro(m\_HEAD\_V[}\ \boxed{1},\ \boxed{2}\ \texttt{],}\quad \begin{bmatrix} \textit{t\_verb} & \\ \text{CATEG} & v \\ \text{VTYPE} & \boxed{1} \\ \text{AGR} & \boxed{2} \end{bmatrix}\texttt{).}$$

(30)

$$\texttt{macro(m\_HEAD\_N[}\ \boxed{1},\ \boxed{2},\ \boxed{3}\ \texttt{],}\quad \begin{bmatrix} \textit{t\_verb} & \\ \text{CATEG} & n \\ \text{NTYPE} & \boxed{1} \\ \text{CASE} & \boxed{2} \\ \text{AGR} & \boxed{3} \end{bmatrix}\texttt{).}$$

Other advantages of modularizing macros can easily be perceived. The different aspects of a linguistic description are easier to locate; the information that is relevant to each lexical entry is much more visible and, therefore, it may be inspected much faster. Furthermore, since only one global statement exists, this can be edited to change all instances throughout the associated lingware. Consider for instance adding new selectional restrictions to nominal complements, we only need to modify the macro 'm_V_FR_Onp/2' that appears in (26), and the rest of the code that uses this macro will be automatically updated.

Table 2.3 shows the number of lexical macros we have for the different categories. Briefly, verbal, nominal and adjectival macros are mainly distinguished on the basis of both the category and number of subcategorized for elements (subject and complements). Pronouns, as can be observed in the table, are not included in the macros for nominal entries, but they are encoded by 'm_LEX_PRON' macros, which in turn distinguish strong personal pronouns, weak personal pronouns (or clitics), relatives, interrogatives, demonstratives, indefinites. Macros for adverbs and prepositions are distinguished according to the element they modify and the elements they subcategorize for. Macros for determiners distinguish demonstratives, indefinites, possessives, interrogatives, relatives, definite articles and indefinite articles. The three macros for auxiliaries are for the auxiliaries *ser* and *estar* (to be) and the verbal marker *haber* (to have). The macros for markers are for the marker *a* (for human direct objects), and for *que* (that) and *si* (whether), respectively. Table 2.4 shows the number of simple macros, by which we build up the lexical macros.

verbs	57
nouns	30
adjectives	10
adverbs	11
prepositions	14
pronouns	10
auxiliaries	3
determiners	7
markers	2
coord conjunctions	1

Table 2.3 Lexical Macros

morph	5
cat	23
head	20
marking	1
subj	9
compl	116
modifies	15
specifies	8
sem	47
nlocal	4

Table 2.4 Simple Macros

2.6.2.2 Lexical Rules

Even though the ALEP formalism does not support Lexical Rules (LRs), the valence changing operations that they perform[32] —movement and removal of complements— can be easily reproduced by means of unary structure rules, thus reducing the number of lexical entries to be manually encoded in the lexicon.

Thus, for example, the rule we present in (31) performs the same operation as the 'Passive Lexical Rule' in HPSG (cf. 24). Similar rules have been implemented to deal with alternations in the surface order of the complements, decausativization, complement extraction and optionality of complements.

(31)

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\mathit{ld}} \\
\ldots & \left[\begin{array}{ll}
\multicolumn{2}{l}{\mathit{t\_local\_morph}} \\
\text{CAT} & \left[\begin{array}{l}
\mathit{t\_subst} \\
\text{HEAD: } \boxed{1}\mathit{t\_verb} \\
\text{ALTERN}\,|\,\text{PASS: } \mathit{yes} \\
\text{SUBJ: } \left\langle \left[\begin{array}{ll}
\multicolumn{2}{l}{\mathit{t\_synsem}} \\
\ldots & \left[\begin{array}{l}
\mathit{t\_subst} \\
\text{HEAD: } \boxed{2}\mathit{t\_noun} \\
\text{...PFORM: } \mathit{none}
\end{array}\right]
\end{array}\right] \right\rangle \\
\text{COMPLS: } \left\langle \boxed{3}\,|\,\boxed{4} \right\rangle \\
\text{BYAGENT: } \langle\rangle \\
\text{MODIFIES: } \langle\rangle
\end{array}\right] \\
\text{SEM: } \boxed{5} &
\end{array}\right]
\end{array}\right]
\rightarrow
\left[\begin{array}{ll}
\multicolumn{2}{l}{\mathit{ld}} \\
\ldots & \left[\begin{array}{ll}
\multicolumn{2}{l}{\mathit{t\_local\_morph}} \\
\text{CAT} & \left[\begin{array}{l}
\mathit{t\_subst} \\
\text{HEAD: } \boxed{1} \\
\text{SUBJ: } \left\langle \boxed{3} \right\rangle \\
\text{COMPLS: } \left\langle \boxed{4} \right\rangle \\
\text{BYAGENT: } \left\langle \left[\begin{array}{ll}
\multicolumn{2}{l}{\mathit{t\_synsem}} \\
\ldots & \left[\begin{array}{l}
\mathit{t\_subst} \\
\text{HEAD: } \boxed{2} \\
\text{...PFORM: } \mathit{por}
\end{array}\right]
\end{array}\right] \right\rangle \\
\text{MODIFIES: } \left\langle \left[\begin{array}{l}
\mathit{t\_modifies} \\
\text{...HEAD: } \mathit{t\_noun}
\end{array}\right] \right\rangle
\end{array}\right] \\
\text{SEM: } \boxed{5} &
\end{array}\right]
\end{array}\right]
$$

Constraints on the application of these rules, which in HPSG are specified by the lexical type hierarchy —the output of a LR is of a different type from that of the input— are stated by means of a set of features defining the type *t_altern*.

The type *t_altern* appears as value of the attribute SYNSEM|LOCAL|CAT|ALTERN.

[32]In [Pollard & Sag 87], LRs are also used to model inflectional paradigms and derivational relationships. Our grammar, however, employs a two–level approach to morphology, as we have seen in section 2.5.2.

$$\begin{bmatrix} \textit{t\_altern} & \\ \textsc{pass} & \textit{atom(yes,no)} \\ \textsc{deca} & \textit{atom(yes,no)} \\ \textsc{perm} & \textit{atom(yes,no)} \\ \textsc{celr} & \textit{atom(yes,no)} \\ \textsc{optc} & \textit{boolean}([\{\textit{none,c1,c2,c3,c12,c13,c23,all}\},\{\textit{of1,of2,of3}\}]) \end{bmatrix}$$

PASS specifies whether a verbal lexical entry undergoes passivization, DECA specifies whether a verbal entry has a decausative reading, PERM is for activating the lexical rule permuting the elements in the COMPL–list,[33] CELR is for activating the complement extraction rule, and OPTC specifies the optional complement (or complements).

In order to reduce backtracking over LRs and to improve processing time, LRs in our grammar are specified to be applied once the parser has been through the morphological component and before phrase structure rules are applied.

2.6.3 Phrase Structure Rules

Based on the lexicalist treatment of linguistic phenomena, phrase structure rules in our grammar are reduced to a small set of binary–branching context–free phrase structure rules closely related to HPSG Immediate Dominance (ID) schemata ([Pollard & Sag 94], in [Pollard & Sag 87] grammar rules).

ID schemata in HPSG are universal constraints on the immediate constituency of phrases (or local phrase structures) from among which languages make a selection. Linear Precedence (LP) is modeled by means of language specific LP–constraints.

In [Pollard & Sag 94] the authors suggest different schemata which are characterized by the kind of non–head daughter accompanying the head–daughter (that is, subject, complement, adjunct, filler, marker and specifier). Here, phrase structure descriptions do not come as structures, but as typed feature structures, i.e. as values of the DTRS feature.

The head–complement schema (or schema 2), for instance, subsumes all phrase

[33]Recursive application of this rule is prevented by specifying that the value for PERM in the output of the rule (the mother node) is *no*.

structure rules that expand a lexical head together with its complements.[34] The general form of the schema is shown in the following AVM representation:

(32)

$$\left[\begin{array}{ll}
\textit{phrase} & \\
\text{SYNSEM} \mid \text{LOC} \mid \text{CAT}: & \left[\begin{array}{l} \text{HEAD: } \boxed{1} \\ \text{SUBJ: } \left\langle \boxed{2} \right\rangle \\ \text{COMPLS: } \langle\rangle \end{array}\right] \\
\text{DTRS}: & \left[\begin{array}{ll}
\textit{head-comp-struc} & \\
\text{HEAD-DTR}: & \left[\begin{array}{ll} \text{SYNSEM} \mid \text{LOC} \mid \text{CAT}: & \left[\begin{array}{l} \text{HEAD: } \boxed{1} \\ \text{SUBJ: } \left\langle \boxed{2} \right\rangle \\ \text{COMPLS: } \left\langle \boxed{3} \right\rangle \end{array}\right] \end{array}\right] \\
\text{COMP-DTRS}: & \left\langle \left[\text{SYNSEM: } \boxed{3} \right] \right\rangle
\end{array}\right]
\end{array}\right]$$

Structure sharing is established by a set of universal linguistic principles. In (32), (i) the Head Feature Principle[35] establishes that the HEAD value of any sign is structure shared with that of its phrasal projections; and, (ii) the flow of subcategorization information up projection paths is handled by the Valence Principle[36]. Other schemata are:

- head–subject schema (or schema 1):
  "A phrase with DTRS value of sort *head–subj–struc* in which the HEAD–DTR value is a phrasal sign." (cf. [Pollard & Sag 94], pp. 347).

- head–subject–complement schema (or schema 3):
  "A [SUBJ < >] phrase with DTRS value of sort *head–subj–comp–struc* in which the head daughter is a lexical sign." (cf. [Pollard & Sag 94], pp. 352).

[34] "A phrase with DTRS value of sort *head–comp–struc* in which the HEAD–DTR value is a lexical sign." (cf. [Pollard & Sag 94], pp. 348).

[35] "In a headed phrase, the values of SYNSEM|LOCAL|CATEG|HEAD and DTR|HEAD-DTR|SYNSEM|LOCAL|HEAD are token identical." (cf. [Pollard & Sag 94], pp. 34).

[36] "In a headed phrase, for each valence feature F, the F value of the head daughter is the concatenation of the phrase's F value with the list of SYNSEM values of the F–DTRS value." (cf. [Pollard & Sag 94], pp. 348).

- head–marker schema (or schema 4):
  "A phrase with DTRS value of sort *head–marker–struc* whose marker daughter is a marker whose SPEC value is structure–shared with the SYNSEM value of the head daughter, and whose MARKING value is structured–shared with that of the mother." (cf. [Pollard & Sag 94], pp. 46).

- head–adjunct schema (or schema 5):
  "A phrase with DTRS value of sort *head–adj–struc*, such that the MOD value of the adjunct daughter is token–identical to the SYNSEM value of the head daughter." (cf. [Pollard & Sag 94], pp. 56).

- head–filler schema (or schema 6):
  "The DAUGHTERS value is an object of sort *head–filler–struc* whose HEAD–DTR|SYNSEM|LOCAL|CATEGORY value satisfies the description [HEAD *verb*[VFORM *finite*, SUBCAT< >], whose HEAD–DTR|SYNSEM|NONLOCAL|INHERITED|SLASH value contains an element token–identical to the FILLER-DTR|SYNSEM|LOCAL value, and whose HEAD-DTR|SYNSEM|NONLOCAL|TO–BIND|SLASH value contains only that element." (cf. [Pollard & Sag 94], pp. 164).

In our grammar, though still performing very general operations, the implementation of a given schema implies the definition of several phrase structure rules. Diversification of grammar rules is mainly motivated by the fact that there is no formal ID–LP separation in ALEP, but both relations are encoded at the same time in the rules. On the one hand, two rules are needed for those schemata where the head–daughter and the non–head–daughter can be permuted e.g. head–complement schema.[37] On the other hand, since LP constraints are category–specific, schemata have to be spelled out in category–specific phrase structure rules. Furthermore, since no implicational constraints are permitted in ALEP, universal principles responsible for structure–sharing, which in HPSG are factored out, have to be repeated in each rule.[38]

Nevertheless, the grammar developer can also benefit from the macro facility supplied by the ALEP linguistic formalism to provide an efficient solution — in terms of development resources— to the phrase structure component of the grammar.

[37]Or, as we have said in section 2.4.3, by one rule where dominated daughters are represented as unordered sets of LDs, by the use of curly brackets, in which case they are compiled out.

[38]Note that, in fact, schemata 4, 5, and 6 include structure–sharing.

2.6.3.1 Structural Macros

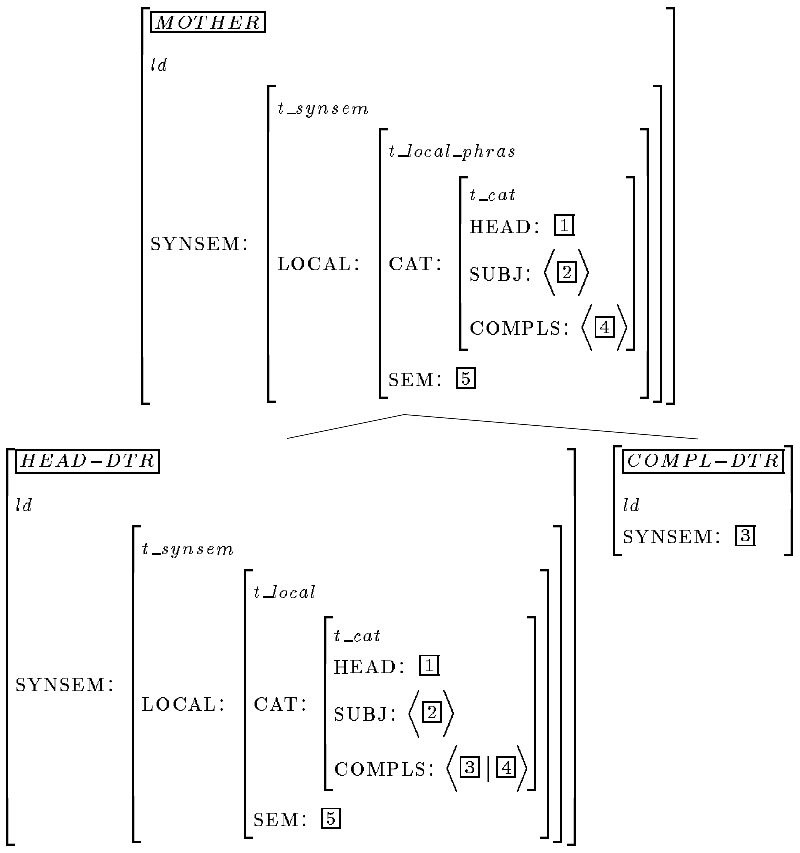
Structural macros in our grammar are used to implement the universal linguistic principles governing ID schemata.[39]

The head–complement schema, for instance, is implemented by the macro 'm_STR_compl_right/3' (33.a), which expands the LS data structure (33.b).

(33)

```
a. m_STR_compl_right[ MOTHER, HEAD-DTR, COMPL-DTR ]
```

b.



[39]Structural macros, like lexical macros, are defined in the type system and apply over all linguistic material compiled under that type system. All macros are expanded at compile time before any defaults are applied.

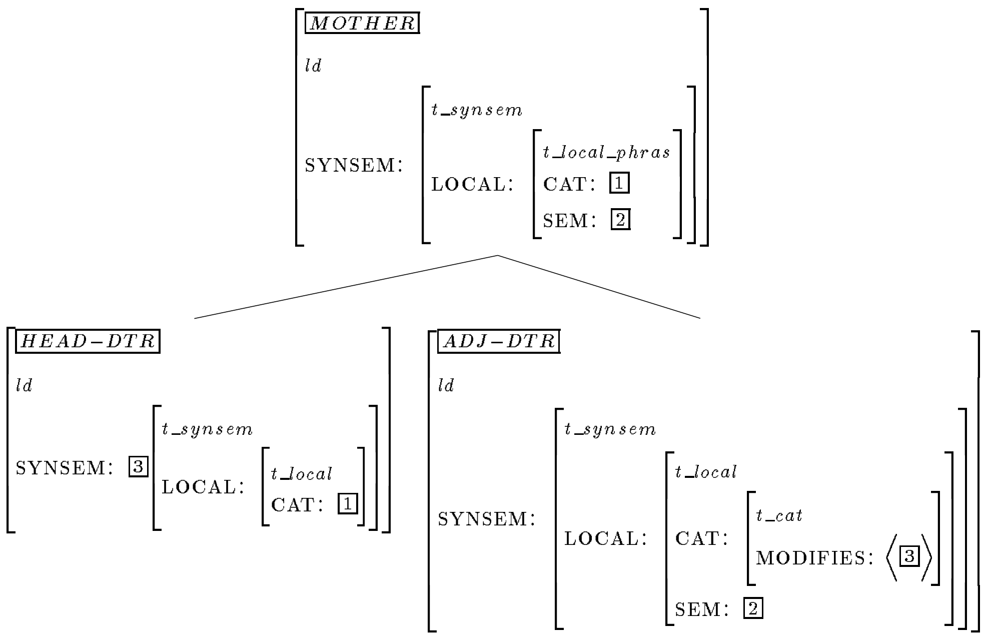
The macro 'm_STR_compl_right/3' expands a binary rule which applies recursively and cancels, form left to right, the complements that are present in the COMPL–list of the head–dtr and which appear to the right.[40] This rule, in addition, encodes all relevant information. It establishes that the feature HEAD is shared between the mother node and the head–daughter node, as required by the Head Feature Principle, so are the attributes SUBJ and SEM according to the Valence Principle and the Semantic Principle[41], respectively.

(34) shows the implementation of the head–adjunct schema. The macro 'm_STR_adjunct_right/3' expands the basic scheme of a rule attaching a constituent in the function of adjunct to the right of the head–dtr.[42]

(34)

a. `m_STR_adjunct_right[ MOTHER, HEAD-DTR, ADJ-DTR ]`

b.



[40]A further macro, 'm_STR_compl_left/3', has been defined to cancel the complements preceding the head–dtr.

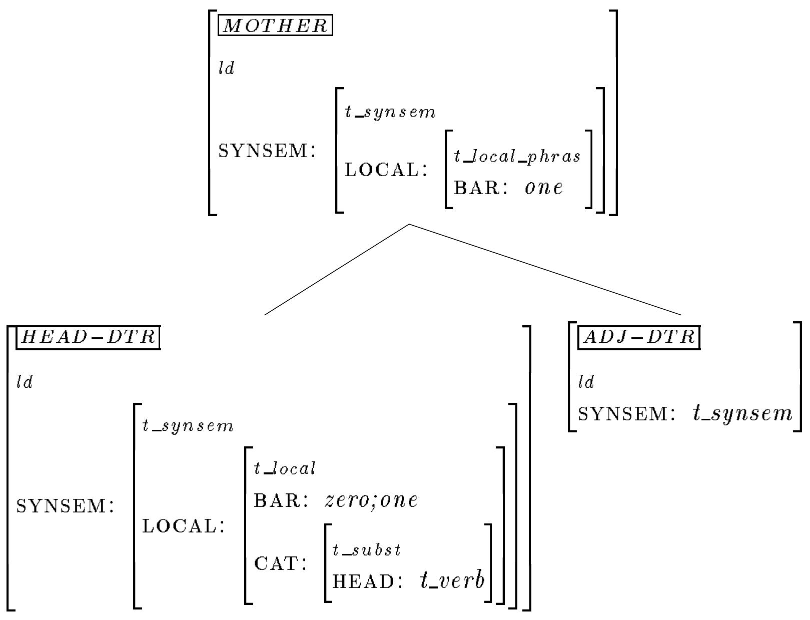
[41]"In a headed phrase the *semantic head* is the ADJUNCT–DAUGHTER if any and the HEAD–DAUGHTER otherwise." (cf. [Pollard & Sag 94], pp. 402).

[42]A further macro, 'm_STR_adjunct_left/3', has been defined to adjuncts preceding the head–dtr.

In accordance with head–adjunct schema, the adjunct selects its head. Thus, the adjunct–dtr is specified to bear a non–empty MODIFIES–list whose element is shared with the *t_synsem* feature structure of the head–dtr. In accordance with the Semantic Principle, the adjunct functions as the semantic head, so that its feature SEM is shared with that of the mother node. The percolation of the syntactic features (i.e. CAT) is due to the Head Feature Principle, the Valence Principle and the Marking Principle[43].

The following phrase structure rule instantiates the head–adjunct schema attaching an adjunct to the right of a verbal head–dtr.

(35)



Note that the information in the rule is very thin. It only specifies the value that both the mother node and the verbal head–dtr node take for the attribute BAR, which we will discuss below.

[43] "In a headed structure, the MARKING value coincides with that of the marker daughter if there is one, and with that of the head–daughter otherwise." (cf. [Pollard & Sag 94], pp. 45).

2.6.3.2 Linear Precedence Constraints

LP constraints, as we have already said in section 2.6.3, are established in category–specific phrase structure rules which constrain word order over sibling constituents, i.e. the daughter nodes that are dominated by one mother. Therefore, while the grammar includes, for instance, phrase structure rules for all major categories which cancel the complements that appear to the right of the head–dtr, there is only one rule for complements that appear to the left of the head–dtr where the head–dtr is specified to be a verb.

However, it is also necessary to capture ordering restrictions beyond our binary trees, i.e. over non–head elements which are not sisters: among complements and subjects, among complements and adjuncts, and among adjuncts,[44] in an account of LP, and, therefore, dealing with the examples in (36), but not with the ones in (37).[45]

(36) a. *Confirmó su asistencia el pasado martes.*
((he/she) confirmed his/her attendance last Tuesday.)

b. *Confirmó el pasado martes su asistencia.*
((he/she) confirmed last Tuesday his/her attendance.)

c. *Afirmó el pasado martes que superará todas las expectativas.*
((he/she) declared last Tuesday that (he/she) will exceed all expectations.)

d. *El monedero rojo que apareció abandonado.*
(The red purse that appeared abandoned.)

e. *El miedo de María a hacer el ridículo.*
(Mary's fear to make a fool of herself.)

(37) a. **Afirmó que superará todas las expectativas el pasado martes.*
((he/she) declared that (he/she) will exceed all expectations last Tuesday.)

[44]Ordering restrictions among complements are captured within the COMPL–list. The order of the elements of the COMPL–list corresponds to the surface order of the complements.

[45]LP constraints may in addition be used to select readings of ambiguous sentences.

b. **El monedero que apareció abandonado rojo.*
(The purse that appeared abandoned red.)

c. ?*El miedo a hacer el ridículo de María.*
(The fear to make a fool of herself of Mary.)

We can find at the literature different mechanisms for dealing with LP constraints. [Engelkamp *et al.* 92, Erbach *et al.* 94b] discuss a mechanism for encoding LP constraints with a binary approach to phrase structure which could be easily integrated into ALEP grammars, since it does not require any changes to the formalism. It is a fully declarative mechanism based on additional features and principles and which holds in a head domain (i.e. a lexical head and its complements and adjuncts). Complements and adjuncts encode LP–constraints by the head features LEFT and RIGHT. Heads and head projections encode information about LP occurring within their head domains in the local feature LP–STORE. This feature is updated by means of the head features LP–IN and LP–OUT of the complements/adjuncts. The feature sharing and propagation is done by two additional principles: the 'Left–Head LP–Principle' and the 'Right–Head LP–Principle', which unify the head's LP–STORE with the non–head daughter's LP–IN and the mother's LP–STORE with the non–head daughter's LP–OUT.[46]

The *Reusability of Grammatical Resources* (LRE–61–061) project has developed LP constraint solvers which have been employed to analyze complex ordering phenomena in German. Surface form of sentences is represented by a set in which the elements are pointers to words together with a collection of ordering constraints between the elements [Erbach *et al.* 95].

In the grammar we are presenting, ordering restrictions over elements which are not sisters are captured by means of an additional local attribute, so–called BAR, indicating the projection level.[47]

$$\begin{bmatrix} \textit{t\_local} & \\ \textrm{BAR} & \textit{boolean(max, one, zero, minus, subj\_r, subj\_r\_v, one\_v, verb, noun, adj, adv, \ldots)} \end{bmatrix}$$

[46]Supplementary to this, [Erbach *et al.* 94b] indicate an approach which expresses and implements LP constraints as finite–state automata, limiting the number of linearizations.

[47]There is no BAR attribute in HPSG, which represents hierarchicality by means of the valence features (COMPLS, SPR and SUBJ) and the distinction between the two subtypes of *sign* (*phrase* and *word*). Basically, lexical items are *words* and all other constituents are *phrases*. The latter are further distinguished on the basis of their valence features: *phrase* [SPR<Y">] (seeking a specifier phrase), *phrase* [SPR< >] (specifier–saturated) [Pollard & Sag 94].

Note that besides the traditional *zero*, *one* and *max* projection levels, our grammar assumes further levels which are assigned on the basis of the category, the function and the position of the non–head nodes.

In particular, the values that the attribute BAR takes are:

- *minus*, which corresponds to morphemes (i.e. lexical entries).
- *zero*, which corresponds to the mother node dominating a word structure rule dealing with either morphological analysis or valence changing operations, i.e. LRs (cf. section 2.6.2.2).
- *adv, neg, prep*, which correspond to the mother node dominating a phrase structure rule dealing with modifiers of verbs preceding the verbal head–dtr.
- *adj, part, prep, verb, noun*, which correspond to the mother node dominating a phrase structure rule dealing with modifiers of nouns following the nominal head–dtr.
- *max*, which corresponds to the mother node dominating a phrase structure rule dealing with subjects and complements preceding the verbal head–dtr, prepositional complements, nominal specifiers, modifiers of adjectives, and modifiers of adverbs.
- *one*, which corresponds to the mother node dominating a phrase structure rule dealing with nominal complements, verbal complements, adjectival complements, adverbial complements, and modifiers of verbs following the head–dtr.
- *subj_r, subj_r_v*, which correspond to the verbal mother node dominating a phrase structure rule dealing with nominal, respectively, verbal subjects following the verbal head–dtr.
- *one_v*, which corresponds to verbal mother node dominating a phrase structure rule dealing with verbal complements of verbal head–dtrs.
- *sbar*, which corresponds to the mother node dominating a phrase structure rule dealing with finite completive clauses.
- *conjunct, coord*, which correspond to the mother node dominating a phrase structure rule dealing with coordinating constructions. *conjunct* is used

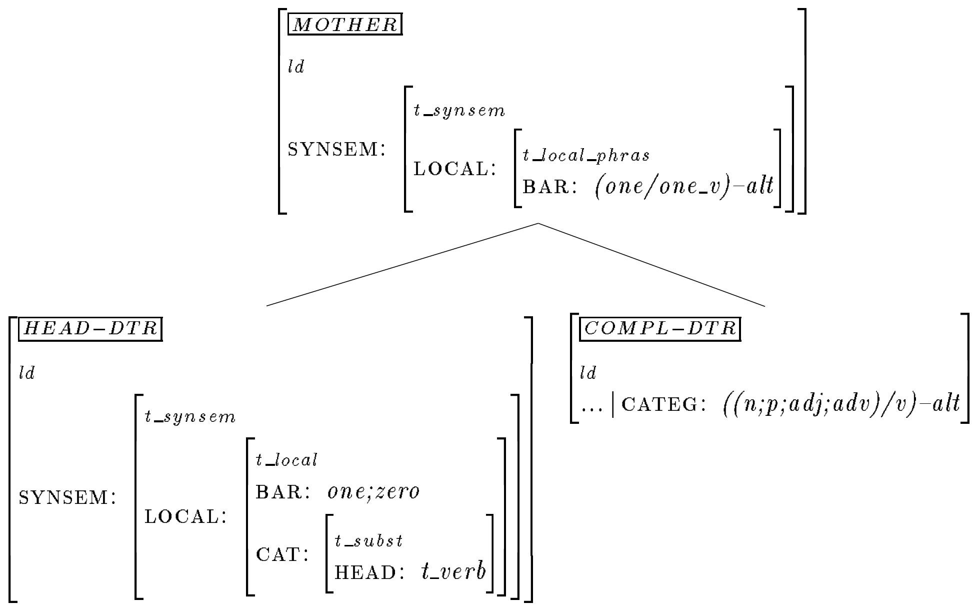
when the coordinated conjunction is attached to the second coordinated element, while *coord* is used for the top node of the construction.

- *root*, which corresponds to the top node.

The rule we illustrate in (38), for example, shows how we assign the two projection levels we distinguish when cancelling the non–verbal complements of a verbal head–daughter ('BAR:*one*'), and when cancelling the verbal complements of a verbal head–daughter ('BAR:*one_v*').[48]

By distinguishing two different projection levels according to the category of the complement the verbal head–daughter takes, we prevent adjuncts from being attached to verbal–head daughters which have cancelled a verbal complement, since, as we have shown in (35), these can only be attached to 'BAR:(*zero;one*)' nodes.

(38) [49]

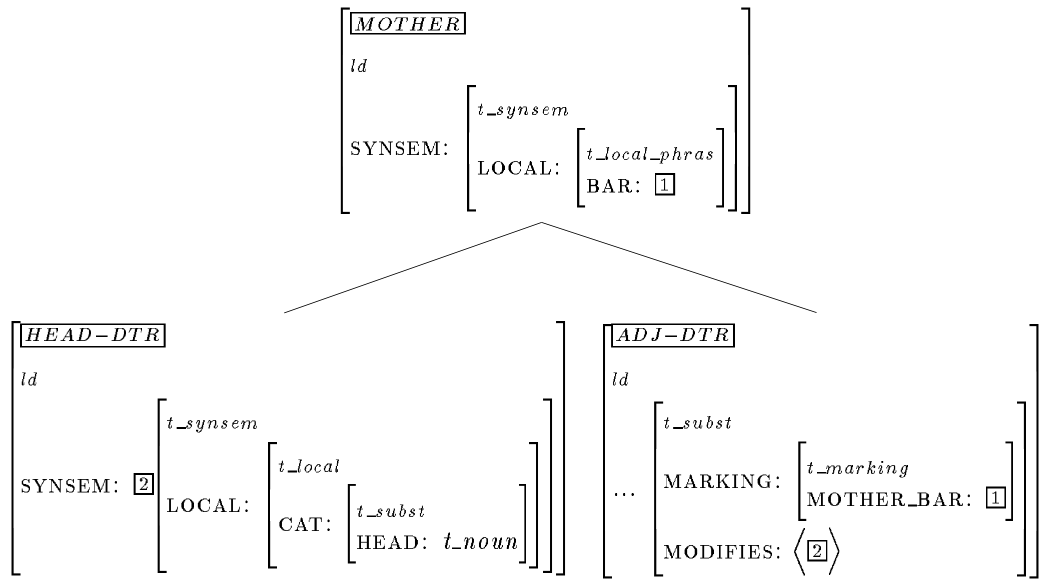


[48]Co-varying values are expressed by distributive disjunction, which is compiled out. Therefore, distributive disjunction, like macros, only facilitates grammar writing, by allowing compact encoding.

[49]Dots (...) abbreviate the path SYNSEM|LOCAL|CAT|HEAD.

The rule we present in (39) shows how we capture ordering restrictions among modifiers of nouns, and among complements and modifiers of nouns, which in Spanish are very rigid (cf. [Hernanz & Brucart 87]).[50] Besides the attribute BAR, the local attribute MOTHER_BAR encodes the projection level that modifying categories percolate to the mother node. Values for MOTHER_BAR are coincident to those values for BAR we have for modifiers of nouns. Lexical entries encode information about the bar level of the head–dtr they may be attached to.

(39) [51] [52]



[50][Hernanz & Brucart 87] propose a hierarchy for complements and adjuncts within the NP, based on a very detailed analysis of word order variation within the NP.

[51]Dots (...) abbreviate the path SYNSEM|LOCAL|CAT.

[52]In fact, such a rule does not specify that the adjunct–dtr bears a non–empty MODIFIES–list, whose element is shared with the *t_synsem* feature structure of the head–dtr, since this information is already encoded in the macro we have shown in (34.b). We have included for the sake of explanation.

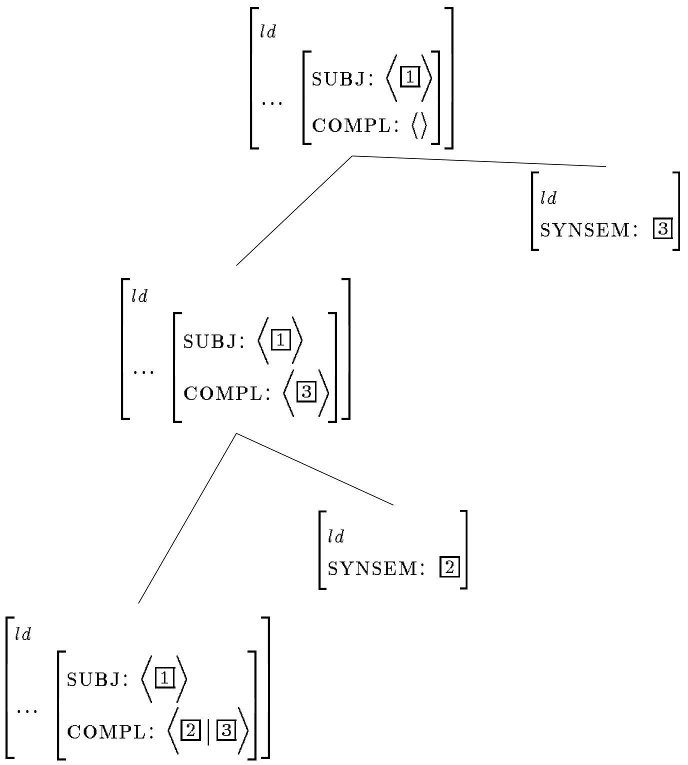
2.6.3.3 The STR Feature

From a grammar engineering point of view, the recursive binary approach to phrase structure rules has some advantages over flat structures.[53]

On the one hand, it avoids rule proliferation according to the different number of complements. Thus, a single rule can be used for realizing every complement (40).[54]

(40) a. *Invitó a sus amigos.*
((he/she) invited his/her friends.)

b. *Invitó a sus amigos a cenar.*
((he/she) invited his/her friends for dinner.)

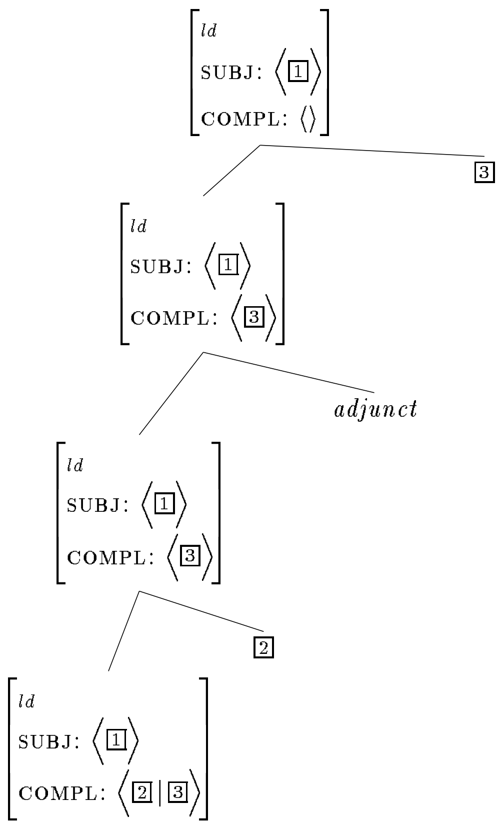


[53]See [Uszkoreit 86, Uszkoreit 91] for a discussion on the linguistic motivation in account of word ordering and fronting phenomena in German.

[54]Dots (...) abbreviate the path SYNSEM|LOCAL|CAT.

On the other hand, the recursive binary approach to phrase structure rules avoids rule proliferation according to the interleaving of complements with an arbitrary number of adjuncts, necessary for languages like Spanish, as examples in (41) show.[55]

(41) a. *Invitó a sus amigos el sábado a cenar.*
((he/she) invited his/her friends on Saturday for dinner.)



[55][Kasper 94] addresses the problem of interleaving adjuncts and complements in a flat structure for German in the framework of HPSG. Basically, [Kasper 94] assumes *head–comp–struc* sorts taking a list–valued attribute ADJ–DTR. Adjuncts select heads by the MOD feature which is split into SYN, unified with the head–dtr's CAT feature, and SEM, unified with the head–dtr and the adjunct which is immediately to the right on the ADJ–DTRS list. Problems, however, arise when getting ADJ–DTRS into the ADJ–DTRS–list representing scope (cf. [Schmidt 98]).

The problems of binary rules, however, are also well–known. There is no control either on the ordering over non–head elements which are not sisters, as we have just seen, nor on the order in which rules apply.

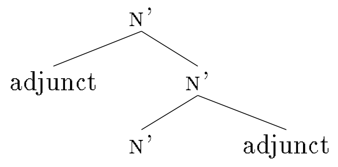
Lack of control on the order in which rules apply may cause spurious ambiguity (i.e. syntactic ambiguity which does not correspond to semantic ambiguity). For example, structure rules in (42.a) may produce both (42.c) and (42.d) in coping with structures like (42.b).

(42)

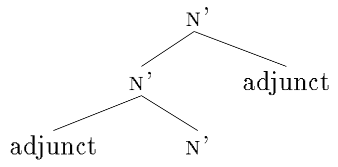
```
a. N' --> N' adjunct
   N' --> adjunct N'
```

```
b. principales mercados europeos.
   (main European markets).
```

c.



d.



We have just seen how by using an attribute encoding the projection level we can efficiently control the ordering of adjuncts and complements beyond our binary trees. Similarly, one may also use additional features to control attachment of constituents to the effect of avoiding overgeneration.

Borrowing from [Bennet & Schmidt 98], our grammar uses two further attributes, so called HEADING and HEADED, as a way to control rule application and to avoid spurious ambiguity.

These attributes are declared in the local attribute STR, and they are used in combination to record information about local headedness and, at the same time, to impose constraints on (or to make prediction about) local headedness.

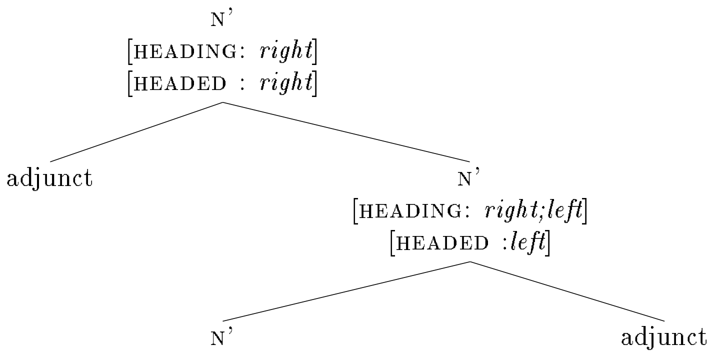
$$\begin{bmatrix} \textit{ld} & \\ \textrm{SYNSEM} \mid \textrm{LOCAL} \mid \textrm{STR} & \begin{bmatrix} \textit{t\_str} & \\ \textrm{HEADING} & \textit{boolean(left,right,solely,no)} \\ \textrm{HEADED} & \textit{boolean(left,right,solely,no)} \end{bmatrix} \end{bmatrix}$$

- The attribute HEADED, when specified for the mother–node, records information about local headedness, whereas when specified for a daughter node, it imposes a constraint on the headedness of this node.
  - *left* the node has a binary immediate dominance sub–structure with the linguistic head occurring to the left.
  - *right* the node has a binary immediate dominance sub–structure with the linguistic head occurring to the right.
  - *solely* the node has a unary immediate dominance sub–structure.
  - *no* the node has no immediate dominance sub–structure.
- The attribute HEADING, when specified for a daughter node, records information about local headedness, whereas when specified at the mother–node, it imposes a constraint on its potential to head a construction.
  - *left* in an immediate dominance structure, the node occurs as a left-peripheral head-daughter.
  - *right* in an immediate dominance structure, the node occurs as a right-peripheral head-daughter.

- *solely* in an immediate dominance structure, the node occurs as the only daughter.
- *no* the node does not occur as the head-daughter in an immediate dominance structure.

Overgeneration we showed in (42.c) and (42.d), thus, may be avoided by specifying that rules attaching a constituent in the function of adjunct to the left of the head–dtr creates a 'HEADING: *right*' node.[56]

(43)



2.6.4 The Parsing/Refinement Distinction

A distinctive feature of the ALEP processing architecture is the division of the analysis task into two sub–tasks: 'parsing', which builds up a complete but shallow phrase structure tree,[57] and 'refinement', which traverses the structure top–down, thus monotonically performing feature decoration, typically with semantic information.[58]

[56]Besides avoiding spurious ambiguity, the HEADING attribute plays the role of a rule application filter, reducing the number of rules to be accessed and thus improving efficient processing.

[57]Shallow in that it does not contain semantic information.

[58]The parsing/refinement distinction in ALEP, thus implements the two-pass parsing approach discussed by [Sikkel 97] where in the first pass a forest of context–free parse trees is constructed and in the second pass these parse trees are decorated. Trees with an inconsistent decoration are discarded.

During the refinement stage of linguistic processing, compositional meaning representations are obtained by recursive embedding of semantic feature structures corresponding to the constituents of the processed input text. In that, predications, argument and modifier relations and functional semantic information are encoded as individual semantic facts, marked by a unique wrapper data type, so called 'SF–terms' (SFs).

Links between SFs are established through sharing of variables figuring as arguments of these terms. This is illustrated by the following example:

(44) *Elaborar propuesta.*
(To elaborate proposal.)

$$\begin{bmatrix} \textit{t\_sem} & \\ \text{INDX} & \begin{bmatrix} \textit{t\_indx\_eve} \\ \text{IND: } \textit{sf(index(event,}\boxed{1}\textit{))} \\ \text{POL: } \textit{sf(polarity('1',}\boxed{1}\textit{))} \\ \text{TNS: } \textit{sf(tense(simul,}\boxed{4}\textit{,}\boxed{5}\textit{))} \\ \text{ASP: } \textit{sf(aspect(perf,}\boxed{1}\textit{,}\boxed{4}\textit{))} \end{bmatrix} \\ \text{PREDARG} & \begin{bmatrix} \textit{t\_predarg} & \\ \text{PRED: } \textit{sf(pred(elaborar,}\boxed{1}\textit{,}\boxed{2}\textit{,}\boxed{3}\textit{))} & \\ \text{ARGS:} & \begin{bmatrix} \textit{t\_args} & \\ \text{ARG1:} & \begin{bmatrix} \textit{t\_sem\_arg} \\ \text{REL: } \textit{sf(rel(arg1,}\boxed{1}\textit{,}\boxed{2}\textit{))} \end{bmatrix} \\ \text{ARG2:} & \begin{bmatrix} \textit{t\_sem\_arg} & \\ \text{REL: } \textit{sf(rel(arg2,}\boxed{1}\textit{,}\boxed{3}\textit{))} & \\ \text{INDX:} & \begin{bmatrix} \textit{t\_indx} \\ \text{IND: } \textit{sf(index(nevent,}\boxed{3}\textit{))} \\ \text{POL: } \textit{sf(polarity('1',}\boxed{1}\textit{))} \end{bmatrix} \\ \text{PREDARG:} & \begin{bmatrix} \textit{t\_predarg} \\ \text{PRED: } \textit{sf(pred(propuesta,}\boxed{3}\textit{))} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{MODS} & \langle\rangle \end{bmatrix}$$

Based on the SF–encoding scheme, a flat list of all SFs representing the core linguistic meaning of an input expression can be extracted from the output structures, which can then be passed into a semantic analysis module.

(45)

```
[pred(elaborar,elaborar,X,propuesta),pred(propuesta,propuesta),
 index(event,elaborar),index(nevent,propuesta),
 rel(arg1,elaborar,X),rel(arg2,elaborar,propuesta),
 tense(simul,R,U),aspect(perf,elaborar,R),
 polarity('1',elaborar),polarity('1',propuesta)].
```

Predicative relations expressed by content words are encoded by the term *pred*. The first argument of the term *pred* encodes the particular concept expressed by the predication, and as the second argument, it introduces the marker of the denoted entity. Every further argument of a *pred*–term of arity larger that 2 introduces a variable corresponding to an entity participating in the predication, an argument of the predicate.[59] Whether denoted entities and entities participating in the predication are events or not is encoded by the *index*–term, which is of arity 2. The first argument encodes the type of index: *event, nevent*. The second argument takes a variable which is shared with the variables corresponding to the entities.

Argument relations are encoded by the term *rel* which is of arity 3. As the first argument, *arg*–terms encode the semantic role the argument takes in the predication. As we have already said in section 2.6.1, a rather small set of argument relation labels is assumed, with external arguments and least oblique arguments being assigned *arg1* and *arg2* respectively, a number of semantically interpreted argument relation labels, such as *arg_place* or *arg_goal*, is reserved for arguments the meaning of which resembles that of adjuncts. As their second and third arguments, *arg*–terms introduce variables, the first of these variables is shared with the variable figuring of the second argument of the *pred*–term, and the second one is shared with the variable corresponding to the entity participating in the predicate relation and expressing the argument itself.[60]

Our account of temporal semantics accounts for two dimensions of temporal information: tense and aspect. In the tradition of [Reichenbach 47], tense is conceived of as a relation between the time of reference and the time of utterance,

[59]We assume that the maximal arity of predicates is 4.

[60]Modifier relations are also encoded by the term *rel*. For adjuncts, a standard set of modifier relations is used including e.g. *place*, *goal*, *origin*, *beneficiary* or *instrument*. The variables introduced as the second and third arguments are shared with the variable variable representing the predicate relation being modified and the variable expressing the modification.

whereas aspect is conceived of as a relation between the time of event and the time of reference. Following this tradition, we encode tense and aspect relations by two terms (*tense*,*aspect*), specifying as their first argument the particular type of tense relation, respectively aspect relation, and introducing as their second and third argument two variables serving as markers of the time of reference and the time of utterance, respectively of the time of event and the time of reference.

The tense relations that may hold between the time of reference R and the time of utterance U are:

`anteriority :`	`R precedes U`
`simultaneity:`	`R equals or includes U`
`posteriority:`	`R follows U`

The aspect relations that may hold between the time of event E and the time of reference R are:

`perfective :`	`E equals R, or R includes E`
`durative :`	`E includes R`
`terminative :`	`E overlaps with R`
`inchoative :`	`R overlaps with E`
`retrospective:`	`E precedes R`
`prospective :`	`R precedes E`

Within the semantic representation of any tensed clause, the variable figuring as the second argument of the main predicate and marking the denoted event will be shared with the variable figuring as the second argument of the *aspect* term, such that this variable is now given the interpretation of a marker of the event time location.The variable figuring as the second argument of the *tense*–term and the variable figuring as the third argument of the *aspect*–term will be shared as the unique marker of the time of reference. Obviously, the variable U marking the time of utterance will be time-stamped. The variable R marking the time of reference, on the other hand, may be constrained by means of locational time adverbials which will be interpreted as (restrictive) modifiers of the time of reference.

Finally, as a means of encoding the fact that some semantic entities are in the scope of negation, we have a 2-ary term *polarity* which encodes, by it first argument, either affirmation ('1') or negation (0) relative to the variable figuring as its second argument being affirmed, respectively negated.

The parsing/refinement distinction in ALEP is based on *specifier* declarations, i.e. a particular set of feature structures which are picked out by a special feature path declaration. *Specifiers* allow grammar developers to encode membership of a grammar rule (lexical entry, structural rule) in one or more type of rules.[61]

[Theofilidis & Schmidt 97] indicate how such internal classification of grammar code was made use of to support distributed grammar engineering in the LS–GRAM project.

In the following we describe how we have deployed the parsing/refinement distinction facility provided by the ALEP system in order to reduce the ambiguity seen by the parser to the effect of developing efficient unification–based grammars.

Prepositions in Spanish are heads of structures which can function both as modifiers (46.a) and as arguments (46.b), each function imposing a different semantic structure to the prepositional entry. Argumental preposition are specified as taking as value a type *t_sem_arg* for the attribute SEM and they instantiate argument–slots within the argument structure of the predicate which subcategorize for them at the level of syntactic description, while modifiers are specified as taking as value a type *t_sem_mod* for the attribute SEM and they insert their semantic features into the semantic MODS–list of the modified element, thus imposing additional restrictions to the index. Therefore, two different entries are needed.

(46) a. *La guerra de las galaxias.* (Star Wars.)

b. *El retorno del Jedi.* (Return of the Jedi.)

In our ALEP grammar, however, such ambiguity has been moved from the parsing to the refinement task. Instead of having two fully specified lexical entries that apply during the tree–building operation, the parsing/refinement distinction allows us to specify the same information in one lexical entry that applies in parsing and two entries that apply in refinement. The parsing entry (47) leaves the value for SEM unspecified, which is fully specified in the two refinement entries, for modifying prepositions (48), and for argumental prepositions (49).[62]

[61]In addition to this 'vertical dimension', grammar partitions may be established along a 'horizontal dimension', i.e. according to different types of structural units being involved in the parsing operation. Structure rules may be specified to be applied only when parsing 'morphemes to words', dealing with morphological analysis, 'words to sentences', or 'sentences to paragraphs', when paragraphs are processed by the grammar.

[62]The relationship between the two sets of lexical entries —and the structure rules we will see below— is that of subsumption (cf. section 2.4.1).

(47)

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{ld}} \\ \multicolumn{2}{l}{\text{SPEC}\,|\,\text{PARSING}:\ \textit{yes}} \\ \text{SYNSEM}\,|\,\text{LOCAL}\,|\,\text{CAT}: & \left[\begin{array}{l} \textit{t\_subst} \\ \text{HEAD}:\ \textit{t\_prep} \\ \text{COMPL}:\ \left\langle \left[\begin{array}{l} \textit{t\_synsem} \\ \text{LOCAL}\,|\,\text{CAT}\,|\,\text{HEAD}:\ \textit{t\_noun} \end{array}\right] \right\rangle \end{array}\right] \end{array}\right]$$

(48)

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{ld}} \\ \multicolumn{2}{l}{\text{SPEC}\,|\,\text{REFINEMENT}:\ \textit{yes}} \\ \text{SYNSEM}\,|\,\text{LOCAL}: & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t\_local}} \\ \text{CAT}: & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t\_subst}} \\ \text{COMPL}: & \left\langle \left[\begin{array}{l} \textit{t\_synsem} \\ \text{LOCAL}\,|\,\text{SEM}:\ \boxed{4} \end{array}\right] \right\rangle \\ \text{MODIFIES}: & \left\langle \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t\_modifies}} \\ \text{MODIFIED}\,|\,\text{LOCAL}\,|\,\text{SEM}: & \left[\begin{array}{l} \textit{t\_sem} \\ \text{INDX}:\ \boxed{1} \\ \text{PREDARG}:\ \boxed{2} \\ \text{MODS}:\ \boxed{3} \end{array}\right] \end{array}\right] \right\rangle \end{array}\right] \\ \text{SEM}: & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t\_sem}} \\ \text{INDX}: & \boxed{1} \left[\begin{array}{l} \textit{t\_indx} \\ \text{IND}:\ \textit{sf(ind(\_,}\boxed{5}\textit{))} \end{array}\right] \\ \text{PREDARG}: & \boxed{2} \\ \text{MODS}: & \left\langle \boxed{3}\,\middle|\, \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t\_sem\_mod}} \\ \multicolumn{2}{l}{\text{REL}:\ \textit{sf(rel(\_,}\boxed{5}\textit{,}\boxed{6}\textit{))}} \\ \text{INDX}: & \left[\begin{array}{l} \textit{t\_indx} \\ \text{IND}:\ \textit{sf(ind(\_,}\boxed{6}\textit{))} \end{array}\right] \\ \multicolumn{2}{l}{\text{PREDARG}\,|\,\text{ARGS}\,|\,\text{ARG2}:\ \boxed{4}} \end{array}\right] \right\rangle \end{array}\right] \end{array}\right] \end{array}\right]$$

(49)

$$\begin{bmatrix} \textit{ld} & \\ \text{SPEC}\,|\,\text{REFINEMENT:} & \textit{yes} \\ \text{SYNSEM}\,|\,\text{LOCAL:} & \begin{bmatrix} \textit{t\_local} & \\ \text{CAT:} & \begin{bmatrix} \textit{t\_subst} & \\ \text{COMPL:} & \left\langle \begin{bmatrix} \textit{t\_synsem} \\ \text{LOCAL}\,|\,\text{SEM: } \boxed{1} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{SEM:} & \begin{bmatrix} \textit{t\_sem\_arg} \\ \text{PREDARG}\,|\,\text{ARGS}\,|\,\text{ARG2: } \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

In [Bredenkamp & Hentze 95], the same strategy is used to avoid ambiguity between prepositions which may modify both nominal and verbal head–dtrs. Note, however, that underspecification of modifying prepositions may result in overgeneration in parsing, when, as in our grammar, the adjunct selects the element it modifies, since they apply more freely.[63]

Similarly, [Theofilidis & Reuther 95] use the parsing/refinement distinction to postpone an account of selectional requirements with respect to the saturation of the SUBJ–list. This avoids lexical disjunction due to frame variation when a governed function e.g. direct object may be realized by more than one syntactic category.

A similar use of the parsing/refinement distinction may be observed in the treatment of possessive determiners.

Besides instantiating the functional semantic information related to determination, possessive determiners may simultaneously qualify as an argument of the head noun, e.g. external argument (50.a), oblique argument (50.b).

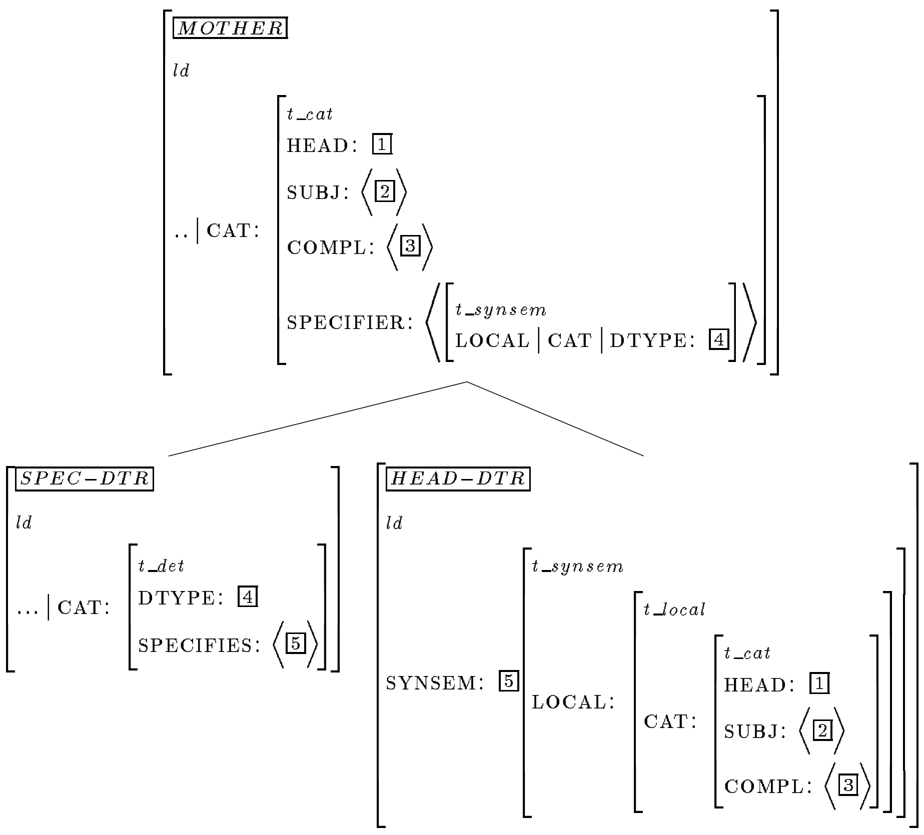
(50) a. *Su boda* = La boda de Murriel
(her wedding = Murriel's wedding)

b. *Su padre* = El padre de la novia
(her father = The bridge's father)

[63]Overgeneration due to semantic unspecification is also observed in [Oepen & Carroll 00].

Semantic distinctions between possessives is captured in the phrase structure rule component of the grammar. We have implemented a parsing rule attaching specifiers to nominal heads (51). Then, the specific semantic contributions characteristic of possessive determiners according to the head noun are accounted for by a number of distinct refinement instantiations of this rule, e.g. (52) for possessives qualifying as external argument, and (53) for possessives qualifying as the least oblique argument.[64]

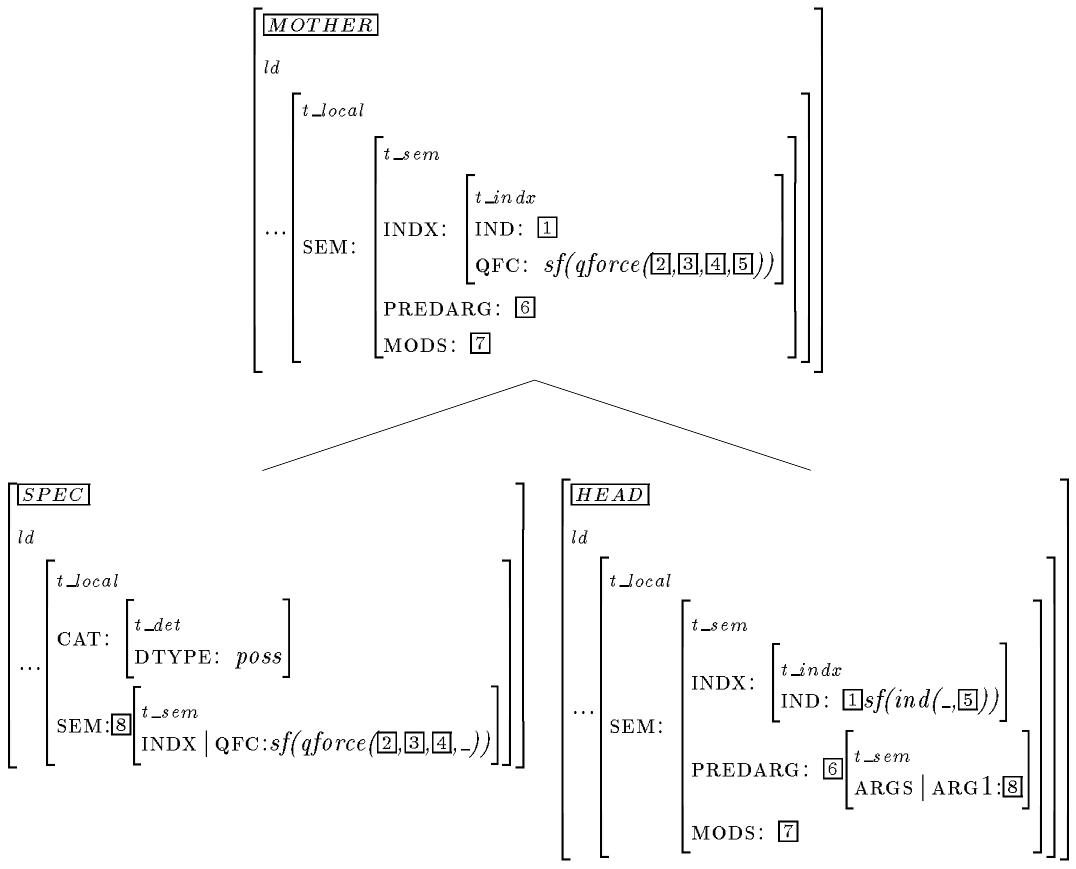
(51) [65]



[64]Note that with bi–valent predicative nouns such as *su venta* ((his/her) sale), two results are provided, unless the least oblique argument is present, e.g. *su venta de acciones* ((his/her) sale of stock shares). The advantage of coping with the specific semantic contributions characteristic of possessive determiners at the refinement component is that overgeneration is produced at the refinement processing stage.

[65]Dots (...) abbreviate the path SYNSEM|LOCAL.

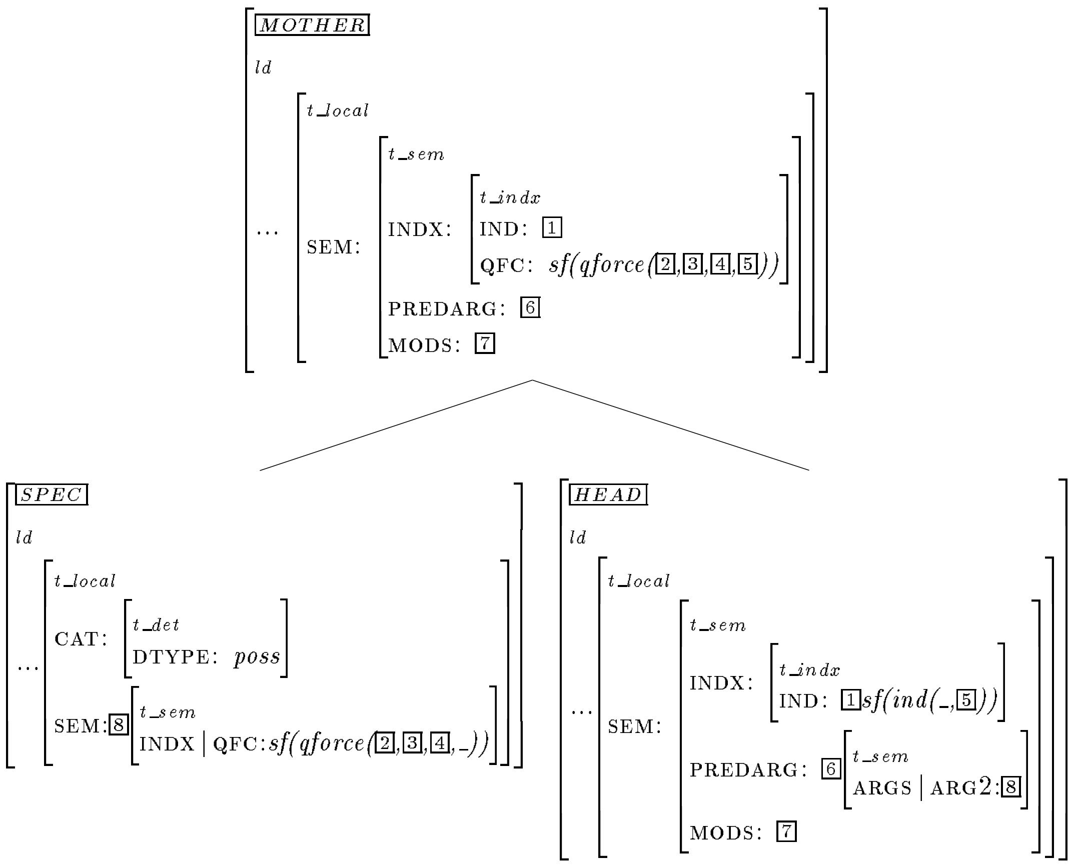
(52) [66] [67]



[66]Dots (...) abbreviate the path SYNSEM|LOCAL

[67]We encode the quantificational force of a determiner —we use the notion of determiner in a broad sense covering articles, possessives, quantifiets, etc.— by the term 'qforce', encoding as its first argument the particular determiner, and as its fourth argument the variable bound by the determiner. Determiners are also characterized as to whether a singularity or plurality is denoted by the bound variable, which we encode in the second argument, and whether the entity is known or unknown, this distinction is captured by the notions of 'def(inite)' and 'indef(inite)' determination and it is encoded in the third argument.

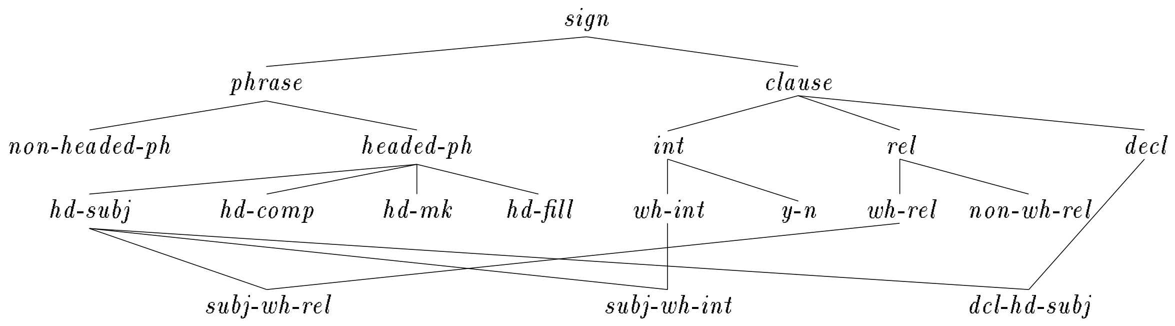
(53) [68]



[68]Dots (...) abbreviate the path SYNSEM|LOCAL.

The parsing–refinement distinction, in addition, allows us to capture the relationships between different types of constructions which in more recent proposals within HPSG theory ([Sag 97]) are captured in a multiple inheritance hierarchy of *phrase* types (cf. (54)), to the main effect of reducing the number of phrase structure rules to be applied during parsing.

(54)



The basic idea behind the analysis proposed in HPSG described by [Sag 97] is that different types of constructions, as the ones in (55),[69] are instances of the same type of headed–phrase (i.e. *hd–subj*), and that differences among them (e.g. the specifications for the non–local attributes REL and QUE) are captured via an additional type hierarchy characterizing clause types.

(55) a. Leslie always drinks milk. (declarative clause)

b. (Leslie,) who always drinks milk. (relative clause)

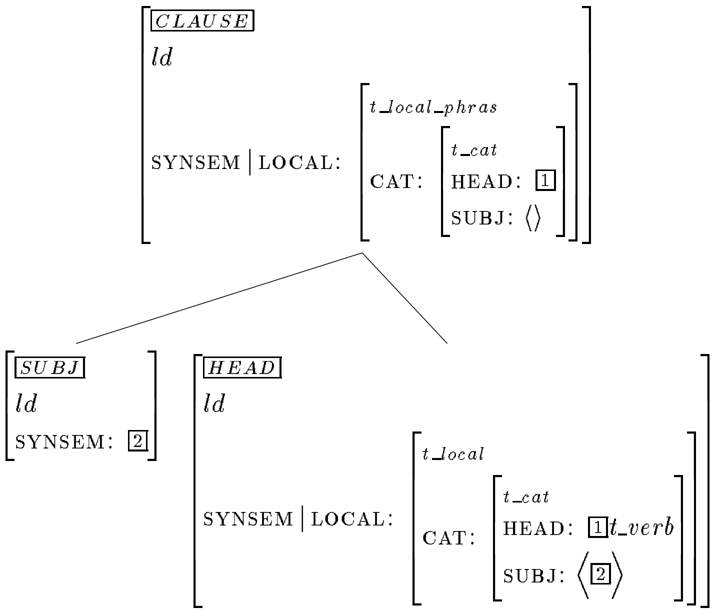
c. Who always drinks milk? (interrogative clause)

Following [Bredenkamp & Hentze 95], we have used the parsing/refinement distinction to avoid rule proliferation at the parsing processing task. Thus, instead of diversifying parsing phrase structure rules on account of the structures in

[69]We take from [Sag 97].

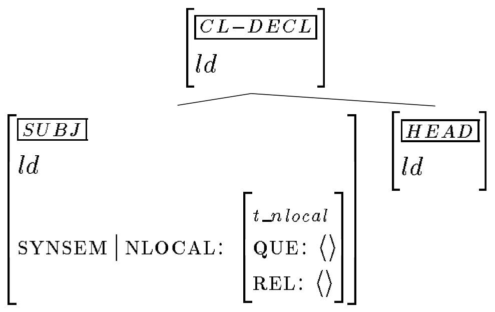
(55), we have a single shallow parsing rule which only encodes the linguistic constraints on *head–subj* type (i.e. head–subj schema) (56). This rule subsumes three refinement rules which impose the requirements on the values for the non–local features of the non–head daughter required by declarative clauses (56), interrogative clauses (57), and relative clauses (58).[70]

(56)

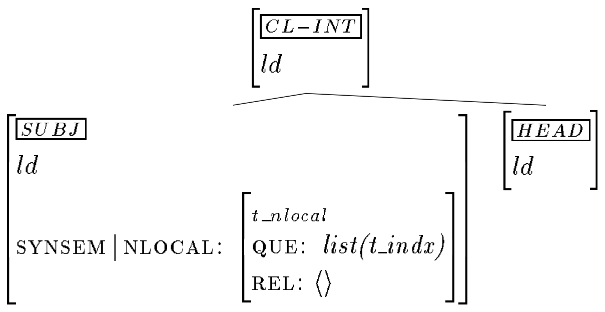


[70]Non–local features are declared in the type *t_nlocal*, which appears as value of the attribute SYNSEM|NLOCAL. The type *t_nlocal* structures the information related to the treatment of long distance dependencies (topicalization, relative clauses and interrogative clauses) and it takes three attributes: SLASH, which takes as value a list of *t_local* types, QUE which takes as value a list of *t_indx* types, and REL which takes as value a list of *t_indx* types.

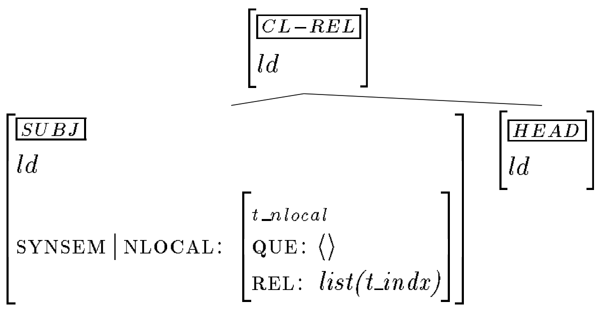
(57)


CL−DECL
ld
SUBJ
ld
SYNSEM | NLOCAL:
t_nlocal
QUE: ⟨⟩
REL: ⟨⟩
HEAD
ld

(58)


CL−INT
ld
SUBJ
ld
SYNSEM | NLOCAL:
t_nlocal
QUE: list(t_indx)
REL: ⟨⟩
HEAD
ld

(59)


CL−REL
ld
SUBJ
ld
SYNSEM | NLOCAL:
t_nlocal
QUE: ⟨⟩
REL: list(t_indx)
HEAD
ld

This section has shown how the ALEP refinement facility allows to shift the computational burden of accounting for semantic ambiguity from the computationally expensive parsing operation to the stage of refinement. The engineering solutions we propose constitute a first step towards distributing the analysis process for efficient processing.

In the following chapters we will describe how the search space of the parser can be further pruned by the integration of shallow techniques. Our goal in that is to release the parser from certain tasks —PoS disambiguation and partial parsing— which may be efficiently and reliably dealt with by less expensive processing computational techniques. We will first describe the shallow processing tools we have used, then, in chapter 4, we will discuss our integrated system.

Chapter 3

A Linguistic Tagger and Chunker of Spanish

This chapter addresses shallow linguistic processing and it presents a linguistic tagger and chunker of Spanish. The tool we present here is the one we have used to integrate shallow and deep processing.[1]

An introductory section presents the existing techniques for PoS tagging and partial parsing and their more representative works. The second section describes the architecture of the tagger and chunker. Sections three and four describe the work done for adapting and extending existing tools and resources for tokenizing input text, morphological analysis and handling unknown words. The last section presents the tool we have used for disambiguation and chunking.

[1]The tool we describe in this chapter was developed by Jordi Porta. The author of this thesis was fully in charge of the design and implementation of both disambiguation and chunking rules.

3.1 Introduction

3.1.1 Part of Speech Tagging

Part of Speech (PoS) tagging basically consists of: (i) attaching lexical morphological analyses (in the form of tags or labels) to words in text, and (ii) disambiguating (choosing the appropriate analyses) on the basis of their contextual constraints.

Introducing morphological analyses is quite straightforward, and it can be performed by means of a morphological analyzer or a simple lexical look–up. Choosing the proper analysis in a given context is not such a trivial task. Even though in many cases morpho–syntactic information about immediate preceding and/or following tokens is enough to disambiguate ambiguous lexical items, neighbouring tokens do not always provide the necessary information to deal with several ambiguity cases, so that contextual constraints must then go far beyond neighbouring tokens. Besides, there may be also cases which can not be solved on the basis of morpho–syntactic information, but need some semantic, pragmatic, or even extra–linguistic information.

Nevertheless, the benefits obtained from a tagged corpus clearly surpass the difficulties. Typical (research and development) applications that can benefit from disambiguated text are: syntactic annotation: partial parsing [Abney 91, Karlsson *et al.* 95] and parsing [Oostdijk 91], semantic annotation [Janssen 90, Wilson & Rayson 93], information extraction: collocations [Church & Hanks 90], frequency lists [Francis & Kucera 82] and idioms [Johansson & Hofland 89], information retrieval [Croft *et al.* 91, Salton *et al.* 90], word processing (e.g. spell and grammar checking) [Carl & Schmidt-Wigger 98], speech processing: speech synthesis [Liberman & Church 91] and speech recognition [Jelinek *et al.* 92], machine translation [Music 96, Music 97], machine aided–translation [Brown *et al.* 88], access to data bases [Kupiec 93b], and lexicography [Summers 96, Church & Gale 91, Klavans & Tzoukermann 90].

Therefore, not surprisingly, the tagging problem has received so much attention in the past —and still a lot of effort is currently being devoted in order to improve the existing techniques. Nowadays, taggers have been developed for a great variety of languages, apart from English, some examples are: [Chang & Chen 93] for Chinese, [Hurskainen 96] for Swahili, [Oflazer & Tür 97, Tür 96, Oflazer & Kuruöz 94] for Turkish, [Cutting 94, Brants & Samuelsson 95] for Swedish, [Hajic & Hladká 98] for Czech, [Feldweg 95, Schmid 95] for German, [Daelemans *et al.* 96] for Dutch, [Dermatas & Kokkinakis 95] for Greek and Italian, [Chanod & Tapanainen 95] for French, [Aduriz *et al.* 95] for Basque, [Marques & Lopes 96] for Portuguese, [Sánchez 97, Padró 98, Sánchez & Nieto 95, Márquez 99] for Spanish. The MULTEXT project has developed taggers for 6 languages [Armstrong *et al.* 95].[2]

Current approaches to PoS tagging can be classified in two broad families on the basis of the the way in which the language model (i.e. the correct sequence of tags for the language to be disambiguated that the tagger uses) is acquired: automatic data–driven taggers and linguistic taggers. Supplementary to them, a third family includes hybrid taggers.

In what follows, we will present these approaches to PoS tagging and their most representative works.[3]

a)– Data–Driven Taggers

In this approach, distributional generalizations are automatically acquired from a corpus by means of statistical inference processes. The derived knowledge may then be represented in different ways.

The first work of this approach was the CLAWS1 (Constituent–Likelihood Automatic Word tagging System, version 1)[4] system [Garside *et al.* 87], which was later improved by using dynamic programming by DeRose in the VOLSUNGA system [DeRose 88].[5] CLAWS1 used a matrix of collocation probabilities, which indicated the relative likelihood of co–occurrence of all ordered pairs of tags, mechanically derived form a pre–tagged corpus. This system reached an accuracy of 96–97%.

In another line of development, Hidden Markov Models (HMMs) were imported from speech recognition and applied to PoS tagging (cf. [Abney 96b]). Basically, a HMM is a probabilistic finite state automaton (i.e. a finite state machine with probabilities determining transitions between the states and con-

[2][Bel *et al.* 96b] describe the work done for using the MULTEXT tagger for Spanish.

[3]See [Halteren 99] for a detailed presentation of PoS tagging techniques and related issues e.g. tagsets, lexicons for tagging, and tagging unknown words. See also [Márquez 99].

[4]This system has currently reached version 4 [Garside & Smith 97].

[5]The VOLSUNGA system was faster, but not more accurate.

trolling the emission of outputs).[6] For the tagging problem, states are tags,[7] transition probabilities are probabilities of a tag given a previous tag, and emission probabilities are probabilities of a word for a given tag. Applying a HMM consists of: (i) estimating the model parameters, and (ii) computing the most likely sequence of state transition (i.e. recovering the most likely tag by selecting the tag with the maximum score from all possible tags). The most–likely sequence of tags can then be calculated using the Viterbi algorithm [Viterbi 67].

In this model, two types of training (or parameters estimation) have been used: supervised training [Bahl & Mercer 76, Church 88, Derouault & Merialdo 84] and unsupervised training [Cutting *et al.* 92, Kupiec 92] —in both cases, though, the corpus will be split into the training part and the validation part. The first one employs a training corpus which has been manually disambiguated to calculate the parameters of the model. *Smoothing techniques* (i.e. counterbalacing techniques) may be used when frequencies are small, so–called sparse data problem.[8] The second method does not require an unambiguous training corpus, in which case the probabilities on the transitions and emission can be re–estimated using the Baum–Welch algorithm (also known as forward–backward algorithm) [Baum 72]. But the best results seem to be obtained with annotated corpora (cf. [Elworthy 94, Merialdo 94]).

The maximum score is calculated on the basis of two probabilities: the language model Pr(T), which establishes the correct sequence of tags, and the communication model Pr(W|T), which establishes the relationship between the tag and its lexical realization:

$$\max_T \Pr(W|T)\Pr(T)$$

The communication model is calculated as the product over the possibilities for its components:

$$\Pr(W|T) \approx \prod_{i=1}^{n} \Pr(w_i|t_i)$$

As for the language model, the simplest technique consists of considering each tag of T independent. The probability of the sequence is thus reduced to the product over the probabilities of each tag:

[6]Standard references for HMMs are [Rabiner 90] and [Kupiec 92].

[7]States are tags in a bigram model, and pairs of tags in a trigram model.

[8][Márquez 99] cites [Church *et al.* 96, Chen & Goodman 96] as surveys and evaluations of several smoothing techniques.

(60)

$$\Pr(T) \approx \prod_{i=1}^{n} \Pr(t_i)$$

With such a simple algorithm in which each word only depends on its own tag, one may achieve slightly over 90% accuracy (cf. [Rodríguez 99]). Better results —95–97% accuracy— are achieved when a word's context is taken into account. The unigram model (60) can thus be extended to a bigram model (61.a) where the probability of each tag will be conditioned by the previous one, or, even, to a trigram model, (61.b) where the probability of each tag will be conditioned by the previous two tags.

(61) a.

$$\Pr(T) \approx \Pr(t_1) \prod_{i=2}^{n} \Pr(t_i|t_{i-1})$$

b.

$$\Pr(T) \approx \Pr(t_1)\Pr(t_2) \prod_{i=3}^{n} \Pr(t_i|t_{i-2}t_{i-1})$$

Inductive machine learning approaches have also been applied to PoS tagging, e.g. connectionist learning algorithms [Schütze 93, Schmid 94], automatic induction of decision trees [Magerman 95], example–based learning techniques [Daelemans *et al.* 96].

Finally, PoS tagging has also investigated the acquisition of disambiguation rules automatically. This method was originally investigated by Hindle in his *Fiddicht* system [Hindle 89], but the best known system is Brill's *transformation–based error–driven learning* algorithm to induce rules from manually tagged corpora [Brill 92].[9]

b)– Linguistic Taggers

In this approach, distributional generalizations used in the disambiguation process are manually encoded by a grammarian in the form of a set of constraint rules which promote or discard appropriate or illegitimate analyses, respectively.

[9]More recently, Brill presented a version with unsupervised training [Brill 95].

The earliest linguistic PoS tagger dates back to the seventies.[10] In 1971, Greene and Rubin developed the tagger TAGGIT for tagging the Brown Corpus [Greene & Rubin 71]. TAGGIT used a set of 3,300 context rules —limited to up to two words on each side of the word to which the rule was applied— to eliminate illegitimate/promote appropriate tags from the list of possible tags for one word.

The bad results of this first rule–based tagger —77% accuracy— were not very encouraging, so this line of development was abandoned in favour of the statistical approaches. However, since the very first taggers until nowadays, several techniques have been developed and a number of experiments for different languages show that the performance of rule–based taggers is currently better than that of the data–driven ones [Chanod & Tapanainen 95, Samuelsson & Voutilainen 97].

The most important work in this direction is the system developed in the early 1990s by a group of researchers from the University of Helsinki: the English Constraint Grammar (EngCG) [Voutilainen *et al.* 92, Voutilainen & Heikkilä 94] based on the Constraint Grammar (CG) framework [Karlsson *et al.* 95], originally proposed by [Karlsson 90]. A CG is a collection of rules that specifies one (or more) context constraints where the tag is illegitimate. The context patterns can be local or global, and they can refer to ambiguous or unambiguous analyses. With this formalism and a grammar of about 1,100 rules, they achieved 99.77% recall and 95.54% precision.[11] In more recent work within the CG formalism they present a system which combines the EngCG disambiguator with a finite–state syntactic parser: the Finite–State Intersection Grammar, originally proposed by [Koskenniemi 90], which resolves remaining part–of–speech ambiguities as a side effect [Koskenniemi *et al.* 92, Voutilainen 94, Voutilainen 95, Voutilainen & Tapanainen 93]. The grammatical representation used in the Finite State framework is an extension of the EngCG syntax where surface–syntactic grammatical relations are encoded with dependency–oriented functional tags (subject, pre–modifier, adverbial, etc.) where the dependency structure of the sentence is expressed by the pointers of the syntactic tags. Here, syntactic analysis also means resolution of structural ambiguities. Morphological, syntactic and clause boundary descriptors are introduced as ambiguities, which are resolved in

[10]Klein and Simmons described an earlier tagger in 1963 [Klein & Simmons 63]. This, however, was an initial categorial tagger rather than a disambiguator, whose primary goal was to avoid "the labor of constructing a very large dictionary". This method reached an accuracy of 90%.

[11]The terms recall, precision and accuracy, which we explain in section 3.5.7, are used to evaluate the resulting disambiguated text.

parallel. The system reached an accuracy of over 99% (cf. [Voutilainen 95]) at the level of morpho–syntactic tags, but the tags for syntactic functions were not evaluated.

More recently, [Oflazer & Tür 97] present a rule–based morphological disambiguator where rules are composed by the feature constraints and a vote — determined by the static properties of the rule. Disambiguation, thus, is performed at the end by selecting the analyses that receive the highest votes, thus making the outcome of the disambiguation independent from the rule sequence.

The use of constraints for morphological ambiguity resolution has also been applied to Finish, Swedish, Danish, Basque, Swahili, German, Portuguese, Spanish, French and Turkish (cf. [Halteren 99]).

c)– Hybrid Taggers

Finally, there are also systems which combine linguistic and data–driven methods in different ways.

The CLAWS4 system uses a hand–coded component dealing with collocations to avoid errors at the statistical module [Leech *et al.* 94]. Tapanainen and Voutilainen present a system which combines the EngCG system and the Xerox HMM tagger, applying the latter to resolve the ambiguities which remain after having applied the former [Tapanainen & Voutilainen 94]. [Samuelsson *et al.* 96] implement a system to induce CGs to create a first version of the grammar. Whereas [Tür 96, Oflazer & Tür 96] investigate the use of hand–crafted rules and rules derived from corpora just the other way round, they use hand–crafted linguistic rules for resolving some of the ambiguities before employing constraint rules automatically derived from corpora. Finally, in [Fligelstone *et al.* 97] linguistic rules are written to correct errors made by a data–driven tagger.

3.1.2 Partial Parsing

Partial parsing is a term covering different techniques for obtaining some of the existing syntactic relationships among the lexical items in a sentence, rather than recovering a complete tree structure, as in traditional syntactic analysis. These parsers aim at reliability and robustness (i.e. extracting the maximum amount of information from unrestricted text trying to avoid risky predictions).

Though not complete, the syntactic information recovered by these systems has proved to be useful in such applications as information extraction used in more complete parsers or data extraction systems.

Following [Rodríguez 99], we have classified partial parsers on the basis of kind of syntactic relationships they recover. In this way, we have co–occurrence parsers, structural parsers, shallow parsers and phrasal parsers.

The different methods for acquiring and representing the language models applied in these systems are almost the same as in PoS tagging we have discussed in the previous section: hand–crafted grammars, finite state techniques, statistical methods, transformation–based learning mechanism.

a)– Co–occurrence Parsers

Co–occurrence parsers aim at deriving syntactical co–occurrence instances between lexical items derived from (unrestricted) text. Here, the different approaches described in the literature differ in the amount of information they use (cf. [Rodríguez 99]).

Examples are Smadja's *XTRACT* system [Smadja 93] and the systems presented in [Calzolari & Bindi 90, Church & Hanks 90, Resnik 93, Yarowsky 92].

b)– Structural Parsers

Structural parsers are parsers which, even though they use a complete grammar and try to produce a complete analysis of the input sentences, they are able to allow a partial analysis in case they can not produce a complete analysis of the sentence, or to leave phrases unattached, when their role can not be determined.

Most relevant works in this direction are Hindle's *Fidditch* deterministic parser [Hindle 83, Hindle 94] and de Marcken's *MITPF* system [Marcken 90].

c)– Shallow Parsers

Systems for shallow analysis usually work on top of PoS taggers. They provide each word with its corresponding part of speech tag plus a syntactic tag indicating their surface–syntactic function (subject, pre–modifier, adverbial, etc.).

These system have been mainly developed in the CG framework (cf. section 3.1.1). Other systems are [Joshi & Srinivas 94] and [Giguet & Vergne 97] for robust syntactic analysis of unrestricted French.

d)– Phrasal Parsers (or Spotters)

Spotters aim at recognizing low–level phrases (NPs, PPs, VGs, ...) where no information is provided with respect to their internal structure nor their function, by means of very simple, but specialized and efficient processors (finite state machines, context free probabilistic grammars, heuristic rules, etc.) (cf. [Rodríguez 99]).

Fall in this group: Bourigault's LEXTER system for extracting terminological noun phrases from French text [Bourigault 92], Church's stochastic system to mark nominal chunks [Church 88], Voutilainen's NPtool [Voutilainen 93], [Hobbs *et al.* 96]'s FASTUS system, Schwarz's Copsy, a dependency parser for extracting multi–word terms [Schwarz 90], Kupiec's finite state NP recognizer for English and French [Kupiec 93b], Ramshaw and Marcus's text chunking system, which uses Brill's transformation–based learning technique [Ramshaw & Marcus 95], and Abney's CASS system, a chunk parsing technique based on finite–state cascades [Abney 96c]. Inspired by Abney's system, Ciravegna and Lavelli have developed an approach to full parsing where sequences of cascades of rules deterministically analyze the text [Ciravegna & Lavelli 99, Ciravegna & Lavelli 00].

3.2 The Architecture of Tagger and Chunker

The tool we have used for shallow processing for Spanish includes the following three modules:

- `The Tokenizer` (`TagIt`), which, on the one hand, identifies and classifies the tokens of the input text, and, on the other hand, performs morphological analysis (section 3.3).
- `The Unknown Word Processor` (`Guesser`), where tokens which have not received an analysis are parsed, thus adding some degree of robustness to the shallow processing tool (section 3.4).
- `The Disambiguator and Chunker` (`Latch`), in charge of both disambiguating PoS tags and marking partial intra–clausal constituents (section 3.5).

The overall architecture of the system is outlined in Figure 3.1.

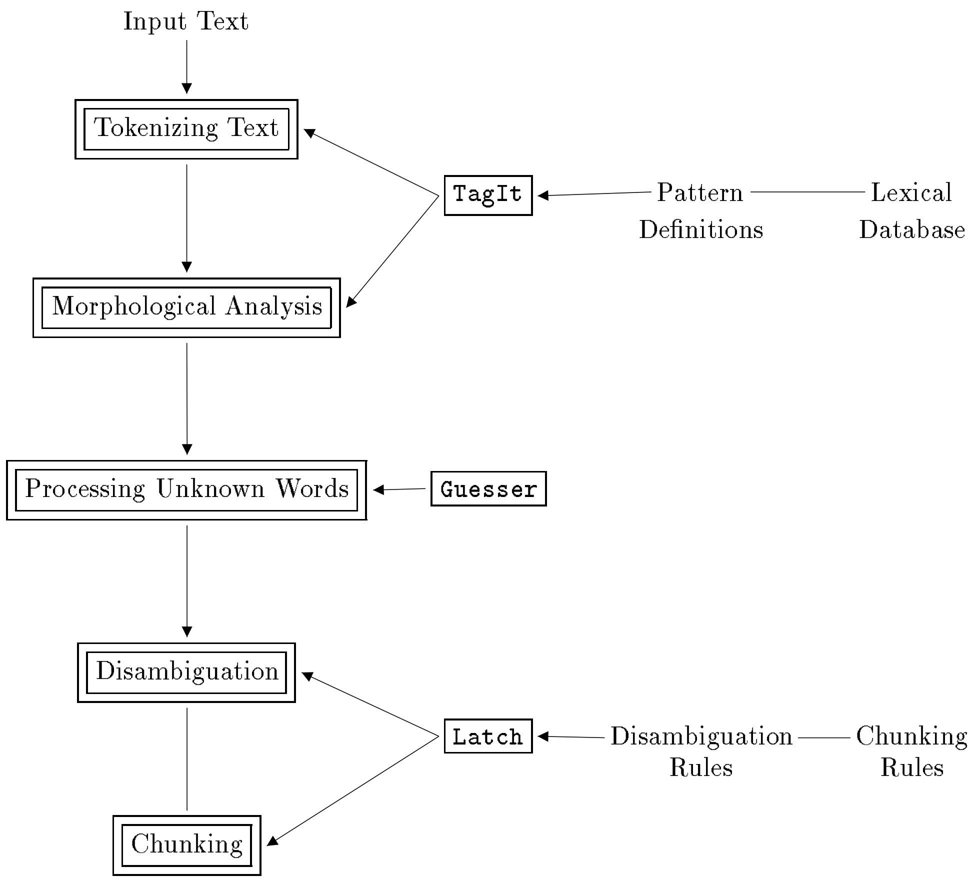


Figure 3.1 Architecture of the tagger and chunker.

3.3 The Tokenizer

Raw text is pre–processed by a tokenizer. The main task of a tokenizer is to identify and to classify the elements of the input text.

Our tokenizer —`TagIt`— is implemented in *perl* (*practical extraction and report language*) and it is designed as a set of modular components which may be activated (or deactivated) when appropriate.

`TagIt` was developed for the MELISSA project[12] (cf. section 2.1) and it was conceived to prepare the input text for deep linguistic analysis within the ALEP framework.[13] The first task of `TagIt` is to delimit the basic textual units and to organize the input text into a three–level tree structure. `TagIt` is configurable with respect to its output format, the default being a SGML mark–up. The highest level unit is the paragraph (<P> ... </P>). Then, the input text is split into sentences (<S> ... </S>). Here, language specific databases containing known abbreviations are consulted to avoid errors when sentences are split. Finally, each sentence is divided into words (<W> ... </W>).

`TagIt` is also used to identify special expressions, such as numbers and dates, and application specific constructs, which have been considered peripheral to a lexicon. These constructs are identified by means of regular expressions or fast look–up at tables and are marked up with an appropriate value for the tag attribute 'TYPE'. When appropriate, they are parsed to produce a normalized value (e.g. an ISO representation for dates).

During the MELISSA project, the list of special items to be covered was basically derived from the user need analysis (i.e. the study of the application specific corpora).[14] In our research work, we extended the phenomena to be covered to deal with specific items that appeared in the corpus we had. The phenomena for Spanish currently treated by `TagIt` are:

`a.` `Numbers`: naturals (14, 7, 987), integers (+8, -7), reals (3'7).

`b.` `Percentages`: numbers followed by '%' or 'por ciento' (33%).

[12]TagIt was developed by Andrew Bredenkamp and Thierry Declerck.

[13]Therefore, the ALEP TH component we have described in section 2.5.1 was already substituted by this external component in the MELISSA project. See [Marimon *et al.* 99] for motivations.

[14]`TagIt` was specifically designed for two applications: ICAD, an administrative purchase and acquirement handling system, used at the Organización Nacional de Ciegos de España (ONCE), and KEWIS, a citizen service and information system with the requirement of multilinguality.

c. `Measures`: numbers followed by a measure unit (*25 litros, 5 kilos*).

d. `Currencies`: 'number + type of currency' (*300 euros*), 'number + de + type of currency' (*300 millones de dólares*).

e. `Dates`: (*14 de julio de 1967, 14 de julio, julio del 67, 14/07/67*).

f. `Periods`: Combinations like 'from + date + to + date' (*del 23 de junio al 17 de septiembre*).

g. `Proper names`:[15] (*Montserrat, Sr. López, RENFE,*[16] ...).

h. `Multi-word units`. Prepositions, conjunctions and adverbs which are formed by more than one lexical item are collapsed together within `TagIt`.

In our research work we have extended the functionality of `TagIt` to include morphological analysis. Morphological analysis is performed by look–up by means of a *perl* script which accesses a hashed lexical database of full–forms and returns the lexical information associated to them —lemma (or lemmata) and lexical tag (or tags)— as value to the attribute 'POS'.

The following is an example of a sentence marked–up by `TagIt` where, for the sake of simplicity, we only present the most relevant features ('POS' and 'TYPE', we have just explained):[17]

(62) a. *Iberia y el Banco Provincial firmaron el martes la compra por 145,5 millones de dólares del 60 por ciento de las acciones.*
(Iberia and the Banco Provincial signed on Tuesday the purchase for 145,5 million dollars of 60% of the stock shares.)

[15]In Spanish there are many proper names which are ambiguous with open class lexical items; examples are the surnames of some of the Ministers of the Spanish Government (e.g. Rato, Cascos, Matas, Piqué). The strategy we follow is to give all items which start in capital letters, wherever they appear in the sentence, the value 'PNAME' for 'TYPE' together with the lexical tag they get from the lexicon, and to let `Latch` disambiguate them on the basis of their contexts. This strategy was mainly motivated by the fact that if the wrong analysis is selected during the pre–processing phase, it will lead to incorrect syntactic analysis.

[16]Acronyms are also typed as 'PNAME'.

[17]Other features are: 'START' and 'END', which are used to get the length of the sentence; 'CONV', which featurizes the canonical format; 'ORIG', taking as value the token as it appears in the input text; and, finally, 'PTR' and 'PTL', for punctuation marks on the right and left of the tokens.

b. <P> <S> <W POS = "iberia\Np—|iberio\Afpfs-" TYPE = "PNAME">iberia</W> <W POS = "y\Cc" TYPE = "WORD">y</W> <W POS = "el\Tdms-" TYPE = "WORD"> el </W> <W POS = "banco_provincial\Np—" TYPE = "PNAME">banco provincial</W> <W POS = "firmar\Vmis3p-" TYPE = "WORD"> firmaron</W> <W POS = "el\Tdms-" TYPE = "WORD">el</W> <W POS = "martes\Ncm.-" TYPE = "WORD"> martes</W> <W POS = "el\Tdfs-|lo\Pp3fsa-" TYPE = "WORD">la </W> <W POS = "compra\Ncfs-|comprar\Vmip3s-|comprar\Vmmp2s-" TYPE = "WORD"> compra </W> <W POS = "por\Sp" TYPE = "WORD">por</W> <W POS = "145'5_millones_de_dólares\currency" TYPE = "CURR" >145'5 millones de dólares</W> <W POS = "de\Ncfs-|de\Sp" TYPE = "WORD"> de</W> <W POS = "el\Tdms-" TYPE = "WORD">el</W> <W POS = "60_por_ciento\perc" TYPE = "PERC" >60 por ciento</W> <W POS = "de\Ncfs-|de\Sp" TYPE = "WORD"> de</W> <W POS = "el\Tdfp-|lo\Pp3fpa-" TYPE = "WORD">las</W> <W POS = "accionar\Vmsp2s-|acción\Ncfp-" TYPE = "WORD"> </W> </S> </P>

The source of the lexical database which performs the lexical look–up and assigns the value to the attribute 'POS' is a full–form lexicon of 575,359 entries (78,535 lemmata) —consisting of about 7,085 verbal entries, 40,265 nominal entries, 15,086 adjectival entries, 15,551 adverbial entries and 548 closed–category entries. The lexical database, in turn, was generated by means of a morphological generator `mmorph` developed during the MULTEXT project.[18]

A MULTEXT–like lexicon is a table with three columns separated with a tabulation. The first column is for the full–form, the second one for the lemma, and the third one for the lexical tag. Examples for the encoding of items of different categories are shown in (63).

(63)

FULL–FORM	LEMMA	LEXICAL TAG	
`estos`	`esto`	`Pd3np--`	(these)
`son`	`ser`	`Vmip3p-`	(are)
`unos`	`un`	`Timp-`	(some)
`ejemplos`	`ejemplo`	`Ncmp-`	(examples)
`de`	`de`	`Sp`	(of)
`entradas`	`entrada`	`Ncfp-`	(entries)
`léxicas`	`léxico`	`Afpfp-`	(lexical)

[18]For a detailed information about the `mmorph` tool or other related tools developed during the MULTEXT project we refer the reader to `http://www.ub.es/gilcub/ingles/projects/european/multext.html`.

Ambiguity implies diversification of the lexical entries, i.e. there is an entry for each analysis. So, for example, a Spanish word like *son* has three entries, one encoding the main verb analysis (they 'are' nice), one encoding the auxiliary verb analysis (they 'are' presented), and one encoding the nominal analysis (sound).[19]

(64)

FULL–FORM	LEMMA	LEXICAL TAG
`son`	`ser`	`Vmip3p-`
`son`	`ser`	`Vaip3p-`
`son`	`son`	`Ncms-`

Lexical tags are Morpho–Syntactic Descriptions (MSDs) which encode morpho–syntactic information as attribute–value pairs for each full–form. The notation format adopted consists of a numbered string of characters —attributes are marked by positions— where: (i) the first character encodes part–of–speech, and (ii) the following characters encode the value of one attribute relevant for each category (e.g. type, gender and number for nouns). If an attribute is not relevant for a language (e.g. case for Spanish nouns) or a lexical item, the corresponding position is underspecified (i.e. it takes '-' as value). Unspecification of one value (i.e. when all values may be relevant for a given lexical item) is expressed by a dot '.'.[20]

Table 3.1 presents the attributes and values for the different categories from which the tagset —consisting of 259 tags— was built where the bold types represent the characters appearing in the tags.[21]

[19]The reader is referred to Appendix A for a detailed classification of the ambiguities in the lexicon.

[20]See [Bel *et al.* 96a] for further details.

[21]We have omitted those attributes which are not relevant for Spanish and which always take '-' as values. These are: case for nouns, adjectives, numerals.

Category	*Attribute*	*Value*
Verb	status	**m**ain, **a**uxiliary
	mood	**i**ndicative, **s**ubjunctive, i**m**perative, **c**onditional, i**n**finitive, **p**articiple, **g**erund
	tense	**p**resent, **i**mperfect, **f**uture, pa**s**t
	person	**1**, **2**, **3**
	number	**s**ingular, **p**lural
	gender	**m**asculine, **f**eminine
Noun	type	**p**roper, **c**ommon
	gender	**m**asculine, **f**eminine
	number	**s**ingular, **p**lural
Adjective	type	**q**ualificative, **o**rdinal, **i**ndefinite, po**s**sessive
	degree	**p**ositive, **c**omparative, **s**uperlative
	gender	**m**asculine, **f**eminine
	number	**s**ingular, **p**lural
ar**T**icle	type	**d**efinite, **i**ndefinite
	gender	**m**asculine, **f**eminine
	number	**s**ingular, **p**lural
nu**M**eral	type	**o**rdinal, **c**ardinal
	gender	**m**asculine, **f**eminine
	number	**s**ingular, **p**lural
Pronoun	type	**p**ersonal, **i**ndefinite, **d**emonstrative, in**t**errogative, **r**elative, refle**x**ive
	person	**1**, **2**, **3**
	gender	**m**asculine, **f**eminine
	number	**s**ingular, **p**lural
	case	**n**ominative, **a**ccusative, **d**ative, **o**blique
	possessor	**s**ingular, **p**lural
Determiner	type	**d**emonstrative, **i**ndefinite, po**s**sessive, in**t**errogative
	person	**1**, **2**, **3**
	gender	**m**asculine, **f**eminine
	number	**s**ingular, **p**lural
	possessor	**s**ingular, **p**lural
Conjunction	type	**c**oordinating, **s**ubordinating
adpo**S**ition	type	**p**reposition
adve**R**b	type	**g**eneral

Table 3.1: Categories, attributes and values.

3.4 The Unknown Word Processor

A lexicon with about 78,000 entries (lemmata) for Spanish does not include all words which might appear in any corpora. However, for a proper disambiguation one needs, at least, to recover a morpho–syntactic description, even if the lemma is unknown.

To deal with this, words which are not found in the lexicon are analyzed by a guesser which identifies the possible tag (or tags) for an unknown token on the basis of inflectional and derivational endings. However, this results in an increase of the ambiguity cases we have between open class items. To name a few:

- `noun/verb`: present indicative and present subjunctive and nouns ending in '-áis', '-éis'.
- `noun/verb/adjective`: present indicative, past indicative, present subjunctive and imperative, and nouns and adjectives ending in '-e'.

3.5 The Disambiguator and Chunker

`Latch` was firstly conceived as a lexical disambiguation tool based on analysis promotion/reduction by means of weighted symbolic context rules [Porta 96].

It is a lean formalism where lexical information, including full–form, lemma and MSD, is expressed by regular expressions. The pivots of the rules, which specify the tokens to be disambiguated, are sequences of lexical elements that receive a vote on their morpho–syntactic analysis. Votes may be positive or negative to promote or to eliminate them, respectively. In addition, a precondition may be expressed in the pivots to specify the type of ambiguity the rule is referred to. Linear generalizations are expressed by means of contextual operators for immediate, unbounded and constrained unbounded contextual conditions.

In a further development state [Marimon & Porta 00], the `Latch` formalism was extended so that it can also be used to mark chunks (or intra–clausal partial constituents) and use that information for PoS disambiguation. This interaction of PoS disambiguation and partial parsing reduces the effort needed for writing rules considerably and improves results.[22]

[22]The tool that we are presenting in this section is currently being used to annotate the 125 million word *Corpus Diacrónico del Español* (*CORDE*) and 125 million word *Corpus de*

3.5.1 I/O Format

The tagged sentence we have presented in section 3.3 shows the default SGML output representation of `TagIt`. However, `Latch` uses a specific I/O format where columns are physically separated by a tabulator character, as shown in the following example:[23]

```
REF  CLASS  ORTH                        LEMMA\MSD(|LEMMA\MSD)*
--------------------------------------------------------------------------------
     (SENT  <S>
1    TOK    Iberia                      iberia\Np---|iberio\Afpfs-
2    TOK    y                           y\Cc
3    TOK    el                          el\Tdms-
4    TOK    Banco Provincial            banco_provincial\Np---
6    TOK    firmaron                    firmar\Vmis3p-
7    TOK    el                          el\Tdms-
8    TOK    martes                      martes\Ncm.-
9    TOK    la                          el\Tdfs-|lo\Pp3fsa-
10   TOK    compra                      compra\Ncfs-|comprar\Vmip3s-|
                                        comprar\Vmmp2s-
11   TOK    por                         por\Sp
12   TOK    145'5 millones de dólares   145'5_millones_de_dólares\currency
16   TOK    de                          de\Ncfs-|de\Sp
17   TOK    el                          el\Tdms-
18   TOK    60 por ciento               60_por_ciento\perc
21   TOK    de                          de\Ncfs-|de\Sp
22   TOK    las                         el\Tdfp-|lo\Pp3fpa-
23   TOK    acciones                    accionar\Vmsp2s-|acción\Ncfp-
     )SENT  </S>
```

Input is identified by searching for <tag> opening and </tag> closing SGML marks in the ORTH column.[24] Marks are also included in the input and can be

Referencia del Español Actual (*CREA*) by the Departamento de Lingüística Computacional de la *Real Academia Española*. Some results on the first version of the tool can be found in [Sánchez *et al.* 99].

[23]When executed with the trace option, some lines are added to the tabular format with information about the rules which have been applied. This information is added below the pivoted items and they constitute the history of the changes.

[24]The default being <S> ... </S>.

referred to by any rule. Only lines with a non-empty REF column are taken into account.

The following illustrates of how this sentence looks like once it has been processed by `Latch`, where chunks are marked by PCHUNKo opening and PCHUNKc closing marks.

REF	CLASS	ORTH	LEMMA\MSD(|LEMMA\MSD)*
	(SENT	<S>	
1	TOK	Iberia	iberia\Np---
2	TOK	y	y\Cc
	PCHUNKo	NX	=\Np---
3	TOK	el	el\Tdms-
4	TOK	Banco Provincial	banco_provincial\Np---
	PCHUNKc	NX	
6	TOK	firmaron	firmar\Vmis3p-
	PCHUNKo	NX	=\Ncm.-
7	TOK	el	el\Tdms-
8	TOK	martes	martes\Ncm.-
	PCHUNKc	NX	
	PCHUNKo	NX	=\Ncfs-
9	TOK	la	el\Tdfs-
10	TOK	compra	compra\Ncfs-
	PCHUNKc	NX	
11	TOK	por	por\Sp
12	TOK	145’5 millones de dólares	145’5_millones_de_dólares\currency
16	TOK	de	de\Sp
	PCHUNKo	NX	=\perc
17	TOK	el	el\Tdms-
18	TOK	60 por ciento	60_por_ciento\perc
	PCHUNKc	NX	
21	TOK	de	de\Sp
	PCHUNKo	NX	=\Ncfp-
22	TOK	las	el\Tdfp-
23	TOK	acciones	acción\Ncfp-
	PCHUNKc	NX	
	)SENT	</S>	

3.5.2 The Formalism

`Latch` is a lean formalism for rule–based disambiguation and chunking where rules take the following syntactic form:[25]

$$
\begin{array}{rcl}
\langle rule \rangle & ::= & \texttt{@TOK}\ \langle rulename \rangle \\
 & & \langle leftcontext \rangle^{*} \langle tpivot \rangle^{n} \langle rightcontext \rangle^{*} \\
 & & [\%\langle score \rangle^{n}] \\
 & | & @\langle chunktype \rangle\ \langle rulename \rangle \\
 & & \langle leftcontext \rangle^{*} \langle cpivot \rangle^{n} \langle rightcontext \rangle^{*}
\end{array}
$$

For the sake of clarity, we will present the disambiguation and the chunking rules in two different sections.

3.5.2.1 The Syntax of Disambiguation Rules

$$
\begin{array}{rcl}
\langle rule \rangle & ::= & \texttt{@TOK}\ \langle rulename \rangle \\
 & & \langle leftcontext \rangle^{*} \langle tpivot \rangle^{n} \langle rightcontext \rangle^{*} \\
 & & [\%\langle score \rangle^{n}]
\end{array}
$$

- $\langle rulename \rangle$ is the (mandatory) rule identifier, which is displayed when the tracing facility is enabled.[26]
- $\langle tpivot \rangle^{n}$ (tag pivot) is a non–empty sequence of elements ($n > 0$) which specifies the lexical elements to be disambiguated by the rule. Its syntax is:

  $$\langle tpivot \rangle \quad ::= \quad < [\langle guard \rangle - >]\ \langle pattern \rangle >$$

  $\langle pattern \rangle$ is a tuple of $\langle fullform \rangle$, $\langle lemma \rangle$ and $\langle MSD \rangle$ —*fullform@lemma @MSD*— which uses regular expressions for each of the three elements,

[25]As usually, [ ] denotes optionality, $\langle sym \rangle$ marks non–terminal symbols, ^{n}, $^{+}$ and * are repetition operators, | separates alternative expressions, and other symbols are terminals.

[26]Therefore, the choice of mnemonic names is strongly recommended.

where omission is interpreted as a regular expression '`.*`'.[27] Rules may thus refer to words (full–forms and/or lemmata), expressing lexical preferences (65), and/or morpho–syntactic analyses (66), where underspecification is expressed by relaxing regular expressions (67).

(65) [28]`<[Pp]ara@paro@>`

(66) `<@@Sp>`

(67) `<@@Ncf.*>`

One of the main features of `Latch` is that it allows the grammar developer to specify a precondition —so–called ⟨*guard*⟩— to the lexical element the rule is referred to. In this way, morpho–syntactic analyses are promoted/rejected not only on the basis of their contextual conditions, but also taking into account such a further constraint. This additional constraint specifies the **ambiguity class** to be solved. As we will see in section 3.5.4, specifying the ambiguity to be solved does not only ensure reliability of the rules (the more restricted, the more reliable rules are), but it also allows us to leave apart those ambiguities that need semantic, pragmatic, or even extra–linguistic information to be reliably solved, as, for instance, the 'noun/adjective' case. The other side of the coin, however, is that the definition of contextual restrictions for each of the ambiguity classes in a lexicon makes rule implementation an almost endless task.[29] In addition, many of the contextual constraints specified for a given ambiguity type are not unique to it, but they are coincident to other ambiguity cases, especially those ones which it subsumes.[30] To avoid unnecessary redundancy, we may specify an **ambiguity superclass**, i.e. an ambiguity class which may subsume other ambiguity classes.

Preconditions are expressed by means of universally (&{⟨*pattern*⟩}&) or existentially (#{⟨*pattern*⟩}#) quantified patterns, which, in turn, may be

[27]Regular expressions are implemented with the GNU regex library.

[28]Paro: someone coming from the Greek island Paros.

[29]Furthermore, by applying disambiguation rules which discard analyses, new ambiguity classes are introduced (cf. section 3.5.4).

[30]An ambiguity class A subsumes an ambiguity class B, if A is included in B, i.e. if the categories in B include those in A.

combined using the logic connectives `&&` (and), `||` (or) and `!!` (not), and which we name $\langle element \rangle$:

$$\begin{array}{lcl} \langle element \rangle & ::= & \&\{\langle pattern \rangle\}\& \\ & | & \#\{\langle pattern \rangle\}\# \\ & | & \langle element \rangle \ \&\& \ \langle element \rangle \\ & | & \langle element \rangle \ || \ \langle element \rangle \\ & | & !!\langle element \rangle \end{array}$$

Thus, an ambiguity class dealing, for example, with the ambiguity 'common noun/preposition' (e.g. *pro, vía*,...) is expressed as follows:

(68) `<#{@@Nc.*}# && #{@@Sp}# && !!#{!!@@Nc.* && !!@@Sp}#->...>`

Whereas its superclass, also covering the 'common noun/preposition/verb' (*sobre*) and the 'common noun/preposition/verb/adjective' (*bajo*) cases, is expressed as:

(69) `< #{@@Nc.*}# && #{@@Sp}# -> ... >`

The `Latch` formalism has been designed to allow **syntactic translation from rules to predicate calculus.**[31] In (70) and (71) we show how the class and superclass in (68) and (69) are expressed and translated into predicate calculus.

(70) $\exists x \; Noun(x) \wedge \exists x \; Prep(x) \;\; \wedge \neg \exists z \; (\neg Noun(z) \wedge \neg Prep(z))$

(71) $\exists x \; Noun(x) \wedge \exists x \; Prep(x) \;\; \forall z \neg \; (\neg Noun(z) \wedge \neg Prep(z))$

- The $\langle context \rangle$ conditions are placed to the left and/or to the right of the rule pivot, indicating the elements preceding and/or following the analyses to be disambiguated. There is no limit to the number of contextual elements nor to the position they occur with respect to the pivot element, i.e. constraints can go beyond neighbouring tokens. Our rules can have one of the following

[31]This is important when the intuitive interpretation of rules is not so clear.

context expressions:

$$\begin{array}{rcl} \langle \mathit{leftcontext} \rangle & ::= & \langle \mathit{element} \rangle \ < \\ & | & \langle \mathit{element} \rangle \ << \\ & | & \langle \mathit{element} \rangle \ ! \ \langle \mathit{element} \rangle \ < \end{array}$$

- The immediate context operator (`<`) states that there is an adjacent element that satisfies the element constraint.
- The unbounded context operator (`<<`) expresses that there is an element somewhere to the left that satisfies the element constraint.
- The constrained unbounded context operator (`!<`) expresses that there is an element somewhere to the left that satisfies the context condition but all elements between boundaries satisfy a constraint.

Similarly, `>`, `>>` and `>!` are defined for right contexts.

Constraint sequences are possible by concatenating context expressions by means of this relative ordering operators, but note that in "C_1 `<` C_2 `<<` C_3 `<<` C_4" backtracking is required in order to search for possible anchoring contexts.

As within the pivot, the concepts of class and superclass may be used to refer to ambiguous contextual elements. In addition, we may also refer to contextual analyses which are either unambiguous or ambiguous between them by means of what we call **ambiguity subclass**. Such a concept is exemplified in (72), and in (73) we show how it is translated into predicate calculus:

(72) `&{@@Nc.* || @@Sp}& <`

(73) $\forall x \ (Noun(x) \vee Prep(x))$

- The last component of lexical disambiguation rules is $\langle score \rangle^n$, an optional sequence of positive or negative scores the grammar developer assigns to a rule in order to promote or to reject the (possibly preconditioned) analyses specified by the pivots on the basis of the contextual elements. The default value is '+1' when no score is provided.

Now that we have presented the syntax of the rules, we show some of the `Latch` rules for PoS disambiguation.

The following rule promotes nominal analyses when they are ambiguous with infinitives (e.g. *amanecer* (to dawn/dawn), *cantar* (to sing/song)) (expressed by the ambiguity class in the rule pivot) following unambiguous determiners, unambiguous indefinite adjectives, or any ambiguity between them (expressed by the ambiguity subclass definition) and preceding the preposition *de* (of).

(74)

```
@TOK VinfNoun_PostDetAiPreDe_Noun
&{@@D.* || @@Ai.*}& <
<#{@@Nc.*}# && #{@@V.n.*}# && !!#{!!@@Nc.* && !!@@V.n.*}# -> @@Nc.*>
> &{@de@Sp}&
```

Infinitives are not only ambiguous with nouns, but they may also be ambiguous with adjectives (e.g. *regular* (regular/to regulate)), and nouns and adjectives (e.g. *militar* (soldier/military/to belong)). This means that to eliminate infinitival analyses in the contexts we have just seen when they are ambiguous with nouns and/or adjectives we need three rules expressing the three types of ambiguities. However, we can reduce the number of rules by making use of the ambiguity superclass concept instead, as we show in (75).[32]

(75)

```
@TOK VinfNoun_PostDetAiPreDe_NegVinf
&{@@D.* || @@Ai.*}& <
<#{@@Nc.*}# && #{@@V.n.*}# -> @@V.n.*>
> &{@de@Sp}&
% -1

@TOK VinfAdj_PostDetAiPreDe_NegVinf
&{@@D.* || @@Ai.*}& <
<#{@@Af.*}# && #{@@V.n.*}# -> @@V.n.*>
> &{@de@Sp}&
% -1
```

[32]Note that in these rules we do not promote the nominal or the adjectival analysis, but we eliminate the infinitival one.

(76) exemplifies the unbounded context operator. It promotes the coordinating conjunction analysis for *bien*[33] when somewhere to the right there is a punctuation mark or the coordinating conjunction *o* (or) followed by *bien*. E.g. *bien por correo, bien por e–mail* (either by mail or by e–mail).

(76)

```
@TOK Bien_PrePunctCc+Bien_Cc
< @bien@Cc >
>> &{@comma@SPUNCT}& || &{@o@Cc}&
> &{@bien@}&
% 40
```

The last rule exemplifies the constrained unbounded context operator. It promotes the (finite) verbal analysis when from the beginning of the sentence to the end of the sentence there is no finite verb.

(77)

```
@TOK Unique_Verb_in_Sentence
&{<S>@@}& ! &{!!@@V.[misc].}& <
< #{@@V.[misc].*}# && #{!!@@V.[misc].*}# -> @@V.[misc].* >
> &{!!@@V.[misc].*}& ! &{</S>@@}&
% 10
```

3.5.2.2 The Syntax of Chunking Rules

The syntax of chunking rules in `Latch` is mainly the same as that of the disambiguation rules:

$$\begin{aligned}\langle rule\rangle \ ::= \ & @\langle chunktype\rangle\ \langle rulename\rangle \\ & \langle leftcontext\rangle^*\langle cpivot\rangle^n\langle rightcontext\rangle^*\end{aligned}$$

- $\langle chunktype\rangle$ distinguishes two types of chunks: $PCHUNK$ and $VCHUNK$. We will see in section 3.5.5 which chunks belong to each type. Basically, the former is for chunks which will remain after the chunk building process, whereas the latter is for chunks which may used to build other chunks but which are eliminated after the chunk construction process.

[33]*Bien* has three MSDs: one encoding the adverbial analysis (well), one encoding the nominal analysis (good), and one encoding the coordinating conjunction (either) (cf. Appendix A).

- $\langle rulename \rangle$ is used to distinguish the different types of chunks along categorial dimension. Here we have: VX for verbal chunks, NX for nominal chunks, AX for adjectival chunks, PARTX for participial chunks, APX for adjectival/participial chunk, RX for adverbial chunks, PX for prepositional chunks, and PC for parenthetical chunks. These rules are described in detail in section 3.5.5.

- $\langle cpivot \rangle^n$ (chunk pivot) is a non–empty sequence of element constraints ($n > 0$) which are grouped into an opaque element.

  `Latch` allows the grammar developer to specify a head element, which must be unique, by marking it with `<<_>>`. The production rule for $\langle cpivot \rangle$ is:

$$\begin{array}{lcl} \langle cpivot \rangle & ::= & < \langle element \rangle > \\ & | & << \langle element \rangle >> \end{array}$$

  If specified, head descriptions are projected to the chunk element so that they can be used by the disambiguation rules both at the rule pivot, to disambiguate the head of the chunk, and at the contextual conditions.

  Here the concepts class, superclass and subclass may also be used to define chunks that will include ambiguous lexical items.

- The last components of a chunk rule are the $\langle context \rangle$ conditions which indicates the elements preceding and or following the elements to be grouped. They hold the same constraints and syntax as the context elements of disambiguation rules.

The following are some examples of chunking rules.

(78) builds up a nominal chunk grouping a feminine plural definite article, the indefinite adjective *demás* (other) and a non–masculine non–singular nominal head. E.g. *Las demás personas* (the other people).

(78)

```
@PCHUNK NX
< &{@@Tdfp-}& > < &{@demás@Ai.*}& > << &{@@Nc[^m][^s]-}& >>
```

The second example shows a rule building up a VCHUNK (or temporal chunk) for coordinated adjectives premodifying a common noun. E.g. *las pequeñas y medianas empresas* (the small and medium companies).

(79)

```
@VCHUNK AX
&{ @@[TD].* }& <
< &{@@Af.*}& > < &{@@Cc}& > < &{@@Af.*}& >
> &{ @@Nc.* }&
```

Finally, (80) presents the rule for verbal chunks collecting dative weak pronouns (or clitics) and finite verbal forms.

(80)

```
@PCHUNK VX
< &{@me|te|le|nos|os|les@P.*}& > << &{@@Vm[ismc].*}& >>
```

3.5.3 Increasing the Formalism Expressivity: Macros

In order to increase the expressivity by rule abstraction, `Latch` is provided with a mechanism for macro processing: `m4`.

Macro definitions may be used to rename the tags in order to make the rules independent from the tagset one may be using, which, in addition, frequently increases their readability. Here, `Latch` tags can be defined as fine grained as desired, also simulating underspecification.

(81) a. `define('N_COMMON','Nc.*')`

b. `define('P_NAME','Np.*')`

c. `define('NOUN_SG','N..s.*')`

Using the same rules by another tagset would just require updating the macro definitions.

Also important is the use of macros to mark some syntactic and/or semantic distinctions which, going beyond the definition of morpho–syntax, may not have been encoded in a given available lexicon, but which may have a relevant role in disambiguation. Examples below include the marking of degree adverbs (82.a) and temporal nouns (82.b).

(82) a. `define('DEG_ADV','más|menos|muy|...')`

b. `define('N_TEMP','segundo|minuto|hora|día|...')`

Other examples are the macros we have used to classify verbal lemmata according to their subcategorization frames. The following are a very reduced version of the macros we have defined for verbs taking nominal subjects, nominal complements and infinitival verbal complements introduced by a marking preposition (83.a) and verbs taking nominal subjects and infinitival complements (83.b).[34]

(83) a. `define('V_SnpOnpPcvp','ayudar|acusar|enviar|...')`

b. `define('V_SnpOvp','deber|poder|...')`

Note that, by recursive macro expansion, hierarchies can be expressed in a very simple way. This is exemplified in (84.a), where the above macro for verbs taking nominal subjects and infinitival complements is substituted by a macro which calls the macro for raising verbs (84.b) and the macro for control verbs (84.c).

(84) a. `define('V_SnpOvp','V_RAIS|V_CTRL')`

b. `define('V_RAIS','deber|...')`

c. `define('V_CTRL','poder|...')`

We may also have parameterized macros. Examples of this type of macros are the ones we have implemented to deal with the concepts of class, subclass and superclass we saw in the previous section, so frequently used in our rules:

$$\texttt{class}(elem_1, \ldots, elem_n)$$
$$\texttt{subclass}(elem_1, \ldots, elem_n)$$
$$\texttt{superclass}(elem_1, \ldots, elem_n)$$

[34]These macros were generated automatically from the lexical resources developed in the PAROLE project (LE2–4017) [Melero & Villegas 98, Villegas *et al.* 98]. We are currently encoding such information in the lexicon, by adding further features to the MSD (cf. section 4.4).

Examples (68), (69) and (72) presented in the section 3.5.2.1 are in fact obtained by macro expansion of (85.a), (85.a) and (85.c).[35]

(85) a. `< class(@@NOUN,@@PREP) -> ... >`

b. `< superclass(@@NOUN,@@PREP) -> ... >`

c. `subclass(@@NOUN,@@PREP) <`

Parameterized macros, in addition, allow us to collect some sequences of contextual elements. We have just seen how macros may be used to classify lexical items according to their subcategorization frame. We have used parametrized macros to encode these subcategorization frames. So, for example, the macro encoding the verbal complement introduced by a marking preposition of (83.a) is:[36]

(86) a. `compl(Pcvp)`

where:

b. `define('Pcvp', 'class(@@PREP) > class(@@V_INF)')`

Therefore, the final expansion of (86.a) in rules such as (87) —promoting the verbal analysis when ambiguous with common nouns[37]— will be as shown in (88):

[35]The macro class generates code of this kind “#{A_1}# && ...&& #{A_n}# &&#{A$_1$ && ...&&A$_n$ }#” when called with arguments $A_1, \ldots, A_n$. But there is a particular very frequent case when the macro is called with just one argument. The optimal expansion of the macro in this case is “&{A}&” because the following sequence of logical equivalences holds:

$$\begin{aligned} &\exists x\ A(x) \wedge \neg\exists x\ \neg A(x) \\ &\quad\equiv\ \exists x\ A(x) \wedge \forall x\ A(x) \\ &\quad\equiv\ \forall x\ A(x) \end{aligned}$$

This optimization is carried out during the macro expansion since the macro takes into account the number of arguments.

[36]Note that we only use parametrized macros to collect those sequences of elements which are not marked up by the chunker or are marked up by the chunker only temporarily (cf. section 3.5.5).

[37]An example of such ambiguity would be ‘ayuda’ (help) in a sentence like *la ayuda a hacer los deberes* ((he/she) helps her/the help to do her homework.).

(87)

```
@TOK VfinNoun_PrePcvp_Vfin
< class(@@N_COMMON,@@V_FIN) -> @V_SnpOnpPcvp@V_FIN >
> compl(Pcvp)
% 1
```

(88)

```
@TOK VfinNoun_PrePcvp_Vfin
<#{@@Nc.*}# && #{@@V.[misc].}# && !!#{!!@@Nc.* && !!@@V.[misc].*}#
 -> @ayudar|acusar|enviar|...@V.[misc].* >
> &{@@Sp}& > &{@@V.n.*}&
% 1
```

Finally, we have used macro definitions to define what we called schemata. An schema has the following syntax:

$$\texttt{schema(}\mathit{expression}, \mathit{var}, \mathit{val}_1, \ldots, \mathit{val}_n\texttt{)}$$

It duplicates *expression* n times, replacing *var* with val_i at iteration ith. We have used schemata to simulate unification in order to reduce the number of rules to be implemented when dealing, for example, with agreement. This is exemplified is (89).

(89) [38]

```
schema( @TOK AiDiAdj_PostArtPreNoun_Ai
        superclass(@@TdAgr.) <
        < class(@@Ai.Agr.,Di.Agr.,@@AfpAgr.) -> @@Ai.Agr.>
        > class(@@NcAgr.)
        % 1,
Agr,fs,fp,ms,mp)
```

which is expanded into (90).

[38]Rule dealing with the ambiguity 'indef. adjective/indef. determiner/qual. adjective' (we have in *rara/as/o/os, distintas/os, diversas/os, escasa/as/o/os, cierta/as/o/os* (cf. Appendix A)) —promoting the indef. adjectival analysis following and article and preceding a common noun.

(90)

```
a. @TOK AiDiAdj_PostArtPreNoun_Ai
   &{ @@Tdfs. }&<
   <#{@@Ai.fs.}# && #{@@Di.fs.}# && #{@@Afpfs.}# &&
    !!#{!!@@Ai.fs. && !!@@Di.fs. && !!@@Afpfs.}# -> @@Ai.fs.>
   > &{ @Ncfs. }&
   % 1


b. @TOK AiDiAdj_PostArtPreNoun_Ai
   &{ @@Tdfp. }&<
   <#{@@Ai.fp.}# && #{@@Di.fp.}# && #{@@Afpfp.}# &&
    !!#{!!@@Ai.fp. && !!@@Di.fp. && !!@@Afpfp.}# -> @@Ai.fp.>
   > &{ @@Ncfp. }&
   % 1


c. @TOK AiDiAdj_PostArtPreNoun_Ai
   &{ @@Tdms. }&<
   <#{@@Ai.ms.}# && #{@@Di.ms.}# && #{@@Afpms.}# &&
    !!#{!!@@Ai.ms. && !!@@Di.ms. && !!@@Afpms.}# -> @@Ai.ms.>
   > &{ @@Ncms. }&
   % 1


d. @TOK AiDiAdj_PostArtPreNoun_Ai
   &{ @@Tdmp. }&<
   <#{@@Ai.mp.}# && #{@@Di.mp.}# && #{@@Afpmp.}# &&
    !!#{!!@@Ai.mp. && !!@@Di.mp. && !!@@Afpmp.}# -> @@Ai.mp.>
   > &{ @@Ncmp. }&
   % 1
```

We will conclude this section by showing how the examples of rules we presented in sections 3.5.2.1 and 3.5.2.2 are, actually, encoded in our grammar. Those examples are, in fact, generated once the macro processor has been executed.

(91)

```
@TOK VinfNoun_PostDetAiPreDe_NegVinf
subclass(@@DET,@@QUANT) <
< superclass(@@N_COMMON,@@V_INF) -> @@V_INF >
> class(@de@PREP)
% -1

@TOK VinfAdj_PostDetAiPreDe_NegVing
subclass(@@DET,@@QUANT) <
< superclass(@@ADJ,@@V_INF) -> @@V_INF>
> class(@de@PREP)
% -1

@TOK Bien_PrePunctCcBien_Cc
< @bien@CC >
>> class(@comma@SPUNCT) || class(@o@CC)
> class(@bien@)
% 40

@TOK Unique_Verb_in_Sentence
class(BOS@@) ! class(!!@@V_FIN) <
< superclass(@@V_FIN,!!@@V_FIN) -> @@V_FIN >
> class(!!@@V_FIN) ! class(EOS@@)
% 10


@PCHUNK NX
< class(@@ART_DEF_FS) > < class(@demás@QUANT) > << class(@@N_COMMON_FP) >>

@VCHUNK AX
class(@@DET) <
< class(@@ADJ) > < class(@@CC) > < class(@@ADJ) >
> class(@@N_COMMON) >

@PCHUNK VX
< class(@me|te|le|nos|os|les@PRON) > << class(@@VFIN) >>
```

3.5.4 Disambiguation Rules

This section describes our disambiguation grammar. We show how it has been designed as a set of modular components defined on the basis of both the ambiguity type to be solved and the contextual information used. In that, our main goal was to implement a grammar which: (i) could be updated efficiently on the basis of new corpus evidence, (ii) could be adapted easily to deal with different types of input texts.

In particular, we distributed our disambiguation rules along the following five modules:[39]

1. The first module includes rules which deal with specific lexical forms, (most of them) independently of the context in which they occur, to eliminate very low frequent readings. An example is the rule eliminating the nominal analysis for *a*.[40] This module also includes a set of rules which is used to eliminate the proper name analyses which are ambiguous with a closed class item when they appear at the beginning of the sentence.

2. The second module includes a very small set of rules which either remove analyses that are always illegitimate in a given context (e.g. verbal analyses following unambiguous determiners) or promote analyses in context where no other analyses are appropriate (e.g. verbs following unambiguous clitics). These are very simple and efficient rules where: (i) contexts are limited to one element either preceding or following the rule pivot, (ii) rule pivots are referred to categories rather than subtypes or lexical instantiation of such categories, and, (iii) no precondition is specified in the focus of the rules, such that they apply whatever ambiguity they have.

[39]One of the main issues when implementing a hand–crafted system is how to organize the grammar so that the developer can efficiently update and adapt it, especially when the grammar is being developed and/or updated by more than one person. We have implemented our disambiguation and chunking rules in a literate programming system, namely `noweb` [Knuth 84]. Briefly, literate programming is a philosophy for writing programs where: (i) you get to write the code in any order you want, independently of the order it will be executed; and (ii) code and documentation can be intermingled. For this, `noweb` comes with two programs: `tangle`, which rearranges the code in the right order to be compiled and executed; and `weave`, which formats the documentation. This has allowed us to group our rules according to the ambiguity classes we found in our lexicon, independently of the module they belong to.

[40]Encoding the reading for the alphabetical letter ‘a’.

3. The third module includes a set of rules which disambiguate on the basis of agreement within phrases. They eliminate analyses which do not agree in gender and/or in number with the tokens preceding and/or following them.

4. The fourth module is the biggest one. While in the previous set of rules we implemented very simple distributional generalizations dealing with a few categories or lexical forms, this module deals with all ambiguity types we found in our lexicon. Our first task here was to classify the ambiguities we had in our lexicon (cf. Appendix A).

   Even though we wanted to avoid as much as possible redundant constraints, since that clearly affected the performance of the grammar, our main goal was to ensure the reliability of the constraints. Therefore, all rules in this module specify a precondition in the focus expressing the ambiguity classes (or superclasses) to be solved.[41] Rules in this module were, in turn, distributed along three sub–modules on the basis of the type of precondition expressed in the pivot.

   Contextual constraints in this module are not limited to a single element, but they may include up to five elements, basically, establishing the distributional requirements within phrases.

   a)– The first sub–module includes those rules whose focus specifies an ambiguity superclass. This sub–module only includes a few rules, since, in fact, to avoid interference between rules the use of ambiguity superclass is recommended when not many ambiguity classes can be subsummed by the precondition.[42]

   b)– The second sub–module includes those rules whose focus specifies an ambiguity class. Note that ambiguity class does not refer to the ambiguities between the different categories (e.g. 'noun/verb'), but it refers to each case we have within such top level cases (e.g. 'first person singular present indicative and masculine singular nouns ending in '–o"). These rules mainly deal with the ambiguities between open

[41]This strategy, in addition, allows us to leave apart the disambiguation of 'adjective/participle' and 'adjective/noun' ambiguities in all those context where morpho–syntactic information about contextual elements is not enough to disambiguate them reliably.

[42]Interference may also be avoided by giving higher votes to more specific rules, since analyses are promoted/rejected on the basis of the score which is calculated once all the rules have been applied (cf. section 3.5.6).

class lexical items, i.e. adjectives, nouns and (finite and non–finite) verbs, and the ambiguities between closed classes, e.g. 'article/clitic'.

c)– The third sub–module includes those rules whose focus specifies a lexical item to be disambiguated and it deals with ambiguities between open and closed classes (e.g. *bajo* 'noun/preposition/verb/adjective'). This strategy was motivated by the fact that in applying the second module, which includes very unspecific rules removing analyses, we changed the ambiguity classes of many of the ambiguities we previously had in the lexicon.

5. Our fifth, and last, module is very similar to the previous one.[43] Again, different contextual constraints were defined for the different ambiguities we found in the lexicon by making use of the class and superclass concepts. The main innovation of this set of rules is that, while in the previous set of rules constraints were limited to the elements within phrases, the rules in this module go far beyond local constituency and disambiguate on the basis of order constraints —or requirements— of sentential constituents. Constraints on sentential constituents are expressed either by parametrized macros collecting sequences of elements or by referring to chunk mark–ups (cf. section 3.5.5).

Table 3.2 shows the number of rules each module has.

Module	*Number of Rules*
1st	16
2nd	55
3rd	66
4th	685
5th	557

Table 3.2 Number of Disambiguation Rules.

[43]Actually, we have a further sub–module heuristically defined to get rid of very low probability analyses. This sub–module has not been included in our experiments, thus no figures will be provided.

Grammar modules are applied to an input text through the following pipeline of processes:[44]

```
$ cat input_file | latch -r pack1 | latch -r pack2 | latch
  -r pack3 | latch -r pack4 | latch -r pack5 > output_file

  where:

  pack1  includes the first module.
  pack2  includes the second module.
  pack3  includes the third module.
  pack4  includes the fourth and the second modules.
  pack5  includes the fifth and the second modules, and the
         chunking rules.
```

3.5.5 Chunking Rules

Having a look at table 3.3 (section 3.5.7) where our results are presented, we see how the figures reflect a well–known problem in the hand–crafted approach to PoS tagging. While very simple distributional constraints may suffice for resolving about 50% of all of the initial ambiguities, it becomes more and more difficult to write new constraints that are as effective as the first ones. The main reason being that, as we have already pointed out at the beginning of the chapter, neighbouring tokens do not always provide the necessary information to deal with several ambiguity cases, so that contextual constrains must then be made about longer sequences. This is clearly reflected in the disambiguation grammar we have just presented where the number of contextual elements increase more and more, till the last module disambiguates on the basis of the order constraints of sentential constituents.

In order to express statements about long tag sequences reliably and efficiently, we extended the `Latch` formalism and integrated into the system a partial parser where rules can collect a sequence of elements, so–called chunks.[45]

[44]The second module of rules which are applied together with the fourth one (pack4) and the fifth one (pack5) only includes a sub–set of the rules in the second module, i.e. it does not include those rules whose contextual elements specify unambiguous patterns.

[45]One can make statements in a less specified context (and, therefore, less reliable context) by means of the unbounded context operator and the constrained unbounded context operator (cf. section 3.5.2.1).

Basically, our notion of chunks follows Abney's proposal [Abney 96a] very closely. Chunks represent intra–clausal partial constituents which are defined on purely syntactic basis, rather than semantically, functionally or lexically.[46] Chunks are defined along categorial dimension, extending from any premodifying (or specifying) element up to the head element (not including post–head complements and modifiers).

Following Abney, we only mark so–called 'maximal chunks' (i.e. chunks which are not contained in another chunk). Nevertheless, in order to reduce the number of rules, we can temporarily collect a sequence of elements which will be included within another one, e.g. adjectival chunks premodifying a nominal head. These chunks are marked as VCHUNK (volative chunks)—to distinguish them from the PCHUNK, or persistent chunks— and are eliminated after the chunk process.

VCHUNKs may also be used to build up (temporally) sequences that are in a higher level, e.g. prepositional chunks and nominal chunks extending to adjectival post–modifiers, as we show in (92) where chunks marked–up by round brackets are VCHUNKs, chunks marked–up by square brackets are PCHUNK.

(92) ($_{\text{NX}}$[$_{\text{NX}}$El Gobierno] argentino) emitirá títulos ($_{\text{PX}}$por [$_{\text{NX}}$otros 200 millones de dólares]) si ($_{\text{VX}}$se confirma) que [$_{\text{NX}}$el interés] ($_{\text{PX}}$de [$_{\text{NX}}$los inversores]) supera [$_{\text{NX}}$el monto] ($_{\text{PX}}$de ($_{\text{NX}}$[$_{\text{NX}}$la colocación] inicial))

Here, contextual conditions allow us to distinguish, for example, adjectival chunks which appear in pre–nominal position and which are marked as VCHUNK, from adjectival chunks functioning as post–modifiers or predicative and which are marked as PCHUNK.

Another difference between our chunks and Abney's is that, as can be observed in the example, we do not have chunks having a unique element.

Chunks are interpreted as lexical items, such that they referred at the contextual conditions. When a head element is specified —marking it with `<<_>>`— the head (morpho–syntactic) description is projected to the chunk element. This allows us to use this information for PoS tagging at the contextual constraints (i.e. contexts may not only refer to chunk type (or category) but also to the morpho–syntactic description of the head element). Furthermore, since we can

[46]Though we have also defined some VCHUNKs to collect support verb constructions (e.g. *dar un paseo* (to have a walk)) and periphrastic constructions (e.g. *echarse a llorar* (to start crying)).

include ambiguous elements in the chunks, such information allows us to define further disambiguation rules to solve ambiguous chunk heads.[47] The chunk grammar is applied together with the fifth module of the disambiguation grammar.

We will now present the chunking rules we have implemented for the different categories. These rules were implemented in two phases. Firstly, we translated the EUROTRA phrase structure rules of the Spanish grammar [EUROTRA 91] into the `Latch` syntax and run our grammar in a 100,000 word corpus, then, on the basis of corpus evidence, we updated our set of rules.

In what follows, we will represent chunks using square brackets instead of `Latch`'s CHUNKo opening and CHUNKc closing marks (cf. section 3.5.1). VCHUNKs will be represented by round brackets.

(a)- **Nominal Chunk (NX).** A NX is a nominal chunk extending from the beginning of a noun phrase to the head noun. Examples are:

(93) a. [$_{\text{NX}}$todas las demandas]
b. [$_{\text{NX}}$las tres primeras cartas]
c. [$_{\text{NX}}$varias personas]
d. [$_{\text{NX}}$la fuerte demanda]

– Nominal chunks can also be headed by a pronoun (94.a) and by one of the following patterns which have already been recognized and marked by `TagIt`: proper names (94.b), percentages (94.c) and measures (94.d).

(94) a. [$_{\text{NX}}$algunas otras]
b. [$_{\text{NX}}$el Banco_de_Francia]
c. [$_{\text{NX}}$el 10_por_ciento]
d. [$_{\text{NX}}$unas 16_toneladas]

– Temporal noun phrases, including dates[48], functioning as adverbials are marked as NX.

[47]However, no disambiguation rule can apply within the chunks, so no ambiguity within non–head elements will be resolved after the chunk process.

[48]Note that for dates, since articles are already included in the date tag by `TagIt` and we do not mark chunks with a unique element, we only include phrases with a pre–modifier, as shown in the example (95.b).

(95) a. juega a tenis [$_{\text{NX}}$cada semana]

b. nacerá [$_{\text{NX}}$el próximo 10_de_julio]

– NXs may be headed by ambiguous items such as: 'noun/adjective' (96.a), 'noun/participle' (96.b), 'noun/participle/adjective' (96.c).

(96) a. [$_{\text{NX}}$los trabajadores]

b. [$_{\text{NX}}$la entrevista]

c. [$_{\text{NX}}$los presupuestos]

– Pre–determiners include articles or determiners and/or numerals and indefinite adjectives.

(97) a. [$_{\text{NX}}$los/unos/varios/algunos casos]

b. [$_{\text{NX}}$los otros/tres casos]

c. [$_{\text{NX}}$los otros tres casos]

– Pre–modifying adjectives and/or participles are also included in a NX, which in turn may be modified, in which case are marked only temporarily (98.b).

(98) a. [$_{\text{NX}}$un gran acuerdo]

b. [$_{\text{NX}}$los ($_{\text{AX}}$más sanos) patrimonios]

(b)- Verbal Chunk (VX). The verbal chunk extends from the auxiliary verb —*ser, estar* (to be) and *haber* (to have)— to the main non–finite verbs, covering: passive gerundive compound forms (99.a), passive compound forms (99.b), gerundive compound forms (99.c), passive gerundive forms (99.d), gerundive forms (99.e), passive forms (99.f), and compound forms (99.g).

(99) a. [$_{\text{VX}}$ha estado siendo generada]

b. [$_{\text{VX}}$ha sido generada]

c. [$_{\text{VX}}$ha estado generando]

d. [$_{\text{VX}}$está siendo generada]

e. [$_{\text{VX}}$está generando]

f. [$_{\text{VX}}$fueron generadas]

g. [$_{\text{VX}}$ha generado]

- Adverbs (or adverbial phrases) (100.a), prepositional phrases (100.b) and temporal nominal phrases (100.c) do not split verbal groups, but are included in a VX when they are trapped between the auxiliary and the main verb.

(100) a. [$_{\text{VX}}$está siempre cantando]

b. [$_{\text{VX}}$estaba ($_{\text{NX}}$el otro día) regando]

c. [$_{\text{VX}}$está ($_{\text{PX}}$desde esta tarde) generando]

- Clitics are included in the VX, except the clitic '*se*', which is ambiguous with the particle '*se*' marking impersonal constructions or reflexive passives:

(101) a. [$_{\text{VX}}$nos/la está generando]

b. [$_{\text{VX}}$nos la está generando]

- The clitic '*se*' is thus included in a verbal VCHUNK.

(102) a. ($_{\text{VX}}$se han reducido)

- Periphrastic constructions are also marked as VCHUNKs. A periphrastic construction is formed by an auxiliary verb and a main verb in a non–finite form: infinitive (103.a), possibly introduced by a preposition (103.b), gerund (103.c)).

(103) a. ($_{\text{VX}}$suele visitarnos)

b. ($_{\text{VX}}$se echó a llorar)

c. ($_{\text{VX}}$anda incordiando)

- Support verb contractions (verb + light noun) are also included in VCHUNKs.

(104) a. ($_{\text{VX}}$tener interés)

b. ($_{\text{VX}}$dar un paseo)

(c)- Adjectival Chunk (AX). Adjectival chunks include adjectives modified by one (105.a) or two (105.b) adverbs or intensifiers. Examples are:

(105) a. [$_{\text{AX}}$muy inteligente]

b. [$_{\text{AX}}$bastante más rápido]

- Pre–nominal adjectives are marked as volative (or temporal) AXs. This allows us to reduce the number of rules for NXs where pre–modifying AXs are included. Here, we include adjectives modified by one adverb (106.a) and coordinated adjectives (106.b) which are preceded by a specifier an followed by an unambiguous noun.

(106) a. Las ($_{\text{AX}}$más grandes) esperanzas

b. Las ($_{\text{AX}}$pequeñas y medianas) empresas

(d)- Participial Chunk (PARTX). Participal chunks include unambiguous participles modified by other adverbs or intensifiers which are not part of a verbal group. Examples are:

(107) a. [$_{\text{PARTX}}$muy expuesto]

b. [$_{\text{PARTX}}$muy poco expuesto]

(e)- Adjectival/Participal Chunk (APX). We do not always disambiguate the ambiguity 'adjective/participle'. APXs include adjectives that are morphologically participles and which have not been disambiguated by the PoS tagger together with their modifying adverbs or intensifiers.

(108) a. [$_{\text{APX}}$muy provistas]

b. [$_{\text{APX}}$bastante más provistas]

(f)- Adverbial Chunk (RX). Adverbial chunks include adverbs modified by one (109.a) or two (109.b) adverbs or intensifiers.

(109) a. [$_{\text{RX}}$muy despacio]

b. [$_{\text{RX}}$bastante más despacio]

- Multi–word adverbs, such as *poco a poco* (slowly), *por el momento* (so far), are already marked by `TagIt` (cf. section 3.3).

(g)- Prepositional Chunk (PX). Prepositional phrases are only marked as VCHUNKs, since they belong to a higher level.

(110) a. [$_{\text{NX}}$el padre] ($_{\text{PX}}$de [$_{\text{NX}}$la novia])

(h)- Parentheticals (PC). Finally, PCs group elements within dashes (111.a) or parentheses (111.b).

(111) a. lo cierto ($_{\text{PC}}$—añade el comunicado—) es que ...

b. estos hipermercados($_{\text{PC}}$(Pryca, Continente y Alcampo)) no

3.5.6 Evaluation Mechanism

The evaluation mechanism operates by cycles of two phases: disambiguation and chunk mark–up.

During the disambiguation phase, the system tries to anchor rule pivots and, if context constraints are satisfied, scores are added to the pivots. Given a threshold based on the maximum score reached by some analyses, analyses scored below are eliminated:

Let S be a sequence of tokens $t_1, \ldots, t_n$ where each element t_i is composed of a word (w_i) and a set of m_i analyses (lemma/MSD):

$$t_i = \langle w_i, \{r_{ij}\}_{1 \leq j \leq m_i} \rangle$$

Let R be a set of rules:

$$R = \{\langle c_k, s_k \rangle\}_{1 \leq k \leq R}$$

where c_k is the context required to apply the rule and s_k is the rule weight.

Let $\delta : R \times S \mapsto S$ be the reduction function that, given a set of rules and a chunk, returns a new chunk where some of the original analyses remain.

The reduction function is evaluated for all analyses of the tokens contained in a chunk. It tries to apply rules and accumulates punctuation for each token $\langle w_i, r_j \rangle$ in the sequence:

$$\forall i : 1 \leq i \leq n : \forall j : 1 \leq j \leq m_i : \delta_{ij} = \sum_{k=0}^{R} \delta_{ijk}$$

where:

$$\delta_{ijk} = \begin{cases} s_k & \text{if } c_k \text{ holds in } \mathcal{S} \text{ for token } \langle w_i, r_j \rangle \\ 0 & \text{i.o.c.} \end{cases}$$

When no rule can apply, analyses are filtered according to a threshold h[49]:

$$f(\langle w_i, \{r_{ij}\}\rangle, h) \equiv \langle w_i, \{r_{ij} | \delta_{ij} \geq max(\{\delta_{ij}\}) - h\}\rangle$$

The process of chunk mark–up is a recursive process, so that chunks can be built up by other chunks. VCHUNKs can be used for other chunk construction, but they are eliminated at the end of this phase and only persistent chunks survive.

Both processes can iterate once or cycle until rules have no effect on text.

3.5.7 Results

The disambiguation grammar was evaluated on a 21,334–word corpus from the newspaper “El Sur”, manually annotated using the MULTEXT annotation scheme. The corpus showed an initial ambiguity of 44.48%.

A disambiguation grammar like the one we have presented that aims at eliminating only illegitimate analyses is likely to leave some ambiguities unresolved. In the grammar we have presented, these ambiguities are:

- ambiguities remaining because the grammar is not complete, in which case, they can be resolved by adding new rules.
- ambiguities which require further (linguistic or extra–linguistic) information to be reliably solved. These ambiguities include ambiguities within the same category, e.g. in nominal entries having different semantic readings and following the same inflection patterns, tense, person and mood ambiguity in verbal forms, as well as ‘adjective/participles’ and ‘adjective/nouns’.

Some statistics on the results are give in table 3.3.

	pack1	pack2	pack3	pack4	pack5
Ambiguity	28.52%	17.82%	17.43%	8.19%	7.95%
Recall	100%	100%	100%	99%	99%
Precision	71%	80%	80%	90%	91%

Table 3.3 Statistics on Results.

[49]The default value of h is *1.0*.

A recall of 100% means that all tokens have received the appropriate analysis— and it is calculated as:

$$Recall = \frac{TokensCorrectlyDisambiguated}{CorrectTokens}$$

A precision of 100% means that there is no superfluous analysis— and it is calculated as:

$$Precision = \frac{TokensCorrectlyDisambiguated}{Tokens}$$

When recall and precision are the same, then this value is called accuracy, which happens when statistical methods are applied so that all tokens have just one analysis.

Chapter 4

Integrating Shallow and Deep Linguistic Processing

This chapter investigates to what extent deep linguistic processing may benefit from shallow processing techniques, and it presents a NLP system which integrates the linguistic tagger and chunker we have described in chapter 3 as a pre–processing module of the unification–based grammar we have described in chapter 2.

An introductory section describes most representative theoretical and engineering proposals for efficient and robust deep processing. The second section presents the integrated architecture. The third section discusses the extensions required by our system in order to transfer the information delivered by the external pre–processing module into the linguistic processing modules of ALEP. Section four describes the default lexical templates we have implemented to provide robustness to the system. In the last section precise results of the system are provided.

4.1 Introduction

Deep linguistic processing produces a complete syntactic and semantic analysis of the sentences it processes, as required for accurate NLP, however, it fails in producing a result when the linguistic structure being processed and/or words in the input sentences fall beyond the coverage of the grammatical resources. NLP systems with monolithic grammars, where all dimensions of linguistic information (morphological, syntactic, semantic, etc.) are interleaved, in addition, have to deal with huge search space due to several sources of non–determinism (i.e. ambiguity). This is particularly true of broad–coverage unification–based grammars.

Nevertheless, deep natural language processing techniques are a very active area of work linked to real–world applications where precise interpretation of linguistic expression is important, for instance NLIs [Marimon *et al.* 99] and speech–to–speech MT systems [Müller & Kasper 00]. More recently, deep processing has been used in the area of industrial applications such as IE and language checking applications [Crysmann *et al.* 02] in which shallow processing has so far played a major role.

Therefore, not surprisingly, a lot of effort has been devoted in the NLP research community, addressing more adequate parsing algorithms,[1] algorithms for the unification operation itself, and feature structure and type hierarchy compilation techniques. Besides, several engineering solutions have been proposed for efficient and robust deep linguistic processing.

This introductory section presents most representative theoretical and engineering proposals for efficient processing. Against this background, we propose an engineering solution which integrates shallow and deep processing. We will present related approaches in the literature and will discuss previous experiments within the ALEP framework.

Parsing Algorithms

Standard parsing algorithms (i.e. top–down or goal–driven)[2] and Prolog search procedure (i.e. depth–first, backtracking) have turned out to be problematic

[1]As well as generation algorithms, which will not be described here.

[2]Top–down algorithm starts with a hypothesis about the structure of the whole expression and proceeds making new hypothesis about their constituency until words are reached.

for NLP because of efficiency (cf. [Erbach & Manandhar 95]). In system using lexicalist grammar formalisms, top–down algorithms may even fail to terminate (cf. [Noord 97, Erbach & Manandhar 95]).

On the one hand, a number of alternative parsing algorithms, as well as variants of existing techniques, have been implemented: bottom–up parsing (or data–driven strategy)[3]; Earley's algorithm [Earley 70], a bottom–up parsing that employs top–down prediction to hypothesize the starting points of possible constituents, and Shieber's extension of Earley's algorithm, which uses restriction to perform top–down filtering [Shieber 85]; head-corner parsing [Bouma & Noord 93, Kay 89, Noord 97], a bidirectional method that supposes there are privileged elements (heads) in productions that guide the parsing process (to their right or to their left); Tomita's generalized LR parsing [Tomita 87], which extended the LR parsing algorithm to handle any sentence generated by CFG.[4] On the other hand, dynamic programming is widely used in parsing to reduce the search space of the parser [Kasami 65, Younger 67, Kay 86]. Dynamic programming is a technique to store and re–use partial results. Particular data structures have been developed to support dynamic programming, most notably the *chart* [Kay 86].[5]

Unification Algorithms

Unification is the primary operation on feature structures. Efficiency of unification, therefore, is crucial for the overall efficiency of the systems that use feature structures to describe linguistic objects. Several algorithms have been proposed to do unification efficiently, specially with respect to graph unification, aiming at reducing both its time and its space cost.[6] Some of the most interesting proposals in the literature are: the *non–destructive* graph unification algorithm described by [Wroblewski 87], which eliminates over copying with incremental copying; the

[3]Bottom–up parsing begins with words and proceeds combining them into larger structures

[4]The LR parsing algorithm could parse input sentences deterministically and efficiently, these, however, were limited to the ones generated by LR grammar (i.e. a subset of CFG).

[5]In another line of research, heuristics have been proposed to reduce the search space of the parser. This method focuses on the most likely analyses, which are acquired using a statistical model.

[6]In many existing implementations, feature structures are represented by directed graphs. Inefficiency of graph unification arises from the destructive feature of the *union/find* algorithm [Aho *et al.* 74], which requires a copy of input feature structure graphs in order to be suitable for use with chart–based parsing. Therefore, in fact, copying requires more time that unification itself.

quasi–destructive variant proposed by [Tomabechi 91], which only copies structures when unification succeeds; the structure sharing approach presented by [Pereira 85], which reduces the amount of copying by allowing graphs to share common parts of their structure; the algorithm based on the *lazy copy*, first suggested by [Karttunen & Kay 85] and more recently by [Kogure 90, Godden 90], where copying is delayed until a destructive change is about to happen; and, Emele's *lazy incremental copy* variant, which combines incremental copying with lazy copying to achieve structure sharing [Emele 91].

Compilation Techniques of Feature Structures

Another active line of research has addressed compilation (or processing) techniques of feature structures and type hierarchies, based on term representations [Alshawi 91, Simpkins *et al.* 93, Erbach 95], bit vectors [Krieger 95], and abstract machines e.g. AMALIA for ALE grammars [Wintner & Francez 99], LILFeS for LinGO grammars [Miyao *et al.* 00], and CHIC for a sub–set of TDL.[7]

Engineering Proposals for Efficient and Robust Processing

Supplementary to this, several engineering solutions for efficient processing have been proposed.

Here are relevant reported works aiming at improving feature structure unification, since in unification–based grammatical frameworks up to 90% of the CPU time expended on parsing goes to it (cf. [Kiefer *et al.* 99, Malouf *et al.* 00]).

[Kiefer *et al.* 99] report on several engineering experiments in the framework of the DISCO system [Uszkoreit *et al.* 94]. Experiments consist of the deployment of filters —a rule application filter and the so–called 'quick check' filter— to avoid failing unification, the use of restrictors to remove features which are only used locally and do not play a role in further derivation, and a technique based on co–occurrence restrictions between lexical items to limit the number of initial chart items (i.e. lexical items).

Filtering techniques have also been employed with the LKB and the LinGO grammar [Copestake & Flickinger 00] together with a checking linear–time algorithm for subsumption relationships and equality between feature structures

[7]See [Porta 00] for a detailed discussion and state of the art of compilation techniques of feature structures used in current NLP systems where the author proposes a novel approach based on objects.

which packs local ambiguities [Malouf *et al.* 00].

Other proposals explore the idea of using CFG filters within high–level grammars, e.g. [Maxwell & Kaplan 93] for LFG–based grammar, [Kiefer & Krieger 00, Kiefer *et al.* 00] for English and Japanese HPSG–based grammars, where they use the CFG first and then let the HPSG deterministically replay the derivations licensed by the CFG. Within this research line [Torisawa *et al.* 00] present a system which automatically extracts a CFG backbone from a given HPSG–based grammar and uses it to eliminate partial trees that do not contribute to the final parse tree.

Inspired by [Abney 90, Abney 91, Abney 92, Abney 95], other proposals investigate hybrid methods to improve processing time. Abney proposes text chunking as a preliminary step to parsing. Abney's chunks (i.e. pieces of phrase structure licensed by immediate constituency) are inspired by [Gee & Grosjean 83]'s *ϕ–phrases*, but they are originally motivated by their computational advantages: efficient and robust parsing of unrestricted text. Efficient and robust parsing arise from resolving morpho–syntactic ambiguities and reducing the number of the elements to be parsed into a complete tree, and from recovering some linguistic information even when sentences can not be completely parsed.

[Srinivas *et al.* 97] present a system which applies a statistical disambiguation technique in the context of a lexicalized grammar framework (Feature–Based Lexical Tree Adjoining Grammar [Joshi *et al.* 75]) prior to parsing, in order to prune the search space of the parser. A (trigram) disambiguation model is used to disambiguate so–called supertags, tags containing information about PoS and (recursive and non–recursive) constructs associated to the lexical items. The task of the parser is thus reduced to establish the dependency links. Reported parsing speed–up is a factor of about 30.

[Ciravegna & Lavelli 97] propose to use text chunking for controlling (i.e. activating/delaying tasks) an agenda–based bottom–up chart parser. Preliminary text chunking allows them to focus directly on the constituents that seem more likely so to reduce the spurious ambiguity seen by the parser. The chunking process is done via finite state automaton, taking the output of a PoS tagger. They claim that first experiments show a reduction of about 68% of edges (constituents) generated and of 78% of time consumed. A detailed check of the results, however, showed that they get 2 wrong analyses out of 31 cases.

[Yoon *et al.* 99] present a system which employs three types of chunking — *eojeol*, the syntactic unit in Korean consisting of a content word and of functional words, N–type chunks, identified by finite state transition networks, and collocations, obtained by statistical methods— and a parsing method in charge of determining dependencies inside N–type chunks and among the head of chunks based on lexical information. Reported results are 87% of dependency decisions and 0'91% dependency, errors though errors increase significantly in sentences having more than two clauses.

[Venkova 00] discusses a model for processing Bulgarian *da*–conjunctions which is intended as a pre–processor for deep analyzer and thus providing it with linguistic information —semantic disambiguation of compound lexemes, marking of adjunct clauses boundaries, semantic classification of adjunct clauses, PP identification— crucial for the efficiency of the analysis procedure.[8]

[Watanabe 00] describes an algorithm for accelerating the CFG–Parsing process by using dependency (modifier–modifiee relationship) information provided by stochastic parsers, interactive systems and linguistic annotations added in the source text. Reported reduction of processing time is about 45% and 15%.

[Prins & Noord 01] show how a PoS tagger can be used to filter the results of lexical analysis of a wide–coverage computational grammar, thus improving parsing efficiency and increasing parsing accuracy.

[Grover & Lascarides 01] describe a method to improve robustness and efficiency by interfacing PoS tag information with the existing lexicon of the Alvey Natural Language Tools (ANLT) system [Carroll *et al.* 91]. If a word exists in the lexicon, the PoS tag is used as a filter, accessing only those entries of the appropriate category, and, thus, cutting down the parser's search space. If the word is unknown to the system, a basic underspecified entry for the PoS tag is used as its lexical entry. An experiment on 200 sentences showed how performance improved a 37.5%; the integrated system parsed 79 sentences, whereas the ANLT itself could only parse 4 cases. A hand–examination of the parsed sentences showed an accuracy of 30.5%.

Finally, [Crysmann *et al.* 02] describe the integration of a high–level HPSG parsing system with a cascade of shallow components performing tokenization, lexico–morphological analysis, PoS filtering, name entity recognition, sentence boundary detection, chunk and subclause recognition, and show how IE and language checking applications for German text benefit from a deep grammatical

[8]Compound *da*–conjunctions in Bulgarian have as a first element one of the following words: *bez, za, vmesto, osven, predi* and as a second element the very frequent and highly polyfunctional word *da* (cf. [Venkova 00]).

analysis. In the system they describe partial analyses from shallow processing can be used to guide the deep parser identify relevant candidates for deep processing. Furthermore, the integrated system they describe improves both parsing efficiency and robustness. In the system they present, however, the accuracy of the deep analysis decreases; 10% of the sentences which were successfully parsed by deep analysis could not be parsed and the number of analysis per sentences dropped from 16.2% to 8.6%.

Hybrid Approaches with ALEP

The idea to integrate shallow processing techniques in the ALEP system to obtain more efficient and robust processing has already been explored.

In the context of the LS–GRAM project, language–specific taggers were developed and integrated into the TH ALEP component for the recognition and mark–up of so–called "*messy details*".[9]

Also in the context of the LS–GRAM project, [Declerck & Maas 97] extend the functionality of this external add–on module and use it to integrate into the ALEP processing components part–of–speech information —the category— delivered by a PoS tagger (the `Mpro` tool [Maas 96]). Even though the PoS tagger simplifies the task of the parser by resolving morpho–syntactic ambiguities, the system they present still needs the ALEP TLM component and the analysis rules concerned with word construction.

On the other hand, [Fouvry & Bredenkamp 97] discuss the extent to which partial parsing can be applied in ALEP in order to provide more robust and efficient high–level processing. The strategy they present, which has not been implemented, resembles that of [Ciravegna & Lavelli 97, Crysmann *et al.* 02] in that it uses the output of partial parsing to guide further processing, by providing the parser the intermediate nodes to be built up. Robust processing is ensured by robust phrase structure rules which use the partial parsing information to attach constituents.

Finally, [Badia & Egea 00] describe a small–scale experiment which uses the results of a Constraint Grammar (i.e. morpho–syntactic and syntactic tags) as

[9]"*Messy details*" are special text constructs, such as dates, numbers or codes, which can occur in many variations, making impossible a complete and efficient account by means of unification–based formalisms like ALEP, but which can be easily identified by pattern–matching techniques based on regular expressions. This strategy was first implemented by [Music 95] and it was extended to all LS–GRAM grammars [Bredenkamp *et al.* 96, Bredenkamp *et al.* 97].

input of a grammar for Catalan implemented in ALEP where "...the robustness of the former remedies some of the deficiencies of the second.". Both robustness and speed–up are said to be accomplished (though no figures are provided) by reducing lexical entries to a single entry per PoS tag and by constraining the application of phrase structure rules according to the syntactic tags obtained from the CG output, which makes it possible to use the same phrase structure rule to build structures for different syntactic relations. Such underspecification, however, clearly increases the search space of the parser and has obvious negative effects on the performance of the parser. On the one hand, underspecification of lexical entries —they do not contain any information about valence, modification, specification or even semantics— leads to structural ambiguity, which, as they already say, can only be solved if rich syntactic and semantic information are taken into account. On the other hand, because of the underspecification of phrase structure rules, their system has no control on the ordering that rules are applied, so that the same rule may be accessed more than once in the analysis process, e.g. the rule cancelling complements on the left is accessed before and after the rule cancelling complements on the right. From the engineering point of view, it would be preferable to integrate syntactic and semantic restrictions into the parsing process for early disambiguation.

In the following sections we present our approach which integrates the linguistic PoS tagger and chunker we described in chapter 3 as a pre–processing module of the unification–based grammar we described in chapter 2. Our goal in that is to improve the efficiency of the grammar by distributing the analysis process such that we can release the parser from certain tasks which may be efficiently and reliably dealt with by these computationally less expensive processing techniques which, in addition, can provide robustness to the high–level linguistic processing components.[10]

[10]First results of the system we describe may be found in [Marimon *et al.* 01, Marimon 02].

4.2 The Integrated Architecture

The integration of shallow processing techniques (PoS tagging and partial parsing) is fully supported by the open architecture of ALEP, which allows easy integration of external modules.

Our system requires some changes to the default architecture of the ALEP system (cf. section 2.3), where both the TH system and the morpho–graphemic analysis component are replaced by a unique external pre–processing module. It also requires the lifting component to be extended in order to transfer the information delivered by the external pre–processing module into the linguistic processing components. The changes to be made in the linguistic processing components, however, are very thin: word structure rules have to be extended, but phrase structure rules and lexical entries can be left untouched.

The overall integrated architecture is outlined in Figure 4.1.

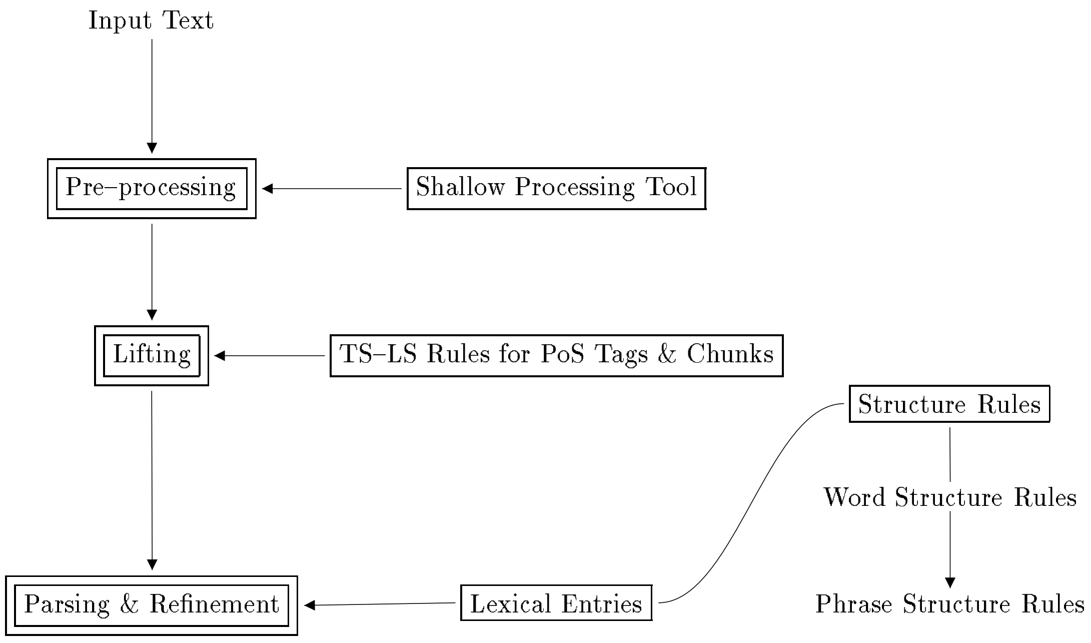


Figure 4.1 The Integrated Architecture

4.3 Integrating Tags and Chunks into the Grammar

The following subsections discuss the extensions required by our system in order to transfer the information delivered by the pre–processing module. Section 4.3.1 and section 4.3.2 discuss the extensions to the lifting component, section 4.3.3 presents the new word structure rules.

4.3.1 Lifting Tags

Integrating morpho–syntactic information in a system like ALEP means defining TS–LS rules propagating (or lifting) the morpho–syntactic information associated to full–forms (i.e. PoS tag and lemma) to the morpho–syntactic features of the lexical component of the grammar.

In a previous step, an external module was developed to convert the tabular format the pre–processing tool produces (112.a) into `ts-chunks` structures (112.b) in order to meet the input requirements of the lifting component.

(112)

```
a.   1      TOK      acciones      accionar\Vmsp2s-

b.   tsn:{ attr => 'S',
           feature_list => [...],
           children => [
              ...
              tsn:{
                    attr => 'W',
                    feature_list => [...],
                    children => [
                       tsl:{
                             attr => 'M',
                             feature_list => [ ft('LEMMA','accionar'),
                                               ft('POS','Vmsp2s-')],
                             data_content => acciones } ] }
              ...] }
```

Note that this external conversion module introduces the three structural levels —'S', 'W', 'M'— which in the default architecture of the ALEP system are produced as an effect of executing the morpho–graphemic analysis operation.

Also note that the morpho–syntactic information is introduced at the lowest level. By using the 'M' tag to lift the lexical information associated to full–forms, we can propagate the ambiguities which can not be reliably solved by the shallow processing tool to the grammar component, thus ensuring that the accuracy of the grammar remains the same.

Ambiguities are introduced as alternative *tsl*, as we show in (113).[11]

(113)

```
ts_alternate:{
     ts_1 => [ tsl:{ attr => 'M',
                     feature_list => [ ft( 'LEMMA', accionar ),
                                       ft( 'POS', 'Vmsp2s-' )],
                     data_content => acciones }],
     ts_2 => [ tsl:{ attr => 'M',
                     feature_list => [ ft( 'LEMMA', acción ),
                                       ft( 'POS', 'Ncfp-' )],
                     data_content => acciones }]}
```

The pre–processing module delivers the part–of–speech information in the form of positional tags or labels, i.e. numbered string of characters where attributes are marked by positions, being the first one for the category (part–of–speech) and the rest for the different attributes which are relevant for each category (cf. section 3.3).

This information, as we show in (114) and (115), in the type system declaration of the grammar is spread along different attribute–value pairs.

(114)

$$\begin{bmatrix} \textit{ld} & \\ \text{SYNSEM}\,|\,\text{LOCAL} & \begin{bmatrix} \textit{t\_local\_morph} & \\ \text{MORPH} & \begin{bmatrix} \textit{t\_morph} & \\ \text{LEMMA} & \textit{acción} \\ \text{MORPHEME} & \textit{acciones} \\ \text{AGR} & \boxed{1}(\textit{fem\&plur}) \end{bmatrix} \\ \text{CAT} & \begin{bmatrix} \textit{t\_subst} & \\ \text{HEAD} & \begin{bmatrix} \textit{t\_noun} & \\ \text{NCLASS} & \textit{common} \\ \text{AGR} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

[11]We only produce the output of the 'record' version of the TLM algorithm.

(115) [12]

$$\begin{bmatrix} \textit{ld} & \\ \text{SYNSEM}\,|\,\text{LOCAL} & \begin{bmatrix} \textit{t\_local\_morph} & \\ \text{MORPH} & \begin{bmatrix} \textit{t\_morph} & \\ \text{LEMMA} & \textit{accionar} \\ \text{MORPHEME} & \textit{acciones} \\ \text{AGR} & \boxed{1}\textit{(p2\&sing)} \\ \text{TNS\_MOOD} & \boxed{2}\textit{(pres\&subj)} \\ \text{VFORM} & \boxed{3}\textit{fin} \end{bmatrix} \\ \text{CAT} & \begin{bmatrix} \textit{t\_subst} & \\ \text{HEAD} & \begin{bmatrix} \textit{t\_verb} & \\ \text{AGR} & \boxed{1} \\ \text{TNS\_MOOD} & \boxed{2} \\ \text{VFORM} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The most straightforward option, then, is to implement as many TS–LS rules as tags we have in the tagset. These rules, rather than propagating information to the target LD, specify the values of the morpho–syntactic attributes of the target LD according to the different tags delivered by the PoS tagger.

To avoid diversification of TS–LS rules, a second option is to distribute the morpho–syntactic information we have in the form of tags along different tag–features on the basis of the different attributes the ALEP grammar foresees for each category (e.g. part–of–speech, noun type and agreement for nouns (116.a); part–of–speech, tense and mood and agreement for verbs (116.b)) and to propagate this information to the LD.[13]

[12]Note that even though there is no attribute encoding the 'status' of the verb (main/auxiliary), this information is encoded in the CAT attribute: main verbs take *t_subst* type, whereas auxiliary verbs take *t_funct* type. In addition, the paths encoding the agreement information are distinct in main verbs and auxiliaries: the former encode it in the SYNSEM|LOCAL|CAT|HEAD|AGR attribute, whereas the latter encode it in SYNSEM|LOCAL|CAT|SPECIFIES|SYNSEM|LOCAL|CAT|HEAD|AGR attribute.

[13]The distribution of the morpho–syntactic information along different tag–features may be done by the previous conversion module.

(116)

```
a. tsl:{
         attr => 'M',
         feature_list =>  [
              ft('LEMMA',acción),
              ft('CATEG','N'),
              ft('NCLASS','c'),
              ft('AGR','fp')],
         data_content => acciones},


b. tsl:{
         attr => 'M',
         feature_list =>  [
               ft('LEMMA',accionar),
               ft('CATEG', 'V'),
               ft('STATUS', 'm'),
               ft('TNS_MOOD','sp'),
               ft('AGR','2s')],
         data_content => acciones},
```

This option, however, does not avoid diversification of TS–LS rules completely, since the tags we have are 'category' positional, rather than 'absolute' positional. This option, in addition, requires to change the values which result from splitting PoS tags in order to match the values as they are specified in the type system declaration of the grammar, e.g. the value 'c' of the tag–feature NCLASS has to be changed into 'common'.[14]

Here, we have an additional restriction. TS–LS rules in ALEP do not allow boolean values to be percolated to LD, as already pointed out by [Declerck & Heyd 97], but only atomic values may be lifted from the tag–features. Such a restriction obliges us to define different TS–LS rules which specify the values the target LD takes for the attributes encoding agreement and tense information, which in our ALEP grammar take boolean values (cf. section 2.6.1), on the basis of the different values which the AGR and TNS_MOOD tag–features take.

Taking into account the above mentioned restrictions, we have followed three different strategies which we describe in the following.

[14]Or the other way round, we have to change the values we have in the type system declaration of the grammar in order to match the values which result from splitting tags.

(a)– We have defined a TS–LS rule for each of the tags we have for common nouns, adjectives and numerals. These rules: (i) propagate the value of the tag–feature LEMMA and the tag data–content to the LD attributes LEMMA and MORPHEME, respectively, and (ii) specify the value of the LD attribute AGR on the basis of the value the tag–feature POS takes.

The following, for example, shows the rule we defined to lift the tag 'Ncfs-'. Note that the target LD, in fact, does not need to co–index the morphological and the head agreement features, as we showed in (114), since this is already done in the lexical entries.

(117)

ts_ls_rule(

$$\begin{bmatrix} \textit{ld} & \\ \text{SPEC} & \textit{t\_lex\_spec} \\ \text{SYNSEM}\,|\,\text{LOCAL} & \begin{bmatrix} \textit{t\_local\_morph} & \\ \text{BAR} & \textit{minus} \\ \text{MORPH} & \begin{bmatrix} \textit{t\_morph} & \\ \text{LEMMA} & \boxed{2} \\ \text{MORPHEME} & \boxed{1} \\ \text{AGR} & \textit{(fem\&sing)} \end{bmatrix} \\ \text{CAT} & \begin{bmatrix} \textit{t\_subst} & \\ \text{HEAD} & \begin{bmatrix} \textit{t\_noun} & \\ \text{NCLASS} & \textit{common} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix},$$

'M', [POS = 'Ncfs-', LEMMA = $\boxed{2}$], $\boxed{1}$).

(b)– For (auxiliary and main) verbal tags, we have adopted a mixed approach. We have maintained the labels we receive from the pre–processing module with respect to agreement and tense and mood information, but we have moved the value encoding the verbal status to a new tag–feature, so–called STATUS (118), thus avoiding diversify TS–LS rules for auxiliary and main verbal tags.

(118)

```
tsl:{
     attr => 'M',
     feature_list =>  [
            ft( 'LEMMA', accionar ),
            ft( 'POS', 'Vsp2s' ),
            ft( 'STATUS', 'm' )],
     data_content => acciones }
```

We have implemented a TS–LS rule for each of the values we have for the POS tag–feature, which specify the values of the LD's morphological attributes VFORM, AGR and TNS_MOOD, and the attributes encoding the semantic tense and aspect. These rules, in addition, propagate the values of the tag–features STATUS and LEMMA and the tag data–content to the LD's attributes STATUS, LEMMA and MORPHEME.

(119) shows the rule we defined to lift the tag 'Vsp2s-'.

(119)

ts_ls_rule(

$$\left[\begin{array}{ll} \textit{ld} & \\ \text{SPEC} & \textit{t\_lex\_spec} \\ \text{SYNSEM}\,|\,\text{LOCAL} & \left[\begin{array}{ll} \textit{t\_local\_morph} & \\ \text{BAR} & \textit{minus} \\ \text{MORPH} & \left[\begin{array}{ll} \textit{t\_vinfl} & \\ \text{MORPHEME} & \boxed{1} \\ \text{LEMMA} & \boxed{2} \\ \text{STATUS} & \boxed{3} \\ \text{VFORM} & \textit{fin} \\ \text{AGR} & \textit{(p2\&sing)} \\ \text{TNS\_MOOD} & \textit{(pres\&subj)} \end{array}\right] \\ \text{SEM} & \left[\begin{array}{ll} \textit{t\_sem} & \\ \text{INDX} & \left[\begin{array}{ll} \textit{t\_indx\_eve} & \\ \text{TNS} & \textit{sf(tense(simul,R,U))} \\ \text{ASP} & \textit{sf(tense(perf,E,R))} \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right],$$

'M', [POS = 'Vsp2s-', LEMMA = $\boxed{2}$, STATUS = $\boxed{3}$], $\boxed{1}$).

Note that these TS–LS rules instantiate the morphological features (VFORM, AGR, TNS_MOOD) defining the morphological type *t_vinfl*. Lexical entries co–index them with their appropriate morpho–syntactic attributes. Entries for main verbs, which take the value *m* for the attribute VSTATUS, co–index them with the (SYNSEM|LOCAL|CAT|HEAD|)VFORM, AGR and TNS_MOOD attributes (120). Entries for auxiliary verbs, which take the value *a* for the attribute VSTATUS, co–index them with the (SYNSEM|LOCAL|CAT|SPECIFIES| SYNSEM|LOCAL|CAT|HEAD|)VFORM, AGR and TNS_MOOD attributes.[15]

(120)

$$\left[\begin{array}{ll}
\textit{ld} & \\
\text{SPEC} & \textit{t\_lex\_spec} \\
\text{SYNSEM}\,|\,\text{LOCAL} & \left[\begin{array}{ll}
\textit{t\_local\_morph} & \\
\text{MORPH} & \left[\begin{array}{ll}
\textit{t\_vinfl} & \\
\text{LEMMA} & \textit{accionar} \\
\text{MORPHEME} & \textit{acciones} \\
\text{VFORM} & \boxed{1}\textit{fin} \\
\text{AGR} & \boxed{2}\textit{(p2\&sing)} \\
\text{TNS\_MOOD} & \boxed{3}\textit{(pres\&subj)} \\
\text{VSTATUS} & \textit{m}
\end{array}\right] \\
\text{CAT} & \left[\begin{array}{ll}
\textit{t\_subst} & \\
\text{HEAD} & \left[\begin{array}{ll}
\textit{t\_verb} & \\
\text{VFORM} & \boxed{1} \\
\text{AGR} & \boxed{2} \\
\text{TNS\_MOOD} & \boxed{3}
\end{array}\right]
\end{array}\right] \\
\text{SEM} & \left[\begin{array}{ll}
\textit{t\_sem} & \\
\text{INDX} & \left[\begin{array}{ll}
\textit{t\_indx\_eve} & \\
\text{TNS} & \textit{sf(tense(simul,R,U))} \\
\text{ASP} & \textit{sf(aspect(perf,E,R))}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

[15]Such redundant information is removed at the phrase structure level where the SYNSEM|LOCAL|MORPH attribute is not percolated.

(c)– Finally, TS–LS rules for closed class items have been defined along categorial dimensions, so that, instead of defining a rule for each of the tags we have for adverbs, prepositions, conjunctions, pronouns, determiners and articles, a single rule is needed for each category. These rules propagate the value of the tag–feature LEMMA and the tag data– content to the LD attributes LEMMA and MORPHEME, respectively. This strategy, however, requires to include an entry for each full–form at the parsing (and refinement) lexicon.

(121) shows the rules we have defined for determiners.

(121)

ts_ls_rule(

$$\left[\begin{array}{ll} \textit{ld} & \\ \text{SPEC} & \textit{t\_lex\_spec} \\ \text{SYNSEM}\,|\,\text{LOCAL} & \left[\begin{array}{ll} \textit{t\_local\_morph} & \\ \text{BAR} & \textit{minus} \\ \text{MORPH} & \left[\begin{array}{ll} \textit{t\_morph} & \\ \text{MORPHEME} & \boxed{1} \\ \text{LEMMA} & \boxed{2} \end{array}\right] \\ \text{CAT} & \textit{t\_det} \end{array}\right] \end{array}\right],$$

`'M'`, [`POS = 'DET'`, `LEMMA =` $\boxed{2}$], $\boxed{1}$).

Besides the TS–LS rules for the tag 'M' we have just presented, our system requires a TS–LS rule accounting for the tag 'W', which we show in (122). This rule lifts tag 'W' of type 'WORD' —for words which are not included in a chunk— into a target LD which is specified as 'SYNSEM|LOCAL|BAR: *zero*' and 'SPEC|LEVEL: *m2w*', i.e. it will be computed when parsing 'morpheme to words' (cf. section 2.6.4).

(122)

ts_ls_rule(

$$\left[\begin{array}{ll} \textit{ld} & \\ \text{SPEC} & \left[\begin{array}{l} \textit{t\_spec\_str} \\ \text{LEVEL: } \textit{m2w} \end{array}\right] \\ \text{SYNSEM} & \left[\begin{array}{ll} \textit{t\_synsem} & \\ \text{LOCAL} & \left[\begin{array}{l} \textit{t\_local\_morph} \\ \text{BAR: } \textit{zero} \end{array}\right] \end{array}\right] \end{array}\right],$$

`'W'`, [`'TYPE' = 'WORD'`]).

4.3.2 Lifting Chunks

Similar to the integration of PoS information, the integration of the structures produced by the chunker in the ALEP system requires TS–LS rules to convert those structures into LD data structures used by the linguistic processing components of ALEP.

In the system proposed by [Fouvry & Bredenkamp 97], these structures are lifted between the root S–node and the W–nodes, based on the assumption that there is this level of partial parsing. This intermediate level is used to guide further processing by providing the analysis component the nodes which it will have to compute.

The system we have implemented does not require any further intermediate level for intra–clausal structures, but it integrates chunk mark–ups at the 'W'–nodes by means of TS–LS rules which lift tags 'W' of type 'CHUNK'.

The system we propose, in addition, can integrate into the linguistic processing components of ALEP linguistic descriptions which do not need to be re-built by phrase structure rules —therefore, avoiding a duplication of efforts—, since, even though they are quite underspecified with respect to the head element of the chunk (they only have information about its part–of–speech), they already specify syntactic and semantic information about the non–head elements that have been attached to the head element.

This strategy, however, requires very specialized TS–LS rules not only with respect to the category of the head element but also the number, category and type of non–head elements.

The information about the elements within the chunk is specified in the feature–list of the structural tag of the `ts-chunk` structure (123.b) which is produced in a previous step by an external conversion module which was developed to convert the tabular output format of the pre–processing module (123.a).[16]

(123)

```
a.
   CHUNKo      NX
   TOK         las         el\Tdfp-
   TOK         próximas    próxima\Afpfp-
   TOK         ventas      venta\Ncfp-
   CHUNKc      NX
```

[16]This external module, in addition, removes the chunk mark–ups of those chunks which contain ambiguous items.

```
b. tsn:{ attr => 'W',
         feature_list => [ ft( 'TYPE', 'CHUNK'),
                           ft( 'CT', 'NX'),
                           ft( 'DT', 'Def'),
                           ft( 'ART', 'ART'),
                           ft( 'AT', 'Qual'),
                           ft( 'ADJ', 'ADJ') ],
         children => [
           tsl:{ attr => 'M',
                 feature_list => [ ft( 'LEMMA', 'el'),
                                   ft( 'POS', 'Tdfp-') ],
                 data_content => las }
           tsl:{ attr => 'M',
                 feature_list => [ ft( 'LEMMA', 'próximo'),
                                   ft( 'POS', 'Afpfp-') ],
                 data_content => próximas }
            tsl:{ attr => 'M',
                  feature_list => [ ft( 'LEMMA', 'venta'),
                                    ft( 'POS', 'Ncfp-') ],
                  data_content => ventas } ] }
```

In the example we show in (123.b), the tag–feature CT is used to specify the type of chunk (NX). The tag–features DT and AT are used to specify the article type (definite) and the adjective type (qualificative), respectively. Finally, the tag–features ART and ADJ specify the base form (or lemma) of the article and the adjective.

The TS–LS rules lifting chunk mark–ups include:

(a)– two rules for adjectival chunks: one for chunks which have the adjectival head element and an adverb (e.g. *muy interesantes* (very interesting)), and one for chunks which have the adjectival head element and two adverbs (e.g. *mucho más interesantes* (much more interesting)).

(b)– two rules for participial chunks: one for chunks which have the participial head element and an adverb (e.g. *suficientemente explicado* (enough explained)), and one for chunks which have the participial head element and two adverbs (e.g. *tan claramente verbalizada* (so clearly expressed)).

(c)– three rules for adverbial chunks: one for chunks which have the adverbial head element and an adverb (e.g. *muy pronto* (very soon)), one for chunks which have the adverbial head element and an appositional noun (e.g. *hoy lunes* (today Monday)), and one for chunks which have the adverbial head element and two adverbs (e.g. *mucho más pronto* (much earlier)).

Rules adjectival, participial and adverbial chunks create a LD which specifies the part–of–speech of the head element (i.e. the value for the attribute HEAD) and which adds the semantic attributes of the modifier into the semantic MODS–list of the head element.

(124) shows the rule for adjectival chunks which have the head element and an adverb.

(124)

ts_ls_rule(

$$\left[\begin{array}{ll} \textit{ld} & \\ \text{STRING} & <\textit{muy interesante}> \\ \text{SPEC}\,|\,\text{LEVEL} & \textit{m2w} \\ \text{SYNSEM}\,|\,\text{LOCAL} & \left[\begin{array}{ll} \textit{t\_local\_morph} & \\ \text{BAR:} & \textit{zero} \\ \text{CAT:} & \left[\begin{array}{l} \textit{t\_subst} \\ \text{HEAD: } \textit{t\_adj} \end{array}\right] \\ \text{SEM:} & \left[\begin{array}{ll} \textit{t\_sem} & \\ \text{INDX:} & \left[\begin{array}{l} \textit{t\_indx} \\ \text{IND: } \textit{sf(index(\_,}\boxed{1}\textit{))} \end{array}\right] \\ \text{MODS:} & < \left[\begin{array}{ll} \textit{t\_sem\_mod} & \\ \text{REL:} & \textit{sf(rel(degree,}\boxed{1}\textit{,}\boxed{2}\textit{))} \\ \text{INDX:} & \left[\begin{array}{l} \textit{t\_indx} \\ \text{IND: } \textit{sf(index(nevent,}\boxed{2}\textit{))} \end{array}\right] \\ \text{PREDARG:} & \left[\begin{array}{l} \textit{t\_predarg} \\ \text{PRED: } \textit{sf(pred(}\boxed{2}\textit{,}\boxed{2}\textit{))} \\ \text{ARGS}\,|\,\text{ARITY: } \textit{zero} \end{array}\right] \end{array}\right] > \end{array}\right] \end{array}\right] \end{array}\right],$$

'W', ['TYPE' = 'CHUNK', CT = 'AX', ADV = $\boxed{2}$]).

If the chunk includes two modifiers, the LD adds the semantic attributes of the first one into the semantic MODS–list of the second one, as we show in the following example.

(125)

ts_ls_rule(

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{ld}} \\
\multicolumn{2}{l}{\text{STRING: } <\textit{mucho más interesante}>} \\
\multicolumn{2}{l}{\text{SPEC}\,|\,\text{LEVEL : } \textit{m2w}} \\
\multicolumn{2}{l}{\text{SYNSEM}\,|\,\text{LOCAL}\,|\,\text{BAR: } \textit{zero}} \\
\text{SYNSEM}\,|\,\text{LOCAL}\,|\,\text{CAT:} & \left[\begin{array}{l} \textit{t\_subst} \\ \text{HEAD: } \textit{t\_adj} \end{array}\right] \\
\text{SYNSEM}\,|\,\text{LOCAL}\,|\,\text{SEM:} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_sem}} \\
\text{INDX:} & \left[\begin{array}{l} \textit{t\_indx} \\ \text{IND: } \textit{sf(index(\_,}\boxed{1}\textit{))} \end{array}\right] \\
\text{MODS:} & < \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_sem\_mod}} \\
\multicolumn{2}{l}{\text{REL: } \textit{sf(rel(degree,}\boxed{1}\textit{,}\boxed{2}\textit{))}} \\
\text{INDX:} & \left[\begin{array}{l} \textit{t\_indx} \\ \text{IND: } \textit{sf(index(nevent,}\boxed{2}\textit{))} \end{array}\right] \\
\text{PREDARG:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t\_predarg}} \\ \multicolumn{2}{l}{\text{PRED: } \textit{sf(pred(}\boxed{2}\textit{,}\boxed{2}\textit{))}} \\ \text{ARGS:} & \left[\begin{array}{l} \textit{t\_args} \\ \text{ARITY: } \textit{zero} \end{array}\right] \end{array}\right] \\
\text{MODS:} & < \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{t\_sem\_mod}} \\
\multicolumn{2}{l}{\text{REL: } \textit{sf(rel(degree,}\boxed{2}\textit{,}\boxed{3}\textit{))}} \\
\text{INDX:} & \left[\begin{array}{l} \textit{t\_indx} \\ \text{IND: } \textit{sf(index(nevent,}\boxed{3}\textit{))} \end{array}\right] \\
\text{PREDARG:} & \left[\begin{array}{l} \textit{t\_predarg} \\ \text{PRED: } \textit{sf(pred(}\boxed{3}\textit{,}\boxed{3}\textit{))} \\ \text{ARGS}\,|\,\text{ARITY: } \textit{zero} \end{array}\right]
\end{array}\right] >
\end{array}\right] >
\end{array}\right]
\end{array}\right],$$

'W', ['TYPE' = 'CHUNK', CT = 'AX', ADV_1 = $\boxed{2}$, ADV_2 = $\boxed{3}$]).

(d)– one rule for verbal chunks. Though verbal chunks may include up to four elements (cf. section 3.5.5), only one rule is required to deal with all kind of verbal chunks. This rule, which we show in (126), creates a LD whose HEAD attribute takes the value *t_verb* and whose semantic MODS–list is empty, in order to distinguish them from participial chunks.

(126)

ts_ls_rule(

$$\left[\begin{array}{ll} \textit{ld} & \\ \text{SPEC}\,|\,\text{LEVEL} & \textit{m2w} \\ \text{SYNSEM}\,|\,\text{LOCAL} & \left[\begin{array}{ll} \textit{t\_local\_morph} & \\ \text{BAR:} & \textit{zero} \\ \text{CAT:} & \left[\begin{array}{l} \textit{t\_subst} \\ \text{HEAD:}\ \textit{t\_verb} \end{array}\right] \\ \text{SEM:} & \left[\begin{array}{l} \textit{t\_sem} \\ \text{MODS:}\ <> \end{array}\right] \end{array}\right] \end{array}\right],$$

`'W', [ 'TYPE' = 'CHUNK', CT = 'VX']).`

(e)– rules for nominal chunks.[17] Nominal chunks may contain up to five elements. The non–head elements are: determiners (indefinite, demonstrative, possessive, relative and interrogative), articles (definite and indefinite), cardinals, adjectives (qualificative, indefinite and ordinal) and participles. They create a LD whose HEAD attribute takes the value *t_noun*. If the chunk includes articles or determiners, the LD specifies the value for the attribute SYNSEM|LOCAL|CAT|SPECIFIER|LOCAL|CAT|DTYPE and the value for the semantic attribute QFORCE, according to the type of determiner or article.[18] If the chunk includes one (or more) adjunct, the LD adds its (or their) semantic features to the semantic MODS–list of the nominal head. Here, indefinite adjectives are characterized as non–predicative adjectives; qualificative and ordinal adjectives are characterized as mono–valent predicative adjectives whose arg1 (external argument) index is co–indexed with

[17]The rules for nominal chunks that we have implemented basically cover the nominal chunks we had in the corpus. We will extend them in future work.

[18]Note that, since the attribute DTYPE takes a boolean value, we can not percolate the value of the tag–feature DT to the LD, but we must diversify rules for the different values it may take.

that of the modified noun; finally, participles are unspecified with respect to the third and fourth argument, their arg2 (least oblique argument) index is shared with that of the modified noun.

Example (127) shows the TS–LS rule lifting the nominal chunk we showed in (123.b). It creates a LD which gives the value *def* to the attribute SYNSEM|LOCAL|CAT|SPECIFIER|LOCAL|CAT|DTYPE, instantiates the value of the semantic attribute QFORCE, and adds the semantic features of the adjective to the semantic MODS–list of the head.

(127)

ts_ls_rule(

$$
\left[\begin{array}{l}
\textit{ld}\\
\text{STRING: } <\textit{las próximas ventas}>\\
\text{SPEC}\,|\,\text{LEVEL: } \textit{m2w}\\
\text{SYNSEM}\,|\,\text{LOCAL: }\left[\begin{array}{l}
\textit{t\_local\_morph}\\
\text{BAR:}\textit{zero}\\
\text{CAT: }\left[\begin{array}{l}
\textit{t\_subst}\\
\text{HEAD: } \textit{t\_noun}\\
\text{SPECIFIER: } <\left[\begin{array}{l}\textit{t\_synsem}\\ \text{LOCAL}\,|\,\text{CAT}\,|\,\text{DTYPE: } \textit{def}\end{array}\right]>
\end{array}\right]\\
\text{SEM: }\left[\begin{array}{l}
\textit{t\_sem}\\
\text{INDX: }\left[\begin{array}{l}\textit{t\_indx}\\ \text{IND: } \textit{sf(index(\_,}\boxed{1}\textit{))}\\ \text{QFORCE: } \textit{sf(qfc(}\boxed{2}\textit{,plur,def,}\boxed{1}\textit{))}\end{array}\right]\\
\text{MODS:}<\left[\begin{array}{l}
\textit{t\_sem\_mod}\\
\text{REL: } \textit{sf(rel(specification,}\boxed{1}\textit{,}\boxed{3}\textit{))}\\
\text{INDX}\,|\,\text{IND: } \textit{sf(index(\_,}\boxed{3}\textit{))}\\
\text{PREDARG: }\left[\begin{array}{l}
\textit{t\_predarg}\\
\text{PRED: } \textit{sf(pred(}\boxed{3}\textit{,}\boxed{3}\textit{,}\boxed{1}\textit{))}\\
\text{ARGS}\,|\,\text{ARG1}\,|\,\text{REL: } \textit{sf(rel(arg1,}\boxed{3}\textit{,}\boxed{1}\textit{))}\\
\text{ARGS}\,|\,\text{ARG1}\,|\,\text{INDX}\,|\,\text{IND: } \textit{sf(index(\_,}\boxed{1}\textit{))}\\
\text{ARGS}\,|\,\text{ARG2: } \textit{t\_sem\_nil}\\
\text{ARGS}\,|\,\text{ARG3: } \textit{t\_sem\_nil}\\
\text{ARGS}\,|\,\text{ARG4: } \textit{t\_sem\_nil}
\end{array}\right]
\end{array}\right]>
\end{array}\right]
\end{array}\right]
\end{array}\right],
$$

`'W', [ 'TYPE' = 'CHUNK', CT = 'NX', AT = 'Def', ART = `$\boxed{2}$`, AT = 'Qual', ADJ = `$\boxed{3}$`]).`

Like the TS–LS rule accounting for words which are not wrapped within a chunk (cf. section 4.3.1), target LDs in these TS–LS rules are specified as:

$$\begin{bmatrix} \textit{ld} & \\ \text{SPEC:} & \begin{bmatrix} \textit{t\_spec\_str} \\ \text{LEVEL: } \textit{m2w} \end{bmatrix} \\ \text{SYNSEM:} & \begin{bmatrix} \textit{t\_synsem} & \\ \text{LOCAL:} & \begin{bmatrix} \textit{t\_local\_morph} \\ \text{BAR: } \textit{zero} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

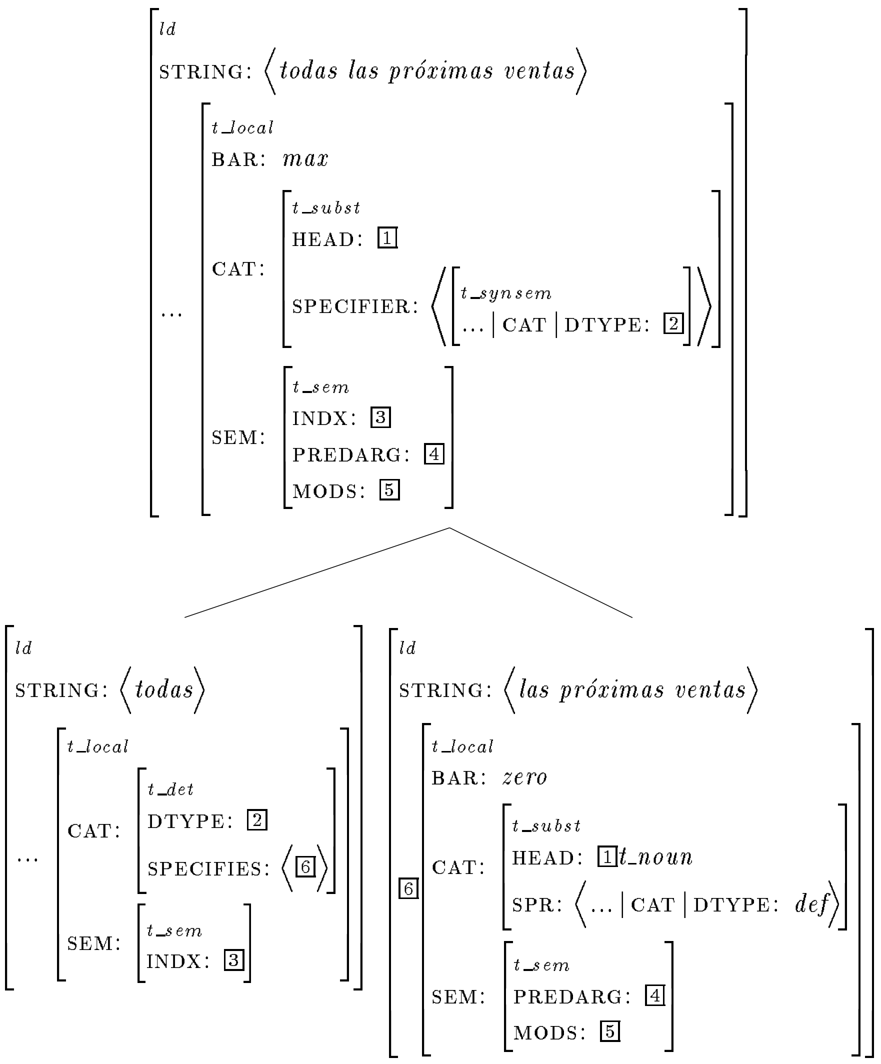
On the one hand, by creating a 'SPEC: *m2w*' node (i.e. it will be computed when parsing 'morpheme to words') we avoid any interference with the set of phrase structure rules which build up the same type of LD, but which are applied when parsing 'words to sentences', since they are specified as 'SPEC|LEVEL: *w2s*'. These rules are maintained to build up nodes which have not been recognized by the pre–processing module. Furthermore, by taking the value *m2w*, unary word structure rules, which in ALEP are used to simulate the operations that LRs perform (cf. section 2.6.2), can be applied on top of these LDs.

On the other hand, by creating a 'SYNSEM|LOCAL|BAR: *zero*' node, we avoid modifying the phrase structure rules which build up a LD on top of these LDs: (i) to attach modifiers and/or specifiers to the left of the head element when the chunk has only been partially recognized (128), (ii) and/or to attach post–head sisters (modifiers and/or complements to the right of the head element) (129).

Note, however, that (129) shows a rather flat type of phrase structure description —[[[Spr Adjct Head] Adjct] Compl]— in contrast with the standard approach to phrase structure description —[Spr [Adjct [[Head Adjct] Compl]]]— referred to as the X'–theory, which, in HPSG is licensed by the ID Schemata (cf. section 2.6.3), according to which: (i) complements are sisters of lexical nodes, (ii) specifiers are attached at *phrase* [SPR<Y">] (i.e. the level X') and have a mother node [SPR< >] (i.e. X"), and (iii) pre–head adjuncts are attached at [SPR<Y">] (the level X') and have a mother node [SPR<Y">] (X').[19]

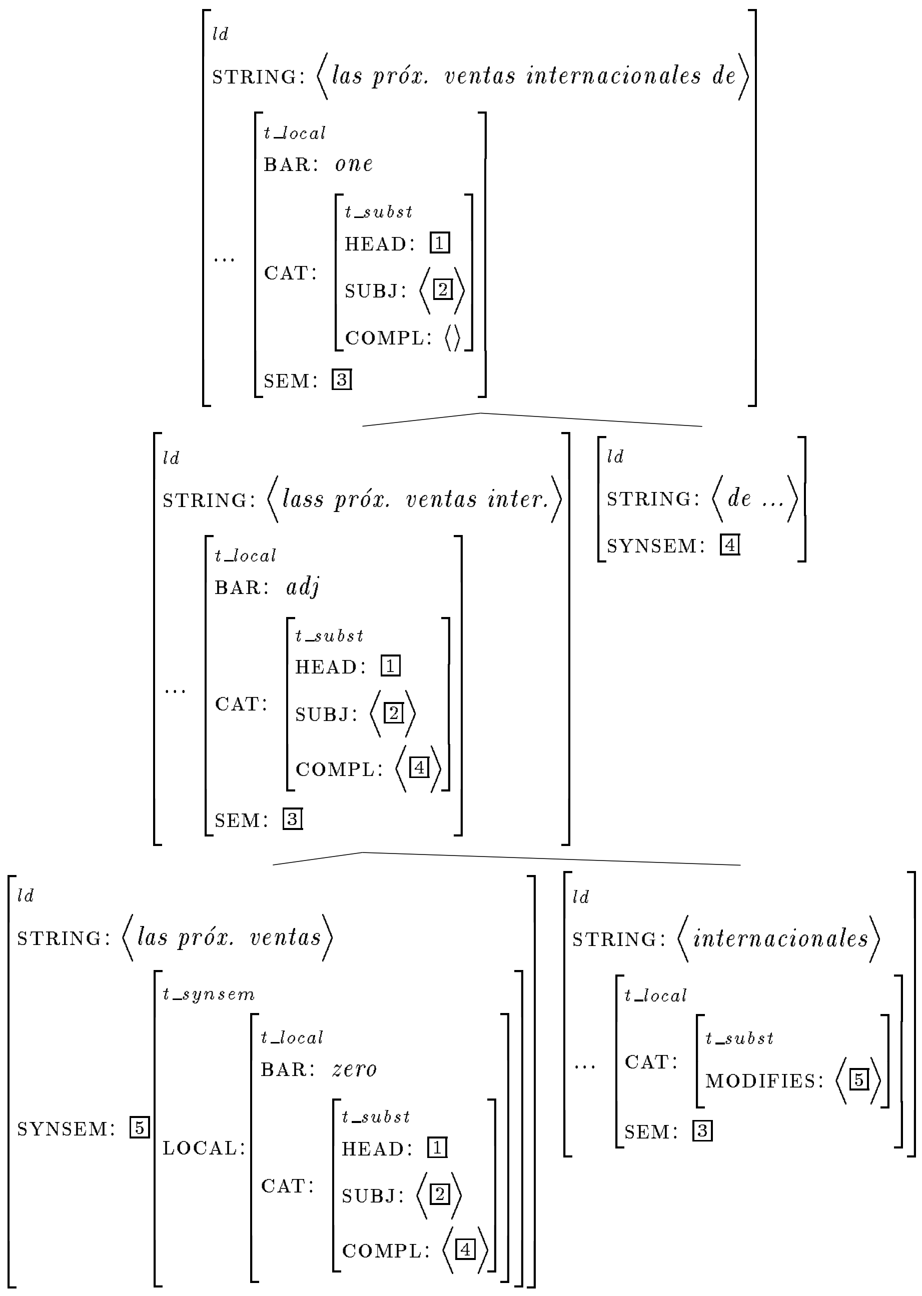
[19]Our proposal, however, does not change the phrase structure description for verbal group constituents, since, following [Hinrichs & Nakazawa 94], auxiliaries and verbal markers are treated as sisters of lexical main verb LDs so that they combine with them forming a constituent before they combine with their complements. See [Badia *et al.* 96] for motivations.

(128) [20]



[20]Dots (...) abbreviate the path SYNSEM|LOCAL.

(129) [21]



[21]Dots (...) abbreviate the path SYNSEM|LOCAL.

4.3.3 Word Structure Rules

Besides the TS–LS rules we have presented, the strategy we use also requires new unary word structure rules which consolidate the output of the lifting operation for tags 'M' and 'W', where 'M' corresponds to daughter node and 'W' corresponds to mother node.

We have two different types of word structure rules:

(a)- Word structure rules where both the LD of the mother node and the LD of the daughter node represent full–forms. These rules deal with lexical items which are not included in a chunk. They project the value of the BAR attribute from *minus* to *zero* and they percolate the local categorial and semantic, and the non–local information of the daughter node to the mother node.

Here we have distinguished two rules:

- one rule for functional categories. It specifies that the attribute SYNSEM|LOCAL|CAT has the value *funct*.
- one rule for substantive categories, which we show in (130). It specifies that: (i) the attribute SYNSEM|LOCAL|CAT has the value *subst*, (ii) the mother node bears an empty (SYNSEM|LOCAL|SEM|)MODS–list, and (iii) the mother node bears an empty (SYNSEM|LOCAL|CAT|)SPR–list.

(130)

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{ld}} \\ \text{SPEC:} & \textit{t\_lex\_spec} \\ \text{SYNSEM:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t\_synsem}} \\ \text{LOCAL:} & \left[\begin{array}{l} \textit{t\_local\_morph} \\ \text{BAR: } \textit{minus} \\ \text{CAT: } \boxed{1} \\ \text{SEM: } \boxed{2} \end{array}\right] \\ \text{NLOCAL:} & \boxed{3} \end{array}\right] \end{array}\right] \rightarrow \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{ld}} \\ \multicolumn{2}{l}{\text{SPEC}\,|\,\text{LEVEL: } \textit{m2w}} \\ \text{SYNSEM:} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{t\_synsem}} \\ \text{LOCAL:} & \left[\begin{array}{l} \textit{t\_local\_morph} \\ \text{BAR: } \textit{zero} \\ \text{CAT: } \boxed{1} \left[\begin{array}{l} \textit{t\_subst} \\ \text{SPR: } <> \end{array}\right] \\ \text{SEM: } \boxed{2} \left[\begin{array}{l} \textit{t\_sem} \\ \text{MODS: } <> \end{array}\right] \end{array}\right] \\ \text{NLOCAL:} & \boxed{3} \end{array}\right] \end{array}\right]$$

(b)- Word structure rules where the mother node represents a chunk and the daughter nodes represents a full–form. These rules deal with chunks and they are distinguished along categorial dimensions. All of them project the value of the BAR attribute from *minus* to *zero* and specify the value for the attribute SYNSEM|LOCAL|CAT|HEAD.

- Rules for adjectival, participial and adverbial chunks, in addition, (i) specify that the mother node bears a non–empty SYNSEM|LOCAL|SEM|MODS–list, and (ii) percolate the non–local, the categorial and the semantic information with respect to predicate argument structure and functional semantics of the daughter node to the mother node.

- Nominal chunks are dealt with two word structure rules which are distinguished according to whether the chunk includes specifiers and/or adjuncts.

  - The rule for nominal chunks which includes adjuncts specifies that the mother node bears a non–empty SYNSEM|LOCAL|SEM|MODS–list and an empty SYNSEM|LOCAL|CAT|SPECIFIER–list.

  - The rule for nominal chunks which includes specifiers specifies that the mother node bears a non–empty SYNSEM|LOCAL|CAT|SPECIFIER–list, but leaves its value for SYNSEM|LOCAL|SEM|MODS unspecified, such that it can also deal with chunks having both specifiers and adjuncts. With this, only two rules are needed.

Both rules percolate the non–local, and the rest of the categorial and semantic information of the daughter node to the mother node.[22]

(131) shows the rules for adjectival chunks.

[22]No word structure rules is needed for verbal chunks.

(131)

$$\begin{bmatrix} \textit{ld} \\ \text{STRING: } \boxed{1} \\ \text{SPEC: } \textit{t\_lex\_spec} \\ \text{SYNSEM: } \begin{bmatrix} \textit{t\_synsem} \\ \text{LOCAL: } \begin{bmatrix} \textit{t\_local\_morph} \\ \text{BAR: } \textit{minus} \\ \text{CAT: } \boxed{2}\begin{bmatrix} \textit{t\_subst} \\ \text{HEAD: } \textit{t\_adj} \end{bmatrix} \\ \text{SEM: } \begin{bmatrix} \textit{t\_sem} \\ \text{INDX: } \boxed{3} \\ \text{PREDARG: } \boxed{4} \end{bmatrix} \end{bmatrix} \\ \text{NLOCAL: } \boxed{5} \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \textit{ld} \\ \text{STRING: } \boxed{1} \\ \text{SPEC} \mid \text{LEVEL: } \textit{m2w} \\ \text{SYNSEM: } \begin{bmatrix} \textit{t\_synsem} \\ \text{LOCAL: } \begin{bmatrix} \textit{t\_local\_morph} \\ \text{BAR: } \textit{zero} \\ \text{CAT: } \boxed{2} \\ \text{SEM: } \begin{bmatrix} \textit{t\_sem} \\ \text{INDX: } \boxed{3} \\ \text{PREDARG: } \boxed{4} \\ \text{PREDARG: } <\_> \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{NLOCAL: } \boxed{5} \end{bmatrix}$$

4.4 Default Lexical Entries

Supplementary to the integration of the shallow processing tool, default lexical entries have been implemented in our ALEP grammar to provide robust deep processing.[23]

Default lexical entries are lexical entry templates that are activated when the system can not find a particular lexical entry to apply. Note that having default lexical entries in such a highly lexicalized grammar increases ambiguity, and, thus, the parsing search space, unless a mechanism is used to restrict as much as possible the templates that are activated. The integration of the tagger,

[23]Default lexical entries for ALEP were already supplied during the LS–GRAM project for the Danish grammar [Music & Navarretta 96]. Besides default entries for parsing and refinement, the Danish implementation integrates into the ALEP system a so–called "Robust TLM processing" functionality. This functionality allows the TLM component to output always a result for every input word. The results, however, are underspecified, words that can not be parsed using the TLM component are simply copied to the output and marked as STATUS=NOK, to distinguish them from normal results, which are marked as STATUS=OK. Special lifting rules convert them into LDs. [Music & Navarretta 96], however, do not show any figures on the system performance.

which supplies the PoS information to the linguistic processing modules of our system, allows us to increase robustness while avoiding increase in part–of–speech ambiguity.

There are two basic ways to define default lexical entries. One is to implement underspecified lexical entry templates assigned to each major word class such that, while parsing, the system fills in the missing information in the lexical entry template of each unknown word by the application of phrase structure rules (or rule schemata and principles, in HPSG–based grammars) [Mitsuishi *et al.* 98, Horiguchi *et al.* 95, Music & Navarretta 96, Grover & Lascarides 01].[24] In the other approach, very detailed default lexical entries for each major word class are defined.

The approach we have followed falls under a middle type. We have defined default lexical entries for the different major word classes —verbs, nouns, adjectives and adverbs— which cover their most frequent subcategorization frames. These templates, however, are unspecified with respect to those features which in the lexical macros of our ALEP grammar are parametrized (i.e. they are encoded in the lexical entries). These features include syntactic head features encoding the word type (verb, noun, adjective and adverb type), the features that constrain the application of unary structure rules and which deal with passivization, decausativization, optionality of complements, etc., the features encoding the lexical restrictions on subcategorized for elements, e.g. semantic properties of subject and complements, and marking preposition, etc. This information is filled by the application of phrase structure rules.[25]

In particular, our lexical entry templates cover:

- verbs
    - intransitive verbs.
    - verbs taking nominal direct objects.
    - verbs taking infinitives/sentential completive clauses.

[24][Music & Navarretta 96] have a single default at analysis for each major word class, and they have several defaults at refinement for some word classes, e.g. verbs have five rules for zero–valent, mono–valent, bi–valent, tri–valent and tetra–valent verbs.

[25]The features encoding semantic properties of subcategorized for element can only be filled by the application of rules if the words that appear in the linguistic expression being processed are encoded in the lexicon.

  - verbs taking oblique nominal complements.
  - verbs taking oblique infinitives/sentential complements.
  - verbs taking nominal direct objects and indirect objects.
  - verbs taking nominal direct objects and oblique nominal complements.
  - verbs taking nominal direct objects and oblique infinitives/sentential complements.

- nouns
  - zero–valent nouns.
  - mono–valent argumental nouns.
- adjectives
  - predicative adjectives taking no complement.
  - predicative adjectives taking a nominal complement introduced by a marking preposition.
- adverbs
  - sentential or verbal modifiers.

As we will show in section 4.5 (cf. table 4.5), first experiments testing the effect of our default lexical entries showed that, by covering the most frequent subcategorization frames, we ensured that the accuracy of the grammar —percentage of input sentences that received the correct analysis— remained the same. The precision of the grammar —percentage of input sentences that received no superfluous (or wrong) analysis—, however, was very low, since we had no information about the subcategorized for elements to restrict the lexical entry templates to be triggered.

To reduce overgeneration without losing coverage, we extended the PoS tags of our external lexicon (i.e. the lexicon we use for morpho–syntactic annotation) so that they included subcategorization information —frames, marking preposition, verb form/mood of subcategorized for elements— information indicating the verb type, and information indicating the copula verb adjectives may take.

Table 4.1 presents the attributes and values for the different categories with which the PoS tagset was extended.[26]. Bold types represent the characters appearing in the tags. Following the MULTEXT notation format, if an attribute is not relevant for a lexical item, the corresponding position takes '-' as value.

Category	*Attribute*	*Value*
verb	v_type	**p**ronominal, **n**on-pronominal, prono**m**inalizable
	f_altern	**p**assivizable, **d**ecausative
	v_frame	(see table 4.2)
	mk_prep	**a**, **c**on, co**n**tra, **d**e, **e**n, ent**r**e, **p**ara, p**o**r, seg**u**n, **s**obre
	v_form_mood	**i**ndicative, **s**ubjunctive, **g**erund, i**n**finitive
noun	n_frame	(see table 4.2)
	mk_prep	**a**, **c**on, co**n**tra, **d**e, **e**n, ent**r**e, **h**acia, **p**ara, p**o**r, **s**obre
adjective	a_frame	(see table 4.2)
	mk_prep	**a**, **c**on, **d**e, **e**n, **p**ara, p**o**r
	copula	ser, estar, both

Table 4.1 Categories, attributes and values.

To encode the different frames the strategy we investigated is inspired by EUROTRA [EUROTRA 91], where frames are coded by a single letter, e.g. 'a' for verbs taking nominal subjects, 'b' for verbs taking nominal subjects and indirect objects, 'c' for verbs taking nominal subjects and direct objects, etc. (cf. table 4.2), and where letter can be combined compositionally to deal with frame alternations. Thus, a verb taking a nominal subject and an optional direct object takes the value 'ac', whereas a verb taking taking a nominal subject, a direct object and an optional indirect object takes the value 'ce'. Following the MULTEXT notation format, combined letters appear between square brackets, such that they only occupy a position in the MSD, e.g. '[ac]'.

[26]This information was not manually encoded, but it was extracted from the lexical resources developed in the project PAROLE (LE2–4017) [Melero & Villegas 98, Villegas *et al.* 98]. We foresee to add semantic information, e.g. lexical semantic restrictions on subcategorized for elements, from the lexical resources developed in the project SIMPLE (LE4–8346) [Lenci *et al.* 00]

Category	*Frame*	*Value*
verb	Empty	-
	Snp	a
	SnpOnp	c
	SnpOcl/vp	d
	SnpIo	b
	SnpPccl/vp	r
	SnpPcnp	j
	SnpLcLOC	k
	SnpOclIo	f
	SnpOclPcnp	g
	SnpOnpAto	l
	SnpOnpIo	e
	SnpOnpLcLOC	n
	SnpOnpPcadj	i
	SnpOnpPcnp	m
	SnpOnpPccl/vp	s
	SnpOnpXc	q
	SnpIoPcnp	h
	SnpPcnpPcnp	o
	SnpOnpPcnpPcnp	p
	SnpOnpIoPcnp	t
	Scl/vp	x
	Scl/vpIo	y
	SclOnp	z
	Pcnp	v
	Pccl/vp	w
	IoPcnp	u
	IoPccl/vp	ñ
noun	Empty	-
	Npcnp (de)	c
	Npcnp	j
	Npcnp(de)Npcnp	m
	Npcvpinf	z
	Npcnp(de)Ppcvpinf	q
	NpcLOC	k
	Npcnp(de)NpcLOC	n
	Npcclind	d
	Npcclsubj	t

adjective	Empty	-
	Apcnp	c
	Apcvpinf	d

Table 4.2 Categories, frames and values.

(132) shows two examples of lexical entries encoding this information, and (133) shows how it is distributed along different tag–features in the `ts-chunk` structure.

(132)

FULL–FORM	LEMMA	LEXICAL TAG	
acciones	accionar	Vmsp2s-mp[ac]--	((you) activate)
acciones	acción	Ncfp---	(stock shares)

(133) [27]

```
tsl:{ attr => 'M',
      feature_list => [
              ft( 'LEMMA',' accionar' ),
              ft( 'POS', 'Vsp2s' ),
              ft( 'STATUS', 'm' ),
              ft( 'V.TYPE', 'pronom' ),
              ft( 'PASS', 'yes' ),
              ft( 'DECA', 'no' ),
              ft( 'V_FRAME', 'ac' ),
              ft( 'MK_PREP', 'x' ),
              ft( 'V_FORM_MOOD', 'x' )],
      data_content => acciones }


tsl:{ attr => 'M',
      feature_list => [
              ft( 'LEMMA',' acción' ),
              ft( 'POS', 'Ncfs' ),
              ft( 'N_FRAME', 'x' ),
              ft( 'MK_PREP', 'x' )],
      data_content => acciones}
```

[27]Note that while splitting tags, we change the values of 'V_TYPE','PASS' and 'DECA' in order to match the values as they are specified in the type system declaration of our ALEP grammar.

Once we added and distributed this information along the different tag–features in the `ts-chunk` structure, we modified our TS–LS rules in order to lift it to the target LD. Here subcategorization information —frame, marking preposition and verb form/mood of subcategorized for elements— is propagated to a new set of features defining the types *t_aframe*, *t_nframe* and *t_vframe*, which appear as values of the attributes (SYNSEM|LOCAL|CAT|HEADJ)AFRAME, NFRAME and VFRAME. The values of the tag–features V_TYPE, PASS and DECA are propagated to the LD's attributes VTYPE, PASS and DECA.

(134)

ts_ls_rule(

$$\begin{bmatrix} \textit{ld} & \\ \text{SYNSEM}\,|\,\text{LOCAL}\,|\,\text{CAT} & \begin{bmatrix} \textit{t\_subst} & \\ \text{HEAD} & \begin{bmatrix} \textit{t\_verb} & \\ \text{VTYPE} & \boxed{1} \\ \text{NFRAME} & \begin{bmatrix} \textit{t\_vframe} & \\ \text{V\_FRAME} & \boxed{4} \\ \text{MK\_PREP} & \boxed{5} \\ \text{V\_FORM\_MOOD} & \boxed{6} \end{bmatrix} \end{bmatrix} \\ \text{ALTERN} & \begin{bmatrix} \textit{t\_altern} & \\ \text{PASS} & \boxed{2} \\ \text{DECA} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix},$$

'M', [..., V_TYPE=$\boxed{1}$, PASS=$\boxed{2}$, DECA=$\boxed{3}$, V_FRAME=$\boxed{4}$, MK_PREP=$\boxed{5}$, V_FORM_MOOD=$\boxed{6}$], ...).

ts_ls_rule(

$$\begin{bmatrix} \textit{ld} & \\ \text{SYNSEM}\,|\,\text{LOCAL}\,|\,\text{CAT}\,|\,\text{HEAD} & \begin{bmatrix} \textit{t\_noun} & \\ \text{NFRAME} & \begin{bmatrix} \textit{t\_nframe} & \\ \text{N\_FRAME} & \boxed{1} \\ \text{MK\_PREP} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix},$$

'M', [..., N_FRAME = $\boxed{1}$, MK_PREP = $\boxed{2}$], ...).

Finally, we specified the values of the features encoding framing information in each default lexical entry, such that a unique default lexical entry is activated.

(135)

```
macro(m_LEX_DEF_V_SnpOnp[],
m_LEX[
     m_LOCAL[
          m_VSTEM[LEMMA,AGR],
          m_CAT_VERB[
               m_HEAD_Vdef[AGR,ac,x,x,x],
               m_SUBJ_NP[AGR,_,ARG1],
               m_V_FR_Onp[_,ARG2]],
          m_SEM_Varg12[LEMMA,ARG1,ARG2,arg1,arg2]]
      m_NLOCAL[]] ).


macro(m_LEX_DEF_N[],
m_LEX[
     m_LOCAL[
          m_NSTEM[LEMMA,AGR],
          m_CAT_NOUN[
               m_HEAD_Ndef[AGR,x,x],
               [],
               []],
          m_SEM_N[LEMMA]]
      m_NLOCAL[]] ).
```

4.5 Experiments and Results

The two experiments described in this section were used to evaluate the performance of the integrated system both with respect to efficient processing and robustness.

In the first experiment, our goal was to perform a comparative study of the processing time of our ALEP grammar before and after the integration of the PoS tagger and chunker. For this experiment, therefore, we required testing cases which were already fully covered by our grammar before the integration of the tagger and chunker. In this experiment, we used a subset of the test suites we have used in the LS–GRAM and the MELISSA projects.

In the second experiment, our goal was to investigate to what extent the ALEP grammar benefited from the default lexical entry templates in terms of robustness. In this experiment, we tested our system on test corpus which was selected randomly and which we did not change so as to be processed (or covered) by our grammar.[28]

a)- Experiment A

To evaluate the efficiency of the system, we defined two test suites and run them with our ALEP grammar both before and after the integration of the shallow processing tools.[29]

The first test suite included short instructive sentences or queries from the corpus of the MELISSA project[30] and sentences we selected from the different test suites we have used for diagnosis and evaluation purposes in the LS–GRAM and the MELISSA projects.[31] Test cases were selected according to: (i) the syntactic function of the chunk, e.g. subject, complement and adjunct, for nominal chunks, complement and adjunct, for adjectival chunks, etc.; (ii) the position of the chunk in the sentence, and (iii) the category and the number of non–head elements. This test suite included 1500 cases (Appendix C.1 shows a few but representative cases).

In running the test suite with the new system, processing time of the overall process improved an average of 65% due to the reduction of both lexical ambiguity and sentence length.[32] The integrated architecture, however, maintained both the precision and the recall of the ALEP grammar.

Once positive results were achieved with such a type of sentential structures,

[28]Test suites and corpora are the two tools traditionally used for evaluating and testing NLP systems. The main properties of test suites are systematicity, control over data, exhaustivity, and inclusion of negative data. Test corpora, by contrast, reflect naturally occurring date (cf. [Lehmann *et al.* 96]).

[29]Experiments have been run in a 128 Mb Ultra Sparc-10. Mean CPU time values in seconds are given for 50 samples.

[30]NL utterances which users made in interacting with ICAD, an administrative purchase and acquirement handling system, employed at ONCE (Organización Nacional de Ciegos de España), dealing with budget proposals and providing information to help decision makers.

[31]These test suites are organized on the basis of a hierarchical classification of linguistic phenomena. Test suites including cases with interaction of phenomena and negative cases are also included.

[32]The reduction of the sentence length is due to the fact that elements that are wrapped together in a chunk by the pre–processing module are lifted to the parsing component of the grammar as a unique element.

we evaluated our system with much more complex sentences, showing a high interaction of phenomena. For this, we used an article —from the newspaper "El Diario Vasco" (cf. Appendix C.2) from the LS–GRAM corpus.

Table 4.3 shows the number of tokens (second column)[33], the initial morpho–syntactic ambiguity (tags/token) (third column) and the number of analyses we get for each sentence (fourth column).

Sentence desc				ALEP	+PoS		+PoS +Chunk	
#	Toks	Amb	#A	CPU	Amb	CPU	Toks	CPU
1	7	1.14	2	10	1.00	10	7	10
2	26	1.38	1	253	1.00	127	20	88
3	18	1.22	1	89	1.00	80	13	55
4	19	1.63	3	72	1.10	62	16	48
5	34	1.50	3	475	1.02	367	26	345
6	22	1.47	2	356	1.00	250	17	205
7	17	1.50	1	458	1.00	158	15	154
8	24	1.58	2	258	1.04	120	19	102
9	24	1.33	1	291	1.04	206	18	154
10	32	1.53	2	830	1.06	314	24	290

Table 4.3 Evaluation of the integrated system.

Two experiments have been carried on, first by integrating the PoS tags into ALEP and then the chunk mark–ups. For the first experiment, the reduction of morpho–syntactic ambiguity (compare the third and the sixth columns) reduces the processing time of the overall process by 45.9% (35.9% on average per sentence). For the second experiment, the system processing time is reduced by 52.6% (an average of 42.7% per sentence). Here, parsing speed–up is due to the fact that by integrating chunk mark–ups, we do not only avoid generating irrelevant constituents not contributing to the final parse tree but we also provide part of the structure that the analysis component has to compute.

A detailed analysis of the results showed us that, while in processing simple sentences, as the ones we included in the first test suite, the most relevant factor for improving processing time was the reduction of the number of tokens of the sentences, in processing complex sentential constructions, e.g. sentences included embedded clauses, efficiency gains were mainly due to the reduction

[33]Tokens in ALEP are words or tagged special constructs.

of the morpho–syntactic ambiguity, since this drastically reduced the structural ambiguity. This is exemplified by case 10 (we show in (136)), where by resolving the morpho–syntactic ambiguity of que (that) we avoid the structural ambiguity of the three embedded clauses.

(136) *Las autoridades argentinas opinan [$_{\mathrm{CL}}$ que los inversores muestran una mayor confianza en el país [$_{\mathrm{CL}}$ en el que impera un drástico plan de ajuste económico [$_{\mathrm{CL}}$ que incluye la privatización de grandes empresas públicas]]].*

a)- Experiment B

The evaluation of the effect of default lexical entries on the ALEP grammar was done with free input text. Here we used an article from "El País" (September 2001) (cf. Appendix C.3).

Even though 67.7% of major words in the article were unknown to the system (i.e. were not encoded in the lexicon) —46% of the verbs, 78% of the nouns, 50% of the adjectives, 50% of the adverbs—, the system did not fail in producing a result because of lack of lexical coverage.

Table 4.4 shows the number of analyses we get for each sentences with our first version of default lexical entries (third column) and how overgeneration is reduced by adding framing information to the PoS tags of our external lexicon (fourth column). Sentences which do not receive an analysis were due to lack of coverage of the linguistic structure.

#	**unknown words**	**DLEs v1 A**	**DLEs v2 A**
1	28.5%	1	1
2	18.8%	2	2
3	21.4%	3	1
4	19.0%	16	1
5	13.5%	0	0
6	25.8%	6	1
7	21.0%	4	3
8	16.6%	24	6
9	31.8%	0	0
10	48.2%	16	4
11	17.6%	2	1

Table 4.4 Evaluation of the integrated system. Test suite 3.

Besides, our system provides structural robustness to the high–level processing. Structural robustness to deep linguistic processing is obtained by integrating into the ALEP system the structures delivered by the shallow processing tool such that they do not need to be re–built by phrase structure rules. This allows us to extend the coverage of the grammar to deal with low frequent constructions.[34] Dealing with such kind of constructions in our grammar, though feasible, would increase drastically the parsing search space and would create spurious ambiguity.[35]

The following are some examples of the type of structures the current version of the grammar can process:

(137) a. No dieron [$_{\text{NX}}$crédito alguno] a ...
((they) did not believe in ...)

b. Se incrementarán en [$_{\text{NX}}$los próximos ocho meses]...
(they will be increased in the following eight months)

(137.a) shows a nominal chunk where the indefinite *alguno* (some) is postponed. This is a type of so–called *término de polaridad negativa* (negative polarity term), referring to phrases which can only appear in negative context (cf. [Sánchez López 99]). Such a construction is not covered by the grammar which only covers indefinites preceding noun heads.

(137.b) shows a nominal chunk where the canonic 'cardinal + adjective' order is inverted. In such a construction, the 'cardinal + noun' forms an inseparable unity (cf. [Bosque 99]). Such a construction is not covered by the grammar which only covers cardinals preceding adjectives.

[34]Future work will include ungrammatical or uncomplete partial constructions.

[35][Steiner & Tsujii 99] present extended ID schemata in an HPSG framework to allow for parsing of normally extra grammatical constructs while preventing an explosion of search space.

Chapter 5

Conclusions

This thesis describes research into the development and deployment of engineered large–scale unification–based grammar to provide more robust and efficient deep grammatical analysis of linguistic expressions in real–world applications, while maintaining the accuracy of the grammar and keeping its precision up to a reasonable level.

This chapter summarizes the approach we have proposed and implemented and presents the main contributions arising from our research work.

1. NLP systems with monolithic grammars, where all dimensions of linguistic information (morphological, syntactic, semantic, etc.) are interleaved, have to deal with huge search space due to several sources of non–determinism (i.e. ambiguity). This is particularly true of broad–coverage unification–based grammars.

   In tacking the efficiency problem in deep processing, our approach is to prune the search space of the parser by integrating shallow and deep processing techniques.

   We propose and implement a NLP system which integrates a linguistic PoS tagger and chunker as a pre–processing module of a broad–coverage unification–based grammar of Spanish implemented in the ALEP platform. Our main goal in that is to distribute the analysis process such that we can release the parser from certain tasks that may be efficiently and reliably dealt with by these computationally less expensive processing techniques, and, therefore, it can perform significantly better.

On the one hand, by integrating the morpho–syntactic information delivered by the PoS tagger, we reduce the number of morpho–syntactic ambiguities of the linguistic expression to be analyzed. Furthermore, by resolving the morpho–syntactic ambiguities, we can reduce the structural ambiguity, thus a large number of unproductive edges are avoided.

On the other hand, by integrating chunk mark–ups delivered by the partial parser, we do not only avoid generating irrelevant constituents which are not to contribute to the final parse tree, but we also provide part of the structure that the analysis component has to compute, therefore, avoiding a duplication of efforts.

2. Deep linguistic processing produces a complete syntactic and semantic analysis of the sentences it processes, as needed for accurate NLP, however, it fails in producing a result when the linguistic structure being processed and/or words in the input sentences fall beyond the coverage of the grammatical resources.

   Besides improving of the overall analysis process, we have showed that the presented system provides both structural and lexical robustness to the high–level processing.

   Structural robustness is obtained by integrating into the ALEP grammar the structures which have already been parsed by the chunker such that they do not need to be re–built by phrase structure rules. The integration of chunk mark–ups allows us to extend the coverage of the grammar to deal with low frequent constructions. Dealing with such kind of constructions by the grammar performing deep analysis, though feasible, would increase drastically the parsing search space and would create spurious ambiguity.[1]

   To provide lexical robustness to the system, we have implemented default lexical entries, i.e. lexical entry templates that are activated when the system can not find a particular lexical entry to apply.

3. The integration of the tagger, which supplies the PoS information to the linguistic processing modules of our system, allows us to increase robustness

[1]Future work will include ungrammatical or uncomplete partial constructions.

while avoiding increase in morphological ambiguity, thus keeping the precision of the grammar —i.e. the percentage of input sentences that received no superfluous analysis— up to a reasonable level.

Furthermore, the results we have presented show how better precision is achieved by extending the PoS tags of our external lexicon so that they include syntactic information, for instance subcategorization information.[2]

4. Another main issue of this thesis is that, for using our system in real–world applications, the accuracy of the grammar —i.e. the percentage of input sentences that receive the correct analysis— should remain the same.

   In order to maintain the accuracy of our ALEP grammar, in the integrated architecture we propose, we propagate the ambiguities which can not be reliably solved by the shallow processing tool to be dealt with by the linguistic components of our ALEP grammar.

5. Finally, note that our approach is in line with the use we make of the parsing/refinement distinction facility provided by the ALEP platform to the effect of reducing the ambiguity seen by the parser. Furthermore, as we have shown, the described system does not need to alter the linguistic proceesing components of the grammar, nor to extend the machinery provided by the ALEP system.

[2]We foresee to add semantic information to PoS tags to further improve precision.

Appendix A

Ambiguity Study

The following shows a classification of the ambiguities found in the lexicon we have used for morpho–syntactic annotation.

1. **Ambiguities within the same category**

   Ambiguities within the same category are found in verbal and nominal lexical entries.

   - `verbal lexical entries`

     Apart from the `auxiliary/main` ambiguity in the verbs *ser* and *estar* (to be) and some of the forms of the verb *haber* (to have), verbal lexical items present:

     – `person ambiguity`, between first and third person singular in all subjunctive tenses and in imperfect indicative, e.g. *cante, cantara, cantase, cantare, cantaba* (cantar).

     – `tense ambiguity`, between present and past indicative in the first person plural, e.g. *abandonamos* (abandonar).

     – `mood ambiguity`, between third person singular present indicative and second person singular imperative, e.g. *ahoga* (ahogar).

`Tense and mood ambiguities` are still more frequent `between different verbs`, i.e. a given lexical item may correspond to two, or even three, different tenses or moods derived from different verbs. Thus, we find, for instance, ambiguities between:

- future and past indicative in the first person singular, e.g. *equiparé* (equiparar, equipar).
- present and imperfect subjunctive in singular persons and third person plural, e.g. *amase, amases, amasen* (amasar, amar).
- present indicative and present subjunctive, e.g. *creamos, creemos* (crear, creer).
- past indicative (second person singular) and present subjunctive (first or third person singular), e.g. *diste* (dar, distar).
- future indicative and present subjunctive in the first and the second person plural, e.g. *equipararemos, equipararéis* (equiparar, equipar).
- present indicative and imperfect subjunctive in the second and third person singular and the third person plural, e.g. *equiparas, equipara, equiparan* (equiparar, equipar).
- conditional and present indicative in the second and third singular person and the third person plural, e.g. *inventaría, inventarías, inventarían* (inventar, inventariar).
- second person singular imperative and first person singular past indicative, e.g. *di* (decir, dar), and third person plural, e.g. *ven* (venir, ver).
- past participles and present indicative, e.g. *visto* (ver, vestir), *rota* (romper, rotar) and *rotas* (romper, rotar).
- third person singular present indicative, second person singular past indicative and imperative, e.g. *viste* (vestir, ver, vestir).
- present indicative, past indicative and present subjunctive in the first person plural, as in *vengamos* (vengar, vengar, venir).

- `nominal lexical entries` present:

  - `gender ambiguity`, in nominal entries having different semantic readings and following the same inflection patterns, e.g. *radio* (el/la radio, los/las radios), and in fullforms, which, being derived from two different base forms, have the same morpho–syntactic description, e.g. *cazadora* (cazadora, cazador).
  - `number ambiguity`, e.g. *as, crisis.*
  - finally, we find a few cases of both `gender and number ambiguity`, e.g. *taxis, res.*

2. **Ambiguities within open classes**

- `noun/adjective`.
- `noun/verb/adjective`, especially between:

  - past participles, nouns and adjectives ending in '-a', '-as', '-o' and '-os' (*cubierta/s, cubierto/s*).
  - infinitives and singular adjectives and nouns ending in '-ar' (*titular*).
  - third person singular present indicative and feminine singular adjectives and nouns ending in '-a' (*asesina*).
  - second person singular present indicative and feminine plural adjectives and nouns ending in '-as' (*asesinas*).
  - first and third person singular present subjunctive and feminine and masculine singular adjectives and nouns ending in '-e' (*clave, presente*).
  - second person singular present subjunctive and feminine and masculine singular adjectives and nouns ending in '-es' (*claves, presentes*).

- second person singular future subjunctive and feminine and masculine plural adjectives and nouns ending in '-es' (*militares*).

- first person singular present indicative and masculine singular adjectives and nouns ending in '-o' (*asesino, piloto*).

- `verb/adjective` (the same ambiguities presented above).

- `verb/noun ambiguities` the same ambiguities presented above plus:

  - second person plural present indicative and masculine singular nouns ending in '-és' (*revés*), and masculine plural nouns ending in '-ís'.

  - imperfect indicative in first/third person singular and feminine singular nouns ending in '-a' (*garantía*), and second person singular feminine plural noun ending in '-as' (*garantías*).

  - past indicative in first person singular and masculine singular nouns ending in '-e' (*traje*), '-é' (*chalé*) and '-í', and third person singular and masculine singular nouns ending in '-o' (*vino*) and '-ó' (*dominó*).

  - first/third person singular future indicative and masculine singular nouns ending in '-é' (*pagaré*).

  - present subjunctive in first/third person singular and feminine and masculine singular nouns ending in '-a' (*cosa, tema*) and '-e' (*calle, accidente*), and second person singular and feminine and masculine plural nouns ending in '-as' (*metas, cometas*) and '-es' (*labores, accidentes*).

  - imperfect subjunctive in first/third person singular and feminine singular nouns ending in '-a' (*citara*), and second person singular and feminine plural nouns ending in '-as' (*citaras*).

  - future subjunctive in first/third person singular and masculine singular nouns ending in '-e', and second person singular and masculine plural nouns ending in '-es' (*cantares*).

- conditional in first/third person singular and feminine singular nouns ending in '-a' (*penitenciaría*), and second person singular and feminine plural noun ending in '-as' (*penitenciarías*).
- imperative in second person singular and masculine and feminine singular nouns (*sal, haz, sostén*), second person plural and masculine and feminine singular nouns (*libertad, abad*).

Noun/verb ambiguity is also present with different tenses of different verbs:

- past participles, first, second and third person singular present indicative and nouns ending in '-o', '-a' and '-as', e.g. *entrevistas, entrevista* (entrevistar, entreveer).
- past participles, first, second and third person singular present subjunctive and nouns ending in '-a' and '-as', e.g. *vista, vistas* (ver, vestir).
- present indicative, present subjunctive feminine nouns ending in '-a' and '-as', e.g. *prenda, prendas* (prender, prendar), and masculine plural nouns ending in '-es', e.g. *pares* (parar, parir).

3. **Ambiguities within closed classes**

- `conjunction/relative pronoun`: *que.*
- `conjunction/preposition`: *cuando.*
- `subordinating conjunction/adverb`: *mientras.*
- `coordinating conjunction/adverb`: *ya, luego.*
- `adverb/preposition`: *durante, incluso.*
- `conjunction/preposition/adverb`: *según.*
- `indefinite pronouns/adverb`: *algo.*

- `indefinite pronoun/indefinite adjective`: *demás.*
- `indefinite pronoun/indefinite adjective/indefinite determiner`: *cuanta/s, otro/a/os/as, mucha/as/os, poca/as/os.*
- `indefinite pronoun/indefinite adjective/indefinite determiner/adverb`: *mucho, poco.*
- `interrogative pronoun/interrogative determiner/adverb`: *cuánto.*
- `ind. determiner/ind. pronoun/adverb`: *bastante, demasiado.*
- `possessive determiner/possessive adjective`: *nuestra/as/o/os, vuestra/as/o/os.*

4. **Ambiguities between open and closed classes**

- `verb/adverb`, which, in turn can be subgrouped as follows:
  - first person singular present indicative: *abajo, adentro.*
  - first/third person singular present subjunctive: *adelante, enfrente, sacramente, fuera.*
  - first person singular past indicative: *así.*
  - second person singular present indicative: *apenas.*
  - third person singular present indicative and second person singular imperative: *afuera, aposta, arriba, encima.*
- `verb/preposition`: *cabe, entre, hace.*
- `verb/conjunction`: *ora.*
- `verb/numeral`: *segunda/as, cuarta/as.*
- `verb/possessive pronoun`: *mío.*
- `verb/indefinite determiner`: *tales.*

- `verb/indefinite pronoun`: *uno.*
- `noun/preposition`: *a, ante, contra, de, pro, so, vía.*
- `noun/adverb`: *alias, amén, mañana, no.*
- `noun/conjunction`: *o, u, sino.*
- `noun/numeral`: *cuartos, décimo/os, octava/as, quinta/as/o/os, segundos.*
- `noun/personal pronoun`: *te.*
- `noun/possessive pronoun`: *tuya/s.*
- `noun/possessive determiner`: *mi/s.*
- `noun/demonstrative determiner/demonstrative pronoun`: *este, ese.*
- `noun/clitic/article`: *la, las.*
- `adverb/adjective`: *asaz.*
- `adverb/noun/verb`: *alerta, aparte, cerca, tarde.*
- `adverb/adjective/noun`: *mal, pronto.*
- `adverb/noun/adjective/verb`: *medio.*
- `numeral/verb/noun`: *segundo, cuarto.*
- `preposition/noun/verb`: *sobre.*
- `preposition/verb/adjective`: *para.*
- `preposition/noun/adjective/verb`: *bajo.*
- `possessive pronoun/noun/verb`: *mía/s.*
- `conjunction/adverb/preposition/verb`: *como.*
- `adjective/reflexive pronoun/indefinite pronoun`: *misma/as/os.*

- adjective/reflexive pronoun/indefinite pronoun/adverb: *mismo.*
- adverb/conjunction/noun: *bien.*
- adverb/reflexive pronoun/noun: *sí.*
- adverb/indefinite pronoun/noun/verb: *nada.*
- adverb/adjective/verb: *harto, junto, presto, salvo, viva.*
- indefinite adjective/adverb/noun: *más, menos.*
- indefinite determiner/indefinite pronoun/verb: *una/as.*
- indefinite determiner/indefinite pronoun/noun: *todo/os.*
- indefinite determiner/indefinite pronoun/adjective: *varias/os.*
- indefinite determiner/indefinite adjective/participle: *determinado/a/os/as.*
- indefinite determiner/indefinite adjective/adjective: *cierta/as/o/os, rara/as/o/os, distintos/as, diversos/as, escasos/as.*
- indefinite determiner/indefinite adjective/indefinite pronoun/noun: *tanta/as/os, cuanto.*
- indefinite determiner/indefinite adjective/indefinite pronoun/noun/adverb: *cuanto.*
- indefinite determiner/indefinite adjective/indefinite pronoun/noun/conjunction/adverb: *tanto.*

Appendix B

Sample Outputs

The following shows an example of the output of the tokenizer, the tagger and chunker, and the ALEP grammar.

B.1 Sample Output of TagIt

```
REF      CLASS   ORTH                       LEMMA\MSD(|LEMMA\MSD)*
                 (PARAG                     <P>
                 (SENT                      <S SNB="1" CTYPE="UNK" PT="NO">
BOS.1    TOK     El                         el\Np---|el\Tdms-
1.2      TOK     gobierno                   gobernar\Vmip1s-|gobierno\Ncms-
2.3      TOK     argentino                  argentino\Afpms-|argentino\Ncms-
3.4      TOK     colocó                     colocar\Vmis3s-
4.5      TOK     títulos                    título\Ncmp-
5.6      TOK     públicos                   público\Afpmp-|público\Ncmp-
6.9      TOK     por valor de               por_valor_de\Sp
9.13     TOK     300 millones de dólares    300_millones_de_dólares\currency
13.14    TOK     en                         en\Sp
14.15    TOK     los                        el\Tdmp-|lo\Pp3mpa-
15.16    TOK     principales                principal\Afp.p-|principal\Ncmp-
16.17    TOK     mercados                   mercado\Ncmp-|mercar\Vmp--pm
17.18    TOK     europeos                   europeo\Afpmp-|europeo\Ncmp-
18.21    TOK     a través de                a_través_de\Sp
21.22    TOK     la                         el\Tdfs-|lo\Pp3fsa-
22.23    TOK     banca                      banca\Ncfs-
23.24    TOK     Morgan                     morgan\Np---
24.25    TOK     estadounidense             estadounidense\Afp.s-|
                                            estadounidense\Nc.s- PTR=,
```

25.26	TOK	anunció	anunciar\Vmis3s- PTL=,
26.27	TOK	hoy	hoy\Rg
27.28	TOK	el	el\Tdms-
28.29	TOK	presidente	presidente\Ncms-
29.30	TOK	de	de\Ncfs-|de\Sp
30.31	TOK	el	el\Tdms-
31.35	TOK	Banco Central de Argentina	banco_central_de_argentina\Np--- PTR=,
35.EOS	TOK	Roque Fernández	roque_fernández\Np--- PTL=,
		)SENT	</S>
		)PARAG	</P>

B.2 Sample Output of Latch

REF	CLASS	ORTH	LEMMA\MSD(|LEMMA\MSD)*
		(PARAG	<P>
		(SENT	<S SNB="2" CTYPE="UNK" PT="NO">
	CHUNKo	NX	
BOS.1	TOK	El	el\Tdms-
1.2	TOK	gobierno	gobierno\Ncms-
	CHUNKc	NX	
2.3	TOK	argentino	argentino\Afpms-
3.4	TOK	colocó	colocar\Vmis3s-
4.5	TOK	títulos	título\Ncmp-
5.6	TOK	públicos	público\Afpmp-
6.9	TOK	por valor de	por_valor_de\Sp
9.13	TOK	300 millones de dólares	300_millones_de_dólares\currency
13.14	TOK	en	en\Sp
	CHUNKo	NX	
14.15	TOK	los	el\Tdmp-
15.16	TOK	principales	principal\Afp.p-
16.17	TOK	mercados	mercado\Ncmp-
	CHUNKc	NX	
17.18	TOK	europeos	europeo\Afpmp-
18.21	TOK	a través de	a_través_de\Sp
	CHUNKo	NX	
21.22	TOK	la	el\Tdfs-
22.23	TOK	banca	banca\Ncfs-
	CHUNKc	NX	
23.24	TOK	Morgan	morgan\Np---
24.25	TOK	estadounidense	estadounidense\Afp.s- PTR=,

```
25.26    TOK      anunció                     anunciar\Vmis3s- PTL=,
26.27    TOK      hoy                         hoy\Rg
         CHUNKo   NX
27.28    TOK      el                          el\Tdms-
28.29    TOK      presidente                  presidente\Ncms-
         CHUNKc   NX
29.30    TOK      de                          de\Sp
         CHUNKo   NX
30.31    TOK      el                          el\Tdms-
31.35    TOK      Banco Central de Argentina  banco_central_de_argentina\Np--- PTR=,
         CHUNKc   NX
35.EOS   TOK      Roque Fernández             roque_fernández\Np--- PTL=,
                  )SENT                       </S>
                  )PARAG                      </P>
```

B.3 Sample Output of ALEP

`Input` : El gobierno argentino colocó títulos públicos por valor de 300 millones de dólares en los principales mercados europeos a través de la banca Morgan estadounidense, anunció hoy el presidente del Banco Central de Argentina, Roque Fernández.

`sfs` : [pred(colocar, colocar, gobierno, título, mercado), pred(anunciar, anunciar, presidente, colocar), rel(time, anunciar, hoy), rel(specification, presidente, type(pname(Banco_Central_de_Argentina))), rel(specification, presidente, type(pname(Roque_Fernández))), rel(quality, mercado, principal), rel(quality, banca, estadounidense), rel(specification, banca, type(pname(Morgan))), rel(quality, mercado, europeo), rel(quality, título, público), rel(quality, título, por_valor_de), rel(quality, gobierno, argentino), rel(manner, colocar, banca), rel(arg_place, colocar, mercado), rel(arg2, por_valor_de, type(currency('300_millones_de_dólares'))), rel(arg2, colocar, título), rel(arg2, anunciar, colocar), rel(arg1, estadounidense, banca), rel(arg1, principal, mercado), rel(arg1, europeo, mercado), rel(arg1, público, título), rel(arg1, argentino, gobierno), rel(arg1, colocar, gobierno), rel(arg1, anunciar, presidente), pred(público, público, título), pred(por_valor_de, por_valor_de, type(currency('300_millones_de_dólares'))), pred(europeo, europeo, mercado), pred(estadounidense, estadounidense, banca), pred(argentino, argentino, gobierno), index(event, colocar), index(event, anunciar)].

Appendix C

Test Suites

The following shows the test suites we have used to test and evaluate our system.

C.1 Test Suite 1

1. Unos socios negocian.
2. Los otros socios negocian.
3. Los otros dos socios negocian.
4. Elabora una propuesta.
5. Elaborar una propuesta.
6. Por favor, elabora una nueva propuesta.
7. Consultar la propuesta 97/4746.
8. Torres es el funcionario.
9. Tengo el puesto de auxiliar.
10. Acceder a la propuesta código 97/014746.
11. El presidente firma la venta.
12. Torres vendió una turbina a los empleados.

13. Torres coloca a los trabajadores en la empresa.
14. El presidente desea efectuar la venta.
15. Torres vió a el presidente comprar las acciones.
16. Los datos generales de la propuesta anterior.
17. Quiero todos los datos de la propuesta 97/014746.
18. El presidente invierte el capital en la empresa.
19. El propio presidente de la empresa firma las ventas.
20. El FIV es un organismo encargado por el gobierno de la privatización de las empresas públicas.
21. Con la participación en VIASA, la compañía aumentó su penetración en el mercado latinoamericano.
22. El pasado año, la compañía garantizó las invesiones.
23. La compañía garantizó las invesiones el pasado año.
24. Desde el año pasado, la compañía garantiza las invesiones.
25. Ir a elaborar una propuesta nueva con el tipo 0102.
26. La compañía opera básicamente en el interior de Venezuela.
27. Quiero elaborar una propuesta.
28. Quiero hacer una nueva propuesta.
29. Quiero identificarme con el cargo de auxiliar.
30. Las diferentes empresas venezolanas desean participar en la compra.
31. Quiero introducir otra propuesta de el mismo tipo.
32. Dame las propuestas pendientes de cumplimentar.
33. Dame las propuestas no tramitadas.

34. Mandar la propuesta 98/7878 a la impresora.

35. Sacar por la impresora la propuesta 98/777.

36. Voy a seguir cumplimentando la propuesta código 97/014746.

37. Deseo acabar de cumplimentar la propuesta con el codigo 97/014746.

38. Deseo preparar para su tramitación la propuesta código 97/014746.

39. Volver a el estado de cumplimentación la propuesta de código 97/014746.

40. Pasar la propuesta 98/11 a el siguiente nivel.

41. Deseo elevar la propuesta 98/11 a mi jefe.

42. Quiero dar mi aprobación a la propuesta 98/12.

43. Quiero solicitar un informe para la propuesta 98/98.

44. La compañía no puede acometer fuertes inversiones de modernización.

45. La compañía competirá contra American Airlines para aumentar sus posiciones.

46. Realización de una solicitud.

47. Lista de las propuestas sin informes.

48. Quiero hacer una copia de la propuesta de código 97/000214.

49. Lista de todas las propuestas pendientes de cumplimentar.

50. Dame una relación de los informes cumplimentados de segundo nivel.

51. Quiero una lista de los comentarios realizados sobre la propuesta 98/12.

52. Imprime los datos correspondientes a la propuesta 98/12.

53. Ir a elaborar una propuesta complementaria a la propuesta de código 1997/014746.

54. Deseo realizar una propuesta complementaria a la propuesta con el número 97/014746.

55. Ver los informes asociados a la propuesta 98/123.

56. Quiero obtener una relación de los informes asociados a la propuesta código 98/123.

57. Elaboramos una propuesta?

58. Podría ver sus datos generales?

59. Qué propuestas no se decidieron?

60. Qué mensajes están pendientes de leer?

61. Qué informes tiene la propuesta 97/000214?

62. Qué propuestas pendientes de informes tienen la fecha 18/03/97?

63. Y cuáles son sus datos generales?

64. Quién es el gestor de la propuesta 98/16111?

65. Cuáles son los informes de la propuesta 98/123?

66. Hay algún mensaje pendiente de leer?

67. Quiero saber qué propuestas no están cumplimentadas.

68. Quiero saber qué mensajes están pendientes de leer.

69. Quiero saber qué informes tiene la propuesta 97/000214.

70. Quiero saber qué propuestas pendientes de informes tienen la fecha 18/03/97.

71. Quiero saber cuáles son las propuestas que no se tramitaron.

72. Quiero saber si hay algún mensaje pendiente de leer.

73. Dime cuáles son las propuestas que están pendientes de informes.

74. Dame las propuestas que están por tramitar.

75. Dame las propuestas que quedan por tramitar.

76. Dame las propuestas que tienen los informes pendientes.

77. El empleado que emitió los bonos efectuó la venta.

78. El puesto que desempeño en la empresa es auxiliar.

79. Quiero crear una propuesta nueva que sea de el tipo 0202.

80. Ir a la pantalla donde se introducen las propuestas.

81. Quiero todas las propuestas que están pendientes de tramitación.

82. El gobierno privatizará la compañía que compró las empresas.

83. El gobierno privatizará las compañías que compró la empresa.

84. El gobierno privatizará la compañía en que participó la empresa.

85. Quiero crear una propuesta nueva que tenga el tipo 0102.

86. Quiero crear una propuesta nueva que sea de el tipo 0102.

87. Quiero crear una propuesta nueva cuyo tipo sea 0102.

88. Dame las propuestas cuya tramitación de crédito nunca se acabó.

89. Dame las propuestas que no tienen todos los informes.

90. Dame los datos generales que pertenecen a la propuesta 97/000214.

91. Quiero la historia que es de la propuesta número 97/000214.

92. Quiero ver la lista de las propuestas que están pendientes de cumplimentar.

93. Quiero hacer una copia de la propuesta que tiene el código 97/000214.

94. Quiero introducir una propuesta complementaria a la propuesta cuyo código es 97/000214.

95. Iberia, que compró su participación, es una aerolínea española.

96. El gobierno privatizará el capital que el presidente no quiero comprar.

97. El gobierno privatizará el capital que el presidente afirma que no quiero comprar.

98. Torres y el presidente negocian.
99. Torres y el principal empleado negocian.
100. Los empleados y el presidente negocian.
101. Los empleados y el presidente de la compañía negocian.
102. Dame el presupuesto y las actividades de la solicitud 98/778.
103. El pesidente negoció y las ventas aumentaron.
104. El pesidente negoció y aumentó las ventas.
105. Torres negoció la participacón de el presidente y efectuó las inversiones.
106. Mi puesto en la empresa es el de auxiliar y la unidad en la que trabajo "xxx".
107. Quiero hacer una nueva propuesta que sea de el tipo 0201 y tenga el título "xxx".
108. El presidente afirma que no efectuará la venta y que no emitirá bonos.
109. Los bonos fueron emitidos.
110. El presidente está comprando bonos.
111. El trabajador ha estado comprando bonos.
112. El presidente ha efectuado la venta.
113. La aerolínea estatal venezolana será privatizada.
114. Los bonos argentinos ya han sido emitidos.
115. El capital fue invertido en bolsa.
116. Los bonos han sido vendidos a Torres.
117. Los bonos serán emitidos por el inversor.
118. Los bonos han sido emitidos por la empresa.

119. Los bonos han estado siendo comprados por los trabajadores.

120. La inversión no ha sido garantizada.

121. La demanda de títulos en los mercados ha aumentado.

122. La compañía ha negociado y ha garantizado las inversiones.

123. La venta ha sido firmada y las empresas han sido privatizadas.

124. El presidente afirma que la compañía será vendida y que las acciones serán privatizadas.

125. Quiero que me hagan un informe de la propuesta 98/98.

126. Dime cuál ha sido la historia de 98/777.

127. Argentina ha recuperado la opción de acceder a el crédito.

128. Los mensajes que he enviado.

129. Los mensajes que han sido enviados.

130. Qué mensajes no he leido?

131. Qué mensajes todavía no he leido?

132. Puedo ver los mensajes que he enviado?

133. Estados que ha tenido la propuesta 98/123.

134. Quiero ver todos los mensajes que no he enviado.

135. Ver las propuestas con el tipo 0201 que no han sido aprobadas.

136. Dame las propuestas que hayan sido realizadas por la unidad A.

137. El empleado es muy drástico.

138. El ajuste está muy próximo.

139. Las vinculaciones de Iberia con Aeropostal son muy estrechas.

140. Dame un informe mucho más completo.

141. Deseo un informe más completo de la propuesta 98/98.

142. Las compañías tienen ideas muy distintas.

143. El presidente está muy orgulloso de la empresa.

144. La empresa es grande y muy fuerte.

145. La empresa es muy grande y muy fuerte.

146. El presidente siempre negocia muy bien.

147. El presidente de Iberia ha negociado muy bien.

148. Muy bien ha negociado el presidente de Iberia.

149. El presidente de Iberia ha negociado la venta de las acciones muy bien.

150. El presidente de Iberia ha negociado muy bien la venta de las acciones.

C.2 Test Suite 2

<s n=1>Argentina colocó bonos por 300 millones de dólares en Europa.</s> <s n=2>"El gobierno argentino colocó títulos públicos por valor de 300 millones de dólares en los principales mercados europeos a través de la banca Morgan estadounidense", anunció hoy el presidente del Banco Central de Argentina, Roque Fernández.</s> <s n=3>Los bonos argentinos tienen un plazo de dos años y un interés anual del 11,2 por ciento, añadió Fernández.</s> <s n=4>Esto es el doble de la rentabilidad que se obtiene en este tipo de operaciones, en los mercados internacionales.</s> <s n=5>El funcionario indicó que los títulos argentinos comenzaron a negociarse el martes en los mercados londinenses, donde se cotizaron al 100,75 por ciento de su valor nominal, debido a la fuerte demanda de los inversores.</s> <s n=6>La colocación de los títulos está garantizada con fondos del Morgan Guaranty Trust estadounidense hasta que las autoridades argentinas emitan formalmente los bonos, el 7 de octubre próximo.</s> <s n=7>"Se prefiere emitir sólo por 300 millones de dólares porque siempre conviene dejar una parte de la demanda insatisfecha", afirmó Fernández.</s> <s n=8>El gobierno argentino emitirá títulos por otros 200 millones de dólares si se confirma que el interés de los inversores supera el monto de la colocación inicial.</s> <s n=9>"Argentina ha

retornado al mercado internacional de capitales y recuperó la opción de acceder al crédito", destacó el presidente del Banco Central.`</s>` `<s n=10>`Las autoridades argentinas opinan que los inversores muestran una mayor confianza en el país en el que impera un drástico plan de ajuste económico que incluye la privatización de grandes empresas públicas.`</s>`

Case : 1.
Input : Argentina colocó bonos por 300 millones de dólares en Europa.
Comment : This sentence has a tri-valent verb (colocó) taking a DO (bonos) and a bound PP (en Europa). Special construction: currency 300 millones de dólares. Overgeneration due to PP attachment (por 300 millones de dólares).

Case : 2. Input : "El gobierno argentino colocó títulos públicos por valor de 300 millones de dólares en los principales mercados europeos a través de la banca Morgan estadounidense", anunció hoy el presidente del Banco Central de Argentina, Roque Fernández.
Comment : Extraposed sentential object and inverted subject. Temporal modifier (hoy) between the main verb (anunció) and the inverted subject (el presidente...). The inverted subject is a complex NP. The head of the NP (presidente) is modified by a PP (del Banco Central de Argentina) and an appositional proper noun (Roque Fernández). The verb in the completive clause (colocó) takes a DO and a bound PP. The head of the DO (títulos) is modified by an adjective (públicos) and a PP which shows an example of special construction, the currency 300 millones de dólares. The bound PP has a complex NP whose head (mercados) is modified by two adjective (principales) and (europeos). The verb in the completive clause, in addition, is modified by a PP introduced by a complex preposition (a través de). The preposition takes a NP whose head (banca) is modified by a proper name (Morgan) and an adjective (estadounidense).

Case : 3.
Input : Los bonos argentinos tienen un plazo de dos años y un interés anual del 11, 2 por ciento, añadió Fernández.
Comment : Extraposed sentential object and inverted subject. The verb in the completive clause (tienen) takes a coordinated NP complement (un plazo ... y un interés). Each coordinated noun is modified by a PP (de dos años, del 11,2 por ciento). Percentage (11,2 por ciento).

Case : 4.
Input : Esto es el doble de la rentabilidad que se obtiene en este tipo de operaciones, en los mercados internacionales. Comment : This sentence has a copula verb (ser) which takes a nominal complement. The head of the nominal complement is a measure noun (doble) which takes a PP complement (de la rentabilidad) and it is modified by a relative clause (que se obtiene...). The verb in the relative clause is a reflexive passive (se obtiene) and it is modified by two locative PPs (en este tipo..., en los mercados...). The first PP has a NP whose head (tipo) is modified by a PP (de operaciones). The second PP has a NP headed by a noun (mercados) modified by an adjective (internacionales). Overgeneration due to PP attachment and RC attachment.

Case : 5.
Input : El funcionario indicó que los títulos argentinos comenzaron a negociarse el martes en los mercados londinenses, donde se cotizaron a 100,75 por ciento de su valor nominal, debido a la fuerte demanda de los inversores.
Comment : This sentence has a verb which takes a completive clause (indicó que). The completive clause has a periphrastic control verb (comenzaron a). The controlled verb is a reflexive passive (negociarse) and it is modified by a temporal NP (el martes) and a locative PP (en los mercados londinenses). The locative PP includes a complex NP. The head of the NP is modified by an adjective (londinenses) and a relative clause where the relative adverb (donde) is the complement of the subordinated verb (cotizaron). In the relative clause we find a reflexive passive (se cotizaron) and two modifiers: the PP introduced by the contracted preposition (al), and the PP introduced by the complex preposition (debido a) which takes an argumental noun (demanda). Finally, this sentence shows a sample of an special construction, the percentage 100,75 por ciento.

Case : 6.
Input : La colocación está garantizada con fondos del Morgan Guaranty Trust estadounidense hasta que las autoridades argentinas emitan formalmente los bonos, el 7 de octubre próximo.
Comment : This sentence has a passive with estar. The participle (garantizada) is modified by a PP (con fondos ...) and a temporal sentential adjunct (hasta que...). The PP has a complex NP. The head of the NP (fondos) is modified by a PP which has a NP headed by (multi-word) proper noun (Morgan Guaranty

Trust) which is modified by an adjective (estadounidense). The temporal sentential adjunct has an adverb (formalmente) placed between verb and complement. The verb in the sentential adjunct is modified by a date, el 7 de octubre próximo.

Case : 7.
Input : "Se prefiere emitir sólo por 300 millones de dólares porque siempre conviene dejar una parte de la demanda insatisfecha", afirmó Fernández.
Comment : Extraposed sentential object and inverted subject. The completive clause has a control verb (preferir). The controlled verb (emitir) is post-modified by an adverb (sólo), a por-PP and a sentential adjunct (porque ...). The verb in the sentential adjunct (conviene) is modified by an adverb (siempre) and it takes an infinitival subject (dejar). The infinitival subject is a tri-valent verb taking a nominal complement and an adjectival complement (insatisfecha). The nominal complement is headed by a partitive noun (parte) which takes a PP complement (de la demanda).

Case : 8.
Input : El gobierno argentino emitirá títulos por otros 200 millones de dólares si se confirma que el interés de los inversores supera el monto de la colocación inicial.
Comment : Bi-valent verb (emitirá) modified by a PP (por otros 200 millones de dólares) and a conditional adjunct (si ...). The conditional adjunct has an impersonal sentence (se confirma). The verb in the impersonal sentence takes a completive clause. Overgeneration due to PP attachment.

Case : 9.
Input : "Argentina ha retornado al mercado internacional de capitales y recuperó la opción de acceder al crédito", destacó el presidente del Banco Central.
Comment : Extraposed sentential object and inverted nominal subject. The head of the subject (presidente) is modified by a PP (del Banco Central de Argentina). The sentential object has two coordinated VPs. The first VP has a di-valent verb in perfect tense (ha retornado) which takes a bound PP complement. The PP has a noun (mercado) which is by modified an adjective (internacional) and a PP (de capitales). The second VP has a transitive verb (recuperó) with complex nominal complement object. The head of the nominal complement (opción) takes a bound infinitival complement. The infinitive (acceder) takes a bound PP complement (al crédito).

Case : 10.
Input : Las autoridades argentinas opinan que los inversores muestran una mayor confianza en el país en el que impera un drástico plan de ajuste económico que incluye la privatización de grandes empresas públicas.
Comment : Di-valent verb taking a completive clause (opinan que). The completive clause has a transitive verb (muestran) with complex nominal complement. The head of the nominal complement is an argumental noun (confianza) modified by an adjective (mayor). The complement of the NP (en el país) has a noun modified by an object relative clause (en el que ...). In the relative clause there is an inverted complex subject: a noun (plan) pre-modified by an adjective (drástico), a PP (de ajuste económico), and a subject relative clause (que incluye...). The verb in the subject relative clause takes a complex NP complement: an argumental noun (privatización de grandes empresas públicas). Overgeneration due to interaction of relative and modifier attachment.

C.3 Test Suite 3

<s n=1>El FBI identifica a los pilotos suicidas.</s> <s n=2> Los principales sospechosos pertenecían presuntamente a la organización que lidera Bin Laden. </s> <s n=3> La policía de Hamburgo registra una vivienda en la que residieron dos presuntos terroristas.</s> <s n=4>El FBI ya conoce la identidad de los terroristas que sumieron en el caos a Nueva York y Washington el pasado martes. </s> <s n=5>Los nombres no han sido pronunciados por los portavoces de la agencia federal, pero extraoficialmente se ha comenzado a hablar de que los suicidas eran Adnan Bukhari, Ahmad Ibrahim Ali al-Hazoumi, Mohamed Atta y Wael Mohammad al-Shehrid, todos relacionados con la organización de Bin Laden, Al Qaeda. </s> <s n=6>Los presuntos terroristas, que habían aprendido a pilotar aviones en la escuela de aviación Huffman de Venice, dejaron sus coches en los aeropuertos de Boston y Daytona Beach (al norte de Miami). </s> <s n=7> Hasta el momento, el FBI ha detenido para su posterior interrogatorio a seis sospechosos interceptados en Boston y Miami. </s> <s n=8> Además, 4.000 agentes y 400 miembros de la oficina de lucha contra crímenes se encuentran desplegados en los lugares donde se cometieron los atentados. </s> <s n=9> El fiscal general, John Ashcroft, ha revelado que algunos de los secuestradores siguieron cursos de pilotaje en el propio territorio estadounidense y que cada aparato llevaba entre tres

y seis suicidas que amenazaron (al pasaje) con cuchillos, objetos cortantes, así como con amenazas de bombas. </s> <s n=10>Además, los agentes están investigando cintas de vídeos, llamadas teléfonicas de los pasajeros y cintas registradas en los aparcamientos de los aeropuertos desde donde los territoristas tomaron los aviones. </s> <s n=11>Entre los objetivos de los terroristas se encontraban también la Casa Blanca y el avión presidencial de George W. Bush.</s>

Bibliography

[Abney 90] S. Abney. Syntactic affixation and performance structures. In D. Bouchard and K. Leffel, editors, *Views on Phrase Structure.* Kluwer Academic Publishers, Dordrecht, 1990.

[Abney 91] S. Abney. Parsing by chunks. In R. Berwick, S. Abney, and C. Tenny, editors, *Principle–Based Parsing.* Kluwer Academic Publishers, Dordrecht, 1991.

[Abney 92] S. Abney. Prosodic structure, performance structure and phrase structure. In *Proceedings of the Workshop on Speech and Natural Language*, Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[Abney 95] S. Abney. Chunks and dependencies: bringing processing evidence to bear on syntax. In J. Cole, G.M. Green, and J.L. Morgan, editors, *Computational Linguistics and the Foundations of Linguistic Theory*, pages 145–164. CSLI, 1995.

[Abney 96a] S. Abney. Chunk Stylebook (work in progress).
`http://www.sfs.nphil.uni-tuebingen.de/ abney/Papers.html#96i`, 1996.

[Abney 96b] S. Abney. Part–of–speech tagging and partial parsing. In K.W. Church, S. Young, and G. Bloothooft, editors, *Corpus–based Methods in Language and Speech.* An ELSNET book. Kluwer Academic Publisher, Dordrecht, 1996.

[Abney 96c] S. Abney. Partial parsing via finite–state cascades. In *Proceedings of the Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information (ESSLLI'96)*, Prague, Czech Republic, 1996.

[Aduriz *et al.* 95] I. Aduriz, I. Alegria, J.M. Arriola, X. Artola, A. Díaz de Ilarraza, N. Ezeiza, K. Gojenola, and M. Maritxalar. Different issues in the desing of a lemmatizer/tagger for Basque. In *Proceedings of the SIGDAT Workshop, 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*, Dublin, Ireland, 1995.

[Aho *et al.* 74] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison–Wesley, Reading, MA, 1974.

[Alshawi 91] H. Alshawi, editor. *The Core Language Engine.* MIT Press, Cambridge, MA, 1991.

[Alshawi *et al.* 91] H. Alshawi, D. Arnold, R. Backofen, D. Carter, J. Lindop, K. Netter, S. Pulman, J. Tsujii, and H. Uszkoreit. *Eurotra ET6/1: Rule Formalism and Virtual Machine Design Study (final report).* Commission of the European Communities, Luxembourg, 1991.

[Armstrong *et al.* 95] S. Armstrong, G. Russell, D. Petitpierre, and G. Robert. An open architecture for multilingual text processing. In *Proceedings of the SIGDAT Workshop, 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*, pages 30–34, Dublin, Ireland, 1995.

[Backofen *et al.* 96] R. Backofen, T. Becker, J. Calder, J. Capstick, L. Dini, J. Dörre, G. Erbach, D. Estival, S. Manandhar, A.-M. Mineur, G. van Noord, S. Oepen, H. Uszkoreit, and A. Zaenen. *The EAGLES Formalisms Working Group.* Technical Report, Final Report, 1996.

[Badia & Colominas 98] T. Badia and C. Colominas. Predicate–argument structure. In F. van Eynde and P. Schmidt, editors, *Studies in Machine Translation and Natural Language Processing, volume 10: Linguistic Specifications for Typed Feature Formalism*, pages 137–208. European Commission, Luxembourg, 1998.

[Badia & Egea 00] T. Badia and T. Egea. A strategy for the syntactic parsing of corpora: from constraint grammar output to unification–based processing. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC–2000)*, pages 575–582, Athens, Greece, 2000.

[Badia *et al.* 95] T. Badia, A. Bredenkamp, T. Declerck, R. Hentze, M. Marimon, M. Melero, P. Schmidt, and A. Theofilidis. *ALEP Rule Coding Manual.* LS–GRAM Deliverable D-WP7, Luxembourg, 1995.

[Badia *et al.* 96] T. Badia, N. Bel, M. Carulla, M. Marimon, M. Melero, and A. Osterhoof. *Final Documentation of the LS–GRAM Spanish Lingware.* LS–GRAM Deliverable E-D8-ES, *gilc*UB/UPF, 1996.

[Bahl & Mercer 76] L.R. Bahl and R. Mercer. Part–of–speech assignment by a statistical decision algorithm. In *IEEE International Symposium on Information Theory*, pages 88–89, Ronneby, Sweden, 1976.

[Baum 72] L.E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process. *Inequalities*, 3:1–8, 1972.

[Bel *et al.* 96a] N. Bel, N. Calzolari, and M. Monachini (coords.). *Common Specifications and Notation for Lexicon Encoding.* MULTEXT D–1.6–B Deliverable, ILC, Pisa, 1996.

[Bel *et al.* 96b] N. Bel, M. Marimon, and J. Porta. Etiquetado morfosintáctico de corpus en el proyecto MULTEXT. In *Sociedad española de lingüística, XXVI simposio sobre morfología y lenguaje científico y técnico*, pages 32–35, Madrid, 1996.

[Bennet & Schmidt 98] P. Bennet and P. Schmidt. Phrase structure. In F. van Eynde and P. Schmidt, editors, *Studies in Machine Translation and Natural Language Processing, volume 10: Linguistic Specifications for Typed Feature Formalism*, pages 70–136. European Commission, Luxembourg, 1998.

[Bolc *et al.* 96] L. Bolc, K. Czuba, A. Kupść, M. Marciniak, A. Mykowiecka, and A. Przepiórkowski. *A Survey of Systems for Implementing HPSG Grammars.* Technical Report 814, Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland, 1996.

[Bosque & Demonte 99] I. Bosque and V. Demonte, editors. *Gramática descriptiva de la lengua española.* Real Academia Española, Colección Nebrija y Bello. Espasa Calpe, Madrid, 1999.

[Bosque 89] I. Bosque. *Las Categorías Gramaticales.* Lingüística. Editorial Síntesis, Madrid, 1989.

[Bosque 99] I. Bosque. El sintagma adjectival. Modificadores y complementos del adjetivo. Adjetivo y participio. In I. Bosque and V. Demonte, editors, *Gramática descriptiva de la lengua española*, Real Academia Española, Colección Nebrija y Bello. Espasa Calpe, Madrid, 1999.

[Bouma & Noord 93] G. Bouma and G. van Noord. Head–driven parsing for lexicalist grammars: experimental results. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL'93)*, pages 101–105, Utrecht, Netherlands, 1993.

[Bouma & Noord 94] G. Bouma and G. van Noord. Constraint–based categorial grammar. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL'94)*, Las Cruces, NM, 1994.

[Bouma 97] G. Bouma. *Valence Alternations without Lexical Rules.* Unpublished Manuscript, Rijksuniversiteit Groningen, 1997.

[Bourigault 92] D. Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING–92)*, pages 977–981, Nantes, France, 1992.

[Brants & Samuelsson 95] T. Brants and C. Samuelsson. Tagging the TELEMAN corpus. In *Proceedings of the 10th Nordic Conference of Computational Linguistics*, 1995.

[Bredenkamp & Hentze 95] A. Bredenkamp and R. Hentze. Some aspects of HPSG implementation in the ALEP formalism. In *Working Papers in Language Processing 46*, The CL/MT Group, University of Essex, UK, 1995.

[Bredenkamp *et al.* 96] A. Bredenkamp, F. Fouvry, T. Declerck, and B. Music. Efficient integrated tagging of word constructs. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING–96)*, pages 1028–1031, Copenhagen, Denmark, 1996.

[Bredenkamp *et al.* 97] A. Bredenkamp, T. Declerck, F. Fouvry, B. Music, and A. Theofilidis. Linguistic engineering using ALEP. In *Proceedings of the Euroconference Recent Advances in Natural Language Procecessing (RANLP'97)*, Tzigov Chark, Bulgaria, 1997.

[Bredenkamp *et al.* 98] A. Bredenkamp, T. Declerck, P. Groenendijk, M. Phelan, S. Rieder, P. Schmidt, H. Schulz, and A. Theofilidis. Natural language access to software applications. In *Proceedings of the Joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING–ACL'98)*, pages 1193–1197, Montreal, Canada, 1998.

[Bresnan 82] J. Bresnan, editor. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, MA, 1982.

[Brill 92] E. Brill. A simple rule–based part–of–speech tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP'92)*, pages 152–155, Trento, Italy, 1992.

[Brill 95] E. Brill. Unsupervised learning of disambiguation rules for part–of–speech tagging. In *Proceedings of the 3rd Workshop on Very Large Corpora (ACL'95)*, pages 1–13. Cambridge, MA, 1995.

[Briscoe & Copestake 96] T. Briscoe and A. Copestake. Controlling the application of lexical rules. In *Proceedings of the SIGLEX Workshop on Breadth and Depth of Semantic Lexicons*, Santa Cruz, CA, 1996.

[Briscoe *et al.* 93] T. Briscoe, V. de Paiva, and A. Copestake, editors. *Inheritance, Defaults and the Lexicon.* Studies in Natural Language Processing, Cambridge University Press, 1993.

[Brown *et al.* 88] P. Brown, J. Cocke, S. DellaPietra, V. DellaPietra, F. Jelinek, R.L. Mercer, and P. Roossin. A statistical approach to language translation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING–88)*, pages 71–76, Budapest, Hungary, 1988.

[Calzolari & Bindi 90] N. Calzolari and R. Bindi. Acquisition of lexical information from a large textual Italian corpus. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING–90)*, volume 3, pages 54–59, Helsinki, Finland, 1990.

[Carl & Schmidt-Wigger 98] M. Carl and A. Schmidt-Wigger. Shallow postmorphological processing with KURD. In *Proceedings of NeMLa/CoNLL98*, Sydney, 1998.

[Carpenter] B. Carpenter. Bob Carpenter's Chapter on Complexity. `http://www.colloquial.com/carp/Publications/complexity.ps`.

[Carpenter & Penn 94] B. Carpenter and G. Penn. *ALE The Attribute Logic Engine User's Guide (Version 2.0).* Carnegie–Mellon University, Pittsburgh, PA, 1994.

[Carpenter 92] B. Carpenter. *The Logic of Typed Feature Structures.* Cambridge University Press, Cambridge, UK, 1992.

[Carroll 93] J. Carroll. *Practical Unification–based Parsing of Natural Language.* PhD thesis, Computer Laboratory, University of Cambridge, Cambridge, UK, 1993.

[Carroll *et al.* 91] J. Carroll, T. Briscoe, and C. Grover. *A Development Environment for Large Natural Language Grammars.* Technical Report 233, Computer Laboratory, University of Cambridge, UK, 1991.

[Chang & Chen 93] C.-H. Chang and C.-D. Chen. HMM–based part–of–speech tagging for Chinese corpora. In *Proceedings of the Workshop on Very Large Corpora*, pages 107–120. Columbus, Ohio, 1993.

[Chanod & Tapanainen 95] J.P. Chanod and P. Tapanainen. Tagging French – comparing statistical and constraint–based methods. In *Proceedings of the 7th Con-*

ference of the European Chapter of the Association for Computational Linguistics (EACL'95), pages 149–156, Dublin, Ireland, 1995.

[Charniak 97] E. Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 1997.

[Charniak *et al.* 94] E. Charniak, G. Carroll, J. Adcock, A. Cassandra, Y. Gotoh, J. Katz, M. Littman, and J. McCann. *Taggers for Parsers.* Technical Report CS–94–06, Brown University, Providence, RI, 1994.

[Chen & Goodman 96] S.F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, 1996.

[Church & Gale 91] K.W. Church and W.A. Gale. Concordances for parallel text. In *Proceedings of the 7th Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research*, 1991.

[Church & Hanks 90] K.W. Church and P. Hanks. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29, March, 1990.

[Church & Mercer 93] K.W. Church and R.L. Mercer. Introduction to special issue on computational linguistics using large corpora. *Computational Linguistics*, 19(1):1–24, March, 1993.

[Church 88] K.W. Church. A stochastic parts program and a noun phrase parser for unrestricted text. In *Proceedings of the 2nd Conference on Applied Natural Language Processing, (ANLP'88)*, pages 136–143, Austin, Texas, 1988.

[Church 92] K.W. Church. *Current Practice in Part of Speech Tagging and Suggestions for the Future.* In Simmons (editors), *Sbornik Praci: In Honor of Henry Kucera*, pages 13–48, Michigan Slavic Studies, Michigan, 1992.

[Church *et al.* 96] K.W. Church, S. Young, and G. Bloothooft, editors. *Corpus–based Methods in Language and Speech.* An ELSNET book. Kluwer Academic Publisher, Dordrecht, 1996.

[Ciravegna & Lavelli 97] F. Ciravegna and A. Lavelli. Controlling bottom–up chart parsers though text chunking. In *Proceedings of the 5th International Workshop on Parsing Technologies (IWPT'97)*, Boston, MA, 1997.

[Ciravegna & Lavelli 99] F. Ciravegna and A. Lavelli. Full text parsing using cascades of rules: an information extraction perspective. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, Bergen, Norway, 1999.

[Ciravegna & Lavelli 00] F. Ciravegna and A. Lavelli. Grammar organization for cascade–based parsing in information extraction. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT'2000)*, Trento, Italy, 2000.

[Copestake & Flickinger 00] A. Copestake and D. Flickinger. An open source grammar development environment and broad–coverage English grammar using HPSG. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC–2000)*, pages 591–600, Athens, Greece, 2000.

[Copestake 99] A. Copestake. *The (new) LKB system.* CSLI, Stanford University, CA, 1999. `http://www.csli.stanford.edu/aac/doc5-2-pdf`.

[Croft *et al.* 91] W. Croft, H. Turtle, and D. Lewis. The use of phrases and structured queries in information retrieval. In A. Bookstein, Y. Chiaramella, G. Salton, and V. Raghavan, editors, *SIGIR Forum*, pages 22–45, 1991.

[Cruickshank 94] G. Cruickshank. *Two Level Morphology User Guide.* Cray Systems, Luxembourg, 1994.

[Cruickshank 95] G. Cruickshank. *Text Handling User Guide.* Cray Systems, Luxembourg, 1995.

[Crysmann *et al.* 02] B. Crysmann, A. Frank, B. Kiefer, H.-U. Krieger, S. Müller, G. Neumann, J. Piskorski, U. Schäfer, M. Siegel, H. Uszkoreit, and F. Xu. An integrated architecture for shallow deep processing. In *Proceedings of Association for Computational Linguistics 40th Anniversary Meeting (ACL'2002)*, University of Pennsylvania, Philadelphia, PA, 2002.

[Cutting 94] D. Cutting. Porting a stochastic part–of–speech tagger to Swedish. In R. Eklund, editor, *Proc. 9:e Nordiska Datalingvistikdagarna*, pages 65–70. Department of Linguistics, Computational Linguistics, Stockholm University, Stockholm, Sweden, 1994.

[Cutting *et al.* 92] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part–of–speech tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP'92)*, pages 133–140, Trento, Italy, 1992.

[Daelemans *et al.* 96] W. Daelemans, J. Zavrel, and P. Berck. Part–of–speech tagging for Dutch with MBT, a memory–based tagger generator. In *Congresboek van de Interdisciplinaire Onderzoeksconferentie Informatiewetenchap*, TU Delft, 1996.

[Declerck & Heyd 97] T. Declerck and S. Heyd. Modifying the internal representation of the TH–LS component of ALEP? In *Proceedings of the 3rd ALEP User Group Workshop*, pages 14–18, Saarbrücken, Germany, 1997.

[Declerck & Maas 97] T. Declerck and H.D. Maas. The integration of a part–of–speech tagger into the ALEP platform. In *Proceedings of the 3rd ALEP User Group Workshop*, pages 19–26, Saarbrücken, Germany, 1997.

[Dermatas & Kokkinakis 95] E. Dermatas and G. Kokkinakis. Automatic stochastic tagging of natural language texts. *Computational Linguistics*, 21(2):137–164, 1995.

[DeRose 88] S.J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, December, 1988.

[Derouault & Merialdo 84] A.-M. Derouault and B. Merialdo. Language modeling at the syntactic level. In *Proceedings of the 7th International Conference on Pattern Recognition*, pages 1373–1375, Montreal, Canada, 1984.

[Dörre *et al.* 94] J. Dörre, M. Dorna, and J. Junger. *The CUF User's Manual.* Institute für Maschinelle Sprachverarbeitung, Universität Stuttgart, Germany, 1994.

[Earley 70] J. Earley. An efficient context–free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[Elworthy 94] D. Elworthy. Does Waum–Welch re–estimation help taggers? In *Proceedings of the 4th Conference on Applied Natural Language Processing (ANLP'94)*, pages 53–58, Stuttgart, Germany, 1994.

[Emele 91] M. Emele. Unification with lazy non–redundant copying. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL'91)*, pages 323–330, Berkeley, CA, 1991.

[Emele 93] M. Emele. *TFS: The Typed Feature Structure Representation Formalism.* Institute für Maschinelle Sprachverarbeitung, Universität Stuttgart, Germany, 1993.

[Engelkamp *et al.* 92] J. Engelkamp, G. Erbach, and H. Uszkoreit. Hanling linear precedence constraints by unification. In *Proceedings of the 30th Annual Meeting* *of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware, 1992.

[Erbach & Manandhar 95] G. Erbach and S Manandhar. Visions for the future of logic–based natural language processing. In *Proceedings of the ILPS '95 Workshop Visions for the Future of Logic Programming*, 1995.

[Erbach & Uszkoreit 90] G. Erbach and H. Uszkoreit. Grammar Engineering: Problems and Prospects. In *Report on the Saarbrücken Grammar Enginnering Workshop. Published as CLAUS Report number 1*, Saarland University, 1990.

[Erbach 94] G. Erbach. Multi–dimensional inheritance. In H. Trost, editor, *Proceedings of KONVENS'94*, pages 102–111, Vienna, 1994.

[Erbach 95] G. Erbach. ProFIT: Prolog with Features, Inheritance and Templates. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*, Dublin, Ireland, 1995.

[Erbach *et al.* 94a] G. Erbach, M.A. Moshier, and H. Uszkoreit. *Multiple Inheritance for ALEP.* Project Deliverable, 1994.

[Erbach *et al.* 94b] G. Erbach, W. Skut, and H. Uszkoreit. *Linear Precedence Constraints in Lean Formalisms.* ET 9.2 Deliverable, 1994.

[Erbach *et al.* 95] G. Erbach, M. van der Kraan, S. Manandhar, H. Ruessink, W. Skut, and C. Thiersch. Extending unification formalisms. In *Proceedings of the 2nd Language Engineering Convention*, London, UK, 1995.

[EUROTRA 91] EUROTRA. *Spanish Final Implementation Report.* Commission of the European Communities, 1991.

[Eynde & Schmidt 98] F. van Eynde and P. Schmidt, editors. *Studies in Machine Translation and Natural Language Processing, volume 10: Linguistic Specifications for Typed Feature Formalism.* European Commission, Luxembourg, 1998.

[Feldweg 95] H. Feldweg. Implementation and evaluation of a German HMM for POS disambiguation. In *Proceedings of the SIGDAT Workshop, 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*, pages 41–46, Dublin, Ireland, 1995.

[Fernández-Ordóñez 99] I. Fernández-Ordóñez. Leísmo, laísmo y loísmo. In I. Bosque and V. Demonte, editors, *Gramática descriptiva de la lengua española*, Real Academia Española, Colección Nebrija y Bello. Espasa Calpe, Madrid, 1999.

[Flickinger 87] D. Flickinger. *Lexical Rules in the Hierarchical Lexicon.* PhD thesis, Stanford University, CA, 1987.

[Flickinger 00] D. Flickinger. On building a more efficient grammar by exploiting types. In D. Flickinger, S. Oepen, J. Tsujii, and H. Uszkoreit, editors, *Natural Language Engineering (6)1 —Special Issue: Efficiency Processing with HPSG: Methods, Systems, Evaluation*, pages 15–28. Cambridge University Press, 2000.

[Flickinger *et al.* 00] D. Flickinger, S. Oepen, J. Tsujii, and H. Uszkoreit, editors. *Natural Language Engineering (6)1 —Special Issue: Efficiency Processing with HPSG: Methods, Systems, Evaluation.* Cambridge University Press, 2000.

[Fligelstone *et al.* 97] S. Fligelstone, M. Pacey, and P. Rayson. How to generalize the task of annotation. In R. Garside, G. Leech, and A. McEnery, editors, *Corpus Annotation*, pages 122–126. Longman, London and New York, 1997.

[Fouvry & Bredenkamp 97] F. Fouvry and A. Bredenkamp. Partial parsing in ALEP. In *Proceedings of the 3rd ALEP User Group Workshop*, pages 33–35, Saarbrücken, Germany, 1997.

[Francis & Kucera 82] N.W. Francis and H. Kucera. *Frequency Analysis of English Usage: Lexicon and Grammar.* Houghton Mifflin, Boston, 1982.

[Garside & Smith 97] R. Garside and N. Smith. A hybrid grammatical tagger: CLAWS4. In R. Garside, G. Leech, and A. McEnery, editors, *Corpus Annotation.* Longman, London and New York, 1997.

[Garside *et al.* 87] R. Garside, G. Leech, and G. Sampson, editors. *The Computational Analysis of English: A Corpus Based–Approach.* Longman, London and New York, 1987.

[Gazdar *et al.* 85] G. Gazdar, E. Klein, G.K. Pullum, and I.A. Sag. *Generalized Phrase Structure Grammar.* Cambridge: Blackwell, Cambridge MA: Harward University Press, 1985.

[Gee & Grosjean 83] J. Gee and F. Grosjean. Performance structures: a psycholinguistic and linguistic appraisal. *Cognitive Psychology*, 15:411–458., 1983.

[Gerdemann & King 94] D. Gerdemann and P. King. The correct and efficient implementation of appropriateness specifications for typed feature structures. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING–94)*, pages 956–960, Kyoto, Japan, 1994.

[Giguet & Vergne 97] E. Giguet and J. Vergne. From part of speech tagging to memory–based deep syntactic analysis. In *Proceedings of the 5th International Workshop on Parsing Technologies (IWPT'97)*, Boston, MA, 1997.

[Godden 90] K. Godden. Lazy unification. In *Proceedings of the 28th Annual Meeting of Association for Computational Linguistics (ACL'90)*, pages 180–187, Pittsburgh, PA, 1990.

[Götz 93] T. Götz. *A normal form for typed feature structures.* Magisterarbeit, Universität Tübingen, Germany, 1993.

[Götz *et al.* 97] T. Götz, D. Meurers, and D. Gedermann. *The ConTroll Manual.* SFS Report, 1997.

[Greene & Rubin 71] B. Greene and G. Rubin. *Automatic Grammatical Tagging of English.* Department of Linguistics, Brown University, Providence, Rhode Island, 1971.

[Groenendijk & Schmidt 97] M. Groenendijk and P. Schmidt. MELISSA — Methods and tools for Natural Language Interfacing to Standard Software Applications. In *Proceedings of the 3rd ALEP User Group Workshop*, pages 36–40, Saarbrücken, Germany, 1997.

[Grover & Lascarides 01] C. Grover and A. Lascarides. XML–based data preparation for robust deep parsing. In *Proceedings of the Joint EACL–ACL Meeting (ACL–EACL 2001)*, pages 252–259, Toulouse, France, 2001.

[Hajic & Hladká 98] J. Hajic and B. Hladká. Tagging inflective languages: prediction of morphological categories for a rich structured tagset. In *Proceedings of the Joint 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING–ACL'98)*, pages 483–490, Montreal, Canada, 1998.

[Halteren 99] H. van Halteren, editor. *Syntactic Wordclass Tagging.* Kluwer Academic Publisher, Dordrecht, Boston, London, 1999.

[Hernanz & Brucart 87] M.L. Hernanz and J.M. Brucart. *La Sintaxis. 1: Principios Teóricos. La Oración Simple.* Editorial Crítica, Barcelona, 1987.

[Hindle 83] D. Hindle. *User Manual for Fidditch.* Technical Memorandum 7590–142, Naval Research Laboratory, Washington, D.C., 1983.

[Hindle 89] D. Hindle. Acquiring disambiguation rules from text. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL'89)*, pages 268–275, Vancouver, 1989.

[Hindle 94] D. Hindle. A parser for text corpora. In A. Zampolli, editor, *Computational Approaches to the Lexicon*, Oxford University Press, New York, 1994.

[Hinrichs & Nakazawa 94] E. Hinrichs and T. Nakazawa. Linearizing AUXs in German verbal complexes. In J. Nerbonne, K. Netter, and C. Pollard, editors, *German in Head–Driven Phrase Structure Grammar*, pages 11–37. CSLI Lecture Notes, Stanford University, CA, 1994.

[Hobbs *et al.* 96] J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M Tyson. FASTUS: A cascaded finite–state transducer for extracting information from natural–language text. In Roche and Schabes, editors, *Finite State Devices for Natural Language Processing*, MIT Press, Cambridge, MA, 1996.

[Horiguchi *et al.* 95] K. Horiguchi, K. Torisawa, and J. Tsujii. Automatic acquisition of content words using an HPSG–based parser. In *Proceedings of the Natural Language Processing Pacific Rim Symposium*, pages 320–325, Seoul, Korea, 1995.

[Hurskainen 96] A. Hurskainen. Disambiguation of morphological analysis in Bantu languages. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING–96)*, pages 568–573, Copenhagen, Denmark, 1996.

[Janssen 90] S. Janssen. Automatic word sense disambiguation in LDOCE. In J. Aarts and W. Meijs, editors, *Theory and Practice in Corpus Linguistics*, pages 105–135. Rodopi, Amsterdam, 1990.

[Jelinek *et al.* 92] F. Jelinek, R. Mercer, and S. Roukos. Principles of lexical language modeling for speech recognition. In S. Furhi and M. Sondhi, editors, *Advances in Speech Signal Processing.* Marcel Dekker Inc., New York, 1992.

[Johansson & Hofland 89] S. Johansson and K. Hofland. *Frequency analysis of English vocabulary and grammar: vol.2, tag combinations and word combinations.* Claredon Press, Oxford, 1989.

[Johnson & Dörre 95] M. Johnson and J. Dörre. Memoization of coroutined constraints. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'94)*, pages 100–107, Cambridge, MA, 1995.

[Joshi & Srinivas 94] A. K. Joshi and B. Srinivas. Disambiguating of super parts of speech (or supertags): almost parsing. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING–94)*, Kyoto, Japan, 1994.

[Joshi *et al.* 75] A.K. Joshi, L. Levy, and M. Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 10:136–63, 1975.

[Karlsson 90] F. Karlsson. Constraint grammar as a framework for parsing running text. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING–90)*, volume 3, pages 168–173, Helsinki, Finland, 1990.

[Karlsson *et al.* 95] F. Karlsson, A. Voutilainen, J. Heikkilä, and A. Anttila, editors. *Constraint Grammar: A Language–Independent System for Parsing Unrestricted Text.* Mouton de Gruyter, Berlin and New York, 1995.

[Karttunen & Kay 85] L. Karttunen and M. Kay. Structure sharing with binary trees. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics (ACL'85)*, pages 133–136, Chicago, IL, 1985.

[Kasami 65] J. Kasami. *An Efficient Recognition and Syntax Algorithm for Context-Free Languages.* Scientific Report AFCRL–65–758, Air Force Cambridge Research Laboratory, Bedford, MA, 1965.

[Kasper 94] R. Kasper. Adjuncts in the Mittelfeld. In J. Nerbonne, K. Netter, and C. Pollard, editors, *German in Head-Driven Phrase Structure Grammar*, pages 91–120. CSLI Lecture Notes, Stanford University, CA, 1994.

[Kay 85] M. Kay. Parsing in Functional Unification Grammar. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 213–278. Cambridge University Press, Cambridge, UK, 1985.

[Kay 86] M. Kay. Algorithm schemata and data structures in syntactic processing. In B. Grosz, K. Sparck-Jones, and B.L Webber, editors, *Readings in Natural Language Processing*, pages 35–70. Morgan Kaufmann Publishers, Inc., Los Angeles, CA, 1986.

[Kay 89] M. Kay. Head–driven parsing. In *Proceedings of the 1st International Workshop on Parsing Technologies (IWPT'89).* Carnegie–Mellon University, Pittsburgh, PA, 1989.

[Kiefer & Krieger 00] B. Kiefer and H.-U. Krieger. A context–free approximation of Head–Driven Phrase Structure Grammar. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT'2000)*, pages 135–146, Trento, Italy, 2000.

[Kiefer *et al.* 99] B. Kiefer, H.-U. Krieger, J. Carroll, and R. Malouf. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual*

Meeting of the Association for Computational Linguistics (ACL'99), pages 473–480, University of Maryland, 1999.

[Kiefer *et al.* 00] B. Kiefer, H.-U. Krieger, and M. Siegel. An HPSG–to–CFG approximation of Japanese. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING–2000)*, pages 1046–1050, Saarbrücken, Luxembourg, Nancy, 2000.

[Klavans & Tzoukermann 90] J. Klavans and E. Tzoukermann. The BICORD system. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING–90)*, volume 3, pages 174–179, Helsinki, Finland, 1990.

[Klein & Simmons 63] S. Klein and R. Simmons. A computational approach to grammatical coding of English words. *JACM*, 10:334–347, 1963.

[Knuth 84] D.E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, May, 1984.

[Kogure 90] K. Kogure. Strategic lazy incremental copy graph unification. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING–90)*, volume 2, pages 223–228, Helsinki, Finland, 1990.

[Koskenniemi 83] K. Koskenniemi. *Two–Level Morphology: A General Computational Model for Word Form Recognition and Production.* PhD thesis, Department of General Linguistics, University of Helsinki, Helsinki, Finland, 1983.

[Koskenniemi 90] K. Koskenniemi. Finite–state parsing and disambiguation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING–90)*, volume 2, pages 229–232, Helsinki, Finland, 1990.

[Koskenniemi *et al.* 92] K. Koskenniemi, P. Tapanainen, and A. Voutilainen. Compiling and using finite–state syntactic rules. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING–92)*, pages 156–162, Nantes, France, 1992.

[Krieger & Nerbonne 93] H.-U. Krieger and J. Nerbonne. Feature–based inheritance networks for computational lexicons. In T. Briscoe, V. de Paiva, and A. Copestake, editors, *Inheritance, Defaults and the Lexicon*, pages 90–136. Studies in Natural Language Processing, Cambridge University Press, 1993.

[Krieger 95] H.-U. Krieger. *TDL — A Type Description Language for Constraint–Based Grammars. Foundations, Implementation and Applications.* PhD thesis, Universität des Saarlandes, Department of Computer Science, Saarbrücken, Germany, 1995.

[Kupiec 92] J. Kupiec. Robust part–of–speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242, 1992.

[Kupiec 93a] J. Kupiec. An algorithm for finding noun phrase correspondences in bilingual corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 17–22, Columbus, Ohio, 1993.

[Kupiec 93b] J. Kupiec. Murax: a robust linguistic approach for question answering using an online encyclopedia. In *Proceedings of SIGIR '93*, pages 181–190, 1993.

[Leech *et al.* 94] G. Leech, R. Garside, and M. Bryant. CLAWS4: the tagging of the British National Corpus. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING–94)*, pages 622–624, Kyoto, Japan, 1994.

[Lehmann *et al.* 96] S. Lehmann, S. Oepen, S. Regnier-Prost, K. Netter, V. Lux, J. Klein, K. Falkedal, F. Fouvry, D. Estival, E. Dauphin, H. Compagnion, J. Baur, L. Balkan, and D. Arnold. TSNLP — test suites for natural language processing. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING–96)*, pages 711–716, Copenhagen, Denmark, 1996.

[Lenci *et al.* 00] A. Lenci, N. Bel, F. Busa, N. Calzolari, M. Monachini, A. Ogonowski, I. Peters, W. Peters, N. Ruimy, M. Villegas, and A. Zampolli. A general framework for the development of multilingual lexicons. *Journal of Lexicography*, 13:249–263, 2000.

[Liberman & Church 91] M. Liberman and K.W. Church. Text analysis and word pronunciation in text–to–speech synthesis. In S. Furni and M. Mohan, editors, *Advances in Speech Signal Processing*, pages 791–832. Marcel Dekker, 1991.

[LSGRAM 93] LSGRAM. *Large-Scale Grammars for EC Languages.* Annex I. Technical and Financial Annex for LRE 61029, 1993.

[Maas 96] H.D. Maas. MPRO - ein System zur Analyse und Synthese deutscher Wörter. In R. Hausser, editor, *Linguistische Verifikation, Sprache und Information, Nr. 34.* Max Niemeyer Verlag, Tübingen, Germany, 1996.

[Magerman 95] D. Magerman. Statistical decision tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, pages 276–283. Cambridge, MA, 1995.

[Malouf *et al.* 00] R. Malouf, J. Carroll, and A. Copestake. Efficient feature structure operations without compilation. In D. Flickinger, S. Oepen, J. Tsujii, and

H. Uszkoreit, editors, *Natural Language Engineering (6)1 —Special Issue: Efficiency Processing with HPSG: Methods, Systems, Evaluation*, pages 29–46. Cambridge University Press, 2000.

[Manandhar 94] S. Manandhar. *User's Guide for CL–ONE.* Centre for Cognitive Science, University of Edinburgh, Scotland, 1994.

[Marcken 90] C. de Marcken. Parsing the LOB corpus. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL'90)*, pages 243–251, Pittsburgh, PA, 1990.

[Marimon & Porta 00] M. Marimon and J. Porta. PoS disambiguation and partial parsing: bidirectional interaction. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC–2000)*, pages 1117–1122, Athens, Greece, 2000.

[Marimon 02] M. Marimon. Integrating shallow linguistic processing into a unification–based Spanish grammar. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING–2002)*, pages 619–625, Taipei, Taiwan, 2002.

[Marimon *et al.* 99] M. Marimon, A. Theofilidis, T. Declerck, and A. Bredenkamp. Linguistic processing modules in ALEP for natural language interfaces. In *Actas del XV Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN'99)*, pages 129–133, Lleida, 1999.

[Marimon *et al.* 01] M. Marimon, J. Porta, and N. Bel. On distributing the analysis process of a broad–coverage unification–based grammar. In *Proceedings of the 3rd Euroconference Recent Advances in Natural Language Procecessing (RANLP–2001)*, Tzigov Chark, Bulgaria, 2001.

[Marques & Lopes 96] N.M. Marques and G.C. Lopes. A neural network approach to part–of–speech tagging. In *Proceedings of the 2nd Workshop on Computational Processing of Written and Spoken Portuguese*, pages 21–22, Brazil, 1996.

[Márquez 99] L. Márquez. *POS Tagging: A Machine Learning Approach Based on Decision Trees.* PhD thesis, Departament Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, 1999.

[Maxwell & Kaplan 93] J. Maxwell and R. Kaplan. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590, December, 1993.

[Melero & Villegas 98] M. Melero and M. Villegas. Issues on the encoding of a computational lexicon. In *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC–1998)*, pages 827–832, Granada, Spain, 1998.

[Melero 95] M. Melero. *Spanish Corpus Study Resources.* LS-GRAM Deliverable D2a, Grup d'Investigació en Lingüística Computacional Universitat de Barcelona (*gilc*UB), Barcelona, 1995.

[MELISSA 95] MELISSA. *Methods and tools for Natural-Language Interfacing to Standard Software Applications.* Technical and Financial Annex for ESPRIT Project 22252. Part II: Description of the RTD Project, 1995.

[Merialdo 94] B. Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172, June, 1994.

[Meurers & Minnen 95] D. Meurers and G. Minnen. A computational treatment of HPSG lexical rules as covariation in lexical entries. In *Proceedings of the 5th International Workshop on Natural Language Understanding and Logic Programming*, Lisbon, Portugal, 1995.

[Meurers 97] D. Meurers. Using lexical principles in HPSG to generalize over valence properties. In *Proceedings of the 3rd Conference on Formal Grammar*, pages 137–146, Aix–en–Provence, France, 1997.

[Mitsuishi *et al.* 98] Y. Mitsuishi, K. Torisawa, and J. Tsujii. HPSG–style underspecified Japanese grammar with wide coverage. In *Proceedings of the Joint 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING–ACL'98)*, pages 876–880, Montreal, Canada, 1998.

[Miyao *et al.* 00] Y. Miyao, Makino, K. Torisawa, and J. Tsujii. The LILFeS abstract machine and its evaluation with the LinGO grammar. In D. Flickinger, S. Oepen, J. Tsujii, and H. Uszkoreit, editors, *Natural Language Engineering (6)1 —Special Issue: Efficiency Processing with HPSG: Methods, Systems, Evaluation*, pages 47–61. Cambridge University Press, 2000.

[Monachini & Calzolari 96] M. Monachini and N. Calzolari. (coordinators) Synopsis and Comparison of Morphosyntactic Phenomena Encoded in Lexicons and Corpora. A Common Proposal and Applications to European Languages. Technical report, Final Report, EAG—CLW—MORPHSYN/R, 1996.

[Müller & Kasper 00] S. Müller and W. Kasper. HPSG analysis of German. In W. Wahlster, editor, *Verbmobil: Foundations of Speech–to–Speech Translation, Artificial Inteligence*, pages 238–258. Berlin: Springer Verlag, New York: Heidelberg, 2000.

[Music & Navarretta 96] B. Music and C. Navarretta. *Final Documentation of the LS–GRAM Danish Lingware.* LS–GRAM Deliverable E–D8–DK, Center for Sprogteknologi, Copenhagen, Denmark, 1996.

[Music 95] B. Music. *Tagging Messy Details in ALEP.* LS-GRAM Supplement to Deliverable E-D5-DK, Center for Sprogteknologi, Copenhagen, Denmark, 1995.

[Music 96] B. Music. Tagging for disambiguation. In *Proceedings of Datalinguistisk Forening*, 1996.

[Music 97] B. Music. ALEP and MT. In *Proceedings of the 3rd ALEP User Group Workshop*, pages 54–61, Saarbrücken, Germany, 1997.

[Nerbonne *et al.* 94] J. Nerbonne, K. Netter, and C. Pollard, editors. *German in Head–Driven Phrase Structure Grammar.* CSLI Lecture Notes, Stanford University, CA, 1994.

[Noord 97] G. van Noord. An efficient implementation of the head–corner parser. *Computational Linguistics*, 23(3):425–456, September, 1997.

[Oepen & Carroll 00] S. Oepen and J. Carroll. Ambiguity packing in constraint–based parsing – practical results. In *Proceedings of 1st Conference of the North American Chapter of the Association of Computational Linguistics (NAACL'00)*, Seattle, WA, 2000.

[Oflazer & Kuruöz 94] K. Oflazer and I. Kuruöz. Tagging and morphological disambiguation of Turkisk text. In *Proceedings of the 4th Conference on Applied Natural Language Processing (ANLP'94)*, pages 144–149, Stuttgart, Germany, 1994.

[Oflazer & Tür 96] K. Oflazer and G. Tür. Combining hand–crafted rules and unsupervised learning in constraint–based morphological disambiguation. In *Proceedings of the ACL SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 69–81, Philadelphia, PA, 1996.

[Oflazer & Tür 97] K. Oflazer and G. Tür. Morphological disambiguation by voting constraints. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, pages 222–229, Madrid, Spain, 1997.

[Oostdijk 91] N. Oostdijk. *Corpus Linguistics and the Automatic Analysis of English.* Rodopi, Amsterdam, 1991.

[Padró 98] L. Padró. *A Hybrid Environment for Syntax–Semantic Tagging.* PhD thesis, Departament Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, 1998.

[Pereira 85] F.C. Pereira. A structure-sharing representation for unification–based grammar formalisms. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics (ACL'85)*, pages 137–144, Chicago, IL, 1985.

[Pollard & Sag 87] C. Pollard and I.A. Sag. *Information–Based Syntax and Semantics, Volume 1: Fundamentals.* CSLI Lecture Notes, Stanford University, CA, 1987.

[Pollard & Sag 94] C. Pollard and I.A. Sag. *Head–Driven Phrase Structure Grammar.* University of Chicago Press and CSLI Publications, Chicago, 1994.

[Porta 96] J. Porta. *Rtag.* Technical Report, Grup de Investigació en Lingüística Computacional Universitat de Barcelona (*gilc*UB), Barcelona, 1996.

[Porta 00] J. Porta. *Representación de estructuras de rasgos en formalismos para el procesamiento del lenguaje natural.* Memoria de investigación. Universidad Autónoma de Madrid, Madrid, Spain, 2000.

[Prins & Noord 01] R. Prins and G. van Noord. Unsupervised post–tagging improves parsing accuracy and parsing efficiency. In *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT'2001)*, Beijing, China, 2001.

[Rabiner 90] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In A. Waibel and K. F. Lee, editors, *Readings in Speech Recognition.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[Ramshaw & Marcus 95] L.A. Ramshaw and M.P. Marcus. Text chunking using transformation–based learning. In D. Yarovsky and K.W. Church, editors, *Proceedings of the 3rd Workshop on Very Large Corpora (ACL'95)*, pages 82–94, Somerset, New Jersey, 1995. Association for Computational Linguistics.

[Reichenbach 47] H. Reichenbach. *Element of Symbolic Logic.* Macmillan, New York, 1947.

[Resnik 93] P.S. Resnik. *Selection and Information: a Class–Based Approach to Lexical Relationship.* PhD thesis, Computer and Information Science Department, University of Pennsylvania, Philadelphia, PA, 1993.

[Rodríguez 99] H. Rodríguez. Técnicas estadísticas en el tratamiento del lenguaje natural. In *Filología e Informática: Nuevas tecnologías en los estudios filológicos*, Seminario de Filología e Informática, Universitat Autónoma Barcelona, Barcelona, 1999.

[Rodríguez 00] H Rodríguez. Técnicas de análisis sintáctico. `http://www.lsi.upc.es/docencia.htmlsoria00.zip`, 2000.

[Sag 97] I.A. Sag. English relative clause constructions. *Journal of Linguistics*, 33:431–484, 1997.

[Salton *et al.* 90] G. Salton, Z. Zhao, and C. Buckely. *A Simple Syntactic Approach for the Generation of Indexing Phrases.* Technical Report 90–1137, Department of Computer Science, Cornell University, 1990.

[Samuelsson & Voutilainen 97] C. Samuelsson and A. Voutilainen. Comparing a linguistic and a stochastic tagger. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, pages 246–253, Madrid, Spain, 1997.

[Samuelsson *et al.* 96] C. Samuelsson, P. Tapanainen, and A. Voutilainen. Inducing constraint grammars. In Miclet and de la Higuera, editors, *Grammatical Inference: Learning Syntax from Sentences*, volume Lecture Notes in Artificial Intelligence 1147, pages 146–155. Springer Verlag, Berlin, 1996.

[Sánchez & Nieto 95] F. Sánchez and A. Nieto. Desarrollo de un etiquetador morfosintáctico para el español. In *Actas del XI Congreso de la Sociedad Española para el Procesamiento del Lenguage Natural, (SEPLN'95)*, pages 14–28, Universidad de Deusto, Bilbao, 1995.

[Sánchez 97] F. Sánchez. *Análisis morfológico y desambiguación en castellano.* PhD thesis, Departamento de Lingüística, Lenguas Modernas, Lógica y Filosofía, Facultad deFilosofía y Letras, Madrid, Spain, 1997.

[Sánchez *et al.* 99] F. Sánchez, J. Porta, J.L. Sancho, A. Nieto, A. Ballester, A. Fernández, L. Gómez, E. Raigal, and R. Ruiz. La anotación de los corpus CREA y CORDE. In *Actas del XV Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN'99)*, pages 175–182, Lleida, 1999.

[Sánchez López 99] C. Sánchez López. La negación. In I. Bosque and V. Demonte, editors, *Gramática descriptiva de la lengua española*, Real Academia Española, Colección Nebrija y Bello. Espasa Calpe, Madrid, 1999.

[Schmid 94] H. Schmid. Part–of–speech tagging with neural netwoks. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING–94)*, pages 172–176, Kyoto, Japan, 1994.

[Schmid 95] H. Schmid. Improvements in part–of–speech tagging with an application to German. In *Proceedings of the SIGDAT Workshop, 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*, Dublin, Ireland, 1995.

[Schmidt 98] P. Schmidt. Formal assumptions. In F. van Eynde and P. Schmidt, editors, *Studies in Machine Translation and Natural Language Processing, volume 10: Linguistic Specifications for Typed Feature Formalism*, pages 11–38. European Commission, Luxembourg, 1998.

[Schmidt *et al.* 96] P. Schmidt, S. Rieder, A. Theofilidis, and T. Declerck. Lean formalism, linguistic theory, and applications: grammar development in ALEP. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING–96)*, pages 286–291, Copenhagen, Denmark, 1996.

[Schütze 93] H. Schütze. Part–of–speech induction from scratch. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL'93)*, pages 251–258, Columbus, Ohio, 1993.

[Schwarz 90] C. Schwarz. Automatic syntactic analysis for free text. *JASIS*, 41(6):408–417, 1990.

[Sells 85] P. Sells. *Lectures on Contemporary Syntactic Theories.* CSLI Lecture Notes, Stanford University, CA, 1985.

[Shieber 85] S. M. Shieber. Using restriction to extend parsing algorithms for complex–feature–based formalisms. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics (ACL'85)*, pages 145–152, Chicago, IL, 1985.

[Shieber 86] S. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes, Stanford University, CA, 1986.

[Shieber *et al.* 83] S. Shieber, H. Uszkoreit, F. Pereira, J. Robinson, and M. Tyson. The Formalism and Implementation of PATR-II. In *Research on Interactive Acquisition and Use of Knowledge*, pages 39–79, AI Center, SRI International, Menlo Park, CA, 1983.

[Sikkel 97] K. Sikkel. *Parsing Schemata.* Springer–Verlag, Berlin, Germany, 1997.

[Simpkins *et al.* 93] N. K. Simpkins, M. Groenendijk, and G. Cruickshank. *ALEP User Guide.* Commission of the European Communities, Luxembourg, 1993.

[Smadja 93] F. Smadja. Retrieving collocations form text: Xtract. *Computational Linguistics*, 19(1):143–177, March, 1993.

[Srinivas *et al.* 97] B. Srinivas, C. Doran, B.A. Hockey, and A. K. Joshi. An approach to robust partial parsing and evaluation metrics. In *Proceedings of the 5th International Workshop on Parsing Technologies (IWPT'97)*, Boston, MA, 1997.

[Steiner & Tsujii 99] R. Steiner and J. Tsujii. Robustness in HPSG via extended ID schemata. In *Proceedings of the Natural Language Pacific Rim Symposium'99*, pages 126–131, Beijin, China, 1999.

[Summers 96] D. Summers. Computer lexicography: the importance of representativeness in relation to frequency. In J. Thomas and M. Short, editors, *Using Corpora for Language Research*, pages 260–266. Longman, London, 1996.

[Tapanainen & Voutilainen 94] P. Tapanainen and A. Voutilainen. Tagging accurately – don't guess if you know. In *Proceedings of the 4th Conference on Applied Natural Language Processing (ANLP'94)*, pages 47–52, Stuttgart, Germany, 1994.

[Theofilidis & Reuther 95] A. Theofilidis and U. Reuther. Dealing with syntactic frame variations by co–representing heads. 2nd ALEP User Group Workshop, Luxembourg, 1995.

[Theofilidis & Schmidt 97] A. Theofilidis and P. Schmidt. ALEP–based distributed grammar engineering. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering, ACL/EACL'97*, pages 84–90, Madrid, Spain, 1997.

[Theofilidis 95a] A. Theofilidis. A boolean approach to saturation - why we want it, how it would work, and why it does not work? 2nd Workshop of the LS-GRAM Project, Barcelona, 1995.

[Theofilidis 95b] A. Theofilidis. What to do with unrealised arguments? The case of optional by–agents. 2nd Workshop of the LS-GRAM Project, Barcelona, 1995.

[Tomabechi 91] H. Tomabechi. Quasi–destructive graph unification. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL'91)*, pages 315–322, Berkeley,CA, 1991.

[Tomita 87] M. Tomita. An efficient augmented context–free parsing algorithm. *Computational Linguistics*, 13(1–2):31–46, 1987.

[Torisawa *et al.* 00] K. Torisawa, K. Nishida, Y. Miyao, and J. Tsujii. An HPSH parser with CFG filtering. In D. Flickinger, S. Oepen, J. Tsujii, and H. Uszkoreit, editors, *Natural Language Engineering (6)1 —Special Issue: Efficiency Processing with HPSG: Methods, Systems, Evaluation*, pages 63–80. Cambridge University Press, 2000.

[Tür 96] G. Tür. *Using Multiple Sources of Information for Constraint–Based Morphological Disambiguation.* M.Sc. Thesis, Submitted to the Department of Computer Engineering and Information Science of Bilkent University, Turkey, 1996.

[Uszkoreit 86] H. Uszkoreit. *Linear Precedence in Discontinous Constituents: Complex Fronting in German.* Report CSLI–86–47, Stanford University, CA, 1986.

[Uszkoreit 91] H. Uszkoreit. Strategies for adding control information to declarative grammars. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL'91)*, pages 237–245, Berkeley, CA, 1991.

[Uszkoreit *et al.* 94] H. Uszkoreit, R. Backofen, S. Busemann, A.K. Diagne, E.A. Hinkelman, W. Kasper, B. Kiefer, H.-U. Krieger, K. Netter, G. Neumann, S. Oepen, and S.P. Spackman. DISCO–an HPSG based NLP system and its application for appointment scheduling. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING–94)*, Kyoto, Japan, 1994.

[Venkova 00] T. Venkova. A local grammar disambiguator of compound conjunctions as a pre–processor for deep analysers. In *Proceedings of Workshop on Linguistic Theory and Grammar Implementation. 12th European Summer School in Logic, Language and Information (ESSLLI–2000)*, pages 239–254, Birmingham, UK, 2000.

[Villegas *et al.* 98] M. Villegas, I. Brosa, and N. Bel. El léxico PAROLE del español. In *Actas del XIV Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN'98)*, pages 84–89, Alicante, Spain, 1998.

[Viterbi 67] A. Viterbi. Errors bounds for convolutional codes and an asymptotically optium decoding algorithm. *IEEE Transactions on Information Theory*, IT–13:260–269, April, 1967.

[Voutilainen & Heikkilä 94] A. Voutilainen and J Heikkilä. An English constraint grammar (ENGCG): a surface–syntactic parser of English. In U. Fries, G. Tottie, and P. Schneider, editors, *Creating and Using English Language Corpora*, pages 189–199. Rodopi, Amsterdam and Atlanta, 1994.

[Voutilainen & Tapanainen 93] A. Voutilainen and P. Tapanainen. Ambiguity resolution in a reductionistic parser. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL'93)*, pages 394–403, Utrecht, Netherlands, 1993.

[Voutilainen 93] A. Voutilainen. NPtool, a detector of English noun phrases. In *Proceedings of the Workshop on Very Large Corpora, ACL*, pages 42–51. Columbus, Ohio, 1993.

[Voutilainen 94] A. Voutilainen. Three Studies of Grammar–Based Surface Parsing of Unrestricted English Text, Department of General Linguistics, University of Helsinki, Helsinki, Finland, 1994.

[Voutilainen 95] A. Voutilainen. A syntax–based part–of–speech analyser. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*, pages 157–164, Dublin, Ireland, 1995.

[Voutilainen *et al.* 92] A. Voutilainen, J. Heikkilä, and A. Anttila. *Constraint Grammar of English. A Performance–Oriented Introduction.* Publication No. 21, Department of General Linguistics, University of Helsinki, Helsinki, Finland, 1992.

[Watanabe 00] H. Watanabe. A method for accelerating CFG–parsing by using dependency information. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING–2000)*, pages 913–918, Saarbrücken, Luxembourg, Nancy, 2000.

[Wilson & Rayson 93] A. Wilson and P. Rayson. Automatic content analysis of spoken discourse: a report on work in progress. In C. Souter and E. Atwell, editors, *Corpus Based Computational Linguistics*, pages 215–226. Rodopi, Amsterdam, 1993.

[Wintner & Francez 99] S. Wintner and N. Francez. Efficient implementation of unification–based grammars. *Journal of Language and Computation*, 1(1):53–92, April, 1999.

[Wroblewski 87] D. Wroblewski. Non–destructive graph unification. In *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 582–587, Seattle, WA, 1987.

[Yarowsky 92] D. Yarowsky. Word–sense disambiguation using statistical models of Roget's categories trained on corpora. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING–92)*, pages 454–460, Nantes, France, 1992.

[Yoon *et al.* 99] J. Yoon, K.-S. Choi, and M. Song. Three types of chunking in Korean and dependency analysis based on lexical association. In *Proceedings of the 18th International Conference on Computer Processing of Oriental Languages (ICCPOL'99)*, Tokushima, Japan, 1999.

[Younger 67] D. Younger. Recognition and parsing of context-free languages in time O(n_3). *Information and Control*, 10(2):189–208, 1967.

[Zeevat *et al.* 87] H. Zeevat, E. Klein, and J. Calder. Unification Categorial Grammar. In N. Haddock, E. Klein, and G. Morrill, editors, *Categorial Grammar, Unification Grammar and Parsing*, volume 1. Edinburgh Working Papers in Cognitive Science, Edinburgh: Centre for Cognitive Science, University of Edinburgh, UK, 1987.