Deep Linguistic Processing of Portuguese Noun Phrases

Francisco Nuno Quintiliano Mendonça Carapeto Costa

DI–FCUL TR–07–34

November 2007

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749–016 Lisboa
Portugal

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Deep Linguistic Processing of Portuguese Noun Phrases

Francisco Nuno Quintiliano Mendonça Carapeto Costa

MESTRADO EM INFORMÁTICA

2007

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Deep Linguistic Processing of Portuguese Noun Phrases

Francisco Nuno Quintiliano Mendonça Carapeto Costa

MESTRADO EM INFORMÁTICA

2007

Dissertação orientada pelo Prof. Doutor António Branco

Resumo

Esta dissertação descreve a implementação de um fragmento do Português numa gramática computacional — LXGram— actualmente em desenvolvimento na Universidade de Lisboa. A LXGram é uma gramática computacional para o processamento linguístico profundo do Português. Como tal, pode ser utilizada para analisar frases do Português, produzindo uma descrição formal do seu significado, ou para gerar frases em Português a partir de representações do significado.

A LXGram é desenvolvida numa plataforma desenhada especificamente para acomodar tais gramáticas — o Linguistic Knowledge Builder (LKB). O LKB implementa algoritmos muito eficientes de análise e geração. Aceita um formalismo que é declarativo e a operação fundamental é a unificação. Adicionalmente emprega um sistema de tipos rígido com herança múltipla, o que proporciona um meio elegante de formular generalizações interessantes e permite verificação estática de tipos. Existem várias outras gramáticas desenvolvidas no LKB, para outras línguas naturais. Algumas destas gramáticas têm vindo a ser integradas em aplicações úteis, como a tradução automática, sistemas de respostas automáticas a correio electrónico, correctores gramaticais e extracção de informação.

Esta dissertação descreve a modelação e a implementação computacional na LXGram de um conjunto de fenómenos linguísticos. Estes fenómenos relacionam-se com as propriedades gramaticais e o significado dos sintagmas nominais em Português (qualquer expressão do Português que pode ocorrer nos contextos onde os pronomes pessoais também ocorrem). A implementação destes fenómenos na LXGram foca-se em alguns aspectos que não estão muito desenvolvidos nas outras gramáticas implementadas no LKB. É novo um modelo computacional que dá conta de diversas interacções interessantes entre eles.

Nos primeiros dois capítulos desta dissertação faz-se uma introdução à tarefa em questão e descrevem-se as ferramentas e o formalismo adoptados. Os três capítulos seguintes apresentam os dados que são cobertos e as soluções que foram adoptadas. O último capítulo inclui uma revisão das ideias principais da dissertação, uma avaliação da implementação levada a cabo e sugestões de trabalho futuro.

PALAVRAS-CHAVE: Processamento de Linguagem Natural, Gramáticas de Unificação, Lógicas de Atributos e Tipos, Processamento Linguístico Profundo.

Abstract

This dissertation describes the implementation of a fragment of Portuguese in a computational grammar — LXGram— currently being developed in the University of Lisbon. LXGram is a computational grammar for the deep linguistic processing of Portuguese. As such, it can be used to parse Portuguese sentences, producing a formal description of their meaning, or to generate Portuguese sentences from meaning representations.

LXGram is developed in a platform that is specifically designed to handle such grammars — the Linguistic Knowledge Builder (LKB). The LKB implements very efficient algorithms for parsing and generation. It accepts a formalism that is declarative and resorts to unification as the fundamental operation. It also employs a strict type system with multiple inheritance, which provides an elegant means of stating interesting generalizations and allows for static type checking. Several other grammars have been developed in the LKB, for other natural languages. Some of these grammars have been integrated in useful applications, like machine translation, automated e-mail responses, grammar checking and information extraction.

This dissertation describes the modeling and computational implementation of a set of linguistic phenomena in LXGram. These phenomena are related to the grammatical properties and the meaning of the Portuguese noun phrase (any Portuguese expression that can appear in the contexts where personal pronouns are allowed). The implementation of these phenomena in LXGram focuses on some aspects that are not very developed in the other LKB grammars. A computational model that accounts for several interesting interactions among them is new.

In the first two chapters of this dissertation we provide an introduction to the task at hand, and we describe the tools and formalism that are adopted. The three chapters that follow present the data to be covered and the solutions that were adopted. The last chapter reviews the main points of the dissertation, includes an evaluation of the resulting implementation and suggests future work.

KEY WORDS: Natural Language Processing, Unification Grammars, Typed Feature Logics, Deep Linguistic Processing .

Acknowledgments

The initial phase of the implementation of LXGram was conducted during my internship at the University of Saarland, Department of Computational Linguistics. I am indebted to Hans Uszkoreit for the conditions made available there to me. During that period I also received support from Valia Kordoni and Berhold Crysmann, and I also thank them.

Some of the material in Chapter 5 has been previously published in a conference paper. I must thank my co-author in that paper and supervisor, António Branco, whose suggestions have been invaluable, during the preparation of that paper and this dissertation.

The work leading to this thesis was part of project GramaXing (FCT/PLUS/PLP/50301/2003), funded by Fundação para a Ciência e Tecnologia, under the coordination of António Branco.

Lisbon, November 21, 2007
Francisco Nuno Quintiliano Mendonça Carapeto Costa

Contents

List of Figures

1 Introduction

1.1 Overview

In this chapter we present the main topics addressed throughout this dissertation. We introduce very briefly the language phenomena that will be covered, the conceptual and development context assumed and the tools that are used to implement a computational model of them. We present a motivation for these topics, and refer the issues and problems that arise in handling them. We discuss why they are important and what applications their implementation can support. Finally, we also present an overview of the remaining chapters in this dissertation.

1.2 Context

The area of Natural Language Processing (NLP) is ultimately concerned with the development of software that can understand text written in a natural human language as humans do, or at least with the development of software that can behave as if it understood it. The other side of the coin, generating meaningful and appropriate text in a natural human language, is another goal.

There are many applications where such behavior is useful. The classical example is HAL, from the movie *2001: A Space Odyssey*, a computer that can dialog with human beings. We are still very far from producing systems that allow humans to communicate with computers as easily as fictioned in that film, but progress in this area has led to a number of less ambitious yet very useful applications: machine translation, question answering, information extraction, chatterbots, automated e-mail responses, grammar checking, automated knowledge acquisition, to name a few.

Machine translation is a particularly important application, with tremendous economical, social and cultural potential.[1] It has been the classic application of the sort of systems we will be talking about in this text. But even current state-of-the-art machine translation cannot compare to top human translation as far as quality is concerned.

Inside NLP several subareas and methods can be found. In this text we will be focusing on deep symbolic processing. By symbolic we mean that machine learning algorithms are not used, but domain knowledge is programmed directly, and the system is rule-based. By deep we mean that we are interested in processing text in order to generate semantic representations of the input, i.e. detailed representations of its meaning. More precisely, we are interested in converting between meaning representations and Portuguese sentences, in both directions.

Grammars for deep linguistic processing try to convert a natural language expression into its syntactic and semantic representation (to parse) and produce a natural language expression from its semantic representation (to generate).

[1]For instance, one of the press releases of the European Union, accessible on the site of the European Commission (http://ec.europa.eu), mentions that "in 2003, before enlargement and with a population of 379 million, expenditure on translation by all the EU institutions came to 549 M EUR. ($\cdots$) Before the 2004 enlargement, translation in all institutions accounted for about 0.55% of the total budget of the EU and about 9% of the administrative expenditure in the EU".

The conception of meaning assumed here is very specific. It is best understood with an example: the meaning representation that a deep grammar assigns to a sentence like *Nietzsche killed God* only says that the ordered pair whose first element is an entity known as *Nietzsche* and whose second is an entity known as *God* is a member of the set of ordered pairs that corresponds to the relation *kill*. That is, we basically capture compositional semantics — how the meaning of parts of a sentence relate to each other.

As far as our example is concerned, a deep grammar does not identify the entities Nietzsche and God in the real world, it does not identify all pairs in that set (the extension of *kill*),[2] and, for example, it does not say that the second elements of these pairs are dead (its intension).[3]

Nevertheless, our meaning representations are actually much more powerful than the previous explanation might suggest. Semantic representations can be viewed as normalized forms of surface strings, that abstract from superficial, grammatical aspects of language (morphology, syntax, etc.). A deep grammar can be viewed as an interface between applications and natural language: the grammar encapsulates language specific knowledge. Moreover, the semantic representations are more than a mere normalization, they are logical formulas and as such have a precise interpretation and support some automated inference.

A grammar must of course ensure that the meaning representations it produces are well-formed, that they adhere to a specific format. It must also ensure that the natural language expressions it manipulates are also grammatical in the language in question. A grammar can be used both for parsing and generation; it is desirable that the strings it produces in generation are grammatical just like it is important that the semantic representations it generates in parsing are well-formed.

Therefore, some computational grammars (also termed as precision grammars) try to account for all and only the expressions that belong to a given language: the ones that are not acceptable should be rejected. For these reasons, precision grammars cannot disregard the little details, and adopting an expressive formalism is therefore inevitable.

The work presented in this dissertation was conducted as part of the development of a computational grammar for Portuguese, LXGram,[4] currently being implemented in the University of Lisbon (Branco and Costa, 2006, 2007a,b; António and Francisco, 2007).

LXGram is integrated in the DELPH-IN initiative, an international collaborative effort to produce precision grammars for several languages, some of which are already integrated in important multilingual applications, like machine translation (Bond *et al.*, 2005). Other grammars exist that are part of DELPH-IN: the LinGO English Resource Grammar for English (Copestake and Flickinger, 2000), Jacy for Japanese (Siegel and Bender, 2002) and the German Grammar for German (Müller and Kasper, 2000) are currently the largest ones. There are other DELPH-IN grammars being developed for French, Modern Greek, Korean, Norwegian, Spanish and other languages.

1.3 Subject Matter

This dissertation is concerned with the modeling and implementation of a non-trivial subset of phenomena in the noun phrase (NP) domain in Portuguese. We propose to (1) describe and model the linguistic knowledge of the noun phrase domain in Portuguese and (2) implement it by expanding a computational grammar for the deep linguistic processing of Portuguese accordingly, addressing the

[2]Also called its denotation or, in Fregean terms, its meaning or reference (*Bedeutung*).

[3]Its connotation or Frege's sense (*Sinn*).

[4]*LX* is used in Portugal to abbreviate *Lisboa* (*Lisbon*), because the intersection of the sets of phonetic sequences representable in Portuguese writing by the strings "x" and "is" is non-empty.

key implementation issues.

A noun phrase is any expression that has the same syntactic distribution as a personal pronoun (i.e. it can replace a personal pronoun without loss of grammaticality) . The following expressions inside square brackets are all examples of NPs, followed by a verb:

(1) a. [$_{\text{NP}}$ Eles] avariaram$_{\text{V}}$.
they broke down
They broke down.

b. [$_{\text{NP}}$ Esses carros] avariaram$_{\text{V}}$.
those cars broke down
Those cars broke down.

c. [$_{\text{NP}}$ Os meus carros] avariaram$_{\text{V}}$.
the my cars broke down
My cars broke down.

d. [$_{\text{NP}}$ Aqueles meus dois carros] avariaram$_{\text{V}}$.
those my two cars broke down
Those two cars of mine broke down.

e. [$_{\text{NP}}$ Ela] saiu$_{\text{V}}$.
she left
She left.

f. [$_{\text{NP}}$ A Ana] saiu$_{\text{V}}$.
the Ana left
Ana left.

The reasons why the topic of NPs was chosen are: (1) a grammar must be able to process NPs, as almost every sentence will contain at least one NP; (2) the topic is not very developed within the grammatical framework that is adopted (see Section 2.2); (3) there are interesting and non-trivial issues to be addressed; (4) NPs exhibit the classical problems of language processing — ambiguity at all levels: syntactic and semantic (they are the locus of the prime example of semantic ambiguity, viz. quantifier scope ambiguity — see Section 2.3).[5]

To illustrate some of the issues involved, consider the difference in meaning between (2a) and (2b).

(2) a. um falso médico chinês
a fake doctor Chinese
a fake Chinese doctor

b. um falso médico que é chinês
a fake doctor that is Chinese
a fake doctor that is Chinese

In (2b) an entity is described as not being a doctor but being Chinese, whereas in (2a) an entity is described as not being simultaneously a doctor and Chinese. These examples contrast with the ones in (3), in that the examples in (3) convey the same meaning.

(3) a. um médico chinês
a doctor Chinese
a Chinese doctor

[5] Also, some the data adduced to show that the syntax of natural languages cannot be described by a finite state machine includes relative clauses inside NPs (Chomsky, 1957).

b.

um	médico	que	é	chinês
a	doctor	that	is	Chinese

a doctor that is Chinese

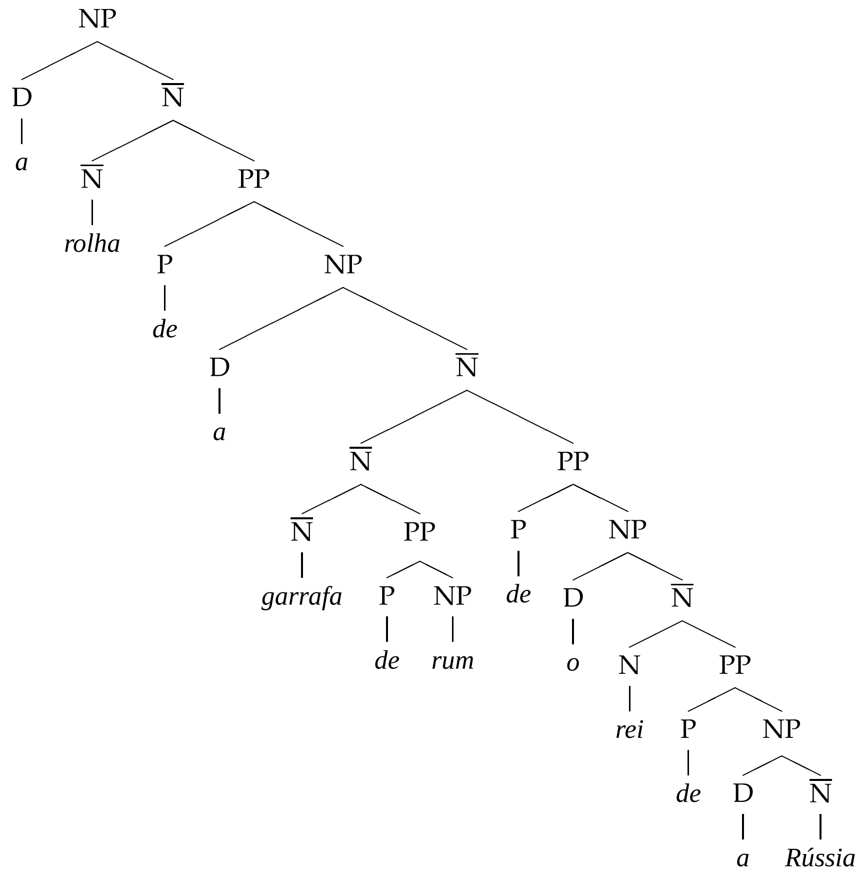
Also, there is no upper bound on the length of an NP, as the following Portuguese tongue-twister illustrates:

(4)

O	rato	roeu	[$_{NP}$ a	rolha	da	garrafa	de	rum	do	rei	da	Rússia].
the	mouse	gnawed	the	cork	of the	bottle	of	rum	of the	king	of the	Russia

The mouse gnawed the cork of the king of Russia's bottle of rum.

The expression labeled with NP in this example can be given the following structure, where PP (preposition phrase) denotes an expression consisting of a preposition followed by an NP (and D stands for determiner, P for preposition, N for noun, and $\overline{\text{N}}$ for a noun with no arguments or a sequence consisting of a noun followed by its arguments):



It shows recursion; in particular a PP (which contains an NP) can be inside an NP.

These and other issues will be addressed in the dissertation.

1.4 Tools

Like the other grammars in DELPH-IN, LXGram is being implemented in the LKB— Linguistic Knowledge Building system (Copestake, 2002) —, with [incr tsdb()] (Oepen, 2001) being used for test suite management, regression testing, and training of maximum entropy models for parse selection.[6]

[6]When the grammar produces more than one solution, a stochastic model is used to choose the fittest one.

These grammars are also compatible with PET— Platform for Experimentation with efficient HPSG processing Techniques (Callmeier, 2000) —, an efficient parser for context free languages.

Whereas there is no separate process of compiling and running a grammar in the LKB, this is so in PET. In any case, a high amount of checking is performed when a grammar is loaded in the LKB: type checking, cyclicity checks,verification of redundant specifications in the source code, etc. PET provides a compiler (flop) and an interpreter (cheap). With PET, a grammar is first compiled to a binary format with flop and the result interpreted with cheap. This allows for efficiency optimizations (like precompiling the lexicon or the results of type unification, as reported in (Kiefer *et al.*, 1999)) that enable a grammar to parse text very fast.

Both the LKB and PET accept grammars written in $\mathscr{TDL}$— Type Description Language (Krieger and Schäfer, 1994). These systems accept code describing typed feature structures (see Chapter 2), which includes code conforming to Head-Driven Phrase Structure Grammar (HPSG, see Section 2.2). LXGram is based on HPSG, too. The syntax of $\mathscr{TDL}$ is very similar to the AVM (attribute value matrices) descriptions in theoretical HPSG, as exemplified in Figure 1.1 and Figure 1.2.

```
some-type := some-supertype & other-supertype &
 [ A.B [ C t,
         D [ E.F #Value-of-F-and-G,
             G   #Value-of-F-and-G ] ] ].
```

Figure 1.1: Example of $\mathscr{TDL}$ code.

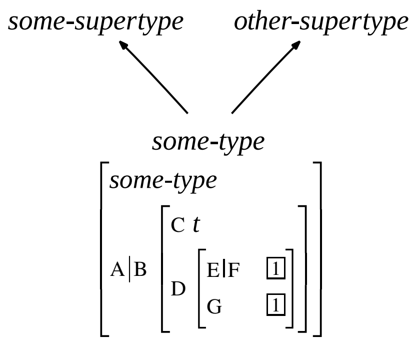
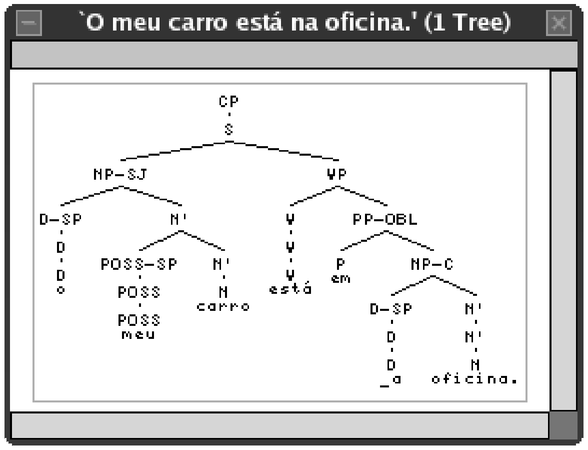
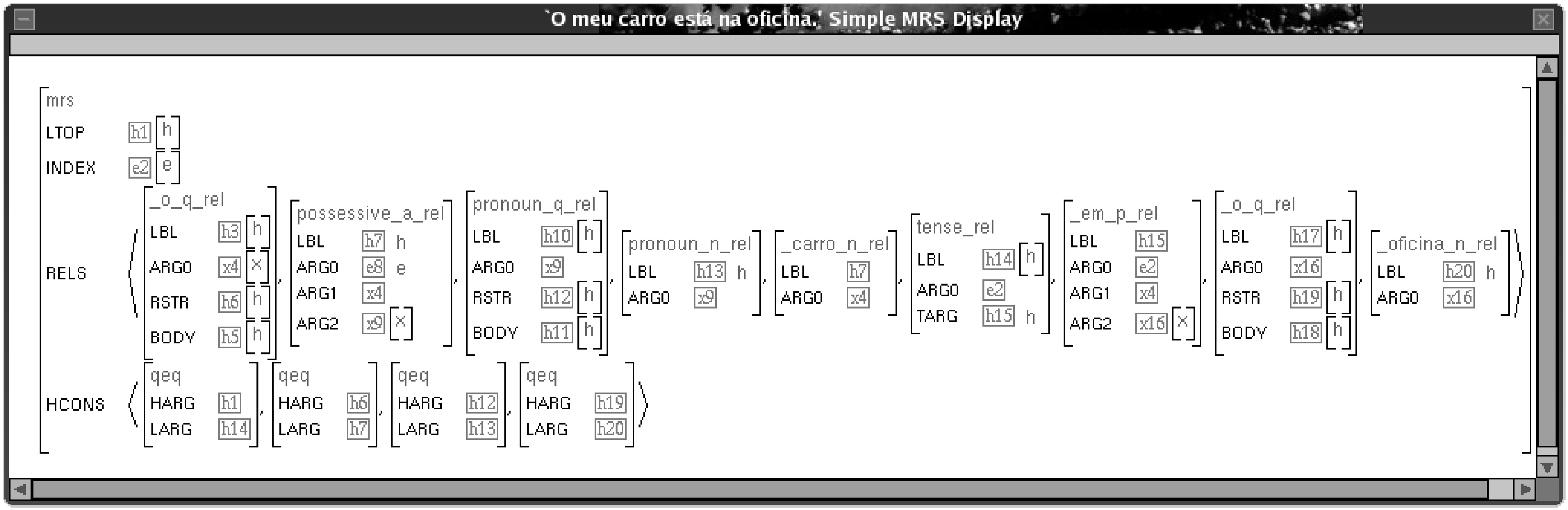


Figure 1.2: Type hierarchy and AVM corresponding to the $\mathscr{TDL}$ code in Figure 1.1. In the type hierarchy, more specific types appear below more general ones.

Unification is denoted by labels starting with a # in $\mathscr{TDL}$ and with boxed integers or letters in AVMs. *Has-a* relations are represented via the symbol "." in $\mathscr{TDL}$ and a vertical bar in AVMs. The $\mathscr{TDL}$ code Attribute1 [Attribute2 type] is equivalent to Attribute1.Attribute2 type. Similarly, the AVM notation $\begin{bmatrix}\text{ATTRIBUTE1}\ \begin{bmatrix}\text{ATTRIBUTE2}\ \textit{type}\end{bmatrix}\end{bmatrix}$ is equivalent to $\begin{bmatrix}\text{ATTRIBUTE1}|\text{ATTRIBUTE2}\ \textit{type}\end{bmatrix}$. Section 2.2 presents a more detailed description of the AVM notation, which will be employed throughout this dissertation.

Figure 1.3 presents a parse window in the LKB for the sentence *o meu carro está na oficina* (*my car is at the mechanic's*). It also shows the semantic representation that LXGram associates with that sentence, using Minimal Recursion Semantics (MRS, see Section 2.3).




`O meu carro está na oficina.' Simple MRS Display

```
mrs
LTOP    h1 [h]
INDEX   e2 [e]
RELS  < [_o_q_rel      LBL h3 [h]   ARG0 x4 [x]   RSTR h6 [h]   BODY h5 [h]],
        [possessive_a_rel   LBL h7 h   ARG0 e8 e   ARG1 x4   ARG2 x9 [x]],
        [pronoun_q_rel   LBL h10 [h]   ARG0 x9   RSTR h12 [h]   BODY h11 [h]],
        [pronoun_n_rel   LBL h13 h   ARG0 x9],
        [_carro_n_rel   LBL h7   ARG0 x4],
        [tense_rel   LBL h14 [h]   ARG0 e2   TARG h15 h],
        [_em_p_rel   LBL h15   ARG0 e2   ARG1 x4   ARG2 x16 [x]],
        [_o_q_rel   LBL h17 [h]   ARG0 x16   RSTR h19 [h]   BODY h18 [h]],
        [_oficina_n_rel   LBL h20 h   ARG0 x16] >
HCONS < [qeq HARG h1  LARG h14],
        [qeq HARG h6  LARG h7],
        [qeq HARG h12 LARG h13],
        [qeq HARG h19 LARG h20] >
```

Figure 1.3: Example parse display and Minimal Recursion Semantics display in the LKB

LXGram is built on top of the LinGO Grammar Matrix (Bender *et al.*, 2002), a starter kit for HPSG grammars in $\mathcal{TDL}$ using MRS. The LinGO Grammar Matrix comes with an interesting hierarchy for many types responsible for basic attributes of the HPSG *sign* (a *sign* is a word or a phrase in HPSG), covering lexical entries, morphological and syntactic rules, composition of semantics and a treatment of basic phrase structure and long distance dependencies (see Section 2.6).

1.5 Organization of the Dissertation

The dissertation proceeds as follows.

Chapter 2 presents a description of how a deep grammar operates and explains the main properties of the formalism that is employed in this dissertation to model Portuguese NP syntax and semantics. We will also describe the format of the semantic representations that are used in LXGram. Additionally, Chapter 2 includes an overview of some of the types and features in the LinGO Grammar Matrix that will be referred to in the rest of the dissertation. They are somewhat different from the HPSG analyses found in the literature in view of the limitations imposed by the systems used.

In Chapter 3 the basic feature geometry that will be employed in the analyses developed in this dissertation is described. We will adopt a specific dialect of HPSG that differs from what is in the LinGO Grammar Matrix, and describe its implementation in LXGram.

Chapter 4 develops an analysis and implementation of a substantial number of elements that can appear in NPs, and constraints among them. We will look at various elements that can make up NPs:

- **predeterminers**

(5) **Todos** os homens são mortais.
all the men are mortal
All men are mortal.

- **determiners**

(6) Todos **os** homens são mortais.
all the men are mortal
All men are mortal.

- **possessives**

(7) a. Chegou a **tua** encomenda.
has arrived the your order
Your order has arrived.

b. Chegou uma encomenda **tua**.
has arrived a order your/yours
An order of yours has arrived.

- **cardinals**

(8) a. Chegaram as **duas** encomendas.
have arrived the two orders
The two orders have arrived.

b. Chegaram **duas** encomendas.
have arrived two orders
Two orders have arrived.

- **ordinals**

(9) O **primeiro** lugar está ocupado.
the first seat is taken
The first seat is taken.

- **elements that mark indefinite specific NPs**

(10) Todos os homens leram um **certo** livro.
all the men read a certain book
All men read a certain book.

- **prenominal and postnominal adjectives**

(11) a. Todos os homens batem num **pobre** burro.
all the men beat on a poor donkey
All men beat a poor donkey.

b. Todos os homens batem num burro **cinzento**.
all the men beat on a donkey gray
All men beat a gray donkey.

- **complements of nouns**

(12) O pai **do Rui** chegou ontem.
the father of the Rui arrived yesterday
Rui's father arrived yesterday.

- **relative clauses**

(13) Todo o homem **que tem um burro** bate-lhe.
every the man who owns a donkey beats it
Every man who owns a donkey beats it.

- **other elements that can follow the noun in an NP**

(14) Era um cão **com três pernas**.
it was a dog with three legs
It was a dog with three legs.

We will discuss word order among these elements, and restrictions on what elements can occur with other elements in the same NP. The formalism presented in Chapter 2 — typed feature structures — will be used to model these phenomena, in Chapter 4.

Chapter 5 focuses on NPs that lack the main noun, as in the following examples (where a "-" marks the place where a noun is expected):

(15) a. [$_{NP}$ Os - que chegarem primeiro] esperam.
the who arrive first wait
The ones who arrive first wait.

b. Há cães com três pernas, mas não há [$_{NP}$ - com cinco].
there are dogs with three legs but not there are with five
There are dogs with three legs, but there are none with five.

We will look at several aspects of these NPs: their semantics, antecedent resolution (in (15b) the noun form *cães* — *dogs* — is recoverable from context), co-occurrence restrictions (not all of the NP elements above can occur in these NPs). We will develop an analysis that can capture the specificities of these constructions by resorting to the material developed in Chapter 4 and combining it in a different way.

Finally, in Chapter 6 a summary of this dissertation is presented. Future work is discussed, and conclusions are drawn.

1.6 Summary

In this chapter, we introduced the main goals and topic of the present dissertation. We discussed the purpose of a deep computational grammar: to encode information regarding natural language details and provide an interface for applications. This interface language is a representation of meaning based on logic. The meaning representation language has a well-known interpretation and supports at least some automated inference. It is also amenable to further processing of other kinds, because it is formal and precise. A deep grammar simply translates between these semantic representations and natural language strings, in both directions, as it is typically written in a declarative language.

We presented LXGram, a computational grammar for Portuguese currently in development, where the material that will be presented in the following Chapters has been implemented.

In this dissertation, we focus on the modeling and implementation of the syntax and semantics of the Portuguese noun phrase in LXGram. This topic is important because noun phrases are very frequent in text, and any natural language processing system must accordingly deal with them. It is also non-trivial, as there are several interactions between the elements that are part of NPs.

Our focus will be the implementation of NPs in a deep computational grammar. We are thus interested in describing precisely the way the various elements that can appear in NPs can combine with each other, their syntax. Furthermore, we are interested in processing natural language expressions (NPs in our case) into representations of their meaning, their semantics.

2 Background

2.1 Overview

In this chapter, we present the framework that is used to model and implement the subject matter of this dissertation, the framework of Head-Driven Phrase Structure Grammar (HPSG). We present the main properties of the formalism used in HPSG.

Afterwards, a description of the format of semantic representations that is employed in LXGram is provided. We start by explaining how semantic representations are produced using the lambda calculus. Next we present some of the difficulties that arise when this method is employed, and present Minimal Recursion Semantics, which has been developed with the purpose of overcoming these difficulties. This is the format employed in LXGram.

We proceed to discuss the properties of the type-feature logic underlying the particular systems used to implement LXGram, which shows some differences with respect to the original HPSG, as in (Pollard and Sag, 1994). This gives a precise meaning to the systems of constraints that are presented in the rest of this dissertation.

LXGram is implemented on top of an open-source module called the LinGO Grammar Matrix. Some parts of the LinGO Grammar Matrix code that LXGram uses are presented, as they will be important for the discussion in the following chapters.

2.2 Head-Driven Phrase Structure Grammar

The linguistic analysis follows the Head-Driven Phrase Structure Grammar (HPSG) framework presented in (Pollard and Sag, 1987), (Pollard and Sag, 1994) and (Sag *et al.*, 2003).

Since its inception, HPSG has been concerned with computational implementations (Sag *et al.*, 2003, p. 537). As a consequence, it has borrowed heavily from computer science. Unsurprisingly, it is well suited to computational implementations, and well known techniques can be used to compile and run HPSG grammars. Among such instruments one finds:

- **Type system** The objects that are used to model the domain are typed. We present an example. All objects representing a syntactic expression are associated with the type *sign*. These expressions can be atomic — words —, of the type *word*, or structured — phrases —, of the type *phrase*. Types are organized in a type hierarchy, that defines *is-a* relations. In this example, the fact that all words (elements of the type *word*) and all phrases (elements of the type *phrase*) are syntactic units (of type *sign*) is captured by positing that the types *word* and *phrase* are subtypes of the type *sign* (in other words, *sign* is a supertype of *word* and *phrase*). This hierarchy can be depicted graphically in the following way:

$$\begin{tikzcd} & \textit{sign} & \\ \textit{word} \arrow[ur] & & \textit{phrase} \arrow[ul] \end{tikzcd}$$

This means that every *word* is a *sign* and every *phrase* is also a *sign*.

Types can be viewed as denoting sets (the set of instances of that type, which we can call its extension or denotation): saying that a certain instance is of the type *sign* is the same as saying that it belongs to the set of *sign*s. Under this perspective, the extension of a subtype of τ, σ, is a subset of the extension of τ, since every instance of σ must be an instance of τ. Since the extension of a type must be a superset of the extensions of all its subtypes, it contains all the elements in the extensions of all its subtypes, i.e. it is a superset of the union of the extensions of its subtypes.

Defining a type hierarchy like the one in our example amounts to saying that the set of *word*s is a subset of the set of *sign*s, and that the set of *phrase*s is also a subset of the set of *sign*s.

Just like set inclusion, the subtype/supertype relation is transitive. Furthermore, there is exactly one type that is a supertype of all types — the top type —, which we will represent with the symbol $\top$ or the string `*top*`.

The term *sort* is also often used for the concept of *type*.

- *Has-a* **relations** We can associate *attributes* (or *features*) to types. For instance, we can postulate that every *sign* has a string representation, and represent this piece of information via the AVM (attribute-value matrix) notation:[1]

$$\begin{bmatrix} \textit{sign} \\ \textsc{orth}\ \textit{string} \end{bmatrix}$$

This AVM means that all elements of the type *sign* have an orthographic representation, which is a string. “String” is also a type, so our mini-hierarchy now looks like this:

$$\begin{tikzcd}
 & \textit{*top*} & & \\
\textit{string} \arrow[ur] & & \textit{sign} \arrow[ul] & \\
 & \textit{word} \arrow[ur] & & \textit{phrase} \arrow[ul]
\end{tikzcd}$$

Sometimes the value of a certain feature cannot be given an intensional definition.

This is the case with our example: the correspondence between semantic/syntactic complexes (instances of *sign*) and their associated surface representations has to be listed (the value of ORTH must be specified for each word), as it cannot be given by rules — it is an arbitrary, or conventional, correspondence. Therefore, the name *sign* is used in HPSG for the supertype of words and phrases, where *sign* means a completely arbitrary and socially agreed upon relation between meaning and form, like a traffic sign. However, in phrases, the relation between meaning and form is arguably predictable from the meaning and form of the elements composing those phrases and the way they are combined (the Principle of Semantic Compositionality, or

[1]In HPSG, this attribute is named PHON. Here we are not concerned with phonological or phonetic representations, since we are processing written text. In this text we will thus use the name ORTH for the orthographic representation of a natural language expression. In several computational implementations it is called STEM (this is also our case), but we will use ORTH here in order to be neutral with respect to morphology. We are also simplifying here in declaring ORTH to be a string instead of a list of strings.

Frege's Principle, a crucial assumption of Natural Language Processing). Ultimately, the need for extensional definitions is the reason why a lexicon is required in deep grammars.

Feature structures can also be depicted as directed graphs. In such representations, a node represents a type and a feature represents an edge from the node representing the type where it is present to the node representing the type that it takes as value.

The following example AVM corresponds to the graph below it:[2]

$$\begin{bmatrix} \textit{sign} & \\ \textsc{orth} & \textit{string} \\ \textsc{dtr} & \begin{bmatrix} \textit{sign} & \\ \textsc{orth} & \textit{string} \end{bmatrix} \end{bmatrix}$$

$$\begin{tikzcd}
\textit{sign} \arrow[r, "\text{ORTH}"] \arrow[dr, "\text{DTR}"'] & \textit{string} & \\
& \textit{sign} \arrow[r, "\text{ORTH}"] & \textit{string}
\end{tikzcd}$$

- **Type Unification** Unification on types can be seen as set intersection of their extensions. The result is completely determined by the type hierarchy, though. For instance, given the following hierarchy:

$$\begin{tikzcd}
& \textit{*top*} & & \\
& \textit{animal} \arrow[u] & & \\
\textit{male-animal} \arrow[ur] & \textit{human} \arrow[u] & \textit{elephant} \arrow[ul] & \textit{female-animal} \arrow[ull] \\
\textit{male-human} \arrow[u] \arrow[ur] & \textit{male-elephant} \arrow[ul] \arrow[ur] & \textit{female-human} \arrow[ul] \arrow[ur] & \textit{female-elephant} \arrow[ul] \arrow[u]
\end{tikzcd}$$

the unification of *elephant* and *male-animal* is *male-elephant*, which can be written as *elephant* $\sqcap$ *male-animal* = *male-elephant*. The result is found by traveling down along the type hierarchy from *elephant* and *male-animal* until a common subtype of the two is found. For instance, according to

[2]This example is not totally well-typed (see Section 2.5).

this hierarchy, all the following propositions hold:

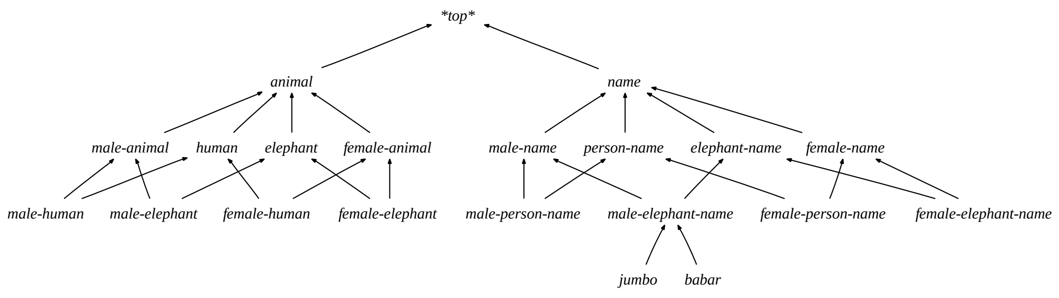
$$\begin{gathered}
\textit{male-animal} \sqcap \textit{human} = \textit{male-human} \\
\textit{female-animal} \sqcap \textit{human} = \textit{female-human} \\
\textit{female-animal} \sqcap \textit{elephant} = \textit{female-elephant} \\
\textit{animal} \sqcap \textit{elephant} = \textit{elephant} \\
\textit{*top*} \sqcap \textit{female-animal} = \textit{female-animal} \\
\textit{human} \sqcap \textit{elephant} = \perp
\end{gathered}$$

In the last example $\perp$ denotes inconsistency (according to this hierarchy, there is no subtype common to *human* and *elephant*, which represents the fact the the intersection of the set of human beings and the set of elephants is empty). The result is unification failure.[3]

- **Multiple Inheritance** A type inherits all features from all its supertypes.

  Suppose that the type *animal* in the hierarchy above is defined as having an attribute NAME of type *name* and this value is further constrained in *elephant* and *male-animal*, using the extended hierarchy below:

$$\begin{bmatrix} \textit{animal} \\ \text{NAME } \textit{name} \end{bmatrix} \begin{bmatrix} \textit{elephant} \\ \text{NAME } \textit{elephant-name} \end{bmatrix} \begin{bmatrix} \textit{male-animal} \\ \text{NAME } \textit{male-name} \end{bmatrix}$$



  These constraints state that all animals have a name, all elephants have an elephant name, and all male animals have a male name.

  These constraints are inherited by all their subtypes. For instance, if no extra constraints are defined for the type *male-elephant*, this type has the inherited constraints:

[3]We use the notation of Copestake (2000) to denote unification, subsumption, the universal unifier and inconsistency:

$\perp$	inconsistency
$\top$	the most general type
$\tau \sqcap \sigma$	the (most general) unifier of τ and σ ($\perp$ if they are incompatible)
$\tau \sqsubseteq \sigma$	σ subsumes τ (σ is more general than τ)

Type hierarchies are displayed with more general types above more specific ones, e.g.:

$$\begin{array}{c} \top \\ | \\ \sigma \\ | \\ \tau \\ | \\ \perp \end{array}$$

It should be noted that it is precisely in the reverse direction of the notation used in Carpenter (1992), another commonly used notation. There, the more general types appear in hierarchies in a position lower than more specific types, with the consequence that there $\perp$ denotes the top type and $\top$ inconsistency, unification is represented by $\sqcup$ and $\tau \sqsubseteq \sigma$ means that τ subsumes σ.

$$\left[\begin{array}{l} \textit{male-elephant} \\ \textsc{name}\ \textit{male-elephant-name} \end{array}\right]$$

because *male-elephant-name* is the unifier for *male-name* (inherited from *male-animal*) and *elephant-name* (inherited from *elephant*).

Inheritance in this type system is actually a case of unification, since the constraints on a type are the result of the unification of all constraints in supertypes and the constraints defined for that type.

Inheritance is monotonic: attributes cannot be redefined in subtypes, but constraints can be added. The added constraints have to be compatible with the inherited constraints: a type has to be unifiable with all of its supertypes.[4]

- **Unification of Feature Structures** The unification of two feature structures A of type a and B of type b is a feature structure C of type c, where c is the unifier of a and b. Additionally, all attributes of A and all attributes of B are present in C and their value is the result of unifying the corresponding attributes of A and B. Also, if there are reentrancies (unified features), they are preserved.

  As an example, suppose we add two features, FATHER to the type *animal* and BEST-FRIEND to the type *human*, and further constrain the inherited FATHER of *human* and *elephant*:[5]

$$\left[\begin{array}{l} \textit{animal} \\ \textsc{father}\ \textit{male-animal} \end{array}\right] \qquad \left[\begin{array}{l} \textit{human} \\ \textsc{father}\ \textit{male-human} \\ \textsc{best-friend}\ \textit{animal} \end{array}\right]$$

  Then the following example illustrates the result of unification of feature structures, where an instance of type *animal* with added constraints (the unification of the features NAME and FATHER|NAME)[6] is unified with an instance of type *male-human*, also with added constraints (the unification of the features FATHER and BEST-FRIEND):[7]

$$\left[\begin{array}{l} \textit{animal} \\ \textsc{name}\ \boxed{1}\ \textit{male-name} \\ \textsc{father}\ \left[\begin{array}{l} \textit{male-animal} \\ \textsc{name}\ \boxed{1} \end{array}\right] \end{array}\right] \sqcap \left[\begin{array}{l} \textit{male-human} \\ \textsc{name}\ \textit{male-name} \\ \textsc{father}\ \boxed{1}\ \textit{male-human} \\ \textsc{best-friend}\ \boxed{1} \end{array}\right] = \left[\begin{array}{l} \textit{male-human} \\ \textsc{name}\ \boxed{1}\ \textit{male-name} \\ \textsc{father}\ \boxed{2}\ \left[\begin{array}{l} \textit{male-human} \\ \textsc{name}\ \boxed{1} \end{array}\right] \\ \textsc{best-friend}\ \boxed{2} \end{array}\right]$$

  The types of the features are often omitted in instances if they are not more specific than the type that that feature is declared to be. For instance, given the above definitions, the following are equivalent:

$$\left[\begin{array}{l} \textit{animal} \\ \textsc{name}\ \boxed{1} \\ \textsc{father}|\textsc{name}\ \boxed{1} \end{array}\right] \qquad \left[\begin{array}{l} \textit{animal} \\ \textsc{name}\ \boxed{1}\ \textit{male-name} \\ \textsc{father}\ \left[\begin{array}{l} \textit{male-animal} \\ \textsc{name}\ \boxed{1} \end{array}\right] \end{array}\right]$$

[4]Some versions of HPSG (e.g. (Sag *et al.*, 2003)) use defaults/non-monotonic inheritance, where constraints can be overridden in subtypes, but we will not use defaults here.

[5]This example is not totally well-typed, as explained in Section 2.5.

[6]The notation A|B is equivalent to A [B].

[7]The features tagged with the same boxed integer point to the same instance (they are unified, reentrant, or structure-shared).

Since FATHER is declared to be of type *male-animal* in the definition of the type *animal*, it can never take a value more abstract than that, so the presence of this value in the example on the right is not informative. By the same token, if FATHER is of the type *male-animal* and *male-animal* is declared to have a feature NAME of type *male-name*, the type of this feature in these instances is also redundant (unless these instances exhibited a more specific value for this feature).

The following token is therefore inconsistent and a type error:

$$\left[\begin{array}{l} \textit{animal} \\ \textsc{name}\ \boxed{1}\,\textit{female-name} \\ \textsc{father}|\textsc{name}\ \boxed{1} \end{array}\right]$$

It describes an animal with a female name that is the same as the name of that animal's father, which is required to be a male name by our system of constraints.

Feature structures with reentrant paths (unified features) correspond to graphs where the edges representing the unified features point to the same node:

$$\left[\begin{array}{ll} \textit{animal} & \\ \textsc{name}\ \boxed{1}\ \textit{male-name} & \\ \textsc{father} & \left[\begin{array}{l} \textit{male-animal} \\ \textsc{name}\ \boxed{1} \end{array}\right] \end{array}\right]$$

$$\begin{tikzcd} \textit{animal} \arrow[rr, bend left, "\text{NAME}"] \arrow[dr, "\text{FATHER}"] & & \textit{male-name} \\ & \textit{male-animal} \arrow[ur, "\text{NAME}"] & \end{tikzcd}$$

Unification therefore represents identity between instances.

- **Polymorphism** An instance can have a feature with a value that is more specific than the type that that feature is declared to be in that instance's type. The following instance of *animal* is valid:

$$\left[\begin{array}{l} \textit{animal} \\ \textsc{name}\ \textit{jumbo} \end{array}\right]$$

Furthermore, a feature F of a type can be more specific than the same feature F in that type's supertypes. The examples constraining the feature NAME in the types *animal*, *male-animal* and *elephant* above illustrate this point.

There are some cosmetic differences with respect to some programming languages that allow polymorphism (like Java). The following instance on the left is actually legal, with the feature NAME instantiated with a type (*name*) more general than the type which is declared to be (*elephant-name*) in that instance's type (*elephant*). However, due to unification with the constraints on its type it is actually equivalent the one on the right (where NAME does not have a type more general that the type of NAME in the definition of *elephant*):

$$\left[\begin{array}{l} \textit{elephant} \\ \textsc{name}\ \textit{name} \end{array}\right] \qquad \left[\begin{array}{l} \textit{elephant} \\ \textsc{name}\ \textit{elephant-name} \end{array}\right]$$

That is, the structure on the left has no type error, but it is in fact more specific (it is as informative as the structure on the right) than what is actually displayed.

The following token on the left is also legal, but given the definitions above, it is equivalent to the structure on the right (because according to the definition of *elephant* above, *elephant*s have *elephant-name*s and, according to the hierarchy, *female-name* and *elephant-name* unify to *female-elephant-name*):

$$\begin{bmatrix} \textit{elephant} & \\ \textsc{name} & \textit{female-name} \end{bmatrix} \begin{bmatrix} \textit{elephant} & \\ \textsc{name} & \textit{female-elephant-name} \end{bmatrix}$$

These differences are cosmetic, because in the LKB the actual values will always be as specific as, or more specific than, the values declared in the types of instances.

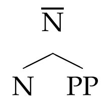
This sort of representation is allowed in the code accepted by the systems used to implement LXGram, but at run time the more specific types are always used.

In Section 2.5 more properties of the specific type system used in the implementation of LXGram and assumed in this dissertation are presented.

Because of these similarities with programming languages, several known techniques can be used. For instance, the type system allows for static type checking: when a grammar is loaded or compiled into a binary format, type errors can be detected. Known algorithms for efficiently computing unification can be used in HPSG processors (Malouf *et al.*, 2000). It is also a declarative formalism, so the same system of constraints (the grammar) can be used both for parsing and generation.

Note that syntactic trees have no theoretical status in HPSG. Trees are merely a visualization device. Because trees are just a special case of graphs, they can be described via feature structures. In the systems employed (LKB and PET), there is a parameter that defines the name of the feature that represents daughters of syntactic nodes. This feature is usually ARGS. Its value is a list. Every element of that list represents a daughter, and the order of the elements in that list represents precedence between these daughters. All other features in the same feature structure describe information about the mother node. As an example, the following feature structure on the left can be abbreviated by the tree on the right.

$$\begin{bmatrix} \textsc{synsem}|\textsc{local}|\textsc{cat} & \begin{bmatrix} \textsc{head} & \boxed{1}\ \textit{noun} \\ \textsc{val}|\textsc{comps} & \langle\rangle \end{bmatrix} \\ \textsc{args} & \left\langle \begin{matrix} \begin{bmatrix} \textsc{synsem}|\textsc{local}|\textsc{cat} & \begin{bmatrix} \textsc{head} & \boxed{1}\ \textit{noun} \\ \textsc{val}|\textsc{comps} & \left\langle \boxed{2} \right\rangle \end{bmatrix} \end{bmatrix}, \\ \begin{bmatrix} \textsc{synsem}\ \boxed{2} & \begin{bmatrix} \textsc{local}|\textsc{cat} & \begin{bmatrix} \textsc{head} & \textit{preposition} \\ \textsc{val}|\textsc{comps} & \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{matrix} \right\rangle \end{bmatrix}$$



In the tree, $\overline{\mathrm{N}}$ represents a node with SYNSEM|LOCAL|CAT|HEAD of type *noun* and SYNSEM|LOCAL |CAT|VAL|COMPS with an empty list as its value, N represents a node with the same HEAD type but a non-empty COMPS, and PP represents a node with a HEAD of type *preposition* and an empty COMPS.

The labels on tree nodes are abbreviations of complex feature structures. They are not types. In the systems used, there are files where one specifies the mapping between feature structures and labels. As mentioned before, trees and labels on tree nodes have no significance (they are not used for computations) — they are just a simplified visualization of feature structures.

As a final remark, syntax rules are merely typed feature structures. They are also organized in a type hierarchy. The leaf types (the ones with no subtypes) are the ones used by the parser and generator (in the LKB and PET they have to be declared in a special file). The more abstract types are a means to encode generalizations across syntactic rules. For instance, the phrase just presented could be produced by a rule with the constraints:

$$\begin{bmatrix} \text{SYNSEM}|\text{LOCAL}|\text{CAT} \begin{bmatrix} \text{HEAD } \boxed{1} \\ \text{VAL}|\text{COMPS } \boxed{2} \end{bmatrix} \\ \text{ARGS} \left\langle \begin{bmatrix} \text{SYNSEM}|\text{LOCAL}|\text{CAT} \begin{bmatrix} \text{HEAD } \boxed{1} \\ \text{VAL}|\text{COMPS} \begin{bmatrix} \text{FIRST} & \boxed{3} \\ \text{REST} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{SYNSEM } \boxed{3} \end{bmatrix} \right\rangle \end{bmatrix}$$

The unification between the HEAD feature of the mother and the HEAD attribute of the head daughter (the Head-Feature Principle), which is the leftmost daughter in this example, occurs in several rules and is an example of a constraint that can be implemented in a general type and inherited by all rules to which it applies.[8]

2.3 Semantic Representations

As mentioned before, a deep grammar's interface for applications is generally the semantic representations it associates with natural language expressions.

MRS — Minimal Recursion Semantics (Copestake *et al.*, 2005) — is employed for the representation of compositional semantics. MRS is not a semantic theory, but rather a way to represent logical formulas.

The standard approach to represent natural language meaning is to translate natural language expressions into logical formulas. This has been done since the work of Montague (1974). This section presents a quick overview of Montague semantics. Dowty *et al.* (1981), Chierchia and McConnell-Ginet (1990), Partee *et al.* (1990), Kamp and Reyle (1993) and de Swart (1998) provide comprehensive introductions to compositional semantics, and Covington (1994) presents its integration in Prolog Definite Clause Grammars.

We cannot present a reasonable description of predicate calculus here, but we show an example of how, despite the limitations, this is useful. For instance, if a grammar translates a sentence like *a black cat ate all my shoes* into:

$$\exists x[black(x) \wedge cat(x) \wedge \forall y[shoe(y) \wedge belonged(y,i) \implies ate(x,y)]]$$

(there is an x such that x is black and x is a cat and for all y, if y is a shoe and y belonged to i, then x ate y) then, assuming it is true, there are ways to manipulate it in order to infer e.g.:

[8]See Section 2.6 for an explanation of the features FIRST and REST in this example.

$\exists x[black(x)]$	(there is at least one black entity)
$\exists x[cat(x)]$	(there is at least one cat)
$\exists x[black(x) \wedge cat(x)]$	(there is at least one black cat)
$\forall x[\neg\exists y[ate(y,x)] \implies \neg[shoe(x) \wedge belonged(x,i)]]$	(everything that was not eaten is not a shoe of mine)

but it is not valid to infer e.g.:

$\exists x[shoe(x) \wedge belonged(x,i)]$ (I had shoes)

and inference can be performed by computers, although in a limited way.

Unary relations, like *black* and *cat* denote sets of entities and binary relations denote sets of ordered pairs of entities. So the first formula describes the following relations: $|BLACK \cap CAT| > 0$ (the intersection of the set of black things and the set of cats has at least one element in it) and $SHOE \cap \{\ x \mid (x,i) \in BELONGED\} \subseteq \{\ y \mid (x,y) \in ATE\ \wedge\ x\ \in BLACK \cap CAT\}$ (the intersection of the set of shoes with the set of all x such that x belonged to i is a subset of the set of all y such that the pair (x,y) is in ATE and x is in the intersection of BLACK and CAT).

With this description we can build a model that satisfies this sentence (i.e. we can imagine a scenario where this sentence is true). Two possible models are the following, where D_e is the set of all entities under consideration:

$$\begin{aligned} D_e &= \{tom, i\} \\ CAT &= \{tom\} \\ BLACK &= \{tom\} \\ SHOE &= \{\} \\ BELONGED &= \{\} \\ ATE &= \{\} \end{aligned} \qquad \begin{aligned} D_e &= \{tom, i, s_1\} \\ CAT &= \{tom\} \\ BLACK &= \{tom\} \\ SHOE &= \{s_1\} \\ BELONGED &= \{(s_1, i)\} \\ ATE &= \{(tom, s_1)\} \end{aligned}$$

The following model does not make our sentence true:

$$\begin{aligned} D_e &= \{tom, i, s_1\} \\ CAT &= \{tom\} \\ BLACK &= \{tom\} \\ SHOE &= \{s_1\} \\ BELONGED &= \{(s_1, i)\} \\ ATE &= \{\} \end{aligned}$$

It is possible to link a grammar to a model builder. However, the simple fact that the semantic representations produced by deep grammars are formal and have a precise meaning makes them amenable to be further processed automatically, in a variety of ways, according to the application.

2.3.1 Expressiveness

There are several limitations. One is that first order logic is not expressive enough to translate all natural language expressions. The classical example is the word *most* in sentences like *most cats meow*. If we assume the meaning of this sentence is $|CAT \cap MEOW| > |CAT|/2$, there is no way to represent it in first order predicate calculus if we do not know the cardinality of the sets involved, or if they are of infinite size (Barwise and Cooper, 1981).

We then need second-order logic.[9] It is customary to use generalized quantifiers, so instead of the universal and existential first order quantifiers (second column in the following examples) or set-

[9] In second order logic, but not in first order logic, relations can themselves be arguments of other relations. Adopting second order logic is not an innocuous decision. First order logic is already undecidable (it is impossible to develop a program that determines the truth value of arbitrary propositions). Second order logic is also incomplete (not all tautologies are theorems; some true propositions cannot be proven, either by a computer or a human).

theoretical expressions (third column), we will employ a variety of generalized quantifiers (fourth column), as in:

All cats meow.	$\forall x[cat(x) \implies meow(x)]$	$CAT \subseteq MEOW$	$all(x, cat(x), meow(x))$
A cat meows.	$\exists x[cat(x) \wedge meow(x)]$	$CAT \cap MEOW \neq \{\}$	$exists(x, cat(x), meow(x))$
Most cats meow.		$\lvert CAT \cap MEOW \rvert > \lvert CAT \rvert / 2$	$most(x, cat(x), meow(x))$

2.3.2 Composition of Meaning

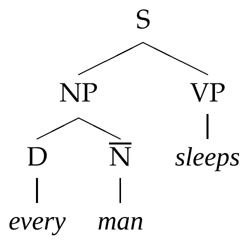
The standard way to compose semantics is to associate lambda expressions to expressions and specify in phrases how they combine.[10] Consider the following small context free grammar (CFG):

$$\begin{array}{lrcl}
(1) & \mathrm{S} & \rightarrow & \mathrm{NP} \quad \mathrm{VP} \\
(2) & \mathrm{NP} & \rightarrow & \mathrm{D} \quad \overline{\mathrm{N}} \\
(a) & \mathrm{D} & \rightarrow & \textit{every} \\
(b) & \mathrm{D} & \rightarrow & \textit{a} \\
(c) & \overline{\mathrm{N}} & \rightarrow & \textit{man} \\
(d) & \overline{\mathrm{N}} & \rightarrow & \textit{cat} \\
(e) & \mathrm{VP} & \rightarrow & \textit{sleeps}
\end{array}$$

where the start symbol is S, the non-terminal symbols are S, NP, VP, D, N, and the terminal symbols are *every, a, man, cat, sleeps*. This grammar generates the strings *every man sleeps, every cat sleeps, a man sleeps* and *a cat sleeps*, by starting with *S* and rewriting symbols according to these rules until only terminal symbols remain (we can replace the symbol on the left-hand side of the rule with all the material on the right-hand side). For instance, the string *every man sleeps* is derived in the following way:

$$\begin{array}{rrrl}
 & & \mathrm{S} & \text{(axiom or start symbol)} \\
 & \mathrm{NP} & \mathrm{VP} & (1) \\
\mathrm{D} & \overline{\mathrm{N}} & \mathrm{VP} & (2) \\
\textit{every} & \overline{\mathrm{N}} & \mathrm{VP} & (a) \\
\textit{every} & \textit{man} & \mathrm{VP} & (c) \\
\textit{every} & \textit{man} & \textit{sleeps} & (e)
\end{array}$$

We can display this derivation via a parse tree, where a node *a* above a node *b* connected to *a* denotes that a rule to rewrite *a* as *b* is used:



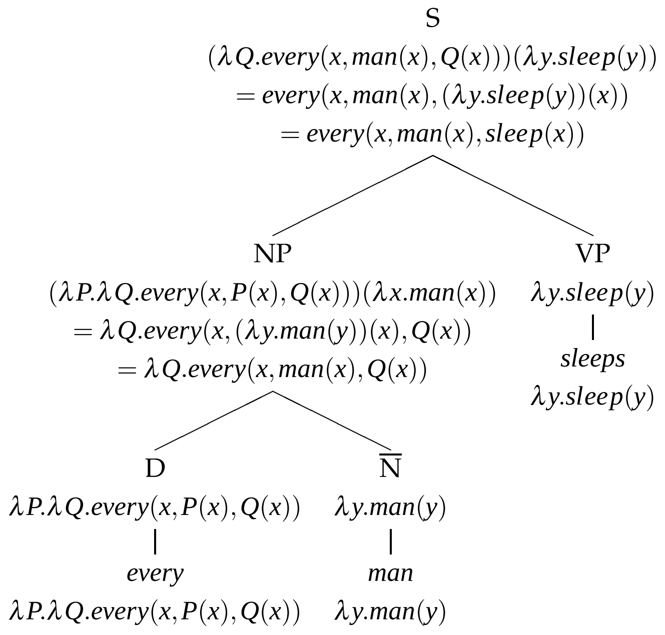
[10] We will not develop on the lambda calculus here. For what follows, it is enough to say that a lambda expression like $\lambda x.P(x)$ denotes a function that given an x returns true if and only if x is in the set $[[P]]$, which is the denotation of the relation P (the characteristic function of the set $[[P]]$ denoted by P). If a and b are two expressions, then ab is the application of a to b (b is the argument of a). For instance, $(\lambda x.P(x))(i)$ is the instantiation of the previous function with the argument i. It is equivalent to $P(i)$: $(\lambda x.P(x))(i)$ reduces to $P(i)$ by removing the leftmost λx and replacing all occurrences of the variable x bound by that operator with the argument i. Therefore, $P(i)$ means $[[i]] \in [[P]]$. Lambda expressions can themselves be arguments of other expressions. For instance, $(\lambda P.\lambda x.Q(x) \vee P(x))(\lambda y.R(y)) = \lambda x.Q(x) \vee (\lambda y.R(y))(x) = \lambda x.Q(x) \vee R(x)$.

We can then associate semantic representations to terminal symbols and to these rules:[11] [12]

(1)	$[[S]]$	$=$	$([[NP]])([[VP]])$	when	S	$\rightarrow$	NP	VP
(2)	$[[NP]]$	$=$	$([[D]])([[\overline{N}]])$	when	NP	$\rightarrow$	D	$\overline{N}$
(a)	$[[every]]$	$=$	$\lambda P_{\in D_{\langle e,t\rangle}}.\lambda Q_{\in D_{\langle e,t\rangle}}.every(x,P(x),Q(x))$					
(b)	$[[a]]$	$=$	$\lambda P_{\in D_{\langle e,t\rangle}}.\lambda Q_{\in D_{\langle e,t\rangle}}.a(x,P(x),Q(x))$					
(c)	$[[man]]$	$=$	$\lambda y_{\in D_e}.man(y)$					
(d)	$[[cat]]$	$=$	$\lambda y_{\in D_e}.cat(y)$					
(e)	$[[sleeps]]$	$=$	$\lambda y_{\in D_e}.sleep(y)$					

The rules that produce terminals keep the semantics of that word.

The parse tree decorated with semantics for *every man sleeps* is:


S
$(\lambda Q.every(x,man(x),Q(x)))(\lambda y.sleep(y))$
$= every(x,man(x),(\lambda y.sleep(y))(x))$
$= every(x,man(x),sleep(x))$
NP
$(\lambda P.\lambda Q.every(x,P(x),Q(x)))(\lambda x.man(x))$
$= \lambda Q.every(x,(\lambda y.man(y))(x),Q(x))$
$= \lambda Q.every(x,man(x),Q(x))$
VP
$\lambda y.sleep(y)$
sleeps
$\lambda y.sleep(y)$
D
$\lambda P.\lambda Q.every(x,P(x),Q(x))$
every
$\lambda P.\lambda Q.every(x,P(x),Q(x))$
$\overline{N}$
$\lambda y.man(y)$
man
$\lambda y.man(y)$

2.3.3 Minimal Recursion Semantics

Minimal Recursion Semantics (MRS — Copestake *et al.* (2005)) is a format of semantic representations that addresses several problems that arise with the composition of semantics. We will mention two key issues in the following subsections and then present a description of MRS.

Quantifier Scope Ambiguity

A problem with natural language semantics is that there is ambiguity concerning quantifier scope. For instance, the sentence *every man read a book* can have the reading $every(x,man(x),a(y,book(y),read(x,y)))$

[11] $[[x]]$ means the denotation of x.

[12] In these examples, D_e denotes the set of all entities under consideration, and $D_{\langle e,t\rangle}$ denotes the set of all sets of entities under consideration. $\lambda y_{\in D_e}.man(y)$ is the characteristic function of the set of men, a function from the set of entities to truth values: given a y that is in D_e it yields true just in case y belongs to the set of men.

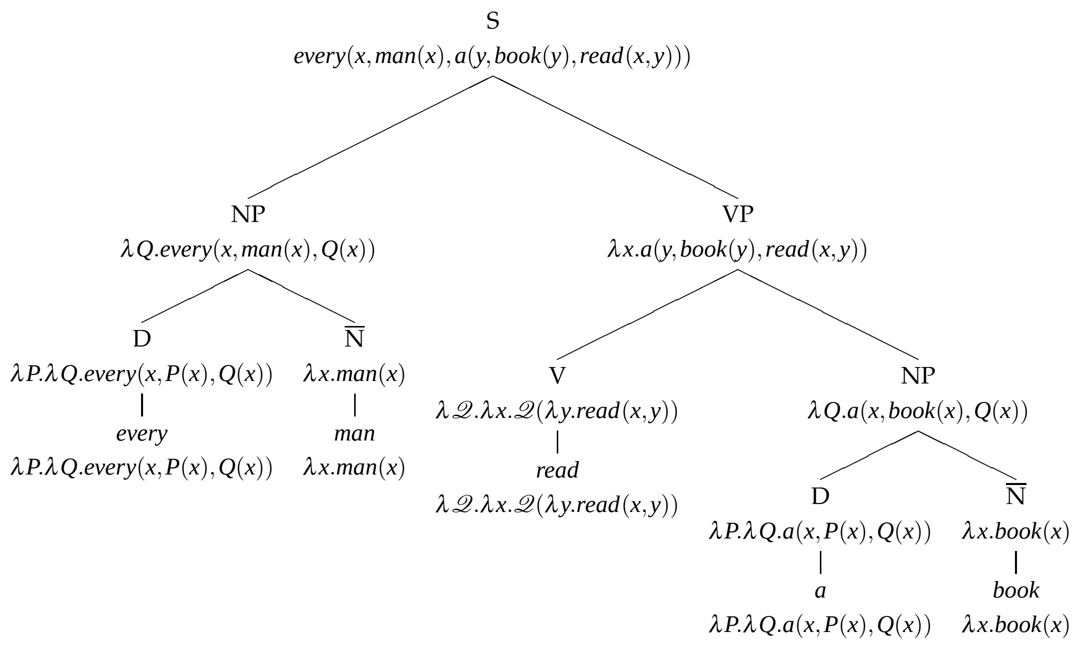
The semantic types we will use are:

- e (an entity);
- t (a truth value);
- if a and b are semantic types, $\langle a,b\rangle$ is also a semantic type, and it is a function from D_a (the set of all elements with the semantic type a) into D_b (the set of all elements with the semantic type b).

as well as $a(y, book(y), every(x, man(x), read(x,y)))$. In the second reading, that there is at least one book that every man has read (it is the same book for all men; this is the meaning in the context: *every man read a book: the Bible*), whereas in the first one, for every man there is at least one book that he has read (but there does not need to be any specific book read by all men).

We cannot account for quantifier scope ambiguity with lambda expressions alone. Consider the following extension to our grammar fragment (rule (3) and the lexical entries (f-g) were added) and the parse tree it generates below for *every man read a book*:[13]

(1)	S	$\rightarrow$	NP VP	[[S]]	=	([[NP]])([[VP]])
(2)	NP	$\rightarrow$	D $\overline{\text{N}}$	[[NP]]	=	([[D]])([[$\overline{\text{N}}$]])
(3)	VP	$\rightarrow$	V NP	[[VP]]	=	([[V]])([[NP]])
(a)	D	$\rightarrow$	*every*	[[*every*]]	=	$\lambda P_{\in D_{\langle e,t\rangle}}.\lambda Q_{\in D_{\langle e,t\rangle}}.every(x, P(x), Q(x))$
(b)	D	$\rightarrow$	*a*	[[*a*]]	=	$\lambda P_{\in D_{\langle e,t\rangle}}.\lambda Q_{\in D_{\langle e,t\rangle}}.a(x, P(x), Q(x))$
(c)	$\overline{\text{N}}$	$\rightarrow$	*man*	[[*man*]]	=	$\lambda x_{\in D_e}.man(x)$
(d)	$\overline{\text{N}}$	$\rightarrow$	*cat*	[[*cat*]]	=	$\lambda x_{\in D_e}.cat(x)$
(e)	VP	$\rightarrow$	*sleeps*	[[*sleeps*]]	=	$\lambda x_{\in D_e}.sleep(x)$
(f)	$\overline{\text{N}}$	$\rightarrow$	*book*	[[*book*]]	=	$\lambda x_{\in D_e}.book(x)$
(g)	V	$\rightarrow$	*read*	[[*read*]]	=	$\lambda \mathcal{Q}_{\in D_{\langle\langle e,t\rangle,t\rangle}}.\lambda x_{\in D_e}.\mathcal{Q}(\lambda y_{\in D_e}.read(x,y))$


S
$every(x, man(x), a(y, book(y), read(x,y)))$
NP
$\lambda Q.every(x, man(x), Q(x))$
VP
$\lambda x.a(y, book(y), read(x,y))$
D
$\lambda P.\lambda Q.every(x, P(x), Q(x))$
every
$\lambda P.\lambda Q.every(x, P(x), Q(x))$
$\overline{\text{N}}$
$\lambda x.man(x)$
man
$\lambda x.man(x)$
V
$\lambda \mathcal{Q}.\lambda x.\mathcal{Q}(\lambda y.read(x,y))$
read
$\lambda \mathcal{Q}.\lambda x.\mathcal{Q}(\lambda y.read(x,y))$
NP
$\lambda Q.a(x, book(x), Q(x))$
D
$\lambda P.\lambda Q.a(x, P(x), Q(x))$
a
$\lambda P.\lambda Q.a(x, P(x), Q(x))$
$\overline{\text{N}}$
$\lambda x.book(x)$
book
$\lambda x.book(x)$

This example illustrates that we can only derive one reading. Several methods have been proposed in the literature to address this problem.

Logical Equivalence

Another problem surfaces with the need to compute logical equivalence. This problem occurs frequently when semantic formulas involve conjunction. Consider the following grammar fragments for English and Portuguese (we added rule (4) to the previous CFG for English, as well as the terminal symbol in (h); the CFG for Portuguese is parallel to this extension):[14]

[13]Here $D_{\langle\langle e,t\rangle,t\rangle}$ denotes the set of all sets of sets of entities (the denotation of an NP is a set of sets of entities).

[14]Rule (4) in the Portuguese fragment exemplifies the need to specify the composition of semantics for each rule: the semantic functor and argument are not predictable from word order.

(1)	S	$\rightarrow$	NP VP	[[S]]	=	([[NP]])([[VP]])
(2)	NP	$\rightarrow$	D $\overline{\text{N}}$	[[NP]]	=	([[D]])([[$\overline{\text{N}}$]])
(4)	$\overline{\text{N}}$	$\rightarrow$	AP $\overline{\text{N}}$	[[$\overline{\text{N}}$]]	=	([[AP]])([[$\overline{\text{N}}$]])
(b)	D	$\rightarrow$	*a*	[[*a*]]	=	$\lambda P.\lambda Q.a(x,P(x),Q(x))$
(d)	$\overline{\text{N}}$	$\rightarrow$	*cat*	[[*cat*]]	=	$\lambda x.cat(x)$
(e)	V	$\rightarrow$	*sleeps*	[[*sleeps*]]	=	$\lambda x.sleep(x)$
(h)	AP	$\rightarrow$	*black*	[[*black*]]	=	$\lambda P.\lambda x.black(x) \wedge P(x)$

(1)	S	$\rightarrow$	NP VP	[[S]]	=	([[NP]])([[VP]])	
(2)	NP	$\rightarrow$	D $\overline{\text{N}}$	[[NP]]	=	([[D]])([[$\overline{\text{N}}$]])	
(4)	$\overline{\text{N}}$	$\rightarrow$	$\overline{\text{N}}$ AP	[[$\overline{\text{N}}$]]	=	([[AP]])([[$\overline{\text{N}}$]])	
(b)	D	$\rightarrow$	*um*	[[*um*]]	=	$\lambda P.\lambda Q.um(x,P(x),Q(x))$	"a"
(d)	$\overline{\text{N}}$	$\rightarrow$	*gato*	[[*gato*]]	=	$\lambda x.gato(x)$	"cat"
(e)	V	$\rightarrow$	*dorme*	[[*dorme*]]	=	$\lambda x.dormir(x)$	"sleep"
(h)	AP	$\rightarrow$	*preto*	[[*preto*]]	=	$\lambda P.\lambda x.P(x) \wedge preto(x)$	"black"

Because the semantic representations for adjectives happened to be stated in a different way in the two grammars (they follow word order, which is different, as depicted in rule (3)), the sentences *a black cat sleeps* and *um gato preto dorme* receive slightly different formulas as translations: $a(x, black(x) \wedge cat(x), sleep(x))$ vs. $um(x, gato(x) \wedge preto(x), dormir(x))$. This is a problem for machine translation based on semantic representations. The problem can be solved by changing one of the grammars. For instance, the Portuguese fragment could give the adjective *preto* the meaning $\lambda P.\lambda x.preto(x) \wedge P(x)$. But this means that the two grammars cannot be developed independently of each other. In practice, it is very difficult to co-ordinate efforts in this way, specially when multiple grammars/languages are involved.

The two formulas are logically equivalent (by commutativity: $P \wedge Q \equiv Q \wedge P$). But note that logical equivalence is in the general case undecidable even for first order predicate calculus (Schieber, 1993).

However, it may be necessary to accommodate logical equivalence, at least in a limited way. Consider the following example. The English grammar could be made more complex in order to allow for *big black cat* but block *black big cat*. The Portuguese grammar would probably want to allow both *gato preto grande* and *gato grande preto*. The last expression would yield $\lambda x.(gato(x) \wedge grande(x)) \wedge preto(x)$ (because the rule (4) in the Portuguese CFG only licenses the syntactic structure $[_{\overline{\text{N}}}\ [_{\overline{\text{N}}}\ \text{gato}_{\overline{\text{N}}}\ \text{grande}_{\text{AP}}\ ]\ \text{preto}_{\text{AP}}\ ]$), which translates to $\lambda x.(cat(x) \wedge big(x)) \wedge black(x)$. However, if an extension of the English grammar blocks *black big cat*, it would only be able to generate from $\lambda x.big(x) \wedge (black(x) \wedge cat(x))$ (the rule (4) in the English CFG licenses the structure $[_{\overline{\text{N}}}\ \text{big}_{\text{AP}}\ [_{\overline{\text{N}}}\ \text{black}_{\text{AP}}\ \text{cat}_{\overline{\text{N}}}\ ]\ ]$). This requires computing associativity of conjunction ($(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$), as well as commutativity.

Brief Description of Minimal Recursion Semantics

MRS addresses these two issues: quantifier scope ambiguity and the necessity to abstract from logical variants given by commutativity and associativity of conjunction.

MRS allows for the underspecification of scope: a single MRS representation can correspond to multiple logical formulas that differ in the relative scope among quantifiers. This is desirable because in several applications scope does not need to be resolved.

For instance, in machine translation, quantifier scope ambiguities can generally be preserved in the translations that are to be produced.

Furthermore, the number of readings is exponential on the number of scope bearing elements (Muskens, 1995): quantifiers or other elements, like modals and negation. Therefore, for the sake of efficiency, it is preferable to delay or avoid resolving scope whenever possible.

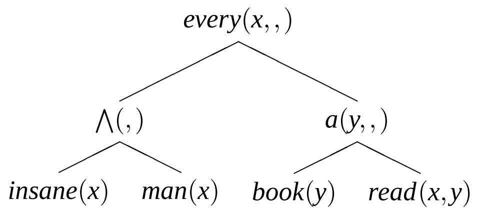
In MRS, conjunction is represented as a bag/multi-set of relations with the same labels (in MRS relations are called elementary predications), which avoids computing associativity and commutativity variants of conjunction.

Here we provide a simplified and informal description of MRS.

Consider the following two semantic representations for the English sentence *All insane men read a book*:[15]

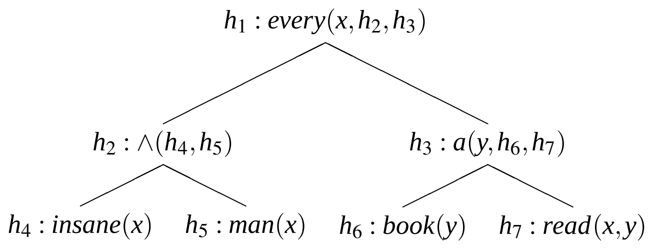
- $every(x, insane(x) \wedge man(x), a(y, book(y), read(x,y)))$
- $a(y, book(y), every(x, insane(x) \wedge man(x), read(x,y)))$

To these formulas, we can associate syntactic trees that describe scope relations between their various non-atomic expressions. For instance, the first formula receives the following tree:



Here we are representing $insane(x) \wedge man(x)$ as $\bigwedge(insane(x), man(x))$. In MRS, conjunction is not necessarily binary. Instead, $\bigwedge$ is generalized conjunction: the number of arguments is arbitrary. The fact that the node $insane(x)$ is the leftmost daughter of the node with $\bigwedge(,)$ denotes the fact that $insane(x)$ is the first argument of $\bigwedge(,)$.

We can decorate all nodes in this tree with labels. In MRS, these labels are called handles. Every node is given a different handle with a name of the form h_n, where $n \geq 0$,[16] and prefixed with its handle and a ":" separating the handle from the formula. Labels can then be used in argument slots to indicate argument positions:

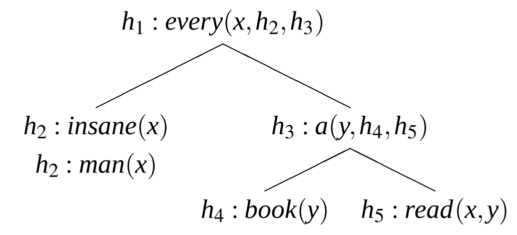


With every sub-formula labeled with a handle and every argument position filled with a variable or a handle, the tree representation is not necessary. We can represent all this information with a pair. The second member of this pair is a bag/multi-set of elementary predications (an elementary predication is a relation and its associated handle). The first member of this pair is the handle that labels the root node (called the global top). The formula under consideration can thus be represented by $\langle h1$, $\{h_1 : every(x,h_2,h_3)$, $h_2 : \wedge(h_4,h_5)$, $h_3 : a(y,h_6,h_7)$, $h_4 : insane(x)$, $h_5 : man(x)$, $h_6 : book(y)$, $h_7 : read(x,y)\} \rangle$.

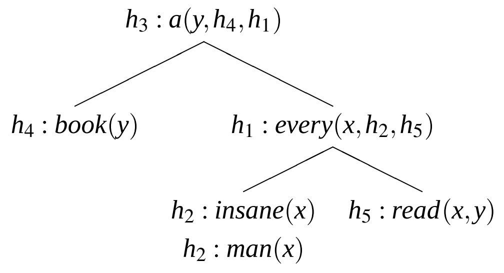
We can simplify this representation by adopting the convention that the arguments of the same conjunction are given identical handles. The new MRS is now $\langle h1$, $\{h_1 : every(x,h_2,h_3)$, $h_2 : insane(x)$, $h_2 : man(x)$, $h_3 : a(y,h_4,h_5)$, $h_4 : book(y)$, $h_5 : read(x,y)\} \rangle$. It corresponds to the following tree:

[15]The generalized quantifier $\lambda P.\lambda Q.every(x, P(x), Q(x))$ is intended to mean $P \subseteq P$, and $\lambda P.\lambda Q.a(x, P(x), Q(x))$ is intended to mean $|P \cap Q| > 0$.

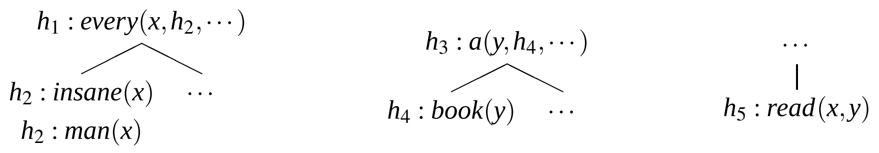
[16]It is usual to start at 1, though.



We still have different MRSs for different quantifier scope possibilities. Consider the MRS for the second reading of the same sentence, depicted graphically as a tree:



These two trees have the following tree fragments in common:



In order to describe these two scope possibilities with a single MRS we need to allow the situation in which handles with different names refer to the same syntactic node. The MRS for this sentence (contemplating both readings) is now $\langle h_0, \{ h_1 : every(x, h_2, h_6), h_2 : insane(x), h_2 : man(x), h_3 : a(y, h_4, h_7), h_4 : book(y), h_5 : read(x,y) \} \rangle$. The first reading obtains when $h_0 = h_1$, $h_6 = h_3$ and $h_7 = h_5$. The second one corresponds to the situation where $h_0 = h_3$, $h_6 = h_5$ and $h_7 = h_1$.

Note that every "hole" must be filled. For instance, it is not possible to set $h_0 = h_1$, $h_2 = h_3$ and $h_4 = h_5$, as this would leave the quantifiers with no body (their last argument, also called nuclear scope).

In order to account for some more interesting examples of quantifier scope ambiguity, MRS resorts to the concept of a *qeq* relation between handles.[17] If h_i and h_j are in a *qeq* relation, written $h_i =_q h_j$, one of the following must hold in all trees represented by a given MRS: (1) h_i and h_j are the same handle; (2) they are different handles, but all nodes in the path between the nodes they label contain quantifier relations, and the lowest handle is in the body of those quantifiers. An MRS is now a triple: the top handle, a multi-set of labeled relations and a set of handle constraints (*qeq* relations).[18]

The previous example can be written as $\langle h_0, \{ h_1 : every(x, h_2, h_6), h_2 : insane(x), h_2 : man(x), h_3 : a(y, h_4, h_7), h_4 : book(y), h_5 : read(x,y) \}, \{h_0 =_q h_5\} \rangle$. That is, the global top, h_0, and the handle that labels the relation introduced by the verb, h_5, are in a *qeq* relation. This makes it explicit that that relation cannot occur in the restrictor of a quantifier.

In order to further illustrate the need for *qeq* constraints, consider the English sentence *Every son of a lion roars* and its semantic representations:

[17] *Qeq* stands for "equality modulo quantifiers".

[18] There are other kinds of relations on handles, but we will not use them in this dissertation, as they are not supported by the systems where LXGram runs.

- $every(x, a(y, lion(y), son(x,y)), roar(x))$
- $a(y, lion(y), every(x, son(x,y), roar(x)))$

The MRS translations for the quantifiers, as presented so far, do not allow for the underspecification of quantifier scope in examples like this one, because the handles on the restrictors of the quantifiers (their first argument if we do not count the bound variable as an argument) are fixed.

We want to say that the restrictor of *every* "includes" *son*, but a quantifier may intervene, as in the first reading, in which case *son* is in the body of the intervening quantifier. The following MRS, with *qeq* constraints between quantifiers and their restrictors, fulfills this requirement: $\langle h_0$, $\{h_1 : every(x,h_2,h_3)$, $h_4 : son(x,y)$, $h_5 : a(y,h_6,h_7)$, $h_8 : lion(y)$, $h_9 : roar(x)\}$, $\{h_0 =_q h_9, h_2 =_q h_4, h_6 =_q h_8\}\rangle$.

The first reading is obtained when $h_0 = h_1, h_2 = h_5, h_6 = h_8, h_7 = h_4, h_3 = h_9$. The second one corresponds to the equalities $h_0 = h_5, h_6 = h_8, h_7 = h_1, h_2 = h_4, h_3 = h_9$. The relation corresponding to the verb, *roar*, can never appear in the restrictor of a quantifier because its handle is not in a *qeq* relation with a handle filling that argument and because *roar* is not a quantifier relation. It will always be in the body of the most embedded quantifier. Also, *lion* cannot be directly in the restrictor position of *every*, because it is not a quantifier relation, and the handle labeling *lion* is not in a *qeq* relation with the handle filling the restrictor of *every*. The relation *lion* can be in the restrictor position of the quantifier *a*, via the equation $h_6 = h_8$. *Mutatis mutandis*, the same is true of *son* and the quantifier *a*.

Note that the formula $every(x, son(x,y), a(y, lion(y), roar(x)))$ is blocked simply because the variable y occurs free in the restrictor of *every*.

In MRS, every quantifier relation contains a *qeq* constraint linking that quantifier with its restrictor. There is no *qeq* constraint between a quantifier and its body. There is always a *qeq* constraint between the global top and the handle of the relation corresponding to the main verb of the sentence associated to the MRS.

For further details on MRS and a more formal description of this format of describing semantic representations, see (Copestake *et al.*, 2005).

An example MRS, in the AVM notation, is in Figure 2.1. It is the MRS produced by LXGram for the Portuguese sentence:

(16)

Todos	os	homens	batem	num	burro	infeliz.
all	the	men	beat	on a	donkey	unhappy

All men beat an unhappy donkey.

$$\begin{bmatrix} \textit{mrs} & \\ \text{LTOP} & \boxed{h1}\ h \\ \text{INDEX} & \boxed{e2}\ e \\ \text{RELS} & \left\langle \begin{bmatrix} \textit{\_todo\_q\_rel} & \\ \text{LBL} & \boxed{h3}\ h \\ \text{ARG0} & \boxed{x6}\ x \\ \text{RSTR} & \boxed{h5}\ h \\ \text{BODY} & \boxed{h4}\ h \end{bmatrix}, \begin{bmatrix} \textit{\_homem\_n\_rel} & \\ \text{LBL} & \boxed{h7}\ h \\ \text{ARG0} & \boxed{x6} \end{bmatrix}, \begin{bmatrix} \textit{\_bater\_v\_-em-\_rel} & \\ \text{LBL} & \boxed{h8} \\ \text{ARG0} & \boxed{e2} \\ \text{ARG1} & \boxed{x6} \\ \text{ARG2} & \boxed{x9}\ x \end{bmatrix}, \begin{bmatrix} \textit{\_um\_q\_rel} & \\ \text{LBL} & \boxed{h10}\ h \\ \text{ARG0} & \boxed{x9} \\ \text{RSTR} & \boxed{h12}\ h \\ \text{BODY} & \boxed{h11}\ h \end{bmatrix}, \begin{bmatrix} \textit{\_burro\_n\_rel} & \\ \text{LBL} & \boxed{h13}\ h \\ \text{ARG0} & \boxed{x9} \end{bmatrix}, \begin{bmatrix} \textit{\_infeliz\_a\_rel} & \\ \text{LBL} & \boxed{h13} \\ \text{ARG0} & \boxed{e14}\ e \\ \text{ARG1} & \boxed{x9} \end{bmatrix} \right\rangle \\ \text{HCONS} & \left\langle \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h1} \\ \text{LARG} & \boxed{h8} \end{bmatrix}, \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h5} \\ \text{LARG} & \boxed{h7} \end{bmatrix}, \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h12} \\ \text{LARG} & \boxed{h13} \end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 2.1: Example MRS. The sentence is *todos os homens batem num burro infeliz* (*all men beat an unhappy donkey*).

This MRS describes the two propositions:

$$\begin{array}{l}
\_um\_q(x9, \_burro\_n(x9) \wedge \_infeliz\_a(e14, x9), \\
\quad\quad \_todo\_q(x6, \_homem\_n(x6), \\
\quad\quad\quad\quad \_bater\_v\_\text{-}em\text{-}(e2, x6, x9)))
\end{array}$$

$$\begin{array}{l}
\_todo\_q(x6, \_homem\_n(x6), \\
\quad\quad \_um\_q(x9, \_burro\_n(x9) \wedge \_infeliz\_a(e14, x9), \\
\quad\quad\quad\quad \_bater\_v\_\text{-}em\text{-}(e2, x6, x9)))
\end{array}$$

This example follows the conventions for the names of predicates in MRS. Predicate names start with an underscore if they are related to a lexical item. They include several fields separated by underscores: the lexical item's lemma, a single character denoting its part-of-speech (n for nouns, v for verbs, etc.: e.g. this allows for a straightforward differentiation between the relations associated with the English noun *fly* and the verb *fly*, which are radically different concepts), an optional field for word sense distinctions that can include prepositions of subcategorized complements (in LXGram we put hyphens around prepositions, in order to easily distinguish between the part-of-speech field with the value a, for adjectives, and the preposition *a*), and the suffix rel, which is omitted in scope-resolved formulas. In MRS, variables over entities start with an *x* followed by an integer.

The *_um_q* relation is meant to express the existential generalized quantifier, and *_todo_q* the universal generalized quantifier. The two readings correspond to the two different scope possibilities between the two quantifiers, i.e. in the first reading there is a single donkey beaten by all men, whereas the second reading can be true if different men beat different donkeys.

In the AVM, the feature LTOP denotes the global top, the feature RELS is the bag of elementary predications, and HCONS is the set of handle constraints. RELS and HCONS are implemented as difference lists, but the LKB MRS display presents them as lists. In the feature structures for relations, LBL is the associated handle, RSTR is the restrictor argument of a quantifier, and BODY is its nuclear scope. The remaining arguments are given by ARG*n*, with $n \geq 0$. In the feature structures for *qeq* constraints, HARG is the left operand (the higher handle), and LARG is the second operand (the lower handle).

A note on MRSs concerns the standard practice of using event arguments. For instance, the relation associated with the verb in the previous example has as its first argument an event represented by $e2$ in that example. Event variables are considered to be implicitly existentially quantified. Its use is inspired in the work of Davidson (see e.g. (Davidson, 1980)). Event variables are useful in order to represent intersective modifiers of verbs. For instance, the verb *chegar* (*arrive*) is given semantics equivalent to $\lambda x2.\_chegar\_v(e1, x2)$. The phrase *chegar hoje* (*to arrive today*) is associated with the semantics $\lambda x2.\_chegar\_v(e1, x2) \wedge \_hoje\_a(e3, e1)$, which is supposed to mean that there is an event of arriving and that event takes place during the time interval denoted by the word *hoje* (*today*). The event associated with the verb (its first argument) provides an argument for the relation associated with the adverb in this example. Events should not be confused with the semantic type *e* (for entities). In MRSs, variables over entities appear with the type *x*, and events with the type *e*.

The MRS display provided by the LKB (similar to the MRS in Figure 2.1) presents relation names as if they were types of relations. In the feature structures manipulated by the grammar, they are not types of relations, but rather the values of a feature PRED present in all relations and omitted in these MRS displays. This feature PRED takes strings as values. For instance, the feature structure for the relation *_burro_n_rel* manipulated by the grammar is actually:

$$\begin{bmatrix} \textit{noun-relation} & \\ \text{LBL} & \textit{handle} \\ \text{PRED} & \text{“\_burro\_n\_rel”} \\ \text{ARG0} & \textit{ref-index} \end{bmatrix}$$

where *noun-relation* is a type for which the features just presented are appropriate, and where they are constrained with the types displayed above.

Similarly, the type names that appear in the MRS displays produced by the LKB are also cosmetic. The type for variables over entities is *x* in MRSs but the real type name in LXGram is *ref-index* (for referential index). The type for handles is *handle* but appears in MRS representations as *h*, and *event* appears as *e*. The mapping between the grammar types and the MRS types is specified in a configuration file.

Since in MRS conjunction is denoted by two or more elementary predications having the same label, the computation of commutativity and associativity of conjunction is reduced to testing for multi-set equality. For instance, a grammar for English would produce an MRS for the corresponding English sentence where the relations corresponding to *_burro_n* and *_infeliz_a* would appear in the reverse order (the MRSs become more readable if they follow word order). The two MRSs, the one presented and the hypothetical MRS for the corresponding English sentence, would nonetheless be equivalent modulo relation names, but in order to match them it is not necessary to produce logical equivalents, but only to compare multi-sets. Note that since conjunction is not represented by a binary operator, associativity is not even an issue: the MRS representation of conjunction is generalized conjunction.

The LKB comes with the functionality of generating scope resolved formulas from MRS representations, under request. The LKB also contains very efficient algorithms for the generation from MRS representations (Carroll *et al.*, 1999; Carroll and Oepen, 2005).

MRS is not the only format of semantic description that allows underspecification of quantifier scope. For instance, Underspecified Discourse Representation Theory (Reyle, 1993) and Hole Semantics (Bos, 1996) explore the same idea. Other approaches include those of Alshawi and Crouch (1992), Egg *et al.* (2001) and Poesio (1994), among others.

2.4 Strong Lexicalism

HPSG is heavily lexical. That is, a large amount of information is encoded in the feature structures associated with the words in the lexicon. Syntactic rules are very general and just combine words according to the lexical stipulations associated with them.

In LXGram a typical lexical entry consists of an identifier for that entry, a lexical type, a string that describes the orthographic form of that item, and the value of the feature PRED, which encodes the name of the semantic relation for that item.

All the constraints associated with that item are given by the lexical type. Lexical types are, like all other types, organized in an inheritance hierarchy.

The type hierarchy for lexical types is, like in the other computational HPSGs, extremely large (with hundreds or thousands of types), since the lexical leaf types (the ones used in the lexical entries) must be cross-classified in several dimensions, taking advantage of multiple inheritance. For this reason, in this text we will not present the names of the lexical types used and just present the constraints associated with these types. Some of these constraints will appear in several items. In LXGram, constraints common to many items are usually defined in abstract lexical types from which many lexical types inherit.

For instance, the lexical entry for the noun *carro* (*car*) in LXGram looks like the following:

```
carro :=
 noun-common-masc-count-0comps-lex &
 [ STEM < "carro" >,
   SYNSEM.LOCAL.CONT.KEYS.KEY.PRED "_carro_n_rel" ].
```

The lexical type contains the constraints that define this noun as a common noun with masculine grammatical gender that is a count noun and has no complements. This type also defines the SYNSEM |LOCAL|CONT|RELS attribute, where the bag of relations associated with this item is defined, to be singleton, and unifies its single element with SYNSEM|LOCAL|CONT|KEYS|KEY. This is the path used in lexical entries to access the relation associated to a word, as in this example. STEM is the feature that has been presented above as ORTH.

2.5 Some Properties of the Formalism in the LKB and PET

Here we present some properties of the logic of typed feature structures assumed by the LKB and PET, focusing on those that are relevant to the subsequent discussion in this dissertation.

- **Open-world semantics** The LKB and PET follow an open-world semantics, because not all the conditions required by a closed-world semantics are satisfied.

  Under a closed-world semantics, the objects that can interpret feature structures are completely defined by the type hierarchy. A logic of typed feature structures is considered to have a closed-world semantics if it satisfies the two following properties (in the systems used here the second one is enforced, but not the first one).

  - **The partition condition** Consider the example of t having exactly two subtypes s and u:

$$\begin{tikzcd} & t & \\ s \arrow[ur] & & u \arrow[ul] \end{tikzcd}$$

    The partition condition says that all objects of type t are either of type s or type u.[19] Consider the case where the features F and G are appropriate for type t and boolean,[20] and constrained in s and u in the following way:

$$\begin{bmatrix} t \\ F\ bool \\ G\ bool \end{bmatrix} \begin{bmatrix} s \\ F\ + \\ G\ - \end{bmatrix} \begin{bmatrix} u \\ F\ - \\ G\ + \end{bmatrix}$$

    There are two consequences. The first is that the following instance is illicit, because it is incompatible with both s and u.

$$\begin{bmatrix} t \\ F\ + \\ G\ + \end{bmatrix}$$

[19]Obviously, it generalizes to any number of subtypes.

[20]As in the LinGO Grammar Matrix, we use + and - as the only subtypes of *bool*, where the first denotes truth and the second falsity. See Section 2.6.

This interpretation of the type hierarchy is sometimes called exhaustive typing.

The second consequence is that e.g. an instance of type t with F of type $+$ is inferred to be of type s, since it is incompatible with u. The kind of inference whereby an object of a type is automatically "demoted" to one of its subtypes in virtue of having constraints incompatible with all other subtypes is sometimes called subtype covering (e.g. in (Melnik, 2005)).

Recall that the extension of a type is a superset of the union of the extensions of all its subtypes. The partition condition means that we consider that the extension of a type is exactly this union (it is a superset, but not a proper superset of that union): it cannot contain elements that are not in the extension of one of its subtypes.[21]

This partition condition is not respected in the LKB or PET: neither exhaustive typing nor subtype covering are performed.

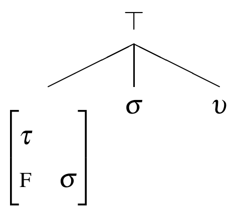
– **The disjoint species condition** If two types τ and σ do not have a common subtype υ (such that $\upsilon \sqsubseteq \tau \wedge \upsilon \sqsubseteq \sigma$, and υ is not required to be different from τ and σ), then $\tau \sqcap \sigma = \bot$, i.e. nothing can be simultaneously of type τ and of type σ. The result of unification is completely determined by the inheritance hierarchy, and unification results in failure if its two operands do not have a common subtype defined there.

  The disjoint species condition means that the extensions of species (species are types that have no subtypes) are all disjoint.

- **Total well-typedness** The LKB and PET require the feature structures that are manipulated to be totally well-typed in the sense of Carpenter (1992). A feature structure is totally well-typed if it is well-typed and satisfies additional properties.

  A feature structure s of type τ is well-typed if all features under s are defined for τ and their value in s is subsumed by the value declared in τ for those features.

  For instance, given the following hierarchy and constraints:



  the following instances are not well-typed:

$$\begin{bmatrix} \tau & \\ \text{F} & \upsilon \end{bmatrix}\begin{bmatrix} \tau & \\ \text{G} & \top \end{bmatrix}$$

  The instance on the left is not well-typed because it presents a value for the feature F, υ, that is incompatible with the type that this feature is constrained to be of in the definition of τ, σ (according to the hierarchy, there is no unifier for σ and υ, so the structure cannot be made well-typed).[22] The structure on the right is also not well-typed, since it has a feature G that is

[21]Drawing a parallel with object-oriented programming languages, it means that all our types that have subtypes are like abstract classes or interfaces there.

[22]If there was an additional type, say π, defined as a subtype of both σ and υ, this instance would be well-typed and equivalent to one where F is of type π, as π would be the unifier of υ and σ.

not declared in τ to be appropriate for instances of this type.[23]

Furthermore, the constraints on a type have to be compatible with the constraints on all their supertypes. Resuming our example, a subtype ρ of τ cannot redefine the feature F to be of the type υ, as υ and σ are incompatible.[24]

A feature structure s of type τ is totally well-typed if it is well-typed and all features declared appropriate for τ are present in s. Type expansion is performed by the system in order to add features that are not stated explicitly.

Consider the following hierarchy:

$$\begin{tikzcd}
& a & \\
b \arrow[ur] & & c \arrow[ul] \\
& d \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

and assume that the types a, b and c are atomic (they have no features), but type d is declared as:

$$\begin{bmatrix} d \\ \text{F}\ a \end{bmatrix}$$

Then the unifier of b and c is a feature structure of type d. For it to be totally well-typed, the feature F has to be added (type expansion), since all instances of d must have this feature:

$$b \sqcap c = \begin{bmatrix} d \\ \text{F}\ a \end{bmatrix}$$

Total well-typedness makes definitions like the following of infinite size after type expansion (but they are detectable statically and generate an error):

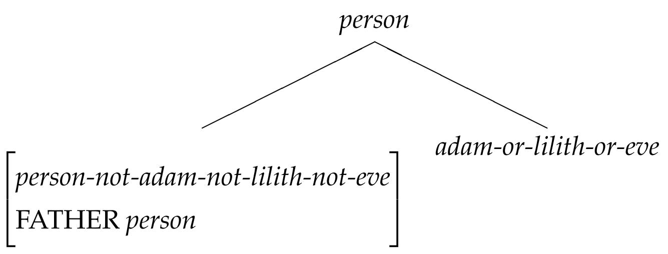
$$\begin{bmatrix} \mathit{person} \\ \text{FATHER}\ \mathit{person} \end{bmatrix}$$

A type cannot be defined to have a feature that is of that same type or more specific, because total well-typedness requires all features appropriate for a type to be present in all instances of that type. In this example, another feature FATHER would have to be present embedded under the topmost FATHER attribute, because it is of type *person*, which, according to this definition has a feature FATHER. But this second feature would be of the same type *person*, so a third feature FATHER would have to be added, and so on. Of course, recursive structures can still be obtained by declaring features to be of a type that is more abstract than, but still compatible with, the type

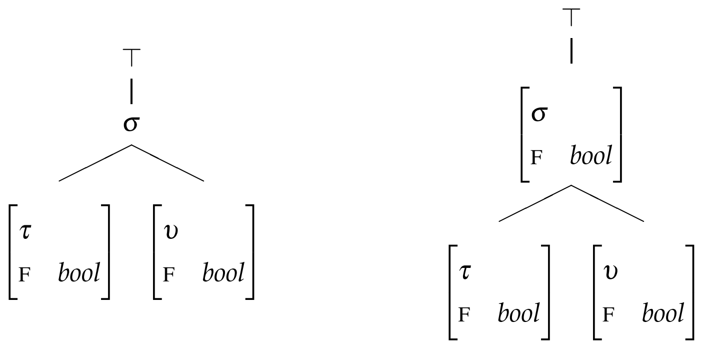
[23]If the hierarchy included a subtype of τ for which the feature G is defined, say the subtype ρ, then this instance would be made well-typed by inferring that it is not only an instance of τ but also an instance of ρ (see the Feature Introduction Condition and Type Inference below.)

[24]However, the LKB supports defaults (PET does not yet). If the constraint on the type of F is declared to be a default constraint (overridable) in τ, then a subtype can define an incompatible value for this feature. As mentioned before, we will not use defaults in this dissertation.

where they are declared. For instance, the following serves the same purpose and is totally well-typed (the type *adam-or-lilith-or-eve* is not required, because the LKB and PET do not perform subtype covering):



- **The feature introduction condition (FIC)** For each feature f, there has to be a single most general type for which f is appropriate. For instance, the hierarchy on the left does not respect the FIC, but the one on the right does:



This essentially means that all feature names must be unique, except for the features that are inherited. More precisely, there is a function, say *Approp*, that for each feature returns the most general type for which it is appropriate; with the hierarchy on the left, it is not possible to determine whether $Approp(F) = \tau$ or $Approp(F) = \upsilon$; with the one on the right, $Approp(F) = \sigma$.

- **Type inference** The kind of type inference of (Carpenter, 1992) is supported by the LKB and PET. It is made possible by enforcing the FIC. For instance, given a type hierarchy containing the hierarchy above on the right, the feature A can be inferred to be of the type σ in the following feature structure, because σ is the most general type for which F is appropriate:

$$\begin{bmatrix} \text{A}|\text{F} & + \end{bmatrix}$$

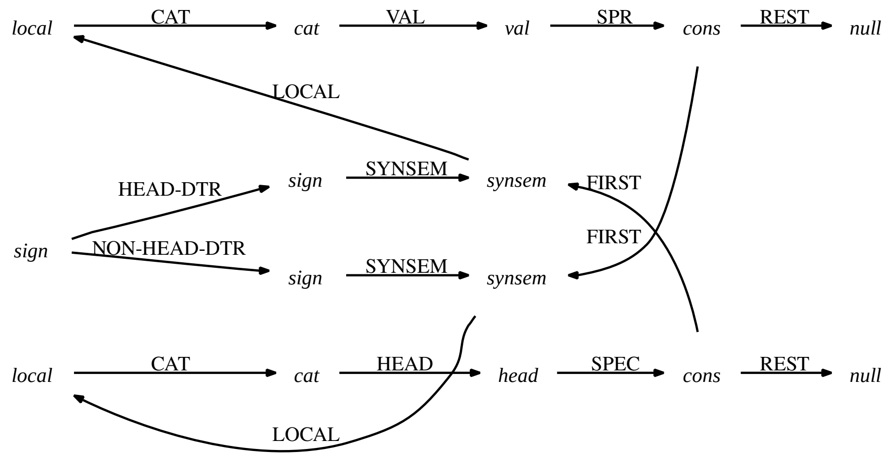
This allows the omission of the types of many features in instances. Note that the type hierarchy must still be fully described in the source.

Type inference interacts with total well-typedness. Suppose that another feature G is declared appropriate for σ, and its type is also *bool*, so that it now has two features, F and G, and σ is still the most general type for which F is appropriate. Then, with type inference and type expansion, the previous feature structure ([A|F +]) is actually the same as:

$$\begin{bmatrix} \text{A} & \begin{bmatrix} \sigma & \\ \text{F} & + \\ \text{G} & \textit{bool} \end{bmatrix} \end{bmatrix}$$

- **Cyclicity** The LKB and PET forbid cyclic structures. For instance, the following familiar construction cannot be implemented exactly as shown, because there would be a cycle in it (i.e. the structure corresponds to the cyclic graph presented below).

$$\begin{bmatrix} \textit{head-specifier-phrase} & & \\ \text{HEAD-DTR}|\text{SYNSEM} & \boxed{1} & \begin{bmatrix} \text{LOCAL}|\text{CAT}|\text{VAL}|\text{SPR} & \left\langle \boxed{2} \right\rangle \end{bmatrix} \\ \text{NON-HEAD-DTR}|\text{SYNSEM} & \boxed{2} & \begin{bmatrix} \text{LOCAL}|\text{CAT}|\text{HEAD}|\text{SPEC} & \left\langle \boxed{1} \right\rangle \end{bmatrix} \end{bmatrix}$$



Feature structures in the LKB and PET are directed acyclic graphs.

- **Greatest lower bound (most general unifier)** Consider the following hierarchy:

$$\begin{tikzcd} a & b \\ c \arrow[u] \arrow[ur] & d \arrow[u] \arrow[ul] \end{tikzcd}$$

It is not possible to determine a single result for the unification of *a* and *b*: both *c* and *d* qualify. The two are lower bounds for *a* and *b*, but none of *c* and *d* is more general than the other. There must be a single greatest lower bound for every pair of compatible types in a hierarchy, in order to guarantee that unification is deterministic.

If the $\mathscr{TDL}$ code describes hierarchies with this shape, automatic types are created by the LKB and PET with names that start with the string `glbtype`, followed by an integer. The result of loading in the LKB a grammar defining a hierarchy like the one just presented might make it look like:

$$\begin{tikzcd}
a & & b \\
& \mathit{glbtype1} \arrow[ul] \arrow[ur] & \\
c \arrow[ur] & & d \arrow[ul]
\end{tikzcd}$$

In the example above, the unification of *a* and *b* is something that can be a *c* or a *d*, hence a member of the union of the extensions of *c* and *d*. Supertypes denote set union, and in that example the glbtype that is created by the system denotes just the union of the extensions of *c* and *d*.

Disjunction and negation are not supported in the LKB and PET, and neither are sets or operations on sets and lists, such as testing for membership or computing set union or list appends. Arbitrary functions cannot be defined either. These limitations make LKB and PET extremely fast in parsing and generation when compared with similar systems.[25] However, the effects of disjunction and negation can be achieved by the type hierarchy alone, and some of these collections and operations on them can be implemented by manipulating typed feature structures directly (see Section 2.6).

2.6 Important Types and Mechanisms Defined in the LinGO Grammar Matrix

LXGram was implemented by taking the LinGO Grammar Matrix code as its base, and developing upon it. The LinGO Grammar Matrix (Bender *et al.*, 2002) is an open-source package where the fundamental types and features used in HPSGs are already defined. It covers several linguistic phenomena. For the phenomena covered, it provides several abstract types whose definitions can be combined by creating subtypes and taking advantage of multiple inheritance, according to the properties of the natural language that is to be modeled.

In this section we present some types that are defined in the LinGO Grammar Matrix and will be referred to in this dissertation.

Boolean types are given by the hierarchy under *bool*, where the descendant + means *true* and - means *false*:

$$\begin{tikzcd}
& \mathit{bool} & \\
+ \arrow[ur] & & - \arrow[ul]
\end{tikzcd}$$

Lists are defined recursively in the usual way. Type *null* is the empty list, *cons* is a non-empty list:

$$\begin{tikzcd}
& \mathit{list} & \\
\mathit{null} \arrow[ur] & & \mathit{cons} \arrow[ul]
\end{tikzcd}$$

[25]Penn (2004) mentions that the ERG, running in the LKB, is 300 times faster than MERGE, a similar grammar developed in TRALE, another typed feature platform that allows for these operations.

In non-empty lists, the feature FIRST points to the head of the list, and the attribute REST has the tail of the list as its value and is declared to be of the type *list*. Non-empty lists are implemented by the type *cons*, with the constraints:

$$\begin{bmatrix} \textit{cons} & \\ \textsc{first} & \textit{*top*} \\ \textsc{rest} & \textit{list} \end{bmatrix}$$

where **top** is $\top$, the most general type. Types *list* and *null* are atomic: they have no features.

The $\mathscr{TDL}$ notation < ···, ··· > is an abbreviation for lists. The empty list can be represented by < >. The following two are equivalent:

$$\begin{bmatrix} \textsc{f} \left\langle a, b \right\rangle \end{bmatrix} \qquad\qquad \begin{bmatrix} \textsc{f} & \begin{bmatrix} \textit{cons} & \\ \textsc{first} & a \\ \textsc{rest} & \begin{bmatrix} \textit{cons} & \\ \textsc{first} & b \\ \textsc{rest} & \textit{null} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Difference lists are a way to implement collections with an append operation that is computationally cheap. They do not have the semantics of sets in that they allow repeated elements and their elements are ordered, but they are used to implement sets, because the most frequent operation on sets for our purposes, set union, can be mimicked efficiently with appends done via difference lists. Difference lists resemble linked lists, with two features pointing to the beginning and the end:

$$\begin{bmatrix} \textit{diff-list} & \\ \textsc{list} & \textit{list} \\ \textsc{last} & \textit{list} \end{bmatrix}$$

An empty difference list is simply one where both features are unified:

$$\begin{bmatrix} \textit{diff-list} & \\ \textsc{list} & \boxed{1}\ \textit{list} \\ \textsc{last} & \boxed{1} \end{bmatrix}$$

They are not constrained to be empty lists (*null*), though.

A non-empty difference list has the most embedded REST of the list in LIST unified with LAST. In this case the most embedded REST is not constrained to be an empty list. The following is a difference list with two elements, *a* and *b*:

$$\begin{bmatrix} \textit{diff-list} & \\ \textsc{list} & \begin{bmatrix} \textit{cons} & \\ \textsc{first} & a \\ \textsc{rest} & \begin{bmatrix} \textit{cons} & \\ \textsc{first} & b \\ \textsc{rest} & \boxed{1}\ \textit{list} \end{bmatrix} \end{bmatrix} \\ \textsc{last} & \boxed{1} \end{bmatrix}$$

There is a $\mathscr{TDL}$ shorthand notation for difference lists: <! $\cdots$, $\cdots$!>. The empty difference list it <! !>.

The following constraints implement different list appends. The difference list append of A and B is C:

$$\begin{bmatrix} \text{A} & \begin{bmatrix} \text{LIST} & \boxed{1} & \textit{list} \\ \text{LAST} & \boxed{2} & \textit{list} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{LIST} & \boxed{2} & \\ \text{LAST} & \boxed{3} & \textit{list} \end{bmatrix} \\ \text{C} & \begin{bmatrix} \text{LIST} & \boxed{1} \\ \text{LAST} & \boxed{3} \end{bmatrix} \end{bmatrix}$$

The following example illustrates the append of <! c, d !> to <! a, b !>:

$$\begin{bmatrix} \text{A} & \begin{bmatrix} \text{LIST} & \boxed{1} \begin{bmatrix} \textit{cons} & \\ \text{FIRST} & a \\ \text{REST} & \begin{bmatrix} \textit{cons} & \\ \text{FIRST} & b \\ \text{REST}\ \boxed{2} & \textit{list} \end{bmatrix} \end{bmatrix} \\ \text{LAST} & \boxed{2} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{LIST} & \boxed{2} \\ \text{LAST} & \boxed{3}\ \textit{list} \end{bmatrix} \\ \text{C} & \begin{bmatrix} \text{LIST} & \boxed{1} \\ \text{LAST} & \boxed{3} \end{bmatrix} \end{bmatrix} \sqcap \begin{bmatrix} \text{A} & \begin{bmatrix} \text{LIST} & \boxed{1}\ \textit{list} \\ \text{LAST} & \boxed{2} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{LIST} & \boxed{2} \begin{bmatrix} \textit{cons} & \\ \text{FIRST} & c \\ \text{REST} & \begin{bmatrix} \textit{cons} & \\ \text{FIRST} & d \\ \text{REST}\ \boxed{3} & \textit{list} \end{bmatrix} \end{bmatrix} \\ \text{LAST} & \boxed{3} \end{bmatrix} \\ \text{C} & \begin{bmatrix} \text{LIST} & \boxed{1} \\ \text{LAST} & \boxed{3} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{A} & \begin{bmatrix} \text{LIST} & \boxed{1} \begin{bmatrix} \textit{cons} & \\ \text{FIRST} & a \\ \text{REST} & \begin{bmatrix} \textit{cons} & \\ \text{FIRST} & b \\ \text{REST}\ \boxed{2} & \begin{bmatrix} \textit{cons} & \\ \text{FIRST} & c \\ \text{REST} & \begin{bmatrix} \textit{cons} & \\ \text{FIRST} & d \\ \text{REST}\ \boxed{3} & \textit{list} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{LAST} & \boxed{2} \end{bmatrix} \\ \text{B} & \begin{bmatrix} \text{LIST} & \boxed{2} \\ \text{LAST} & \boxed{3} \end{bmatrix} \\ \text{C} & \begin{bmatrix} \text{LIST} & \boxed{1} \\ \text{LAST} & \boxed{3} \end{bmatrix} \end{bmatrix}$$

Note that, under the attribute C, the feature LAST points to the most embedded tail under LIST, as desired.

We will use the notation of sets for difference lists in this thesis, largely for convenience. Therefore $\{\cdots,\cdots\}$ can be viewed as notation for difference lists.

The basic feature geometry of the HPSG *sign* (a word or a phrase), is implemented in the LinGO Grammar Matrix in a way that replicates the usual organization of these features in HPSG:

$$\left[\begin{array}{ll}\textsc{synsem} & \left[\begin{array}{ll} \textsc{local} & \left[\begin{array}{ll} \textsc{cat} & \left[\begin{array}{ll}\textsc{head} & \textit{head}\\ \textsc{val} & \left[\begin{array}{ll}\textsc{subj} & \textit{list}\\ \textsc{spr} & \textit{list}\\ \textsc{comps} & \textit{list}\end{array}\right]\end{array}\right]\\ \textsc{cont} & \left[\begin{array}{ll}\textsc{hook} & \left[\begin{array}{ll}\textsc{ltop} & \textit{handle}\\ \textsc{index} & \textit{individual}\end{array}\right]\\ \textsc{rels} & \textit{diff-list}\\ \textsc{hcons} & \textit{diff-list}\end{array}\right]\end{array}\right]\\ \textsc{non-local} & \left[\begin{array}{ll}\textsc{slash} & \textit{diff-list}\\ \textsc{que} & \textit{diff-list}\\ \textsc{rel} & \textit{diff-list}\end{array}\right]\end{array}\right]\end{array}\right]$$

The features under SYNSEM contain syntactic and semantic information about the syntactic node represented by this instance. The features under CAT describe syntactic information. The attribute HEAD is of type *head*,[26] which has subtypes like *noun* and *verb*. It denotes the sort of head of a phrase, abstracting from saturation of arguments: nouns, $\overline{\text{N}}$ s and noun phrases all have a HEAD of type *noun*; verbs, verb phrases and sentences all have a HEAD of type *verb*, etc. The valence features under VAL — SUBJ, SPR and COMPS — encode information about the syntactic arguments of a word or phrase. These lists are lists of SYNSEM objects, and therefore all information that can be selected for by a word must be under SYNSEM. They take the empty list as value for words that do not have the relevant argument or for phrases where that argument has already been discharged.

The features under CONT contain information about semantics. The attributes RELS and HCONS contain the MRS information, and HOOK contains information that is used for the composition of semantics (they are our lambdas). The type *individual* is the supertype of *ref-index* (for variables over entities) and *event* (for events). More information about composition of semantics with MRS is in Section 3.4.2.

The features under NON-LOCAL are used for the treatment of long distance dependencies. We will not be concerned with the linguistic phenomenon of unbounded dependencies in this dissertation.

Other important types in the LinGO Grammar Matrix are the ones for the feature SYNSEM. We present a version of the hierarchy, simplified in that only the types mentioned in the dissertation are presented. It is in Figure 2.2.[27]

The Principle of Canonicality (Ginzburg and Sag, 2000) (all projected constituents have a SYNSEM of type *canonical-synsem*) is enforced by constraints on the types for phrases that force all realized elements to have SYNSEM of type *canonical-synsem*. The two subtypes of *canonical-synsem* in the hierarchy in Figure 2.2, *lex-synsem* and *phrase-synsem*, are the types of the SYNSEMs of words and phrases respectively. That is, the type *word*, the subtype of *sign* from which all lexical types inherit, has its SYNSEM constrained to be of type *lex-synsem*, and similarly for phrases.

It follows from the Principle of Canonicality that elements with a SYNSEM of a type subsumed by *non-canonical-synsem* cannot be realized, since there is no unifier for *canonical-synsem* and *non-canonical-synsem* according to the hierarchy in Figure 2.2.

[26]The type it is declared to be in the LinGO Grammar Matrix is actually called *head-min*.

[27]Some types in this figure have names slightly different from their names in the LinGO Grammar Matrix. For instance, the type *unexpressed-synsem* is actually called *unexpressed* in the LinGO Grammar Matrix, and *phrase-synsem* is named *phr-synsem*.

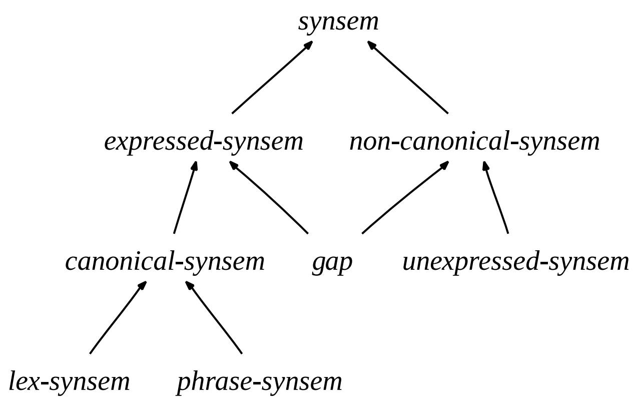


Figure 2.2: Simplified hierarchy of types for SYNSEM objects.

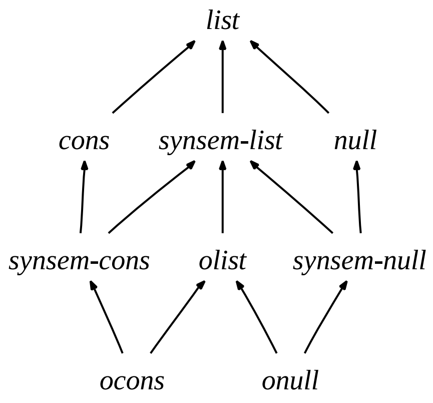


Figure 2.3: Hierarchy of list types.

Type *gap* is used for the treatment of long distant dependencies, whereas *unexpressed-synsem* is for elements that are not realized either locally or non-locally, e.g. null arguments. For instance, complements with a synsem of a type unifiable with *unexpressed-synsem* are thus optional; obligatory ones are constrained to be incompatible with *unexpressed-synsem*.[28] Since obligatory complements can nevertheless be extracted, *gap* inherits from *expressed-synsem*.

An important set of types is the one for lists of *unexpressed-synsem*s. Since the LKB lacks parameterized/generic types, one has to create subtypes of *list* in order to enforce the type of the elements of a list. The hierarchy in Figure 2.3 allows constraining a list to have all elements of type *synsem* or *unexpressed-synsem*.

The types *list*, *null* and *cons* are the ones presented before for lists in general.

Type *synsem-list* represents a list of synsems. One of its specialized types, *synsem-cons*, includes additional constraints:

$$\begin{bmatrix} \textit{synsem-cons} & \\ \text{FIRST} & \textit{synsem} \\ \text{REST} & \textit{synsem-list} \end{bmatrix}$$

The types *synsem-list* and *synsem-null* have no constraints.

[28]In the Matrix, a feature OPT is used to this end, but it is not required, the type hierarchy being enough: obligatory arguments can be specified to have a SYNSEM of type *expressed-synsem*.

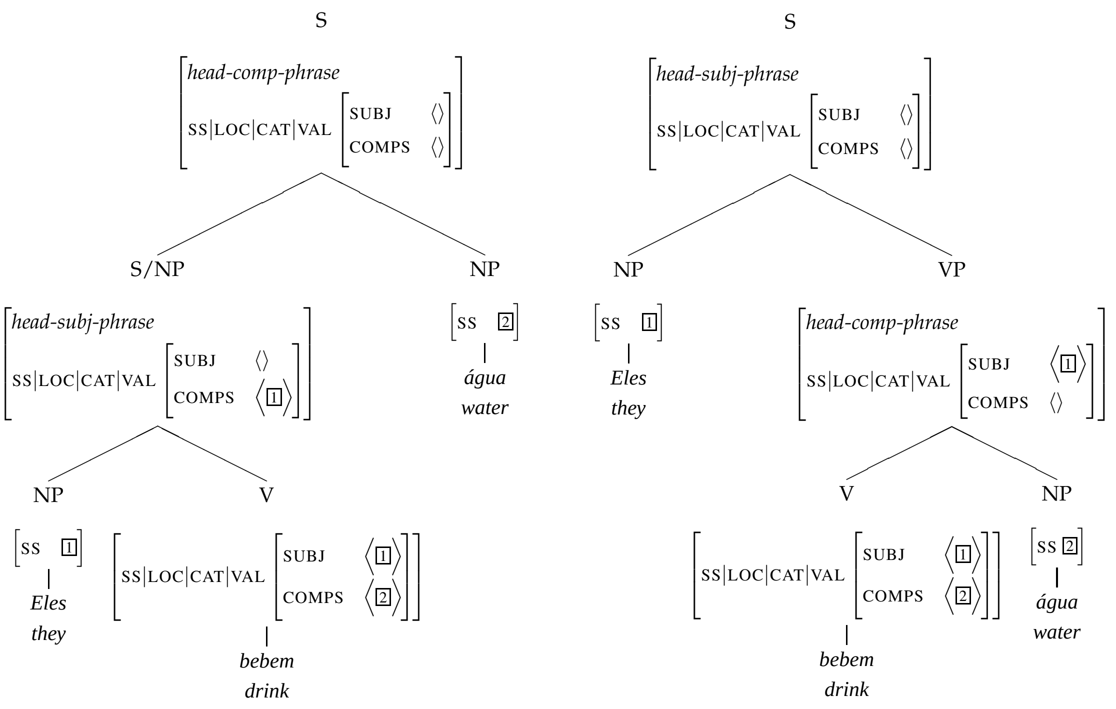
Finally, *olist*s are lists of "unexpressed" elements:

$$\begin{bmatrix} \textit{ocons} & \\ \text{FIRST} & \textit{unexpressed-synsem} \\ \text{REST} & \textit{olist} \end{bmatrix}$$

The types *olist* and *onull* have no constraints.

Although the LinGO Grammar Matrix includes type *olist*, no *synsem-list* is given. It is an addition in LXGram that allows more type checking at grammar compile/load time.[29]

An interesting property of *olist* is that it can often be used to constrain lists like COMPS instead of requiring them to be empty (see (Flickinger, 2000), where this list type is called *optlist*). For instance, if subjects are projected higher in a tree than complements, it is usual to constrain the head daughter of Head-Subject constructions to have an empty COMPS list. This rules out the tree on the left and allows for the one on the right:[30]


S
head-comp-phrase
SS|LOC|CAT|VAL SUBJ ⟨⟩ COMPS ⟨⟩
S/NP
head-subj-phrase
SS|LOC|CAT|VAL SUBJ ⟨⟩ COMPS ⟨1⟩
NP
SS 2
água
water
NP
SS 1
Eles
they
V
SS|LOC|CAT|VAL SUBJ ⟨1⟩ COMPS ⟨2⟩
bebem
drink
S
head-subj-phrase
SS|LOC|CAT|VAL SUBJ ⟨⟩ COMPS ⟨⟩
NP
SS 1
Eles
they
VP
head-comp-phrase
SS|LOC|CAT|VAL SUBJ ⟨1⟩ COMPS ⟨⟩
V
SS|LOC|CAT|VAL SUBJ ⟨1⟩ COMPS ⟨2⟩
bebem
drink
NP
SS 2
água
water

If *olist* is used to constrain the COMPS of Head-Subject constructions instead, the same structure obtains (the one on the right). The relevant constraints of Head-Subject and Head-Complement phrases are the following:

[29]This is useful because several features are lists of synsems, like SUBJ or COMPS, while others are lists of signs, like ARGS (which can also be typed similarly).

[30]Here, SS abbreviates SYNSEM, and LOC abbreviates LOCAL.

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{head-subj-phrase}} \\
\textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{val} & \left[\begin{array}{ll} \textsc{subj} & \langle\rangle \\ \textsc{comps} & \boxed{1}\ \textit{olist} \end{array}\right] \\
\textsc{head-dtr}\ \boxed{4} & \left[\textsc{synsem}\ \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{canonical-synsem}} \\ \textsc{local}|\textsc{cat}|\textsc{val} & \left[\begin{array}{ll} \textsc{subj} & \left\langle\boxed{2}\right\rangle \\ \textsc{comps} & \boxed{1} \end{array}\right] \end{array}\right]\right] \\
\textsc{non-head-dtr}\ \boxed{3} & \left[\textsc{synsem}\ \ \boxed{2}\ \textit{canonical-synsem}\right] \\
\textsc{args}\ \left\langle\boxed{3},\boxed{4}\right\rangle &
\end{array}\right]$$

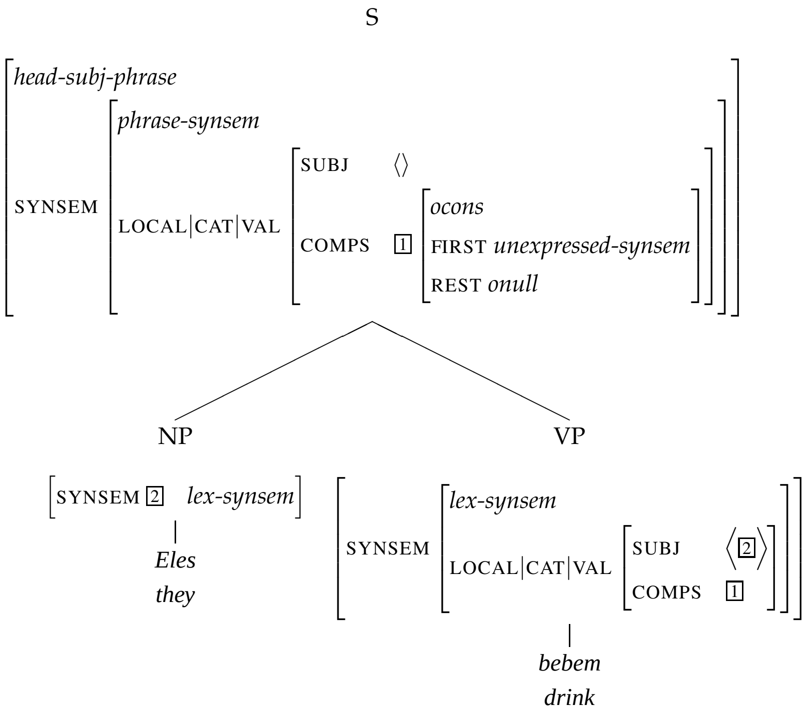
$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{head-comp-phrase}} \\
\textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{val} & \left[\begin{array}{ll} \textsc{subj} & \boxed{1} \\ \textsc{comps} & \boxed{2} \end{array}\right] \\
\textsc{head-dtr}\boxed{4} & \left[\textsc{synsem}\ \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{canonical-synsem}} \\ \textsc{local}|\textsc{cat}|\textsc{val} & \left[\begin{array}{ll} \textsc{subj} & \boxed{1} \\ \textsc{comps} & \left[\begin{array}{ll} \textsc{first} & \boxed{3} \\ \textsc{rest} & \boxed{2} \end{array}\right] \end{array}\right] \end{array}\right]\right] \\
\textsc{non-head-dtr}\ \boxed{5} & \left[\textsc{synsem}\ \ \boxed{3}\ \textit{canonical-synsem}\right] \\
\textsc{args}\ \left\langle\boxed{4},\boxed{5}\right\rangle &
\end{array}\right]$$

Types *olist* and *null* (*null* is the tail of a singleton list, percolated from the head daughter in Head-Complement constructions) unify to *onull*, as can be seen in the hierarchy in Figure 2.3, which allows the analysis on the right.

The analysis on the left is blocked by the interaction with the Principle of Canonicality: *unexpressed-synsem* does not unify with *canonical-synsem*. Note that the type of COMPS of the head daughter of *head-comp-phrase* is inferred to be *cons*, because it is the most general type for which the features FIRST and REST are appropriate (it is actually *synsem-cons* if COMPS is declared to be of type *synsem-list*, since the most general unifier of *synsem-list* and *cons* is *synsem-cons*). A *head-subj-phrase* cannot be the head daughter of a *head-comp-phrase*: since *head-subj-phrase* constrains its COMPS to be an *olist*, it would be an *ocons*, the most general unifier for *olist* and *cons* (or *synsem-cons*), which constrains its elements to be *unexpressed-synsem*s. But *head-comp-phrase* requires the first element of the COMPS of its head daughter to be a *canonical-synsem*, which does not unify with *unexpressed-synsem*.

The advantage of using *olist* instead of *null* is that if the complement is not realized, it does not have to be removed from COMPS by another rule:[31]

[31]There are complications when there are multiple complements and an optional complement precedes another complement, but they need not concern us here. See (Flickinger, 2000) for a possible solution.



2.7 Summary

We described Head-Driven Phrase Structure Grammar, which is employed in LXGram and will be used in the discussion in the following chapters. HPSG is a grammatical framework that is suited to computational implementations, since its fundamental mechanisms are shared with programming languages. We reviewed several of these aspects, like the type system, *has-a* relations, and unification. We presented the specific properties of the formalism as it is accepted by the systems where LXGram is implemented, like the Feature Introduction Condition and type inference.

The mechanism used to compose sentential semantics from the basic meanings of words was described. In the original approach, lambda expressions are given as the meaning of words and semantic rules, where argument and functor are specified, are associated to the syntactic rules of a context free grammar. The way the semantics is composed in HPSG is very similar. Here we will use Minimal Recursion Semantics instead of the lambda calculus for this purpose, but the idea is fundamentally the same. MRS is used in several unification grammars, as it addresses some of the shortcomings of the lambda calculus in the composition of semantics, in particular efficiency ones. MRS was conceived with computational efficiency in mind. For instance, since it allows for underspecification of quantifier scope, it avoids generating multiple analyses for sentences that are ambiguous in this respect.

Finally, we presented some of the types and features that are defined in the LinGO Grammar Matrix and will be referred to in the rest of this dissertation. We also took this opportunity to present the mostly used attributes and types of HPSG.

The use of structured information allows for very general rules in HPSG, that just combine information coming from the daughters. Ultimately, the relevant information comes from constraints on lexical items (terminal symbols). HPSG is strongly lexicalized. Syntax rules and their associated constraints for the composition of semantics are very general. We provided brief illustrations of some important syntax rules of HPSG.

3 Functors

3.1 Overview

This chapter begins with a brief review and illustration of the various syntactic relations that have been considered in the HPSG literature as holding between syntactic constituents.

We present the repertoire of syntactic relations that is used in LXGram, which contains an extension with respect to standard HPSG. The motivation for that extension is discussed, as well as the types and features that will be involved in the architecture thus obtained. This architecture will be employed in the analyses to be worked out in the remaining chapters.

3.2 Syntactic Relations in HPSG

In (Pollard and Sag, 1994), the syntactic relations that hold between two syntactic constituents are Head-Specifier, Head-Subject, Head-Complement, Head-Adjunct and Head-Marker. Figure 3.1 shows an example sentence with these relations identified.

The tendency since then has been to reduce this inventory. In (Sag *et al.*, 2003) they are Head-Specifier, Head-Complement and Head-Adjunct, with the previous Head-Subject configurations considered instances of Head-Specifier relations and the former Head-Marker constructions as instances of Head-Complement configurations. Figure 3.2 presents these relations for the same sentence.

In the LinGO Grammar Matrix, four configurations are employed: Head-Specifier, Head-Subject, Head-Complement and Head-Adjunct. This is also true of many computational HPSGs (e.g. the LinGO English Resource Grammar, the German Grammar or Jacy). Figure 3.3 exemplifies them.

The work of Allegranza (1998a,b) and Van Eynde (2003a,b) proposes a reduction in a different way: specifiers, adjuncts and markers are merged into the category functor. In this system the configurations are the ones in Figure 3.4.

In LXGram we adapted the LinGO Grammar Matrix so as to use subjects, functors (merging of

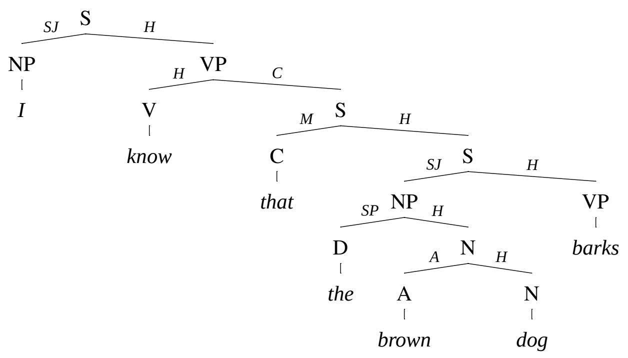


Figure 3.1: Syntactic relations in (Pollard and Sag, 1994). *H* denotes a head, *SJ* denotes a subject, *SP* a specifier, *C* a complement, *A* an adjunct and *M* a marker.

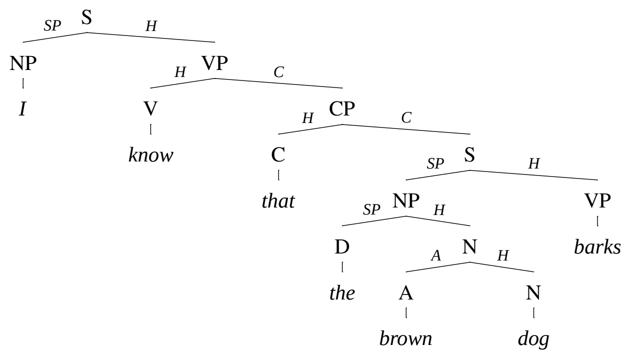


Figure 3.2: Syntactic relations in (Sag *et al.*, 2003). *H* denotes a head, *SP* a specifier, *C* a complement and *A* an adjunct.

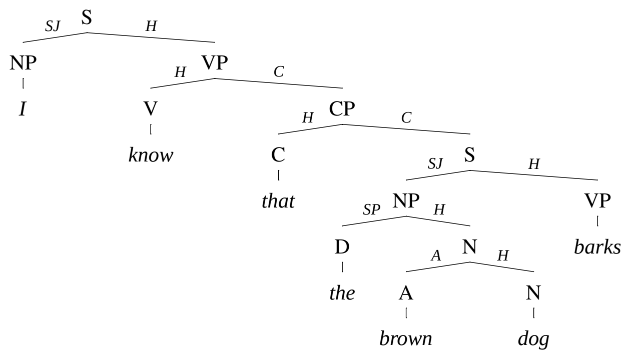


Figure 3.3: Syntactic relations in the LinGO Grammar Matrix. *H* denotes a head, *SJ* denotes a subject, *SP* a specifier, *C* a complement and *A* an adjunct.

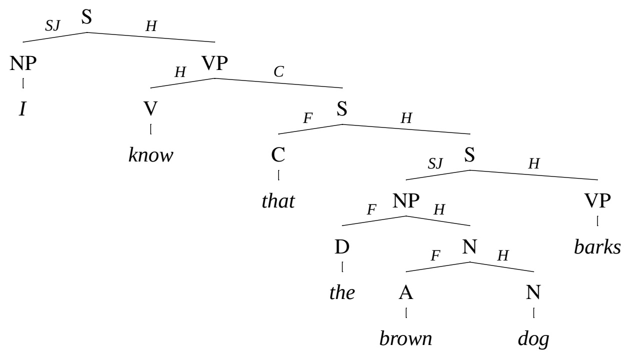


Figure 3.4: Syntactic relations in (Van Eynde, 2003b). *H* denotes a head, *SJ* denotes a subject, *C* a complement and *F* a functor.

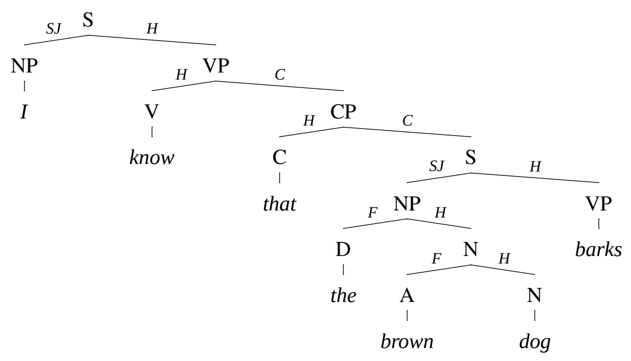


Figure 3.5: Syntactic relations in LXGram. *H* denotes a head, *SJ* denotes a subject, *C* a complement and *F* a functor.

specifiers and adjuncts) and complements, but we are treating Head-Marker constructions as Head-Complement constructions, like Sag *et al.* (2003), as depicted in Figure 3.5.

The implementation of Head-Functor configurations in LXGram is presented in the following Sections.

Within HPSG there are different approaches also with respect to the status of NPs, since the DP hypothesis (according to which NPs are rather DPs, i.e. the head of an NP is the determiner and not the noun, the sister node of the determiner being its complement) is followed in some work (e.g. Beavers (2003b)). In LXGram and here, we will treat these phrases as NPs and accordingly consider the noun to be the head.

The inventory of syntactic relations used has implications in the general feature geometry of the grammars. Different relations are implemented via different features in the lexical entries, and syntactic rules look at them in order to produce phrasal constituents. Therefore, the number and nature of the relations considered have an impact on the number of syntactic rules necessary in a grammar, as well as on the number of features.

With the conceptual organization of Pollard and Sag (1994), the features SUBJ (where the subject of a head is stated), COMPS (the complements of a head), SPR (the specifier of a head), SPEC (the head of a specifier), MOD (the head of an adjunct) and MARKING (for markers) are necessary. Five schemata are also required to project subjects, complements, specifiers, adjuncts and markers, because selection information is found in different attributes.

3.2.1 Motivation for Functors

Before explaining the functor architecture (in the following sections), we provide some motivation in favor of it.

Within the NP domain, the status of several constituent items challenge a sharp divide between specifiers and adjuncts and therefore the full relevance of those notions. We will look at possessives as an example.

Possessives have specifier-like properties. Since they cannot iterate, the combinatorial potential of a possessive plus noun sequence is different from that of a noun, because the former cannot further combine with a possessive to its left. Also, possessives can realize an argument of the head noun, as in the expression *meu pai* (*my father*). So it seems natural to encode them in a valence feature of nouns.

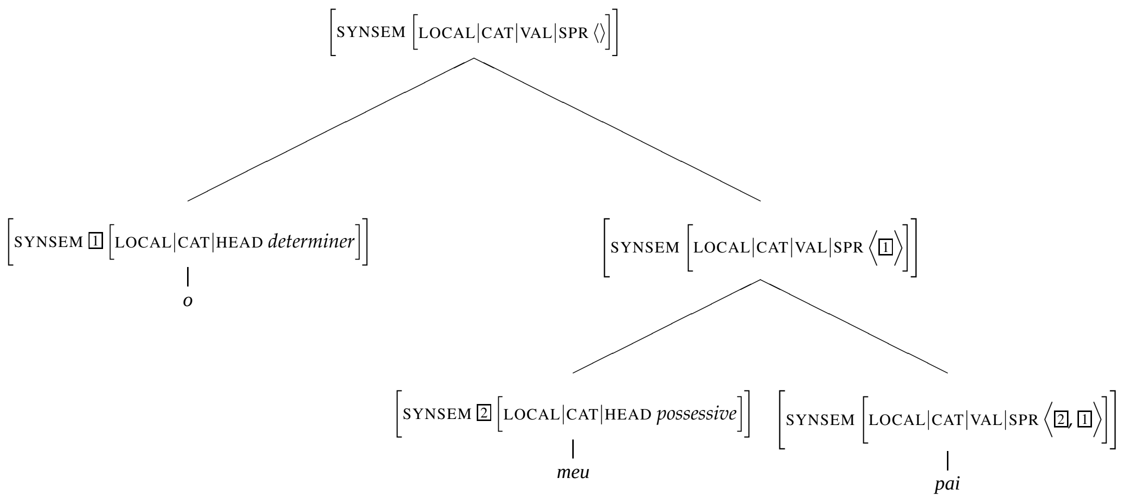
There are two possibilities: to use a valence list that already exists, or to create a separate attribute for possessives. If the first possibility is considered, the best candidate is the feature SPR, since in many contexts possessives precede the head.

But in Portuguese possessives co-occur with determiners, as in (17).

(17) a. o meu pai
the my father
my father

b. * meu o pai
my the father

The consequence is that SPR should have more than one element. Since we only have access to the heads of lists (not to e.g. their last element), and the first element to be discharged will be the one closest to the noun, the elements of SPR have to be in the order reverse to word order. In this example, the possessive would be the first element, and the determiner would be the second. Head-Specifier constructions simply project the first element of a head's SPR list feature to the left of that head, and pass that list's tail up:


[SYNSEM [LOCAL|CAT|VAL|SPR ⟨⟩]]
[SYNSEM 1 [LOCAL|CAT|HEAD *determiner*]]
o
[SYNSEM [LOCAL|CAT|VAL|SPR ⟨1⟩]]
[SYNSEM 2 [LOCAL|CAT|HEAD *possessive*]]
meu
[SYNSEM [LOCAL|CAT|VAL|SPR ⟨2, 1⟩]]
pai

The problems start with the fact that possessives do not have to be present in every NP. Either a unary rule would have to be employed to eliminate the first element of SPR or another Head-Specifier rule that projects the second element would be necessary. In the first case this rule would have to be different from the rule that accounts for bare NPs (NPs lacking a determiner), because the latter also has to add quantifier semantics.

Furthermore, in some contexts the possessive follows the head, as in (18).

(18) um irmão meu
a brother my
a brother of mine

To account for this case yet another Head-Specifier construction would be needed in a grammar implemented in the LKB, because the order between head daughter and non-head daughter is different, and the LKB does not support the linear precedence constraints of theoretical HPSG, that are independent of phrasal types.

The second possibility consists of using a separate feature to encode saturation of possessives. This would also require two extra syntactic rules to project elements in this feature, one for each relative word order between possessive and noun.

The multiplication of syntactic rules in the second possibility is aggravated by the existence of other adnominal elements that cannot be iterated, like cardinals or ordinals. If each of these elements is encoded in a different valence list, the number of syntactic rules necessary to realize them is inevitably large.

The fact that possessives can precede and follow the noun approximates them also to adjuncts. However, the facts that they cannot iterate, that they may realize noun arguments and that nonetheless word order between them and their head is somewhat more constrained than the general behavior of adjuncts in Portuguese makes it problematic to simply group them with adjuncts.

With the general schema assumed for adjuncts, it is not possible to prevent these elements from iterating, as their combination with heads cannot give rise to nodes with HEAD or VAL features different from the head daughter. Another feature under CAT can be used to control iteration, though, and this is precisely what the functor approach puts in place.

3.3 General Feature Geometry

In Allegranza (1998a,b) and Van Eynde (2003a,b), Head-Functor relations are a cover of Head-Specifier and Head-Adjunct configurations. Functors, like adjuncts, select their head via a dedicated feature. All treatments put this feature that encodes selection requirements under the attribute HEAD, but the name varies (here it is SELECT). As a consequence, this feature SELECT percolates in all headed constructions (in all constructions with a head daughter, the feature HEAD of the mother node is token identical to the same feature in the head daughter). Like Head-Specifier configurations, information about saturation of the resulting node (i.e. its combinatorial potential) may be different from the combinatorial potential of the head daughter. So, for instance, a noun can combine with a determiner on the left to form a phrase with a determiner and noun, and this phrase cannot combine with another determiner on the left.

In Head-Specifier configurations, information about this kind of combinatorial potential is determined by the head noun, not by the specifier. Nouns select their specifier in their SPR attribute, which is list-valued (and usually either the empty list or a singleton list). Head-Specifier constructions unify the synsem in that attribute with the synsem of the other daughter, and the SPR of the mother node is the tail of the SPR list in the head daughter.

In Head-Functor schemata, saturation of the mother node is determined by the functor daughter, not the head (details are below).

With functors replacing specifiers and adjuncts, the features MOD, SPEC and SPR are no longer necessary, and neither are Head-Specifier phrases and Head-Adjunct phrases.

LXGram uses the features SELECT, MARKING and MARK to implement Head-Functor schemata. MARKING is used to describe combinatorial potential other than saturation of a head's arguments. For instance, since cardinals cannot iterate, the top node of a phrase like *three cars* would have a different value of MARKING from the one of *cars*.

The features MARK and SELECT are relevant for functors. The value of MARKING of the mother node in Head-Functor phrases comes from the MARK attribute of the functor daughter. The attribute SELECT is where functors state the heads they can attach to.

Because the attributes SELECT and MARK are appropriate for functors and functors only, they are grouped under the MARKER attribute, which is put directly under HEAD. In LXGram there is also an abstract subtype of *head*, *functor*, where this subfeature MARKER is introduced. Figure 3.6 depicts it with some other subtypes of *head*.

The feature MARKER is thus present only in the signs that can be the functor daughter of a Head-Functor phrase, viz. those that have a HEAD type that inherits from *functor*. According to Figure 3.6, nouns, which have their HEAD feature of type *noun*, do not have the MARKER feature, which auto-

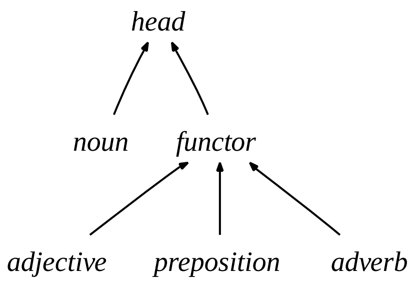


Figure 3.6: Minimal type to introduce the feature MARKER, and some other subtypes of *head*

matically excludes them from unifying with the functor daughter of Head-Functor phrases, as will be shown.

3.4 Constraints on Head-Functor Phrases

The main properties of Head-Functor schemata are presented in Figure 3.7.

$$\begin{bmatrix} \textit{basic-head-functor-phrase} & \\ \text{SYNSEM}|\text{LOCAL}|\text{CAT} & \begin{bmatrix}\text{HEAD} & \boxed{1}\\ \text{VAL} & \boxed{2} \\ \text{MARKING} & \boxed{3}\end{bmatrix} \\ \text{HEAD-DTR}|\text{SYNSEM}\ \boxed{4} & \begin{bmatrix}\text{LOCAL}|\text{CAT} \begin{bmatrix}\text{HEAD} & \boxed{1}\\ \text{VAL} & \boxed{2}\end{bmatrix}\end{bmatrix} \\ \text{NON-HEAD-DTR}|\text{SYNSEM}|\text{LOCAL}|\text{CAT} & \begin{bmatrix}\textit{saturated-cat} & \\ \text{HEAD}|\text{MARKER} & \begin{bmatrix}\text{MARK} & \boxed{3}\\ \text{SELECT} & \boxed{4}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Figure 3.7: Outline of Head-Functor schemata

It is a headed construction, so the HEAD of the mother is token-identical to the HEAD of the head daughter. This constraint — the Head Feature Principle — does not have to be stated explicitly in this type, since it is inherited from a supertype in the LinGO Grammar Matrix, which implements this principle. This type is *headed-phrase*, and a simplified hierarchy of phrasal types is in Figure 3.8. In that figure, *basic-binary-headed-phrase* is where the feature NON-HEAD-DTR is introduced, with HEAD-DTR being appropriate for *headed-phrase*. The types *head-initial* and *head-final* are discussed in the next section.

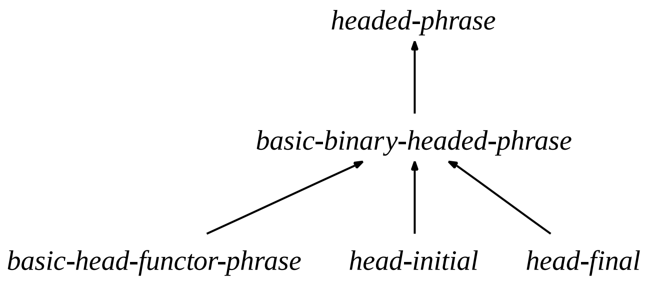


Figure 3.8: Type *basic-head-functor-phrase* in the hierarchy of phrase types

Valence is also shared since functors do not discharge the subject or a complement of the head.

Because functors have access to information about valence and marking of the head (the functor daughter's SELECT feature is unified with the head daughter's SYNSEM), control on the level of saturation of the head in these constructions is reduced to lexical specifications in functors.

Functors can cause the saturation described by the feature MARKING on the mother node to be different from the one on the head daughter — the mother node's MARKING feature is structure-shared with the functor's MARK feature.

The functor daughter is constrained to be saturated: its feature CAT is defined to to be of the type *saturated-cat*. The type *saturated-cat* encapsulates the information of what a saturated phrase is. It is defined in the following way:

$$\begin{bmatrix} \textit{saturated-cat} \\ \text{VAL}|\text{COMPS } \textit{olist} \\ \text{MARKING } \textit{saturated} \end{bmatrix}$$

The purpose of the constraint on MARKING will be made clear in the next chapter. This means that the functor daughter of Head-Functor constructions is required to have its complements already discharged when it feeds this rule (cf. the constraint on COMPS). It is a simplification. For instance, (19) shows an example where the phrase following the noun (*do que as proferidas pelo Comissário*) can arguably be analyzed as the complement of the adjective, but the adjective precedes the noun while that phrase follows it.

(19)

Não	encontraria	melhores	palavras	do que	as	proferidas	pelo	Comissário.
not	would find	better	words	than	the	pronounced	by the	Commissary

I wouldn't find better words than the ones pronounced by the Commissary.

This example can be analyzed as involving the percolation of the complement of the adjective (*melhores*) to the *basic-head-functor-phrase* node, and projecting it as its sister. In order for this to work, the COMPS of the functor daughter cannot be empty or a list of unexpressed elements. The account presented here does not allow for this analysis. However, there are alternative analyses for data like this one. For instance, Bouma *et al.* (2001) sketch an analysis for extraposed complements of adjectives involving a feature EXTRA that is compatible with the constraints presented here.

Section 3.4.1 presents the features involved in constraining word order possibilities between the functor and the head. Section 3.4.2 presents the constraints on Head-Functor phrases necessary to compose the semantics. Long distance dependencies are beyond the scope of this dissertation, and we will not present here the constraints on the NON-LOCAL features of Head-Functor constructions.

3.4.1 Word Order in Head-Functor Phrases

In the LKB, the actual daughters of a rule are configured with the parameter ***args-path***. Its value is usually ARGS, a feature of *sign* instances.[1] It must be list-valued (it is a list of *sign*s), and the position of an element in that list correlates with linear precedence. In many computational grammars and in the LinGO Grammar Matrix, features like the attribute HEAD-DTR used in Figure 3.7 can be viewed as pointers to specific elements of that list.

It is often the case that abstract types for phrases are employed to constrain these pointer features, and then different subtypes implement different word order possibilities by linking them to different

[1]The feature ARGS is declared in the LinGO Grammar Matrix as appropriate for *sign*s, but it does not make sense to speak of the ARGS of lexical items, so in LXGram ARGS is declared in type *phrase-or-lexrule* instead. This type also comes in the LinGO Grammar Matrix. It is the supertype of all phrases and lexical rules (which for instance account for morphological inflection), but not of lexical items.

$$\begin{bmatrix}\textit{head-initial} & \\ \text{HEAD-DTR} & \boxed{1} \\ \text{NON-HEAD-DTR} & \boxed{2} \\ \text{ARGS} & \left\langle \boxed{1}, \boxed{2} \right\rangle\end{bmatrix}\begin{bmatrix}\textit{head-final} & \\ \text{HEAD-DTR} & \boxed{1} \\ \text{NON-HEAD-DTR} & \boxed{2} \\ \text{ARGS} & \left\langle \boxed{2}, \boxed{1} \right\rangle\end{bmatrix}$$

Figure 3.9: Implementation of word order in binary headed phrases

elements in ARGS. Figure 3.9 shows the constraints for two abstract types that define word order between the head daughter and the non-head daughter in binary headed phrases: *head-initial* constrains the head daughter to precede the non-head daughter, and *head-final* defines the non-head daughter to precede the head daughter. These types are in the LinGO Grammar Matrix.

For instance, there is a general type in the LinGO Grammar Matrix where the basic constraints for Head-Subject constructions are stated. This type is *basic-head-subj-phrase*. Among the constraints in *basic-head-subj-phrase* are the following: [2]

$$\begin{bmatrix}\textit{basic-head-subj-phrase} & \\ \text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{VAL} & \begin{bmatrix}\text{SUBJ} \ \langle\rangle \\ \text{COMPS} \ \boxed{2}\end{bmatrix} \\ \text{HEAD-DTR}|\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{VAL} & \begin{bmatrix}\text{SUBJ} \ \left\langle \boxed{1} \right\rangle \\ \text{COMPS} \ \boxed{2}\end{bmatrix} \\ \text{NON-HEAD-DTR}|\text{SYNSEM} \ \boxed{1} & \end{bmatrix}$$

In order to parse a sentence like the one in (20a), where the NP subject precedes the head VP, one needs a syntactic rule that inherits from *basic-head-subj-phrase* and *head-final*. In order to parse a sentence like the one in (20b), where the NP subject follows the VP, one can create a syntax rule that inherits from *basic-head-subj-phrase* and *head-initial*.

(20) a.

[$_{NP}$ Este	governo]	[$_{VP}$ não	cairá]	porque	não	é	uma casa.
this	government	not	will fall down	because	not	is	a house

This government will not fall because it is not a house.

b.

[$_{VP}$ Sairá]	[$_{NP}$ o	mesmo]	com benzina,	porque	é	uma nódoa.
will go away	the	same	with benzine	because	is	a stain

It will go away with benzine, because it is a stain.

The same is done here. The essential properties of Head-Functor phrases are stated in the abstract type *basic-head-functor-phrase*, already presented in Figure 3.7. Two subtypes implement the two word order possibilities between the daughters, by inheriting from *head-initial* or *head-final*. The result is shown in Figure 3.10. All headed phrases with two daughters inherit from *basic-binary-headed-phrase*. The two lowest types are the only ones that are used by the system as syntactic rules.

As stated, all functors can feed both rules. This situation is not desirable, since specific pairings of head and functor can be restricted to occur in a specific order.

For instance, a preposition comes in the lexicon with the information that it must attach to an NP

[2]In this rule there is no need to recursively project the first element of SUBJ, as there can be only one subject. Instead of stating that the SUBJ of the mother is the tail of the SUBJ of the head daughter, the constraints simply state that the SUBJ of the mother is empty and the SUBJ of the head daughter is singleton.

$$\begin{tikzcd}
 & & \textit{basic-binary-headed-phrase} & & \\
\textit{head-initial} \arrow[urr] & & \textit{basic-head-functor-phrase} \arrow[u] & & \textit{head-final} \arrow[ull] \\
 & \textit{head-functor-phrase} \arrow[ul] \arrow[ur] & & \textit{functor-head-phrase} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Figure 3.10: Organization of Head-Functor phrases.

on its right (forming a PP)[3]and then modify nouns and verbs. PPs can attach to either side of a verb headed constituent (21a, 21b), but only to the right of nouns (21c, 21d).

(21) a.

Isso	sai	[$_{PP}$ com	benzina.]
that	goes away	with	benzine

That goes away with benzine.

b.

Isso	[$_{PP}$ com	benzina]	sai.
that	with	benzine	goes away

That goes away with benzine.

c.

Era	um	chapéu	[$_{PP}$ com	uma	antena.]
was	a	hat	with	an	antenna

It was a hat with an antenna.

d.

* Era	um	[$_{PP}$ com	uma	antena]	chapéu.
was	a	with	an	antenna	hat

The way word order is controlled is by using more features to denote word order restrictions. These restrictions are seen as properties of functors, i.e. it is assumed that word order restrictions are lexical properties of functors.

The two features that are used are also under MARKER: PREHEAD and POSTHEAD. They contain constraints that must be satisfied when a functor precedes or follows the head daughter, respectively. These constraints are put on SELECT and MARK attributes under PREHEAD and POSTHEAD. The basic organization of features under MARKER is in Figure 3.11. The AVM in this figure does not correspond to a type definition. It is rather a schematic view of the features we will use. More details on the types involved are provided in Section 3.5.

$$\begin{bmatrix} \text{MARKER} & \begin{bmatrix} \text{SELECT} & \textit{synsem} \\ \text{MARK} & \textit{marking} \\ \text{PREHEAD} & \begin{bmatrix} \text{SELECT} & \textit{synsem} \\ \text{MARK} & \textit{marking} \end{bmatrix} \\ \text{POSTHEAD} & \begin{bmatrix} \text{SELECT} & \textit{synsem} \\ \text{MARK} & \textit{marking} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 3.11: Organization of the features under the HEAD of functors.

The head type for prepositions could have the constraints in Figure 3.12, where *noun-or-verb* is a supertype of *verb* and *noun*.

[3]This is achieved by constraining the element in the COMPS of prepositions to be of type *canonical-synsem*, as in Portuguese comple-

$$\begin{bmatrix} \textit{preposition} & \\ \textsc{marker} & \begin{bmatrix} \textsc{select}|\textsc{local}|\textsc{cat}|\textsc{head} & \textit{noun-or-verb} \\ \textsc{prehead}|\textsc{select}|\textsc{local}|\textsc{cat}|\textsc{head} & \textit{verb} \end{bmatrix} \end{bmatrix}$$

Figure 3.12: Constraints on the HEAD of prepositions. *noun-or-verb* is a supertype of *noun* and *verb*, the head types of nouns and verbs, respectively.

$$\begin{bmatrix} \textit{head-functor-phrase} & \\ \textsc{non-head-dtr}|\textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{head}|\textsc{marker} & \begin{bmatrix} \textsc{select} & \boxed{1} \\ \textsc{mark} & \boxed{2} \\ \textsc{posthead} & \begin{bmatrix} \textsc{select} & \boxed{1} \\ \textsc{mark} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textit{functor-head-phrase} & \\ \textsc{non-head-dtr}|\textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{head}|\textsc{marker} & \begin{bmatrix} \textsc{select} & \boxed{1} \\ \textsc{mark} & \boxed{2} \\ \textsc{prehead} & \begin{bmatrix} \textsc{select} & \boxed{1} \\ \textsc{mark} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 3.13: Constraints on *head-functor-phrase* and *functor-head-phrase*

In order for these constraints to play the intended role, the two syntactic rules *head-functor-phrase* and *functor-head-phrase*, depicted in Figure 3.10, must be further refined. Their definitions are in Figure 3.13.

Because the higher SELECT attribute is already unified with the SYNSEM of the head daughter, and the higher MARK with the MARKING of the mother node, the homonymous features under PREHEAD and POSTHEAD will also be unified with these, but only when the relevant syntax rule is used.

We show an example involving a PP preceding a verbal projection in Figure 3.14.

3.4.2 Composition of Semantics in Head-Functor Phrases

The composition of semantics comes as expected and is presented in Figure 3.15.[4]

As all semantic information comes from the daughters, constructional content (C-CONT) is vacuous — these constructions do not add any semantics to the semantics conveyed by the daughters —, and the RELS and HCONS of the mother are the union of the homonymous features of the daughters.

We present a simplified scenario regarding the information under the HOOK of the mother node. As discussed elsewhere (Kasper, 1996; Copestake *et al.*, 2005), there are issues (concerning the feature LTOP) regarding the interaction between intersective and scopal modifiers. The next paragraphs describe the problem at hand.

Consider an example like *possibly brown cat*, where the adjective *brown* is an intersective modifier of the noun *cat*, and the adverb *possibly* is a scopal modifier of the adjective *brown*.

Intersective modifiers unify their LTOP feature with the LTOP of the synsem they select via their SELECT attribute in their lexical entries, so that the relations for the modifying element and the modified

ments of prepositions cannot be null (hence they cannot unify with *unexpressed-synsem*), and they cannot be extracted, either (hence they must be incompatible with *gap*).

[4]Once again, set notation is adopted instead of the implementation-level difference lists. But here the attributes RELS and HCONS are understood as multi-sets (bags).

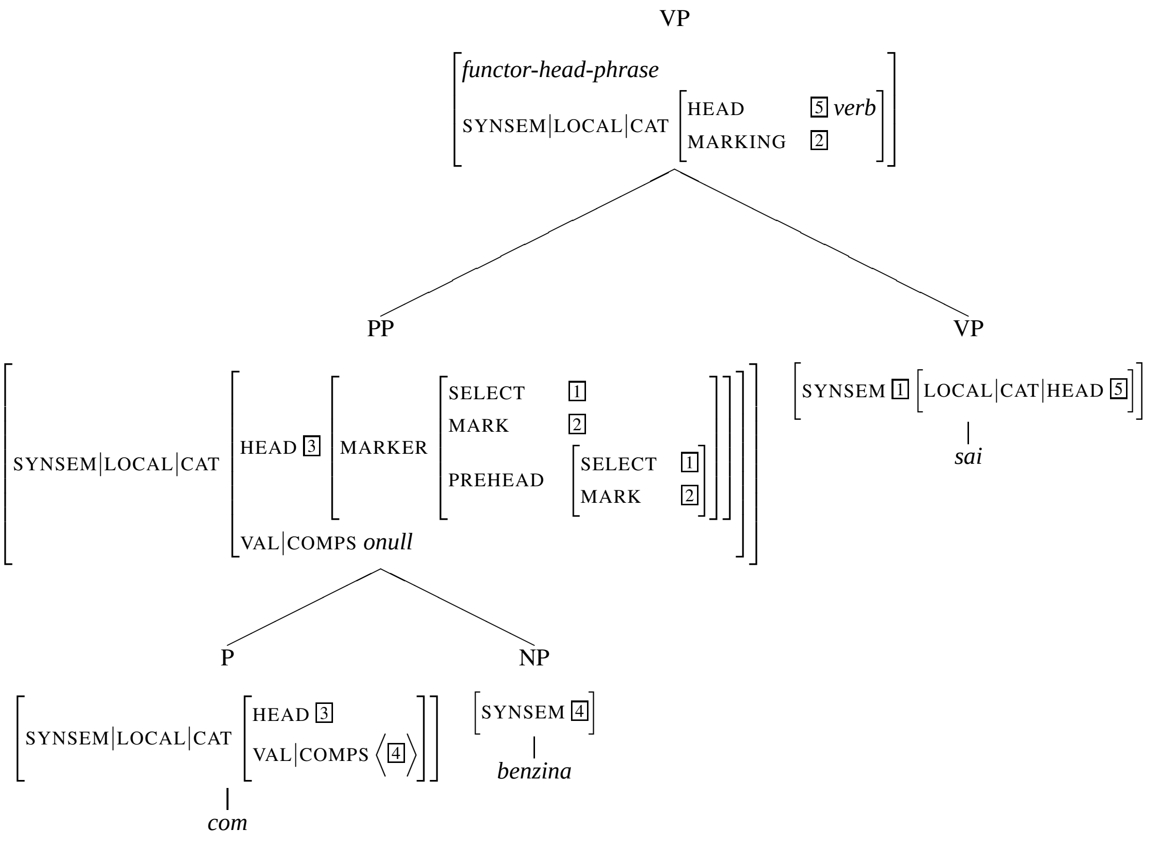

VP
functor-head-phrase
SYNSEM|LOCAL|CAT HEAD 5 verb MARKING 2
PP
SYNSEM|LOCAL|CAT HEAD 3 MARKER SELECT 1 MARK 2 PREHEAD SELECT 1 MARK 2
VAL|COMPS onull
VP
SYNSEM 1 LOCAL|CAT|HEAD 5
sai
P
SYNSEM|LOCAL|CAT HEAD 3 VAL|COMPS ⟨4⟩
com
NP
SYNSEM 4
benzina

Figure 3.14: Example parse tree for a VP modified by a preceding PP. The Portuguese phrase is *com benzina sai* (*goes away with benzine*) (see (21b) for a sentence where it can occur).

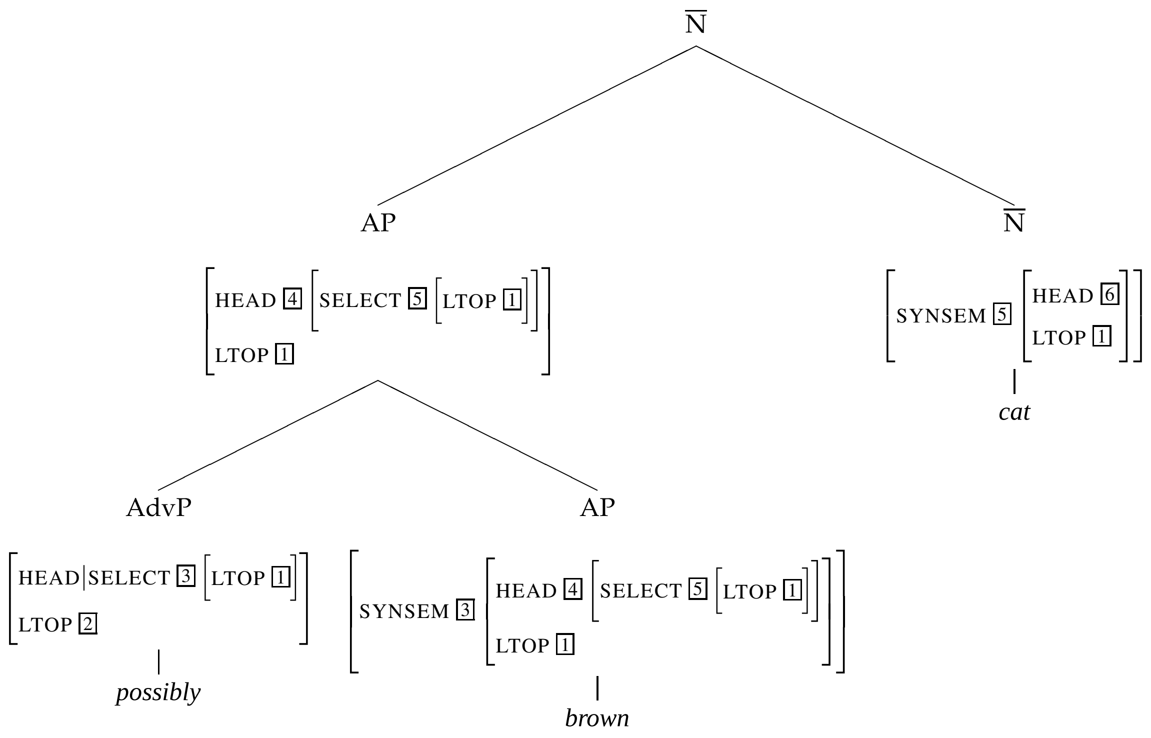
$$\begin{bmatrix} \textit{basic-head-functor-phrase} & \\ \text{SYNSEM}|\text{LOCAL}|\text{CONT} & \begin{bmatrix} \text{HOOK} & \begin{bmatrix} \text{LTOP} & \boxed{1} \\ \text{INDEX} & \boxed{2} \end{bmatrix} \\ \text{RELS} & \boxed{A} \cup \boxed{C} \cup \boxed{E} = \boxed{A} \cup \boxed{C} \\ \text{HCONS} & \boxed{B} \cup \boxed{D} \cup \boxed{F} = \boxed{B} \cup \boxed{D} \end{bmatrix} \\ \text{HEAD-DTR}|\text{SYNSEM}|\text{LOCAL}|\text{CONT} & \begin{bmatrix} \text{HOOK}|\text{INDEX} \ \boxed{2} \\ \text{RELS} \ \boxed{A} \\ \text{HCONS} \ \boxed{B} \end{bmatrix} \\ \text{NON-HEAD-DTR}|\text{SYNSEM}|\text{LOCAL}|\text{CONT} & \begin{bmatrix} \text{HOOK}|\text{LTOP} \ \boxed{1} \\ \text{RELS} \ \boxed{C} \\ \text{HCONS} \ \boxed{D} \end{bmatrix} \\ \text{C-CONT} & \begin{bmatrix} \text{RELS} & \boxed{E} & \{\} \\ \text{HCONS} & \boxed{F} & \{\} \end{bmatrix} \end{bmatrix}$$

Figure 3.15: Semantic constraints on Head-Functor phrases

one end up with the same LBL in the MRS (as this situation denotes conjunction of the two relations). Note that in general the LTOP of a lexical entry will be unified with the LBL of a relation in that item's RELS, so unifying LTOPs amounts to unifying LBLs in MRSs.

As for scopal modifiers (as well as determiners, which now, as functors, combine via the same rules), they do not identify these values but rather use the LTOP of the selected constituent as the value for one of the arguments of the relation they introduce, often mediated by a *qeq* constraint in the functor's HCONS — and all of this is done in the lexical entries for functors.

This yields the wrong semantics for phrases like *possibly brown cat*. What is produced is equivalent to $\lambda x.possible(brown(x) \wedge cat(x))$, but what is correct is $\lambda x.possible(brown(x)) \wedge cat(x)$ (feature paths are shortened in the derivation tree):


N̄
AP
HEAD 4 [SELECT 5 [LTOP 1]]
LTOP 1
N̄
SYNSEM 5 [HEAD 6, LTOP 1]
cat
AdvP
HEAD|SELECT 3 [LTOP 1]
LTOP 2
possibly
AP
SYNSEM 3 [HEAD 4 [SELECT 5 [LTOP 1]], LTOP 1]
brown

As can be seen, because LTOPs are unified in the lexicon, semantic scope and syntactic scope do not match.

There are two solutions in the literature. One is in Copestake *et al.* (2005): LTOP features are not unified in the lexicon, but in the syntax rules. This requires separate rules for intersective modification (where LTOP attributes are unified) and scopal modification (where these features are not unified). The other is in (Kasper, 1996): in order to obtain the desired result, the number of features used for the composition of semantics is enriched.

In order to simplify our exposition, we will not be concerned with this issue in the remainder of this text, as its solutions are known. These solutions are compatible with the analyses that will be developed. We will assume that the LTOP of the mother node is the LTOP of the functor daughter, and that LTOPs are unified in the lexical entries for intersective modifiers.

3.5 Implementation Details

As presented so far, the feature geometry under MARKER allows the attributes SELECT and MARK to occur at different levels of embedding (cf. Figure 3.11).

It is worth noting that it does not make sense to have yet another attribute PREHEAD or POSTHEAD under an attribute PREHEAD or POSTHEAD, because the two syntactic rules for functors only constrain the paths MARKER|PREHEAD and MARKER|POSTHEAD that are directly under the HEAD feature of the functor daughter. In fact, if the feature MARKER were declared to be of the type *marker* and this type were defined as:

$$\begin{bmatrix} \textit{marker} & \\ \textsc{select} & \textit{synsem} \\ \textsc{mark} & \textit{marking} \\ \textsc{prehead} & \textit{marker} \\ \textsc{posthead} & \textit{marker} \end{bmatrix}$$

one would get structures of infinite size in order to satisfy total well-typedness (the features PREHEAD and POSTHEAD must be present in every instance of *marker*, and these features are themselves of this type — see the paragraph on total well-typedness in Section 2.5).

The problem of structures of infinite size is avoided by resorting to two types, say *marker* and *marker-full*, where *marker-full* is the only subtype of *marker*, and these types are defined as:

$$\begin{bmatrix} \textit{marker} & \\ \textsc{select} & \textit{synsem} \\ \textsc{mark} & \textit{marking} \end{bmatrix} \qquad \begin{bmatrix} \textit{marker-full} & \\ \textsc{prehead} & \textit{marker} \\ \textsc{posthead} & \textit{marker} \end{bmatrix}$$

In the head type *functor*, MARKER is declared to be of type *marker*. With this design, one gets totally well-typed feature structures of finite size, but it is still possible to have a constraint on a feature PREHEAD or POSTHEAD under another feature PREHEAD or POSTHEAD, because the two types, *marker* (the type that these features are declared to be) and *marker-full*, (the most general type for which they are appropriate) are compatible. That is, the following instance would be valid:

$$\begin{bmatrix} \textsc{marker} & \begin{bmatrix} \textit{marker} \sqcap \textit{marker-full} = \textit{marker-full} & \\ \textsc{select} & \textit{synsem} \\ \textsc{mark} & \textit{marking} \\ \textsc{prehead} & \begin{bmatrix} \textit{marker} & \\ \textsc{select} & \textit{synsem} \\ \textsc{mark} & \textit{marking} \end{bmatrix} \\ \textsc{posthead} & \begin{bmatrix} \textit{marker} \sqcap \textit{marker-full} = \textit{marker-full} & \\ \textsc{select} & \textit{synsem} \\ \textsc{mark} & \textit{marking} \\ \textsc{prehead} & \begin{bmatrix} \textit{marker} & \\ \textsc{select} & \textit{synsem} \\ \textsc{mark} & \textit{marking} \end{bmatrix} \\ \textsc{posthead} & \begin{bmatrix} \textit{marker} & \\ \textsc{select} & \textit{synsem} \\ \textsc{mark} & \textit{marking} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

In this example, the presence of the most embedded features PREHEAD and POSTHEAD makes the sort of type inference discussed in Section 2.5 determine that the topmost feature POSTHEAD must be of the type *marker-full*, as *marker-full* is the most general type for which PREHEAD and POSTHEAD are appropriate. The type of the topmost POSTHEAD feature is then the unifier of *marker-full* (the inferred type) and *marker* (the type that this feature PREHEAD is declared to be), which is *marker-full*.

The situation of a PREHEAD or POSTHEAD feature embedded under another PREHEAD or POSTHEAD attribute would not occur in parsing (as a result of unifying parsed nodes with the daughters of syntactic rules), but it can occur in the code, in which case it is an error.[5]

A very simple type hierarchy for the values that the feature MARKER can take, such as the one presented in Figure 3.16, can be devised to turn such errors into type errors, therefore detectable statically.

$$\begin{tikzcd}
 & \textit{marker} & \\
\textit{marker-full} \arrow[ur] & & \textit{marker-simple} \arrow[ul]
\end{tikzcd}$$

Figure 3.16: Type hierarchy for values of MARKER (version 1/3). Final version on p. 61.

The feature MARKER is declared in type *functor* (see Figure 3.6) to be of type *marker*, where the attributes SELECT and MARK are declared:

$$\begin{bmatrix} \textit{marker} & \\ \text{SELECT} & \textit{synsem} \\ \text{MARK} & \textit{marking} \end{bmatrix}$$

The features PREHEAD and POSTHEAD are defined in *marker-full*. They are both defined to be of type *marker-simple*:

$$\begin{bmatrix} \textit{marker-full} & \\ \text{PREHEAD} & \textit{marker-simple} \\ \text{POSTHEAD} & \textit{marker-simple} \end{bmatrix}$$

Since *marker-full* is a subtype of *marker*, it inherits the features SELECT and MARK, and it can occur as a value of a feature declared to be of type *marker* (the feature MARKER in this example). The type *marker-simple* has no additional constraints, but it also inherits the features SELECT and MARK from *marker*. There is no unifier for *marker-simple* and *marker-full*, so the attributes PREHEAD and POSTHEAD cannot be embedded under other features PREHEAD or POSTHEAD.

If POSTHEAD and PREHEAD are not constrained in a feature structure under the respective MARKER feature (e.g. it shows constraints on SELECT and MARK only), PREHEAD and POSTHEAD will not appear in that definition, since MARKER is of type *marker*, which does not include these features. If PREHEAD or POSTHEAD are constrained, type inference determines that MARKER must be of type *marker-full*, since that is the most general type for which these features are appropriate.

3.5.1 Addition of Minimal Types

Problem

Consider once again the constraints on the HEAD of prepositions presented above in Figure 3.12. With the type hierarchy and the type declarations for the features MARKER (*marker*) and PREHEAD (*marker-*

[5]For instance, it can occur as a consequence of a constraint of the form ...|POSTHEAD|POSTHEAD|..., instead of the intended form ...|POSTHEAD|...

simple) that have just been presented, type inference determines the following types in the structure (where *marker* is the type that the attribute MARKER is declared to be and *marker-full* is the most general type for which the attribute PREHEAD is appropriate):[6]

$$\begin{bmatrix} \textit{preposition} & \\ \textsc{marker} & \begin{bmatrix} \textit{marker} \sqcap \textit{marker-full} = \textit{marker-full} & \\ \textsc{select}|\textsc{local}|\textsc{cat}|\textsc{head}\ \textit{noun-or-verb} & \\ \textsc{mark}\ \textit{marking} & \\ \textsc{prehead} & \begin{bmatrix} \textit{marker-simple} & \\ \textsc{select}|\textsc{local}|\textsc{cat}|\textsc{head} & \textit{verb} \\ \textsc{mark}\ \textit{marking} & \end{bmatrix} \\ \textsc{posthead} & \begin{bmatrix} \textit{marker-simple} & \\ \textsc{select} & \textit{synsem} \\ \textsc{mark} & \textit{marking} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Because of type expansion, the feature POSTHEAD is also present under MARKER: the presence of the feature PREHEAD makes the type of MARKER be *marker-full*. Since *marker-full* is declared to have a feature POSTHEAD, this feature has to be added for the structure to be totally well-typed. Additionally, a feature MARK is added by type expansion under MARKER (it is a feature of *marker* and *marker-full*) and under PREHEAD (inherited by *marker-simple* from *marker*).

The features POSTHEAD and MARK in such a structure are completely uninformative, since they take as values the most general types that are valid for them. The problem is that, because PREHEAD and POSTHEAD are declared in the same type, when one of them is present in a feature structure, the other one has to be present as well in order to satisfy total well-typedness. This is a result of the interaction between type inference and type expansion. A similar situation holds for the features SELECT and MARK.

This consequence does not affect correction, but it has a potential negative impact on efficiency, as unification of larger feature structures is more expensive than unification of smaller ones (because every feature and subfeature must be unified). It also obscures feature structures, by making them contain information that is irrelevant.

Approach

The type hierarchy just presented for the feature MARKER can be extended with so-called minimal types, in order to reduce the size of feature structures in the lexicon and the feature structures manipulated by the grammar at run time. Minimal types are used in several LKB grammars, like the LinGO English Resource Grammar, and in the LinGO Grammar Matrix.

The idea behind the use of minimal types is to enrich the type hierarchy so that features are present in a feature structure only if they are constrained. Flickinger (2000) proposes this kind of strategy.

Consider a revised hierarchy for MARKER objects in Figure 3.17.

[6]In this AVM and in the following ones we do not expand the feature structures under the SELECT attributes, which will be much larger than shown after type expansion.

$$\begin{tikzcd}
 & & \textit{marker-min} & \\
 & \textit{marker} \arrow[ur] & & \textit{marker-simple-min} \arrow[ul] \\
\textit{pre-marker} \arrow[ur] & \textit{post-marker} \arrow[u] & & \textit{marker-simple} \arrow[ull] \arrow[u] \\
\textit{pre-or-post-marker} \arrow[u] \arrow[ur] & & &
\end{tikzcd}$$

Figure 3.17: Type hierarchy for values of MARKER (version 2/3). Previous version on p. 56. Final version on p. 61.

With this hierarchy, SELECT and MARKER are still declared in *marker*, as before:

$$\begin{bmatrix} \textit{marker} \\ \text{SELECT } \textit{synsem} \\ \text{MARK } \textit{marking} \end{bmatrix}$$

In *functor*, the feature MARKER is defined to be *marker-min*, instead of being of type *marker*:

$$\begin{bmatrix} \textit{functor} \\ \text{MARKER } \textit{marker-min} \end{bmatrix}$$

The type *marker-min* is declared with no features.

The attributes PREHEAD and POSTHEAD are declared in different types: PREHEAD in *pre-marker* and POSTHEAD in *post-marker*. They are now both of the type *marker-simple-min*:

$$\begin{bmatrix} \textit{pre-marker} \\ \text{PREHEAD } \textit{marker-simple-min} \end{bmatrix} \qquad \begin{bmatrix} \textit{post-marker} \\ \text{POSTHEAD } \textit{marker-simple-min} \end{bmatrix}$$

The type *marker-simple-min* has no features. The features SELECT and MARK can however be used under PREHEAD or POSTHEAD: e.g. if a constraint mentions the feature SELECT under PREHEAD the type of that PREHEAD instance is inferred to be *marker-simple*, the unifier of *marker*, where SELECT is declared, and *marker-simple-min*, the type that the feature PREHEAD is declared to be.

The revised hierarchy and type definitions produce the feature structure in Figure 3.18 for the HEAD of prepositions once type inference and type expansion are performed (from the definition presented above in Figure 3.12).

In this structure, the type of the feature MARKER is the unifier of the types *marker-min*, *marker* and *pre-marker*, which is *pre-marker*. The type *marker-min* is the type that this feature is declared to be, *marker* is the type the feature structure under MARKER is inferred to be because of the presence of SELECT, and *pre-marker* is the type the same structure is inferred to be because of the presence of the feature PREHEAD.

The type of the feature structure under PREHEAD is the unifier of the types *marker-simple-min* and *marker*, which is *marker-simple*. The type *marker-simple-min* is the type this feature is declared to be, and *marker* is the type it is inferred to be because of the constrained SELECT feature under it. The attributes

$$\begin{bmatrix} \textit{preposition} & \\ \textsc{marker} & \begin{bmatrix} \multicolumn{2}{l}{\textit{marker-min} \sqcap \textit{marker} \sqcap \textit{pre-marker} = \textit{pre-marker}} \\ \multicolumn{2}{l}{\textsc{select}|\textsc{local}|\textsc{cat}|\textsc{head}\ \textit{noun-or-verb}} \\ \multicolumn{2}{l}{\textsc{mark}\ \textit{marking}} \\ \textsc{prehead} & \begin{bmatrix} \textit{marker-simple-min} \sqcap \textit{marker} = \textit{marker-simple} \\ \textsc{select}|\textsc{local}|\textsc{cat}|\textsc{head}\quad \textit{verb} \\ \textsc{mark}\ \textit{marking} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 3.18: Feature structure for the type *preposition* after type expansion

MARK under MARKER and PREHEAD are still added by type expansion, because of the presence of the feature SELECT, as SELECT and MARK are still declared in the same type, *marker*.

We see a reduction in the size of the feature structure, as the attribute POSTHEAD is no longer added by type expansion.[7]

Sometimes the reduction in the size of the feature structures only occurs in the feature structures of lexical items. However, sometimes the feature structures manipulated at run time are reduced in size, too. The constraints on prepositions just presented illustrate this point. Because a constraint is put inside the PREHEAD attribute in the definition of the type *preposition*, the type for MARKER of prepositions is inferred to be *pre-marker*, and the feature PREHEAD will always be present. In the cases where a preposition follows the head, unification with the functor daughter in *head-functor-phrase* will cause the type of MARKER to be *pre-or-post-marker*, in which case both PREHEAD and POSTHEAD will be present. In this case no reduction in the size of feature structures is obtained at run time.[8] However, in the cases where the preposition precedes the head, it remains of this type (and the corresponding feature structure is still like that in Figure 3.18), and the feature POSTHEAD is not present.

On the other hand, when the definition of a functor does not constrain PREHEAD or POSTHEAD, a reduction in the size of the feature structures is always obtained: PREHEAD will only be present in the feature structures when that functor feeds the *functor-head-phrase*, and POSTHEAD will only be present when that functor feeds the *head-functor-phrase*.

Discussion

The advantage of using minimal types is that feature structures are smaller. Because unification of feature structures is recursive, smaller structures make unification operations more efficient. It also has the nice side effect that the features that are left completely unconstrained are not present in the final feature structures, which makes them more succinct.

The disadvantage is that more type unifications must be performed at run time. Consider the example of prepositions.

With the first version of the hierarchy of values for the feature MARKER, presented in Figure 3.16, both the MARKER of prepositions and the MARKER of the non-head daughter of *head-functor-phrase* and *functor-head-phrase* are of the type *marker-full* after type inference. For the two structures to unify (the structure under the MARKER of the preposition and the structure under the MARKER feature of the

[7]We also see a reduction in the feature structures of the two syntactic rules for functors presented above in Chapter 3.4.1. With the previous hierarchy for MARKER values, the features PREHEAD and POSTHEAD would be present in both rules, as the presence of one of them would cause the addition of the other after type inference and type expansion. With the new hierarchy, they are as shown in Figure 3.13, as far as the feature structure under MARKER is concerned, and once again ignoring type expansion of the *synsem* objects under the SELECT features.

[8]In any case, a preposition must combine with an NP to its right before the resulting node attaches to the element it modifies, via the Head-Functor rules. Before the Head-Functor rules are used, the feature structures that are manipulated by the parser are also smaller.

non-head daughter of one of these rules), their type must be unifiable and the features they contain must be unifiable. The first part — type unification (i.e. comparison of two types to determine their most general unifier, or greatest lower bounds) — is trivial to compute in this example (*marker-full* ⊓ *marker-full*), as unification is idempotent.

With the revised hierarchy, the MARKER attribute of prepositions is of type *pre-marker*, as described above. The type of the MARKER feature under the non-head daughter of *functor-head-phrase* is also *pre-marker* (because PREHEAD is constrained there), and in this case type unification is also trivial. But in *head-functor-phrase* the same feature is of type *post-marker*. In order to unify the two structures (the structure under the MARKER feature of the preposition's HEAD and the structure under the MARKER attribute of the non-head daughter of *head-functor-phrase*) it is thus necessary to find a subtype common to *pre-marker* and *post-marker* (which is *pre-or-post-maker*). In this case, it is necessary to travel along the type hierarchy, and type unification will be more costly.

However, the LKB algorithm that computes type unification has approximately constant complexity at run time (Malouf *et al.*, 2000).

It would be truly constant at run time if the LKB and PET precomputed the unification of all pairs of types in the type hierarchy at grammar compile/load time and stored them in a data structure with constant time random access, like a hash table. In the LKB this is not done, because unification of most pairs of types in the type hierarchy would fail and is never going to be tried, as they appear in completely different contexts. For instance, consider that no bug-free grammar is going to try to unify a feature HEAD with a feature VAL, as they take completely different values (the most general types appropriate for these features are incompatible, therefore unification is deemed to fail).

Instead, the LKB stores the results of previous unifications in a hash table. Malouf *et al.* (2000) mention that there is "no appreciable runtime performance cost compared with full table lookup" and that in fact around 99% of the types compared are found in this cache when the grammar used is the LinGO English Resource Grammar. Therefore, the overhead of more type unifications should not be detrimental in practice. Flickinger (2000) reports improvements in performance associated with the use of minimal types, which means that more efficient unification of feature structures (they are smaller) compensates for more type unifications.

Of course, the price to pay for using this technique is having more complicated hierarchies (there must be unifiers for all types introducing the features that can co-occur), besides more unification operations at run time.

The type hierarchy under *marker-min* could be further elaborated, with the purpose of reducing feature structures. For instance, if the feature SELECT is constrained, the attribute MARK will also be added to the relevant feature structure, and vice-versa, as a consequence of type inference and total well-typedness, because both features are introduced in the same type. This situation would not arise if different types were used to introduce them. As we will see in the following chapters, SELECT and MARK are usually constrained together, so such an optimization would possibly have little effect.

In several type hierarchies of the grammar, we use this technique in order to reduce the size of feature structures. In the following chapters, we will sometimes present hierarchies that are simplified in that minimal types are not shown. The hierarchies presented are equivalent to the ones used in the grammar, but we omit minimal types whenever possible, as they obscure type hierarchies and we have already presented their purpose here.

3.6 Lexical Constraints on Rule Application

A final extension to the type hierarchy of MARKER values produces the hierarchy in Figure 3.19, which is the one that will be used.

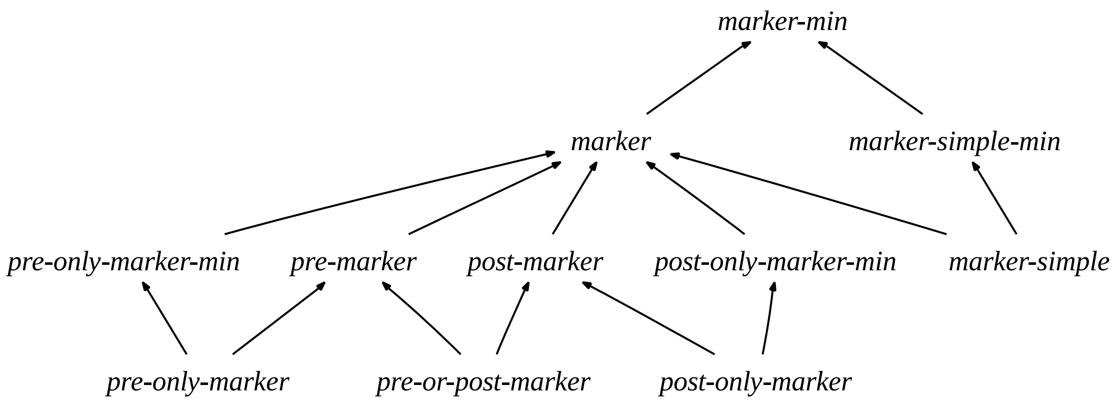


Figure 3.19: Type hierarchy for values of MARKER (final version — 3/3). Previous version on p. 58.

The types *pre-only-marker* and *post-only-marker* in the hierarchy in Figure 3.19 serve a purpose different from the one of the minimal types. They are there to block rule application of *functor-head-phrase* or *head-functor-phrase*, making it possible to prevent specific functors from appearing in one of the two syntactic rules for functors.

The types *pre-only-marker-min* and *post-only-marker-min* are the corresponding minimal types (they do not inherit from *pre-marker* or *post-marker*, so they lack the features PREHEAD and POSTHEAD). These new types have no associated constraints, besides the constraints they inherit from their supertypes.

The new types are useful in cases where a functor cannot precede or follow its head. For instance, cardinals never follow the noun they modify in Portuguese:

(22) a. as duas lentes
the two lenses
the two lenses

b. * as lentes duas
the lenses two

Accordingly, their MARKER feature can be constrained to be of type *pre-only-marker-min* (among other constraints — more details in Chapter 4):

$$\left[\text{MARKER}\ \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{pre-only-marker-min}} \\ \text{SELECT}|\text{LOCAL}|\text{CAT}|\text{HEAD} & \textit{noun} \end{array}\right]\right]$$

The type *pre-only-marker-min* does not have a feature POSTHEAD. The feature POSTHEAD is declared in type *post-marker*. According to the hierarchy in Figure 3.19, there is no unifier for *pre-only-marker-min* and *post-marker*. Therefore, POSTHEAD cannot be under the MARKER attribute of a cardinal. This makes it impossible for a cardinal to be in the functor daughter (the non-head daughter) of a *head-functor-phrase*, where this feature is constrained. Type inference determines that the MARKER feature of the functor daughter of *head-functor-phrase* must be of type *post-marker*, which is not unifiable with *pre-only-marker-min*.

Similarly, *post-only-marker-min* does not have a feature PREHEAD: PREHEAD is declared in the type *pre-marker*, and according to the same hierarchy there is no unifier for *post-only-marker-min* and *pre-marker* either. Elements with a MARKER feature constrained to be *post-only-marker-min* are thus not eligible to be the non-head daughter of *functor-head-phrase*.

After type expansion, the attribute PREHEAD will not be present under the MARKER feature of cardinals, given the above definition: it is not constrained in that definition, and the feature PREHEAD is appropriate for type *pre-marker*, which is not a supertype of *pre-only-marker-min*. Type expansion will add the feature MARK, as it is also appropriate for type *pre-only-marker-min* (it inherits this feature from *marker*):

$$\begin{bmatrix} \textsc{marker} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \textsc{select}|\textsc{local}|\textsc{cat}|\textsc{head} & \textit{noun} \\ \textsc{mark}\ \textit{marking} & \end{bmatrix} \end{bmatrix}$$

Although PREHEAD is not present, a functor with such a head is nonetheless eligible to feed the *functor-head-phrase* rule, as *pre-only-marker-min* (the type of MARKER in the lexical item) and *pre-marker* (the type of MARKER in the functor daughter of *functor-head-phrase*) are unifiable (they unify to *pre-only-marker* according to the hierarchy in Figure 3.19).

In the lexicon, the feature structures for cardinals are thus reduced in size, as the feature PREHEAD is not present in these structures (using the constraints just presented). At run time, when a cardinal is matched with the non-head daughter of *functor-head-phrase*, the feature PREHEAD is added to the structure corresponding to the cardinal. The feature POSTHEAD is never present.

3.7 Example

As a way of illustration of the technical details discussed so far, we present the analysis with functors for the NP in (23). Full details are to be discussed in Chapter 4, where the machinery described in the present chapter is employed to model NP structure.

(23)	os	quatro	naipes
	the	four	suits
	the four suits		

Figure 3.20 includes a type hierarchy under *head* with all the head types mentioned so far.

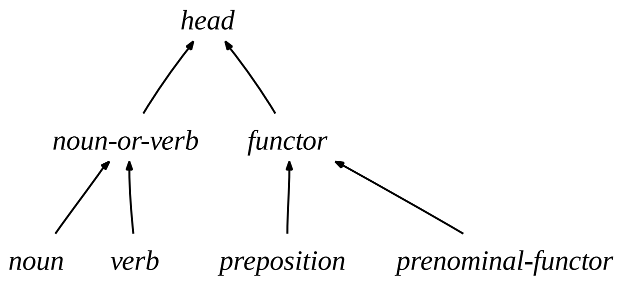
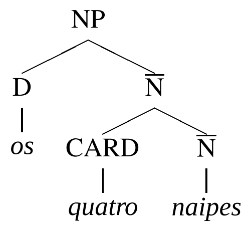


Figure 3.20: Type hierarchy under *head*

The type for the HEAD of determiners and cardinals can be defined as:

$$\begin{bmatrix} \textit{prenominal-functor} & \\ \text{MARKER} & \begin{bmatrix}\textit{pre-only-marker-min} & \\ \text{SELECT}|\text{LOCAL}|\text{CAT}|\text{HEAD} & \textit{noun}\end{bmatrix}\end{bmatrix}$$

The parse for the NP in (23) produced by this analysis is in Figure 3.21.



$$\begin{bmatrix} \textit{functor-head-phrase} & \\ \text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD}\ \boxed{1}\ \textit{noun} & \\ \text{NON-HEAD-DTR}|\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD} & \begin{bmatrix}\textit{prenominal-functor} & \\ \text{MARKER}|\text{SELECT} & \boxed{2}\end{bmatrix} \\ \text{HEAD-DTR} & \begin{bmatrix} \textit{functor-head-phrase} & \\ \text{SYNSEM}\ \boxed{2}\ \begin{bmatrix}\text{LOCAL}|\text{CAT}|\ \text{HEAD}\ \boxed{1}\end{bmatrix} & \\ \text{NON-HEAD-DTR}|\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD} & \begin{bmatrix}\textit{prenominal-functor} & \\ \text{MARKER}|\text{SELECT} & \boxed{3}\end{bmatrix} \\ \text{HEAD-DTR}|\text{SYNSEM}\ \boxed{3}\ \begin{bmatrix}\text{LOCAL}|\text{CAT}|\ \text{HEAD}\ \boxed{1}\end{bmatrix} & \end{bmatrix}\end{bmatrix}$$

Figure 3.21: AVM and corresponding tree for the NP *os quatro naipes* (*the four suites*).

PPs can already be blocked from occurring on the left of nouns. The result of a PP attaching to the left of noun-headed constituents is displayed in Figure 3.22. The two daughters of *functor-head-phrase* have incompatible constraints. The head daughter is constrained to have a HEAD attribute with the value *noun*, but the constraint under the PREHEAD attribute of the preposition requires an element with the value *verb* for this feature.

$$\begin{bmatrix} \textit{functor-head-phrase} & \\ \text{HEAD-DTR}|\text{SYNSEM}\ \boxed{1}\ \begin{bmatrix}\text{LOCAL}|\text{CAT}|\text{HEAD}\ \textit{noun} \sqcap \textit{noun-or-verb} \sqcap \textit{verb} = \bot\end{bmatrix} & \\ \text{NON-HEAD-DTR}|\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD}|\text{MARKER} & \begin{bmatrix}\text{SELECT}\ \boxed{1}\ \begin{bmatrix}\text{LOCAL}|\text{CAT}|\text{HEAD}\ \textit{noun-or-verb}\end{bmatrix} \\ \text{PREHEAD}|\text{SELECT}\ \boxed{1}\ \begin{bmatrix}\text{LOCAL}|\text{CAT}|\text{HEAD}\ \textit{verb}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Figure 3.22: Unification failure when trying to parse a PP and noun sequence via *functor-head-phrase*.

So far, nothing prevents parsing or generating the impossible NP in (24), and nothing so far allows a distinction between NPs and $\overline{\text{N}}$ s. This situation will be corrected in Chapter 4.

(24)	* quatro	os	naipes
	four	the	suits

The features MARKING and MARK are not list-valued, their type is *marking*. A hierarchy under this type is crucial to define phrase-structure. Chapter 4 develops a full analysis of NP structure for

Portuguese defined in terms of a type hierarchy of *marking* and constraints on the features MARKING and MARK using the values in that hierarchy.

3.8 Comments on the Functor Architecture

With the setup just presented, functors have visibility over the constituents they attach to through the SELECT features, but these constituents do not have any visibility over functors. This presents no problem.

For the purpose of the composition of semantics, it is necessary that determiners have access to their sister node, which is what happens with this functor approach. It is not necessary that nouns have visibility over determiners.

Consider the example in Figure 3.23, for the Portuguese NP *um falso carro* (*a false car*).

In this example, the RSTR of the quantifier relation introduced by the determiner is equated via a *qeq* constraint in the determiner's lexical entry with the LTOP of the phrase *falso carro* (*false car*), which in that figure is tagged with $\boxed{h3}$. This is also the LTOP of the adjective, not of the noun (the noun's LTOP is tagged with $\boxed{h5}$ in that figure), and the noun does not have access to the adjective's LTOP. The semantics for the NP is equivalent to $\lambda P.\_um\_q(x, \_falso\_a(e, \_carro\_n(x)), P(x))$, which translates to first-order logic as $\lambda P.\exists x[\neg\_carro\_n(x) \wedge P(x)]$.

If determiners happened to not have access to their sister node, the only way to get the semantics right would be to create a special syntactic rule for determiner attachment, that fills the LARG of the *qeq* introduced by the determiner with the appropriate value. This value cannot be filled in the lexical entries of nouns. In this example, the LTOP tagged with $\boxed{h3}$ is not visible in the noun's lexical entry. The only LTOP visible there is the noun's LTOP, with the tag $\boxed{h5}$. One could think of filling the LARG of the determiner's *qeq* constraint with this value (the noun's LTOP) in the feature structure for nouns, as in the following constraints (assuming the SPR feature and Specifier-Head constructions):

$$\begin{bmatrix} \text{SYNSEM}|\text{LOCAL} & \begin{bmatrix} \text{CAT}|\text{VAL}|\text{SPR} & \left\langle \begin{bmatrix} \text{LOCAL}|\text{CONT}|\text{HCONS} & \left\{ \begin{bmatrix} \text{LARG} \ \boxed{1} \end{bmatrix} \right\} \end{bmatrix} \right\rangle \\ \text{CONT}|\text{HOOK}|\text{LTOP} \ \boxed{1} & \end{bmatrix} \end{bmatrix}$$

This would produce an MRS where it is stated that both the determiner and the adjective relations must scope over the noun relation with the possibility of quantifier relations in between. Since the relation contributed by the adjective is not a quantifier relation, one would not get the desired formula. Instead one would get the equivalent of $\lambda P.\_falso\_a(\_um\_q(x, \_carro\_n(x), P(x)))$, which is not what is intended (e.g. *A false car was sold* does not mean *It is false that a car was sold*).

This problem exists in every NP where the LTOP of the noun and the LTOP of the determiner's sister node are different.

This is what motivated the SPEC feature in specifier approaches (Pollard and Sag, 1994, p.50, adapted here to the MRS universe). In the specifiers approach, nouns have visibility over determiners (via the SPR feature) and determiners also have visibility over their sister node (via the SPEC feature). Nouns select the syntactic properties of the determiners they can co-occur with, and determiners look at semantic properties of their sister node.

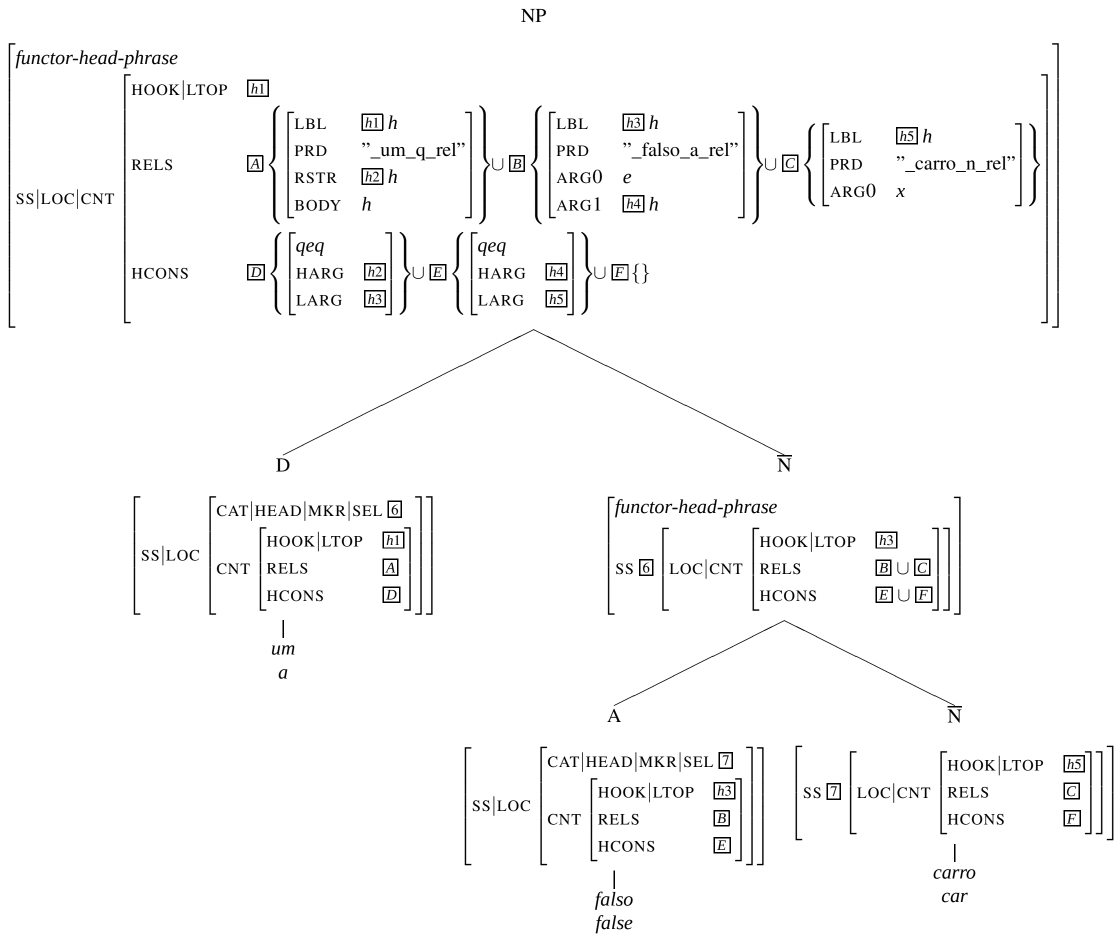

NP
functor-head-phrase
SS|LOC|CNT
HOOK|LTOP h1
RELS A { [LBL h1 h, PRD "_um_q_rel", RSTR h2 h, BODY h] } ∪ B { [LBL h3 h, PRD "_falso_a_rel", ARG0 e, ARG1 h4 h] } ∪ C { [LBL h5 h, PRD "_carro_n_rel", ARG0 x] }
HCONS D { [qeq, HARG h2, LARG h3] } ∪ E { [qeq, HARG h4, LARG h5] } ∪ F {}
D
SS|LOC [CAT|HEAD|MKR|SEL 6, CNT [HOOK|LTOP h1, RELS A, HCONS D]]
um
a
N̄
functor-head-phrase
SS 6 [LOC|CNT [HOOK|LTOP h3, RELS B ∪ C, HCONS E ∪ F]]
A
SS|LOC [CAT|HEAD|MKR|SEL 7, CNT [HOOK|LTOP h3, RELS B, HCONS E]]
falso
false
N̄
SS 7 [LOC|CNT [HOOK|LTOP h5, RELS C, HCONS F]]
carro
car

Figure 3.23: Visibility of determiners over nouns. SS abbreviates SYNSEM, LOC abbreviates LOCAL, MKR abbreviates MARKER, SEL abbreviates SELECT, CNT abbreviates CONT, PRD abbreviates PRED.

With the functors approach, in turn, syntactic selection is enforced from determiners by way of their feature structures. This is possible as long as the relevant information to be selected is encoded in the feature structures of nouns and percolated up to the determiner's sister phrase.

The functors approach is therefore more parsimonious than the specifiers approach with respect to the number of selection features: we only use one feature, SELECT, through which determiners (as well as all other functors) select the head. This feature is required, since determiners must have access to their sister node, for the reasons related to the composition of semantics that have just been described. A feature similar to the SPR feature is not necessary in the functors approach, because the saturation of the head is encoded in a different feature, MARKING.

The functors design does have one consequence — agreement must be enforced in the functors. Assuming agreement is encoded in the feature AGR, as in the LinGO Grammar Matrix, if a functor exhibits agreement with the head noun (e.g. most Portuguese determiners, adjectives, etc.) the unification of the AGR feature of the noun with that of the functor must be stated in the lexical entry for the functor.

For instance, with the LinGO Grammar Matrix setup, AGR is a feature under LOCAL, and has subfeatures that encode information about person, number and gender. In LXGram, the gender of nouns is specified in lexical entries for nouns, and the value of the feature where number information is present is instantiated by inflectional rules. For the other morpho-syntactic classes that show variation in number and gender (adjectives, determiners, etc.), this information is also filled in by inflection rules. The feature AGR is percolated (passed up to the mother node from one of the daughters via unification, in this case from the head daughter) in all headed constructions.[9] In our approach, a functor that exhibits agreement with its head noun must unify its AGR with that of the noun it selects. This is achieved by constraints like:

$$\begin{bmatrix} \text{SYNSEM}|\text{LOCAL} \begin{bmatrix} \text{CAT}|\text{HEAD}|\text{MARKER}|\text{SELECT}|\text{LOCAL}|\text{AGR } \boxed{1} \\ \text{AGR } \boxed{1} \end{bmatrix} \end{bmatrix}$$

In the exposition in the following chapters, agreement constraints are not shown, since they are simply implemented via unification of the AGR features, when appropriate.

3.9 Summary

We considered the various syntactic relations that are used to model dependencies between syntactic constituents. In HPSG an attribute SUBJ encodes information of what constitutes a possible subject of a word. An attribute COMPS points to information about its complements. Some other attributes are used in the literature.

We resorted to a feature SELECT, based on the work of Allegranza (1998a,b) and Van Eynde (2003a,b). This attribute is appropriate for elements that can serve the syntactic function of functor, which is an abstraction over the more usual linguistic concepts of adjunct and specifier. The feature SELECT of a functor item is where it is stated what that item's syntactic sister node in a derivation tree must be like. Examples include adjectives, prepositional phrases, cardinals, ordinals, determiners, etc.

[9]For this reason, and because AGR is meaningless to some items (e.g. Portuguese prepositions do not show any inflection, so their AGR would always be completely underspecified), we chose to change the matrix.tdl (the file that contains the Matrix definitions) we are using and make AGR a feature under HEAD. AGR is percolated in headed constructions, because HEAD is percolated in these constructions. The constraints we are using are somewhat different from the ones presented below, because AGR is in a different place, but the differences are immaterial.

We also defined the rules that pick this information and combine these elements with other constituents. The architecture accounts for word order possibilities between the different elements involved. The features PREHEAD and POSTHEAD are used to this end. In PREHEAD additional requirements are implemented that describe the constraints that must be satisfied when one of these elements precedes the element it selects for via the SELECT feature. Similarly, POSTHEAD contains the constraints that must be satisfied when the functor follows its head.

The way the semantic representations are built was also discussed. With the MRS mode of composing semantics, the semantics of Head-Functor constructions is simply the combination of the relations and the handle constraints of the two daughters of this construction, the head and the functor. No extra information is needed.

The constraints on the syntax rules that are employed to combine functor and head were presented. In HPSG, syntax rules are also types, and can be organized in a type hierarchy just like any other types. We organized the Head-Functor constructions in a type hierarchy, taking advantage of inheritance in order to factor out the constraints that are common to several constructions. Very general types that are responsible for word order constraints and are already defined in the LinGO Grammar Matrix are used as supertypes of specific constructions that differ only with respect to word order.

It is also possible to state the specific constraints of Head-Functor constructions in a single type, that abstracts from word order. Using all these types, specific Head-Functor constructions with constrained word order are implemented via multiple inheritance.

Additionally, we provided implementation details that are justified by efficiency reasons. The specific logic of typed feature structures accepted by the systems where LXGram is run (the LKB and PET) can be exploited for efficiency reasons. Certain properties of the type system, like type inference, can be taken into consideration when the shape of a type hierarchy is defined, so that the data structures defined by the grammar and manipulated by the parsing and generation algorithms can have a reduced size, with a positive impact on performance (Flickinger, 2000). We illustrated this implementation strategy with the types and the attributes that are used in the functor architecture.

4
NP Syntax and Semantics

4.1 Overview

In this chapter we present a system of constraints that captures Portuguese NP structure and meaning, based on the functor architecture presented in Chapter 3. We will present a functor-based treatment of the syntax of several NP constituents, and we will also describe their semantics when appropriate. This analysis was adopted in the implementation of LXGram and as such has been thoroughly tested.

In the next section, an overview of the data under consideration is presented, and each of the remaining sections will cover a position inside the NP.

4.2 Data

The table in Appendix A presents a summary of the NP structure to be modeled, with examples of the relevant data. It is broken down into nine positions, that reflect word order.

Many of the elements in this table do not need to be atomic. For instance, adjectives can be modified or have complements of their own.

The category Predeterminers (Position I) in this table contains elements like *todo* (*all*).

The category Determiners (Position II) includes the definite and indefinite articles, the demonstratives and other items, like *bastante(s)* (*much*, *several*).

The category Possessives (Position III) contains prenominal possessives, which in European Portuguese are preceded by determiners.

The category Cardinals (in Position IV) includes the cardinal numerals, either atomic (*dois*, *two*) or complex (*vinte e dois*, *twenty two*).

The category Ordinals (in Position IV) includes the ordinal numerals, atomic (*primeiro*, *first*) and complex ones (*vigésimo primeiro*, *twenty-first*).

The category Vague Quantifiers (in Position IV) contains elements like *muitos* (*many*), *poucos* (*few*). The distinction between determiners like *bastantes* and vague quantifiers is not semantic but syntactic. Consider their different behavior with respect to a preceding definite article, exemplified in the following sentences:

(25) a. [$_{\text{NP}}$ Muitas espécies de sapos da Amazónia] já estão extintas.
many species of frogs of the Amazon Rainforest already are extinct
Many species of frogs of the Amazon Rainforest are already extinct.

b. [$_{\text{NP}}$ Bastantes espécies de sapos da Amazónia] já estão extintas.
several species of frogs of the Amazon Rainforest already are extinct
Several species of frogs of the Amazon Rainforest are already extinct.

c. [$_{\text{NP}}$ As muitas espécies de sapos da Amazónia] já estão extintas.
the many species of frogs of the Amazon Rainforest already are extinct
The many species of frogs of the Amazon Rainforest are already extinct.

d.	* [$_{NP}$	As	bastantes	espécies	de	sapos	da	Amazónia	]	já	estão	extintas.
		the	several	species	of	frogs	of the	Amazon Rainforest		already	are	extinct

The category Indefinite Specifics (in Position IV) contains elements like *certo* and *determinado* (*certain*), that mark NPs with exclusively indefinite specific readings, as in the first example below (26a):

(26)	a.	Todas	as	pessoas	leram	um	certo	livro.
		all	the	people	have read	a	certain	book

All people have read a certain book.

$\exists y[book(y) \wedge \forall x[person(x) \rightarrow read(x,y)]]$

b.	Todas	as	pessoas	leram	um	livro.
	all	the	people	have read	a	book

All people have read a book.

$\forall x[person(x) \rightarrow \exists y[book(y) \wedge read(x,y)]]$

$\exists y[book(y) \wedge \forall x[person(x) \rightarrow read(x,y)]]$

The most interesting property of these elements is that their contribution to the meaning of the sentences where they occur consists in merely restricting the relative scope possibilities between the quantifiers in these sentences. The example Portuguese sentence in (26b) is ambiguous between the two readings shown below it. In contrast, the example sentence in (26a) is not ambiguous and only has the reading where the existential quantifier has wide scope — its specific reading. This issue is explored in Section 4.10.1.

The category Prenominal Adjective Phrases (Position V) includes adjective phrases (APs) that precede the noun, and the slot named Head Noun (Position VI) represents the position where the noun surfaces.

The slot for Adjectival Arguments (Position VII) represents the position where adjectives that realize arguments of nouns surface. In the example in the table repeated below, the adjective form *americana* (*American*) realizes one of the arguments of the noun *invasão* (*invasion*). The semantics of this NP is quite similar to the semantics of a sentence like *The U.S. invaded Iraq*. More specifically, the arguments of the semantic relations for the noun *invasão*/*invasion* and the verb *invade* are the same in these examples.

(27)	a	invasão	americana	do	Iraque
	the	invasion	American	of the	Iraq

the American invasion of Iraq

In Position VIII one finds APs that do not saturate noun arguments, prepositional phrase (PP) adjuncts (not realizing noun arguments) and complements (realizing noun arguments), adverbial phrase (AdvP) adjuncts of nouns, postnominal demonstratives and postnominal possessives (adjuncts or complements). Not all adverbs can occur in this context (as noun modifiers). Among the adverbial phrases that can modify nouns one finds *aqui, aí, ali, dentro (de NP), fora (de NP), junto (a/de NP)* respectively *here, there, there, inside (NP), outside/out of NP, nearby/near NP*.

The last slot is for relative clauses (Position IX).

Elements occupying the same position in the table in Appendix A generally show free word order among themselves (but, depending on the category of these elements, there are some restrictions that will be presented in the following Sections). For instance the relative word order between cardinals and ordinals (both in Position IV) is unconstrained:

(28)	a.	Os	primeiros	dois	filmes	foram cancelados.
		the	first	two	films	were canceled

The first two films were canceled.

b.	Os	dois	primeiros	filmes	foram	cancelados.
	the	two	first	films	were	canceled

The two first films were canceled.

The numbering of these positions reflects precedence constraints among these elements: to give an example, prenominal adjectives (Position V) cannot precede cardinals (Position IV):

(29)	a.	Os	adeptos	entusiasmaram-se	depois de	[$_{NP}$ duas	grandes	vitórias	do	clube.]
		the	fans	got excited	after	two	great	victories	of the	club

The fans got excited after two great victories of their club.

b.	* Os	adeptos	entusiasmaram-se	depois de	[$_{NP}$ grandes	duas	vitórias	do	clube.]
	the	fans	got excited	after	great	two	victories	of the	club

The elements in some slots can iterate; the elements in others cannot. This sort of information is given in the following sections. This table is an approximation of the NP structure that is covered in this chapter. More detailed descriptions of linear precedence constraints and co-occurrence restrictions among the several NP constituents are provided in the following sections.

The presentation in the following sections does not respect the order of the elements in the table in Appendix A. The simple cases are presented first. For instance, the discussion on the implementation of determiners (Position II) is presented before the one on predeterminers (Position I), because the discussion about determiners is important for the discussion about predeterminers.

4.3 General Constraints

For ease of exposition, we will use a very flat hierarchy under *head* (for values of the feature SYNSEM |LOCAL|CAT|HEAD), that does not exploit inheritance in order to factor out the constraints common to multiple types. A simplified version of the type hierarchy under *head* that will be used is in Figure 4.1. All *head* subtypes that will be presented containing the attribute MARKER inherit from the type *functor*, where this feature is declared. We assume all of these types are a direct descendant of *functor*.

$$\begin{tikzcd}
 & \textit{head} & & \\
\textit{functor} \arrow[ur] & & \textit{noun-or-verb} \arrow[ul] & \\
 & \textit{noun} \arrow[ur] & & \textit{verb} \arrow[ul]
\end{tikzcd}$$

Figure 4.1: Simplified type hierarchy under *head*

Figure 4.2 shows a first version for a type hierarchy for *marking* and the following paragraphs describe the necessary constraints that are employed to model the NP structure assumed. In this hierarchy, the types *saturated* and *non-saturated* provide a way to abstract from the implementation of NP structure: parts of the grammar that are not directly relevant to the noun phrase use these types to refer to information about NPs. Their subtypes are employed in the definitions of NP constituents.

With this setup, items that select for NPs constrain them to have a MARKING with value *saturated* (instead of requiring their SPR feature to be empty). For instance, an item subcategorizing for exactly

$$\begin{tikzcd}[column sep=small]
 & \textit{marking} & \\
\textit{saturated} \arrow[ur] & & \textit{non-saturated} \arrow[ul] \\
 & & \textit{no-det-marking} \arrow[u] \\
 & & \textit{basic-marking} \arrow[u]
\end{tikzcd}$$

Figure 4.2: Type hierarchy under *marking* — (version 1/6). Final version on p. 118.

one NP complement will be constrained as follows:

$$\begin{bmatrix} \textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{val}|\textsc{comps} \left\langle \begin{bmatrix} \textsc{local}|\textsc{cat} \begin{bmatrix} \textsc{head} & \textit{noun} \\ \textsc{val}|\textsc{comps} & \textit{olist} \\ \textsc{marking} & \textit{saturated} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

In LXGram we encapsulate the information of what exactly constitutes a saturated constituent (in this case the constraints on COMPS and MARKING). To that end we create a subtype of *cat* (the type of the feature CAT) called *saturated-cat*, with the constraints:

$$\begin{bmatrix} \textit{saturated-cat} & \\ \textsc{val}|\textsc{comps} & \textit{olist} \\ \textsc{marking} & \textit{saturated} \end{bmatrix}$$

With this type, items that select exactly one NP complement are constrained in the following manner instead:

$$\begin{bmatrix} \textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{val}|\textsc{comps} \left\langle \begin{bmatrix} \textsc{local}|\textsc{cat} \begin{bmatrix} \textit{saturated-cat} \\ \textsc{head}\ \textit{noun} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

For instance, intransitive verbs, selecting an NP subject and no complements, have lexical entries with the following constraints, among others:

$$\begin{bmatrix} \textsc{synsem}|\textsc{local}|\textsc{cat} \begin{bmatrix} \textsc{head}\ \textit{verb} \\ \textsc{val} \begin{bmatrix} \textsc{subj} \left\langle \begin{bmatrix} \textsc{local}|\textsc{cat} \begin{bmatrix} \textit{saturated-cat} \\ \textsc{head}\ \textit{noun} \end{bmatrix} \end{bmatrix} \right\rangle \\ \textsc{comps}\ \langle\rangle \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Nouns come in the lexicon with [MARKING *basic-marking*]. This type is incompatible with *saturated*. Therefore, nouns need to combine with a functor with a MARK value unifiable with *saturated* in order for the resulting constituent to be able to occur in NP contexts.

We will simplify our presentation as far as the status of the feature SUBJ for noun-headed constituents is concerned. We assume that it is always empty for nouns, which can be stated in a general type for nominal lexical entries, from which all lexical types for nouns inherit. If this is the case, it does not need to be constrained in any way elsewhere, but it will be percolated in the relevant phrases. This is accounted for in our treatment of Head-Functor constructions, where the entire VAL feature of the head daughter is passed up. But also note that SUBJ can be used to account for co-occurrence restrictions between nouns in a predicative context and their subject. The following examples illustrate that a noun like *facto* (*fact*) can take a sentential subject (more specifically, a complementizer phrase introduced by the complementizer *que*) or an NP subject, whereas a noun like *lápis* (*pencil*) can only take an NP subject.

(30) a.

$[_{\text{NP-SUBJ}}$ Isso]	é	$[_{\text{NP}}$ um	facto.]
that	is	a	fact

That is a fact.

b.

$[_{\text{NP-SUBJ}}$ Isso]	é	$[_{\text{NP}}$ um	lápis.]
that	is	a	pencil

That is a pencil.

c.

É	$[_{\text{NP}}$ um	facto]	$[_{\text{CP-SUBJ}}$ que continuam	a	ser	vagos	relativamente	a	muitos	detalhes.]
is	a	fact	that they continue	to	be	vague	relatively	to	many	details

It is a fact that they remain vague about many details.

d.

* É	$[_{\text{NP}}$ um	lápis]	$[_{\text{CP-SUBJ}}$ que continuam	a	ser	vagos	relativamente	a	muitos	detalhes.]
is	a	pencil	that they continue	to	be	vague	relatively	to	many	details

In order to contemplate these cases in future versions of LXGram, the constraints on the feature SUBJ would have to be made different. We do not develop further on that here. In any case, as the examples above make clear, the projection of subjects of nouns is more peripheral than the NP boundaries. Within the NP, the SUBJ feature of nouns should therefore simply percolate and not be constrained to be empty. The implementation presented below respects this requirement.

4.4 Determiners

Determiners select a constituent with a value of MARKING incompatible with the value of their MARK feature, so that they do not iterate. Their HEAD is of type *determiner*, (in a preliminary version to be worked out further in the discussion below) defined as:

$$\begin{bmatrix} \textit{determiner} & \\ \text{MARKER} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{MARKING} & \textit{no-det-marking} \end{bmatrix} \\ \text{MARK}\ \textit{saturated} & \end{bmatrix} \end{bmatrix}$$

They are responsible for introducing quantifier semantics. The type used in the lexical entries for determiners thus has the following constraints, among others:

$$\begin{bmatrix} \text{SYNSEM|LOCAL} & \begin{bmatrix} \text{CAT|HEAD} & \begin{bmatrix} \textit{determiner} & \\ \text{MARKER|SELECT|LOCAL|CONT|HOOK} & \begin{bmatrix}\text{LTOP} & \boxed{1} \\ \text{INDEX} & \boxed{2}\end{bmatrix}\end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{HOOK|LTOP} & \boxed{3} \\ \text{RELS} & \left\{\begin{bmatrix}\text{LBL} & \boxed{3}\\ \text{ARG0} & \boxed{2} \\ \text{RSTR} & \boxed{4}\end{bmatrix}\right\} \\ \text{HCONS} & \left\{\begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{4} \\ \text{LARG} & \boxed{1}\end{bmatrix}\right\}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

There are other constraints in lexical types that we will systematically ignore in the following presentation, for the sake of simplicity. These include constraints on the NON-LOCAL features and the VAL features. In the case of determiners and many nominal constituents, all of them are empty, and we omit them in order to present more concise feature structures.

The feature PRED in the relation introduced by the determiner (the sole element of RELS in the above AVM) is filled in in each lexical entry (it is not shown in that AVM — see Section 2.4).[1] For the definite article its value is "_o_q_rel" in LXGram. The semantics of the definite article is equivalent to $\lambda P.\lambda Q.\_o\_q(x, P(x), Q(x))$.

4.4.1 Example

The example in Figure 4.3 displays the syntactic structure and the semantics for the sentence *o carro avariou* (*the car broke down*). The MRS representation there is equivalent to $\_o\_q(\ x_2,\ \_carro\_n(x_2),\ \_avariar(e, x_2))$.

In order to understand that example, it is necessary to consider the semantic constraints on the lexical type for intransitive verbs (in addition to the syntactic constraints on the same kind of verbs, presented above):

$$\begin{bmatrix} \text{SYNSEM|LOCAL} & \begin{bmatrix} \text{CAT|VAL|SUBJ} & \left\langle \begin{bmatrix}\text{LOCAL|CONT|HOOK|INDEX} \ \boxed{3}\end{bmatrix} \right\rangle \\ \text{CONT} & \begin{bmatrix} \text{HOOK} & \begin{bmatrix}\text{LTOP} & \boxed{1}\ h \\ \text{INDEX} & \boxed{2}\ e\end{bmatrix} \\ \text{RELS} & \left\{\begin{bmatrix}\text{LBL} & \boxed{1} \\ \text{ARG0} & \boxed{2} \\ \text{ARG1} & \boxed{3}\end{bmatrix}\right\} \\ \text{HCONS} & \{\} \end{bmatrix}\end{bmatrix}\end{bmatrix}$$

These constraints ensure that the INDEX of the subject is the second argument of the verb's relation, the first being an event variable (see Section 2.3.3).

[1]The feature RSTR is appropriate for the type *quantifier-relation*, which is a subtype of *relation*, which is defined to have a feature PRED (denoting the relation's name). The type *quantifier-relation* also has the feature BODY. After type inference and type expansion, the features PRED and BODY are added to the relation that is the sole member of RELS in the above AVM, because of the constrained RSTR.

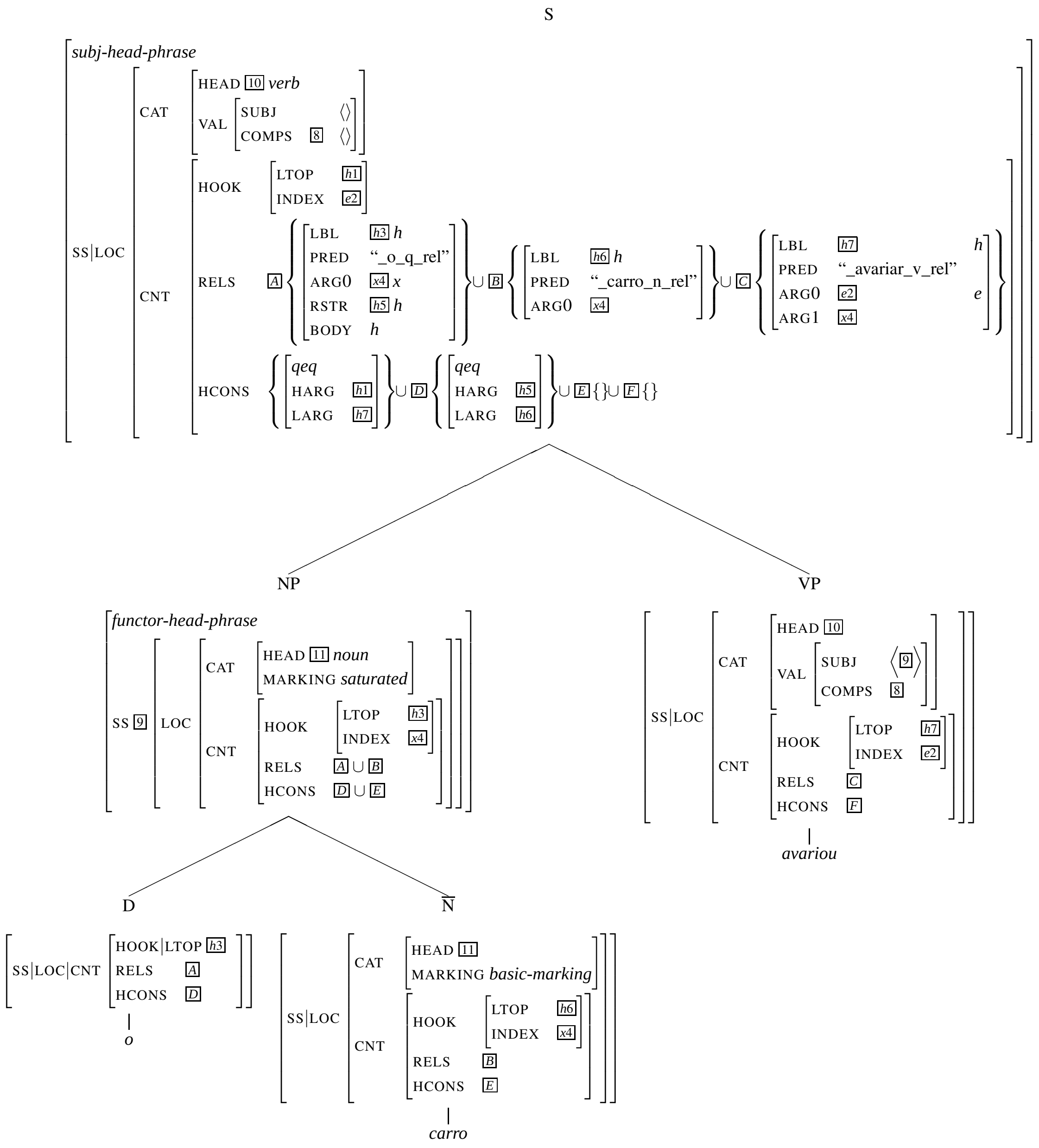

S
subj-head-phrase
SS|LOC
CAT
HEAD 10 verb
VAL SUBJ ⟨⟩ COMPS 8 ⟨⟩
CNT
HOOK LTOP h1 INDEX e2
RELS A { LBL h3 h, PRED "_o_q_rel", ARG0 x4 x, RSTR h5 h, BODY h } ∪ B { LBL h6 h, PRED "_carro_n_rel", ARG0 x4 } ∪ C { LBL h7 h, PRED "_avariar_v_rel", ARG0 e2 e, ARG1 x4 }
HCONS { qeq HARG h1 LARG h7 } ∪ D { qeq HARG h5 LARG h6 } ∪ E {} ∪ F {}
NP
functor-head-phrase
SS 9 LOC
CAT HEAD 11 noun MARKING saturated
CNT HOOK LTOP h3 INDEX x4
RELS A ∪ B
HCONS D ∪ E
VP
SS|LOC
CAT HEAD 10 VAL SUBJ ⟨9⟩ COMPS 8
CNT HOOK LTOP h7 INDEX e2
RELS C
HCONS F
avariou
D
SS|LOC|CNT HOOK|LTOP h3 RELS A HCONS D
o
N̄
SS|LOC
CAT HEAD 11 MARKING basic-marking
CNT HOOK LTOP h6 INDEX x4
RELS B
HCONS E
carro

Figure 4.3: Example of the semantics of determiners. The sentence is *o carro avariou* (*the car broke down*). SS abbreviates SYNSEM, LOC abbreviates LOCAL, and CNT abbreviates CONT.

The constraints on the lexical type for nouns with no arguments relevant for this example are:

$$\left[\begin{array}{ll}\textsc{synsem}|\textsc{local} & \left[\begin{array}{ll}\textsc{cat} & \left[\begin{array}{ll}\textsc{head} & \textit{noun}\\ \textsc{val} & \left[\begin{array}{ll}\textsc{subj} & \langle\rangle\\ \textsc{comps} & \langle\rangle\end{array}\right]\\ \textsc{marking} & \textit{basic-marking}\end{array}\right]\\ \textsc{cont} & \left[\begin{array}{ll}\textsc{hook} & \left[\begin{array}{ll}\textsc{ltop} & \boxed{1}\ h\\ \textsc{index} & \boxed{2}\ x\end{array}\right]\\ \textsc{rels} & \left\{\left[\begin{array}{ll}\textsc{lbl} & \boxed{1}\\ \textsc{arg0} & \boxed{2}\end{array}\right]\right\}\\ \textsc{hcons} & \{\}\end{array}\right]\end{array}\right]\end{array}\right]$$

These constraints make the variable that is in the argument of the noun's relation available in the INDEX feature of the noun.

Some of the constraints on *subj-head-phrase* assumed here are also relevant for that example:

$$\left[\begin{array}{ll}\textsc{synsem}|\textsc{local} & \left[\begin{array}{ll}\textsc{cat} & \left[\begin{array}{ll}\textsc{head} & \boxed{1}\ \textit{verb}\\ \textsc{val} & \left[\begin{array}{ll}\textsc{subj} & \langle\rangle\\ \textsc{comps} & \boxed{2}\end{array}\right]\end{array}\right]\\ \textsc{cont} & \left[\begin{array}{ll}\textsc{hook} & \left[\begin{array}{ll}\textsc{ltop} & \boxed{3}\\ \textsc{index} & \boxed{4}\end{array}\right]\\ \textsc{rels} & \boxed{A}\cup\boxed{B}\\ \textsc{hcons} & \left\{\left[\begin{array}{l}\textit{qeq}\\ \textsc{harg}\ \boxed{3}\\ \textsc{larg}\ \boxed{5}\end{array}\right]\right\}\cup\boxed{C}\cup\boxed{D}\end{array}\right]\end{array}\right]\\ \textsc{non-head-dtr}|\textsc{synsem}\ \boxed{6} & \left[\textsc{local}|\textsc{cont}\ \left[\begin{array}{ll}\textsc{rels} & \boxed{A}\\ \textsc{hcons} & \boxed{C}\end{array}\right]\right]\\ \textsc{head-dtr}|\textsc{synsem}|\textsc{local} & \left[\begin{array}{ll}\textsc{cat} & \left[\begin{array}{ll}\textsc{head} & \boxed{1}\\ \textsc{val} & \left[\begin{array}{lll}\textsc{subj} & \left\langle\boxed{6}\right\rangle & \\ \textsc{comps} & \boxed{2} & \textit{olist}\end{array}\right]\end{array}\right]\\ \textsc{cont} & \left[\begin{array}{ll}\textsc{hook} & \left[\begin{array}{ll}\textsc{ltop} & \boxed{5}\\ \textsc{index} & \boxed{4}\end{array}\right]\\ \textsc{rels} & \boxed{B}\\ \textsc{hcons} & \boxed{D}\end{array}\right]\end{array}\right]\end{array}\right]$$

4.5 Predeterminers

Saturated NPs can be introduced by a determiner (Position II) or a predeterminer (Position I):

(31) a. [$_{\text{NP}}$ Os$_{\text{D}}$ seres humanos] são livres.
the human beings are free
Human beings are free.

b. [$_{\text{NP}}$ Todos$_{\text{PreD}}$ os$_{\text{D}}$ seres humanos] são livres.
all the human beings are free
All human beings are free.

In the first case the quantifier relation of the NP comes from the determiner, but in the second case it comes from the predeterminer. When a predeterminer introduces an NP, a determiner must be present (32).

(32) a. todas as pessoas
all the people
all (the) people

b. todas aquelas pessoas
all those people
all those people

c. * todas pessoas
all people

The last example is actually a possible NP in Brazilian Portuguese. More on this is said below.

Determiners that co-occur with predeterminers must thus be different from determiners introducing an NP, since the former contribute no quantifier semantics but the latter do. Multiple lexical items are required in view of the fact that it is not possible to underspecify the number of elementary predications that a given lexical item contributes to the MRS representation.

In order to model these restrictions, the type hierarchy under *marking* can be made to look like in Figure 4.4.

$$\begin{tikzcd}
 & \textit{marking} & & \\
\textit{saturated} \arrow[ur] & & \textit{non-saturated} \arrow[ul] & \\
 & \textit{no-det-marking} \arrow[ur] & & \textit{non-saturated-det-marking} \arrow[ul] \\
 & \textit{basic-marking} \arrow[u] & &
\end{tikzcd}$$

Figure 4.4: Type hierarchy under *marking* (version 2/6). Previous version on p. 72. Final version on p. 118.

The lexical entries for determiners that contribute quantifier semantics and appear at the left edge of NPs (the form *os* in (31a)) are constrained to have a head type like:

$$\begin{bmatrix} \textit{determiner} & \\ \textsc{marker} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \textsc{select}|\textsc{local}|\textsc{cat} & \begin{bmatrix} \textsc{head} & \textit{noun} \\ \textsc{marking} & \textit{no-det-marking} \end{bmatrix} \\ \textsc{mark}\ \textit{saturated} & \end{bmatrix} \end{bmatrix}$$

The constraints on the head of determiners that follow predeterminers and contribute no semantics (the form *os* in (31b)) are:

$$\begin{bmatrix} \textit{determiner} & \\ \textsc{marker} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \textsc{select}|\textsc{local}|\textsc{cat} & \begin{bmatrix} \textsc{head} & \textit{noun} \\ \textsc{marking} & \textit{no-det-marking} \end{bmatrix} \\ \textsc{mark}\ \textit{non-saturated-det-marking} & \end{bmatrix} \end{bmatrix}$$

We can factor out the commonalities between the two kinds of determiners by defining the HEAD type *determiner* as:

$$\begin{bmatrix} \textit{determiner} & \\ \textsc{marker} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \textsc{select}|\textsc{local}|\textsc{cat} & \begin{bmatrix} \textsc{head} & \textit{noun} \\ \textsc{marking} & \textit{no-det-marking} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

With this definition, the value of MARK is then further specified in the lexical types, yielding the results just presented.

Summing up, determiners that follow predeterminers carry no semantics and have a MARK feature with the value *det-marking-nonsat*. They also have the HOOK of their sister node (so that the LTOP of the mother node in *functor-head-phrase*s and in *functor-head-phrases*, which comes from the functor daughter, is the same as the head daughter's LTOP):

$$\begin{bmatrix} \textsc{synsem}|\textsc{local} & \begin{bmatrix} \textsc{cat}|\textsc{head} & \begin{bmatrix} \textit{determiner} \\ \textsc{select}|\textsc{local}|\textsc{cont}|\textsc{hook}\ \boxed{1} \\ \textsc{mark}\ \textit{non-saturated-det-marking} \end{bmatrix} \\ \textsc{cont} & \begin{bmatrix} \textsc{hook} & \boxed{1} \\ \textsc{rels} & \{\} \\ \textsc{hcons} & \{\} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Determiners that do not follow predeterminers have quantifier semantics and produce saturated noun-headed phrases immediately:

$$\begin{bmatrix} \text{SYNSEM}|\text{LOCAL} & \begin{bmatrix} \text{CAT}|\text{HEAD} & \begin{bmatrix} \textit{determiner} \\ \text{SELECT}|\text{LOCAL}|\text{CONT}|\text{HOOK} \begin{bmatrix}\text{LTOP} & \boxed{1} \\ \text{INDEX} & \boxed{2}\end{bmatrix} \\ \text{MARK } \textit{saturated}\end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{HOOK}|\text{LTOP} & \boxed{3} \\ \text{RELS} & \left\{\begin{bmatrix}\text{LBL} & \boxed{3}\\ \text{ARG0} & \boxed{2} \\ \text{RSTR} & \boxed{4}\end{bmatrix}\right\} \\ \text{HCONS} & \left\{\begin{bmatrix}\textit{qeq} & \\ \text{HARG} & \boxed{4} \\ \text{LARG} & \boxed{1}\end{bmatrix}\right\}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Predeterminers have a head type like:

$$\begin{bmatrix} \textit{predeterminer} & \\ \text{MARKER} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \text{SELECT}|\text{LOCAL}|\text{CAT}|\text{HEAD } \textit{noun} & \\ \text{PREHEAD} & \begin{bmatrix}\text{SELECT}|\text{LOCAL}|\text{CAT}|\text{MARKING } \textit{non-saturated-det-marking} \\ \text{MARK } \textit{saturated}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

They require the presence of a semantically vacuous determiner (therefore they select for a sister node with MARKING of type *non-saturated-det-marking*). They produce a saturated phrase, since their feature MARK is of type *saturated*.

Figure 4.5 shows the syntactic analysis and the semantic representation derived for the NP *as pessoas* (*the people*). Figure 4.6 presents the same pieces of information for the NP *todas as pessoas* (*all people*).

Predeterminers can also appear postnominally, as in (33). As the second example shows, they occupy an NP internal position (Position VIII). This means that syntactic and semantic scope do not match in such structures, and more features are therefore needed to pass the relevant information along the syntax tree. We will not elaborate on this issue, as this is left to future work.

(33) a. as pessoas todas
the people all
all (the) people

b. as pessoas todas dessa aldeia
the people all from that village
all (the) people from that village

To account for Brazilian Portuguese *todo* (32c), we can resort to positing more than one lexical entries for *todo*. The constraints associated with the HEAD attribute of this item only differ from the ones of the head type *predeterminer* above in that, instead of selecting for a constituent with MARKING *non-saturated-det-marking*, this item selects for an element with MARKING *no-det-marking* (i.e. this item is encoded as a determiner).

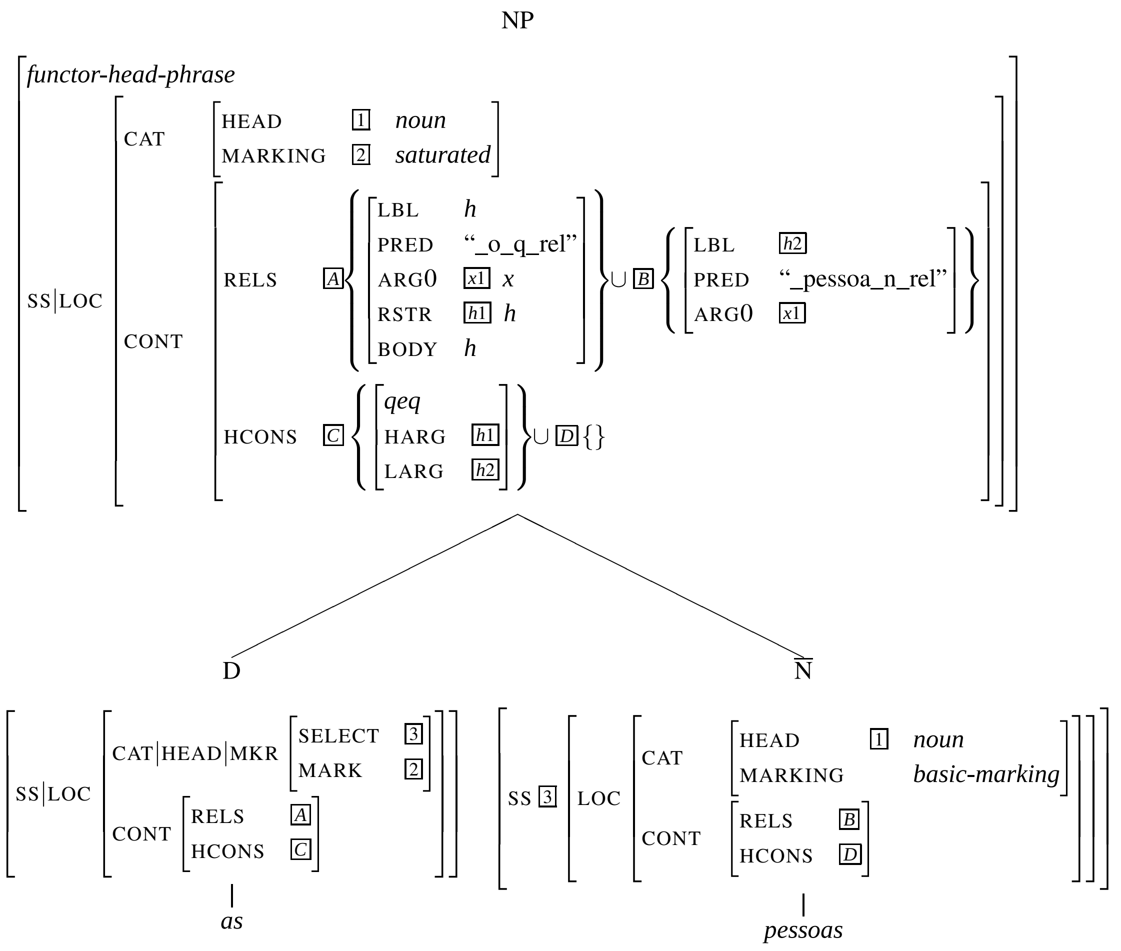


Figure 4.5: Example of a determiner in NP initial position. The NP is *as pessoas* (*the people*). SS abbreviates SYNSEM, LOC abbreviates LOCAL, MKR abbreviates MKR. The MRS is equivalent to $\lambda P.\_o\_q(x, \_pessoa\_n(x), P(x))$, with the possibility of other quantifiers inside the restrictor (second argument) of the *_o_q* quantifier relation.

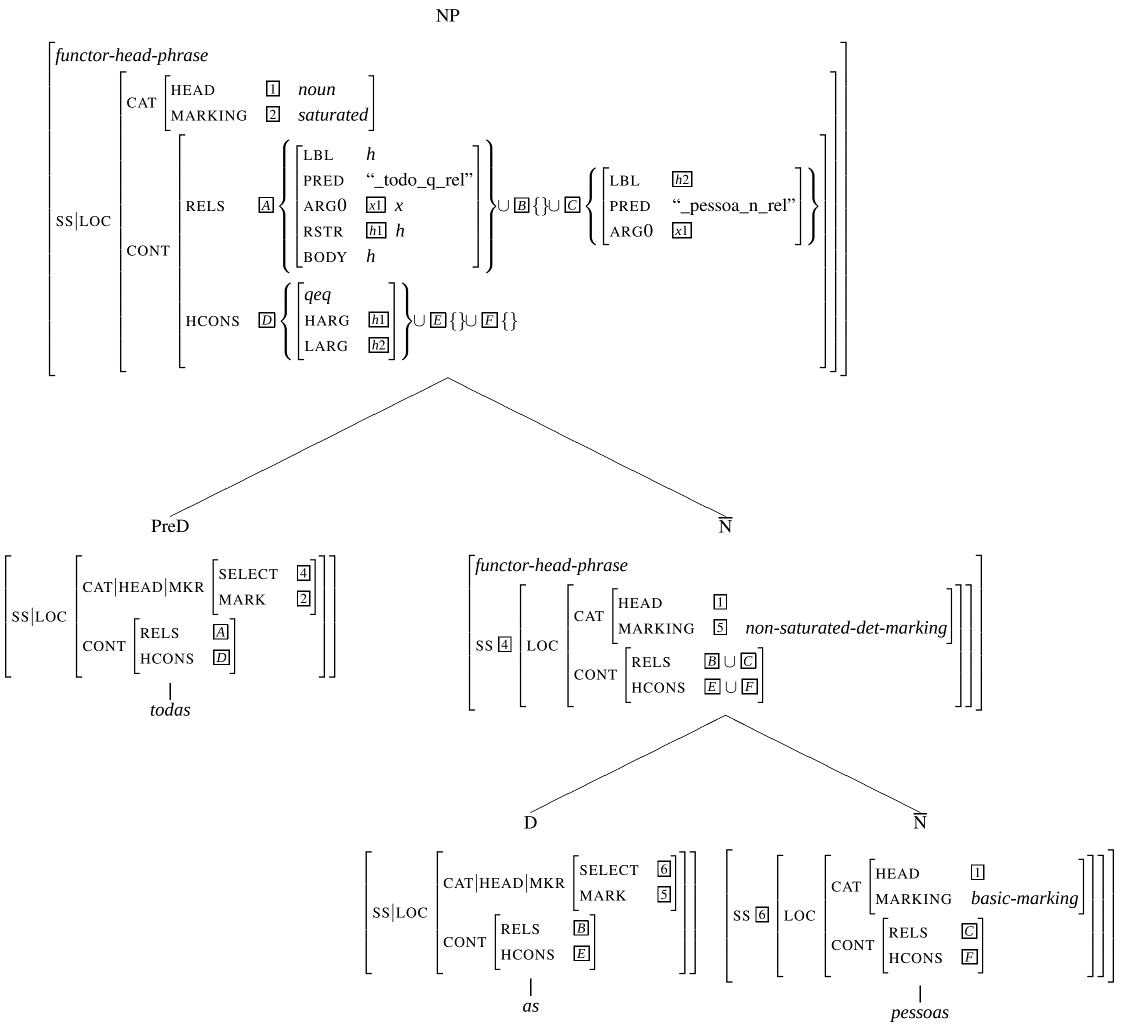


Figure 4.6: Example of a determiner preceded by a predeterminer. The NP is *todas as pessoas* (*all people*). SS abbreviates SYNSEM, LOC abbreviates LOCAL, MKR abbreviates MKR. The MRS is equivalent to $\lambda P.\_todo\_q(x, \_pessoa\_n(x), P(x))$, with the possibility of other quantifiers inside the restrictor (second argument) of the *_todo_q* quantifier relation.

4.6 Modifying Adjectives

On a first approximation, adjectives select for a constituent with [MARKING *basic-marking*] and produce a node with the same level of saturation:

$$\begin{bmatrix} \textit{adjective} & \\ \text{MARKER} & \begin{bmatrix} \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{MARKING} & \textit{basic-marking} \end{bmatrix} \\ \text{MARK } \textit{basic-marking} & \end{bmatrix} \end{bmatrix}$$

As a consequence, they are allowed to recur.

Portuguese has prenominal and postnominal adjectives. Potentially spurious attachment ambiguities will be produced for a sequence AP_1-Noun-AP_2: [AP_1 [Noun AP_2]] and [[AP_1 Noun] AP_1]. Although spurious ambiguity is innocuous, it is also a source of inefficiency, as it causes the parser to perform more computations than needed. It is straightforward to complicate the type hierarchy of *marking* to control this, too.

Examples like the one in (34) argue in favor of the structure [AP_1 [Noun AP_2]], since this NP can describe someone who is not Chinese. Accordingly, we want to provide to such NP semantics like $\lambda P.\_um\_q(x,\_falso\_a(e_1,\_médico\_n(x) \wedge \_chinês\_a(e_2,x)),P(x))$.[2] It does not describe a Chinese person who is a fake doctor (i.e. $\lambda P.\_um\_q(x,\_falso\_a(e_1,\_médico\_n(x)) \wedge \_chinês\_a(e_2,x),P(x))$). Assuming syntactic scope and semantic scope match, the structure [AP_1 [Noun AP_2]] is justified.[3]

(34)	um	falso	médico	chinês
	a	fake	doctor	Chinese
	a fake Chinese doctor			

We can add types to the type hierarchy under *marking* to force this structure. Figure 4.7 displays a revised hierarchy under *marking*.

Prenominal adjectives can be specified to have the constraint [MARK *prenom-adj-marking*] and select for nominal projections with [MARKING *prenom-adj-or-basic-marking*], while postnominal adjectives select for sister nodes with [MARKING *basic-marking*] and also bear the value *basic-marking* for their MARK attribute. The type of HEAD in adjectives looks like:

$$\begin{bmatrix} \textit{adjective} & \\ \text{MARKER} & \begin{bmatrix} \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{MARKING} & \textit{prenom-adj-or-basic-marking} \end{bmatrix} \\ \text{MARK } \textit{prenom-adj-or-basic-marking} & \end{bmatrix} \end{bmatrix}$$

In the lexical types for adjectives, we distinguish between the adjectives that can only precede the noun, the ones that can only follow it and the ones that can occur in either position. The following examples illustrate these three classes. An adjective like *mero* (*mere*) can only precede the noun, an adjective like *japonês* (*Japanese*) can only follow the noun, and an adjective like *falso* (*false*) can precede or follow it.

[2] We can assume that the semantic representation of *falso* (*fake*), $\lambda P_{\in D_{\langle e,t\rangle}}.\lambda x_{\in D_e}.\_falso\_a(e,P(x))$, means $\lambda P.\lambda x.\neg P(x)$.

[3] It is not required that syntactic and semantic scope match, because it is possible to manipulate feature structures, but it is desirable that they do, since implementation becomes more straightforward if they match. We thus assume that syntax and semantics match in the absence of a compelling argument against it.

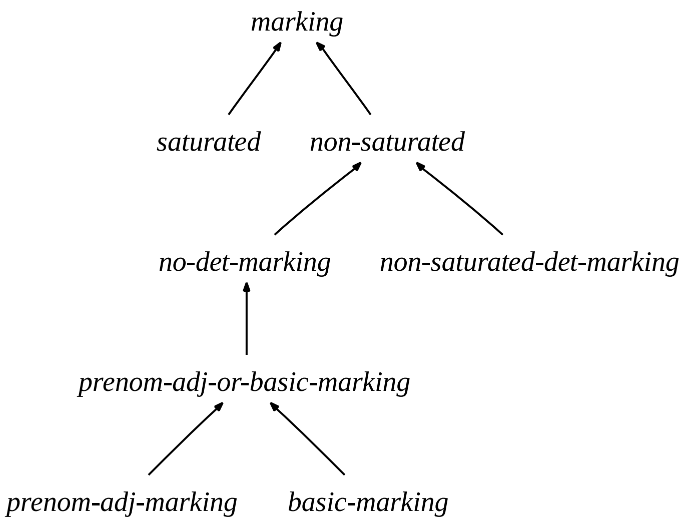


Figure 4.7: Type hierarchy under *marking* (version 3/6). Previous version on p. 77. Final version on p. 118.

(35) a. Atacaram um mero inspector.
they attacked a mere inspector
They attacked a mere inspector.

b. * Atacaram um inspector mero.
they attacked an inspector mere

c. * Atacaram um japonês inspector.
they attacked a Japanese inspector

d. Atacaram um inspector japonês.
they attacked an inspector Japanese
They attacked a Japanese inspector.

e. Atacaram um falso inspector.
they attacked a false inspector
They attacked a false inspector.

f. Atacaram um inspector falso.
they attacked an inspector false
They attacked a false inspector.

The lexical types for the adjectives that can precede the noun have the constraints:

$$\left[\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD}\ \begin{bmatrix}\textit{adjective}\\ \text{MARKER}|\text{PREHEAD}|\text{MARK}\ \textit{prenom-adj-marking}\end{bmatrix}\right]$$

The lexical types for the ones that can follow the noun are constrained with:

$$\left[\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD}\ \begin{bmatrix}\textit{adjective}\\ \text{MARKER}|\text{POSTHEAD}\ \begin{bmatrix}\text{SELECT}|\text{LOCAL}|\text{CAT}|\text{MARKING}\ \textit{basic-marking}\\ \text{MARK}\ \textit{basic-marking}\end{bmatrix}\end{bmatrix}\right]$$

The adjectives that can follow or precede the noun inherit all these constraints. The ones that can only precede it are given a lexical type that inherits from the type where the constraints on PREHEAD are stated and is further constrained with:

$$\left[\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD}|\text{MARKER}\ \textit{pre-only-marker-min}\right]$$

Likewise, the lexical type for the adjectives that can only follow the noun inherits from the type above that has a constrained POSTHEAD feature and is defined to also bear:

$$\left[\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD}|\text{MARKER}\ \textit{post-only-marker-min}\right]$$

All lexical types for nouns have [MARKING *basic-marking*], as before, since nouns have the same syntactic distribution of noun-adjective sequences: they can combine with another adjective to their right, or with a prenominal adjective. Under this analysis, nouns have a syntactic distribution different from adjective-noun sequences, as the latter cannot combine with an adjective to their right.

With this system of constraints, the noun phrase *um médico chinês falso* receives a semantic representation equivalent to *um falso médico chinês*, equivalent to the lambda formula presented above. On the other hand, a noun phrase like *um médico falso Chinês* (*a fake doctor who is Chinese*) receives semantics equal to $\lambda P.\_um\_q(x, \_falso\_a(e_1, \_médico\_n(x)) \wedge \_chinês\_a(e_2, x), P(x))$, based on the syntactic structure [*um* [[*médico falso*] *chinês*]].

Adjectives are allowed to iterate in both positions (prenominal and postnominal). This is borne out by data like:

(36) a.

Era	um	grande,	grande	filme.
it was	a	great	great	movie

It was a great, great movie.

b.

Era	um	filme	chato,	chato.
it was	a	movie	boring	boring

It as a boring, boring movie.

It is worth pointing out that we cannot properly capture the meaning difference between an $\overline{\text{N}}$ like *filme chato* (*boring movie*), which receives semantics equivalent to $\lambda x.\_filme\_n(x) \wedge \_chato\_a(e1, x)$, and an $\overline{\text{N}}$ like *filme chato, chato* (*boring, boring movie*), which is assigned an MRS representation equivalent to $\lambda x.\_filme\_n(x) \wedge \_chato\_a(e1, x) \wedge \_chato\_a(e2, x)$: the two formulas are logically equivalent due to idempotence of conjunction if we ignore the different event variables. It is not clear that the difference is truly semantic, anyway. It may simply be a pragmatic effect.

The syntactic analysis produced by LXGram for the NP in (34) (*um falso médico chinês — a false Chinese doctor*) is in Figure 4.8, with abridged feature structures.

The structure [[AP_1 N] AP_2] is blocked, because the phrase with the form [AP_1 N] has MARKING with the value *prenom-adj-marking* but postnominal adjectives select for a sister node with the value *basic-marking* for that feature. There is no unifier for *basic-marking* and *prenom-adj-marking*, as can be seen in Figure 4.7.

4.7 Argumental Adjectives

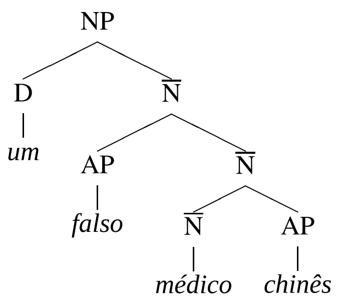
Semantics

Adjectives that are used as an argument of nouns (37a) display drastically different semantics from adjectives that modify a noun (37b). Consider the two examples:

(37) a.

Viram	[$_{\text{NP}}$ a	alunagem	americana]	na	televisão.
they saw	the	moon landing	American	on the	television

They saw the American moon landing on TV.



$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{functor-head-phrase}} \\
\textsc{orth} & \left\langle \text{“um”, “falso”, “médico”, “chinês”} \right\rangle \\
\textsc{synsem}|\textsc{local}|\textsc{cat} & \left[\begin{array}{ll} \textsc{head} & \boxed{1}\ \textit{noun} \\ \textsc{marking} & \boxed{2}\ \textit{saturated} \end{array}\right] \\
\textsc{non-head-dtr} & \left[\begin{array}{ll} \textsc{orth} & \left\langle \text{“um”} \right\rangle \\ \textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{head} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{determiner}} \\ \textsc{marker} & \left[\begin{array}{ll} \textsc{select} & \boxed{3} \\ \textsc{mark} & \boxed{2} \end{array}\right] \end{array}\right] \end{array}\right] \\
\textsc{head-dtr} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{functor-head-phrase}} \\
\textsc{orth} & \left\langle \text{“falso”, “médico”, “chinês”} \right\rangle \\
\textsc{synsem}\ \boxed{3} & \left[\textsc{local}|\textsc{cat} \left[\begin{array}{ll} \textsc{head} & \boxed{1} \\ \textsc{marking} & \boxed{4}\ \textit{prenom-adj-marking} \end{array}\right] \right] \\
\textsc{non-head-dtr} & \left[\begin{array}{ll} \textsc{orth} & \left\langle \text{“falso”} \right\rangle \\ \textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{head} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{adjective}} \\ \textsc{marker} & \left[\begin{array}{ll} \textsc{select} & \boxed{5} \\ \textsc{prehead} & \left[\begin{array}{ll} \textsc{select} & \boxed{5} \\ \textsc{mark} & \boxed{4} \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \\
\textsc{head-dtr} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{head-functor-phrase}} \\
\textsc{orth} & \left\langle \text{“médico”, “chinês”} \right\rangle \\
\textsc{synsem}\ \boxed{5} & \left[\textsc{local}|\textsc{cat} \left[\begin{array}{ll} \textsc{head} & \boxed{1} \\ \textsc{marking} & \boxed{6}\ \textit{basic-marking} \end{array}\right] \right] \\
\textsc{non-head-dtr} & \left[\begin{array}{ll} \textsc{orth} & \left\langle \text{“chinês”} \right\rangle \\ \textsc{synsem}|\textsc{local}|\textsc{cat}|\textsc{head} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{adjective}} \\ \textsc{marker} & \left[\begin{array}{ll} \textsc{select} & \boxed{7} \\ \textsc{posthead} & \left[\begin{array}{ll} \textsc{select} & \boxed{7} \\ \textsc{mark} & \boxed{6} \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \\
\textsc{head-dtr} & \left[\begin{array}{ll} \textsc{orth} & \left\langle \text{“médico”} \right\rangle \\ \textsc{synsem}\ \boxed{7} & \left[\textsc{local}|\textsc{cat} \left[\begin{array}{ll} \textsc{head} & \boxed{1} \\ \textsc{marking} & \textit{basic-marking} \end{array}\right] \right] \end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure 4.8: Syntactic analysis for an NP with a prenominal and a postnominal adjective

b. Viram [$_{\text{NP}}$ um carro americano] naquela rua.
they saw a car American on that street
They saw an American car on that street.

The NP in the first example has semantics quite similar to a sentence like *Os americanos alunaram* (*The Americans landed on the moon*). The semantics for this sentence could be

$$\_o\_q(x, \_americano\_n(x), \_alunar\_v(e,x))$$

For the NP in the first example (*a alunagem americana* — *the American moon landing*) we could thus think of the semantics

$$\lambda P_{\in D_{\langle e,t\rangle}}.\_o\_q(x, \_o\_q(y, \_americano\_n(y), \_alunagem\_n(x,y)), P(x))$$

The semantics for the noun *alunagem* is

$$\lambda \mathscr{Q}_{\in D_{\langle\langle e,t\rangle t\rangle}}.\lambda x_{\in D_e}.\mathscr{Q}(\lambda y_{\in D_e}.alunagem(x,y))$$

Assuming that semantically, the noun is the functor and the adjective is the argument, the semantics for the argumental adjective in (37a) would have to be

$$\lambda P_{\in D_{\langle e,t\rangle}}.\_o\_q(z, \_americano\_n(z), P(z))$$

The most simple semantics for the modifying adjective *americano* in (37b) is

$$\lambda P_{\in D_{\langle e,t\rangle}}.\lambda x_{\in D_e}.P(x) \wedge \_americano\_a(e,x)$$

The semantics for the NP in (37b) is thus

$$\lambda P_{\in D_{\langle e,t\rangle}}.\_um\_q(x, \_carro\_n(x) \wedge \_americano\_a(e,x), P(x))$$

The same adjective in these two contexts presents very different semantics. There are two options: to have multiple lexical entries for the adjectives that can occur as modifiers and as arguments; to use an optional lexical rule to change the meaning and syntactic properties of such adjectives, producing one of the versions from the other, which would be in the lexicon.

The lexical rule approach is certainly more appealing, since adjectives that can occur as arguments would simply receive a special lexical type in their lexical entry, denoting this property. The problem is that we cannot produce one of the semantic representations from the other with the machinery in the LKB, because we cannot manipulate strings, and the mapping between the relation names *_americano_n* and *_americano_a* requires string manipulation.

We address this by providing a slightly different semantics to these adjectives when they are used as modifiers:

$$\lambda P_{\in D_{\langle e,t\rangle}}.\lambda x_{\in D_e}.\_o\_q(y, americano\_n(y), P(x) \wedge abstract\_a(e,x,y))$$

Using the MRS format, this semantics can be easily produced from the semantics the adjective displays when it occurs as an argument. Therefore, the lexical entries for the adjectives that can occur in both positions have argumental semantics, and an optional lexical rule adds the *abstract_a* relation.

Under this model, the NP *o carro americano* receives the semantics:

$$\lambda P_{\in D_{\langle e,t\rangle}}.\_o\_q(x, \_o\_q(y, americano\_n(y), \_carro\_n(x) \wedge abstract\_a(e,x,y), P(x))$$

The relation named *abstract_a* denotes a relation we cannot determine systematically. In this example, it links "car" with "Americans" and can be understood as "produced by". The relation $\lambda x_{\in D_e}.\lambda y_{\in D_e}.abstract\_a(e,x,y)$ is intended to mean $\lambda x_{\in D_e}.\lambda y_{\in D_e}.\exists R_{\in D_{\langle e,\langle e,t\rangle\rangle}} R(x,y)$.

Adjectives that cannot be used as arguments of nouns (e.g. *amarelo* — *yellow*) still receive standard adjective semantics:

$$\lambda P_{\in D_{\langle e,t\rangle}}.\lambda x_{\in D_e}.P(x) \wedge \_amarelo\_a(e,x)$$

Syntax

A PP complement cannot intervene between a noun and an adjectival complement:

(38)	a.		a	invasão	americana	do	Iraque
			the	invasion	American	of the	Iraq
			The American invasion of Iraq				
	b.	*	a	invasão	do	Iraque	americana
			the	invasion	of the	Iraq	American

Also, the remaining elements in Position VIII cannot appear before an adjective argument either. An example with a PP adjunct follows:

(39)	a.		a	alunagem	americana	de	1969
			the	moon landing	American	of	1969
			the American moon landing of 1969				
	b.	*	a	alunagem	de	1969	americana
			the	moon landing	of	1969	American

In LXGram, we use a dedicated syntactic rule similar to Head-Complement constructions that requires the head daughter to be a noun that selects for a PP complement (cf. "the American moon landing" with "the moon landing by the Americans") and the non-head daughter to be an adjective with argumental semantics.

We can resort to two important subtypes of *canonical-synsem* (see Figure 2.2 in Section 2.6): *lex-synsem* and *phrase-synsem*. The SYNSEM feature of all words (terminal symbols and lexical rules) is of type *lex-synsem*, and the SYNSEM of phrases is of type *phrase-synsem*. These types are incompatible: they have no common subtype.

In order to force strict adjacency between the noun head and the adjective argument, all that is necessary is that the head daughter of this syntactic rule dedicated to project adjectival arguments to the right of a noun be constrained to have a SYNSEM of type *lex-synsem*. Because of this constraint, nothing can intervene between the noun and this type of adjective, because, if that happened, a phrasal node would have to be the head daughter of this construction. This constraint also has the nice side effect of blocking two adjectival arguments of the same noun. This blocks the following ungrammatical example:

(40)	a.	*	a	invasão	americana	iraquiana
			the	invasion	American	Iraqi

In this special rule, the MARKING value of the mother node is also *basic-marking*.

4.8 Noun Complementation

Many nouns subcategorize for one or more complements, that can be of different kinds. For the sake of illustration, here we will focus only on nouns with a single PP complement.

The standard HPSG approach to project complements is assumed: subcategorized for complements are members of a list-valued attribute COMPS in the lexical entry of the corresponding head, and a syntactic rule projects elements in that list, producing a mother node with a reduced COMPS.

Following many computationally implemented HPSGs, like the LinGO English Resource Grammar or the LinGO Grammar Matrix, strict binary branching is assumed — in the case of multiple complements, they are discharged one at a time. The Head-Complement syntactic rule or rules therefore unify the SYNSEM of the non-head daughter with the first element in the COMPS of the head daughter, and the COMPS of the mother node is the tail of the COMPS of the head daughter.

An issue in focus here is the relative scope between complements and the various functors. In Portuguese, the relative order between complements and several adnominal constituents (the ones in Position VIII in the table in Appendix A) is free. Consider the examples in (41).

(41)	a.	o consumo	galopante$_{AP}$	[$_{PP}$ de petróleo]	
		the consumption	ever increasing	of oil	
		the ever increasing consumption of oil			
	b.	o consumo	[$_{PP}$ de petróleo]	galopante$_{AP}$	
		the consumption	of oil	ever increasing	
		the ever increasing consumption of oil			

These examples show that word order between postnominal adjunct adjectives and PP complements is arbitrary. Similar data can be presented for the other elements in Position VIII.

With other functors, however, word order is not free.

Indeed, PP complements must surface before restrictive relative clauses:

(42)	a.	o consumo	[$_{PP}$ de petróleo]	[$_{RelCl}$ que continua a crescer]
		the consumption	of oil	that continues to increase
		the consumption of oil that continues to increase		
	b.	* o consumo	[$_{RelCl}$ que continua a crescer]	[$_{PP}$ de petróleo]
		the consumption	that continues to grow	of oil

The exact constraints on the position of relative clauses within an NP, as they are implemented in LXGram, are presented in Section 4.13.

Since complements occupy the same word order slot as the functors that give rise to constituents with *basic-marking*, the relative syntactic scope between complements and the remaining functors must be the same as the relative scope between *basic-marking* functors and the rest.

Obviously, if complement placement is not constrained, many attachment ambiguities will surface. There will be no corresponding differences in the semantics produced, because the semantic constraints that link the MRS representation for the noun and the MRS representation for its complements are completely lexical (given in the lexical entries for nouns) and not affected by syntax.

There are two possible solutions to prevent this spurious overgeneration. The first one is to have all functors except the ones that occur in Position VIII select for a projection with empty COMPS (or with COMPS of type *olist*, as presented in Section 2.6). Since the ones that occur in this slot are the ones most deeply embedded, if a complement is projected, it can only occur also in this position.

The second solution involves constraining the value of MARKING in the mother node of Head-Complement rules to be of type *basic-marking*.

In either case, Head-Complement rules unify the MARKING value of the head daughter with the MARKING value of the mother node:

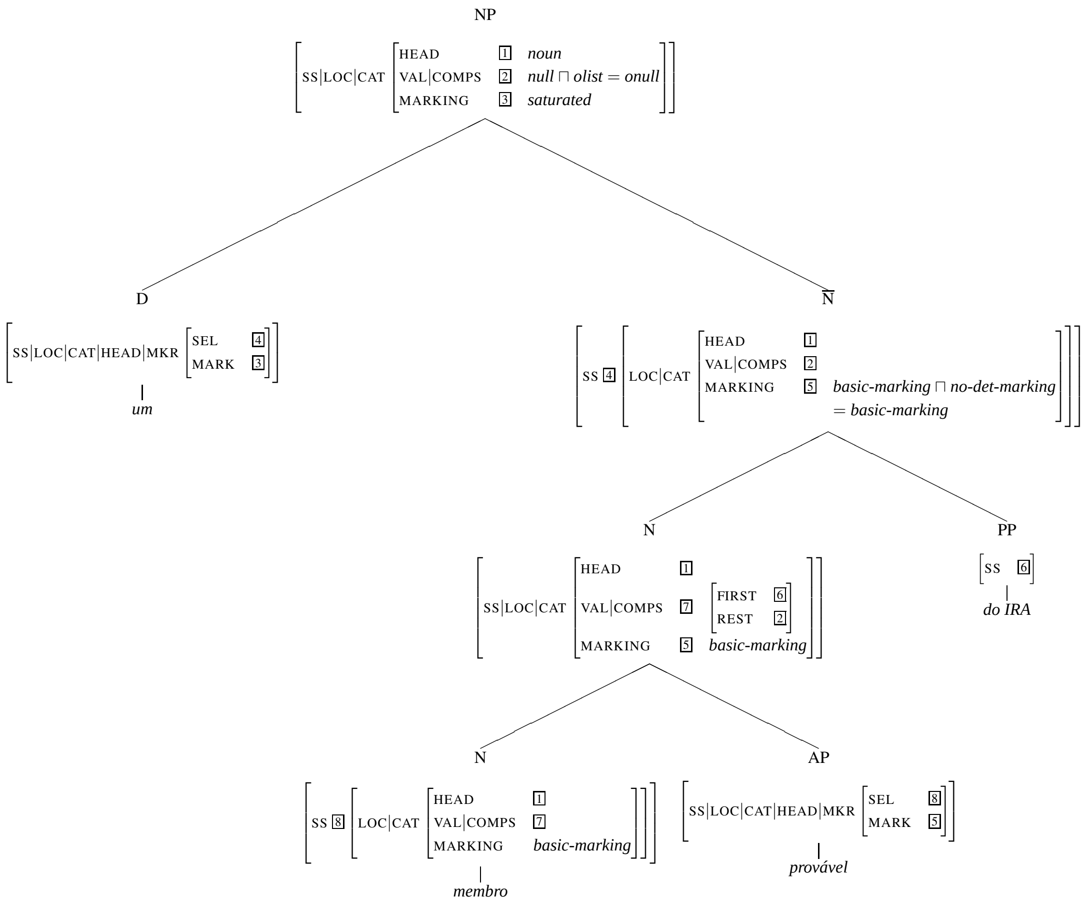

NP
SS|LOC|CAT HEAD 1 noun VAL|COMPS 2 null ⊓ olist = onull MARKING 3 saturated
D
SS|LOC|CAT|HEAD|MKR SEL 4 MARK 3
um
N̄
SS 4 LOC|CAT HEAD 1 VAL|COMPS 2 MARKING 5 basic-marking ⊓ no-det-marking = basic-marking
N
SS|LOC|CAT HEAD 1 VAL|COMPS 7 FIRST 6 REST 2 MARKING 5 basic-marking
PP
SS 6
do IRA
N
SS 8 LOC|CAT HEAD 1 VAL|COMPS 7 MARKING basic-marking
membro
AP
SS|LOC|CAT|HEAD|MKR SEL 8 MARK 5
provável

Figure 4.9: Syntactic analysis for *um membro provável do IRA* (*a probable member of the IRA*). SS abbreviates SYNSEM, LOC abbreviates LOCAL, MKR abbreviates MARKER, and SEL abbreviates SELECT.

$$\begin{bmatrix} \text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{MARKING}\ \boxed{1} \\ \text{HEAD-DTR}|\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{MARKING}\ \boxed{1} \end{bmatrix}$$

The second solution has an important advantage over the first one: the level of saturation at which complements attach is stated in a single place, a type for Head-Complement constructions. This is the solution used in LXGram.

Figure 4.9 shows the analysis of an NP with an adjective intervening between the head and the complement. The NP is *um membro provável do IRA* (*a probable member of the IRA*). In this example the node labeled $\overline{\text{N}}$ is produced via a Head-Complement construction. The remaining phrasal nodes are produced via *functor-head-phrase* or *head-functor-phrase*.

The semantic representation for this NP produced by LXGram is equivalent to

$$\lambda P_{\in D_{\langle e,t\rangle}}.proper\_q(x_1, named(x_1, \text{“}IRA\text{''}), \_um\_q(x_2, \_provável(e, \_membro\_n(x_2, x_1)), P(x_2)))$$

4.9 Prenominal Possessives

In definite NPs, possessives can appear prenominally (43a), while postnominal NPs can occur in indefinite NPs (43b). However, demonstratives license both prenominal and postnominal possessives (44).

(43) a. A minha bicicleta tem um pneu furado.
the my bicycle has a tire flat
My bicycle has a flat tire.

b. Uma bicicleta minha tem um pneu furado.
a bicycle my/mine has a tire flat
A bicycle of mine has a flat tire.

(44) a. Aquela tua bicicleta tem um pneu furado.
that your bicycle has a tire flat
That bicycle of yours has a flat tire.

b. Aquela bicicleta tua tem um pneu furado.
that bicycle your/yours has a tire flat
That bicycle of yours has a flat tire.

Other contexts allow prenominal possessives. Examples are vocatives (48) and predicative nominals lacking a determiner (45b).

(45) a. Minha senhora, eu quero a mala.
my lady, I want the bag
I'd like the bag, Miss.

b. É teu irmão?
is your brother
Is he your brother? (a brother of yours)

Postnominal possessives are covered in Section 4.14, as well as the mechanism to control relative word order between noun and possessive.

Prenominal possessives always occur after the determiner (article, demonstrative, ...), if it is present, and they always precede cardinals, if both occur (46).

(46) a. as minhas duas bicicletas
the my two bicycles
my two bicycles

b. * minhas as duas bicicletas
my the two bicycles

c. * as duas minhas bicicletas
the two my bicycles

The syntax of prenominal possessive can be accounted for via an extension of the type hierarchy of *marking*, presented in Figure 4.10. Two types have been added: *poss-marking* and *no-poss-marking*. The first one is the MARKING value of a constituent that contains a prenominal possessive, the second type is the value of a constituent with no such element.

A value of HEAD for possessives can then be like:

$$\begin{bmatrix} \textit{possessive} & \\ \text{MARKER} & \begin{bmatrix} \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{MARKING} & \textit{no-poss-marking} \end{bmatrix} \\ \text{MARK } \textit{poss-marking} & \end{bmatrix} \end{bmatrix}$$

This accounts for prenominal possessives after a definite article or demonstrative determiner.

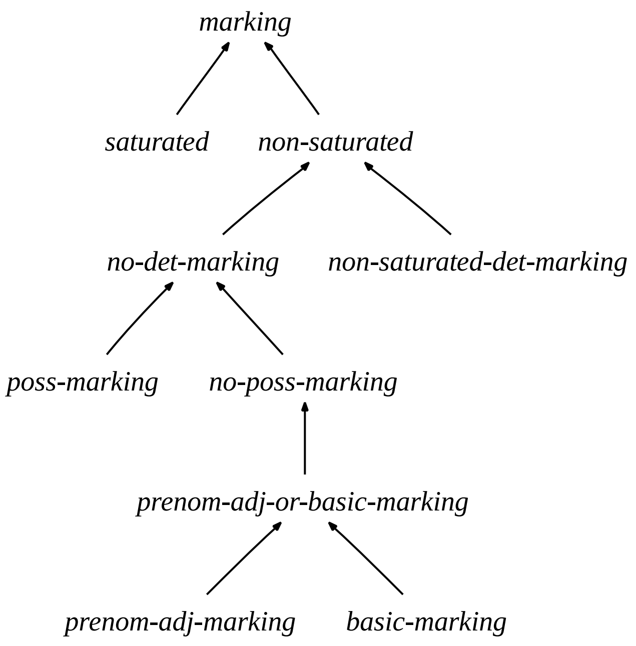


Figure 4.10: Type hierarchy under *marking* (version 4/6). Previous version on p. 83. Final version on p. 118.

We will consider that the other contexts where prenominal possessives are allowed to occur (with no preceding determiner) are not contexts where full NPs occur. For instance, definite articles and demonstratives are impossible in vocatives, as (47) shows. Possessives occur prenominally in vocatives (48).

(47) a. (* A) senhora, eu quero a mala.
the lady, I want the bag
I'd like the bag, Miss.

b. * Essa senhora, eu quero a mala
that lady, I want the bag

(48) a. Minha senhora, eu quero a mala.
my lady I want the bag
I'd like the bag, Madam.

b. * Senhora minha, eu quero a mala.
lady my/mine I want the bag

Noun predicates can also not be full NPs. In (49) a singular count noun occurs in this position, but bare singular NPs are generally not acceptable in European Portuguese (50). Possessives also occur prenominally in such contexts (51).

(49) Ele é pianista.
he is pianist
He is a pianist.

(50) * Ele viu pianista.
he saw pianist

(51) a. É teu irmão?
is your brother
Is he your brother?

b. * É irmão teu?
is brother your/yours

These cases can thus be viewed as not involving a full (saturated) NP. For instance, predicative *ser* (*be*) can be defined as selecting for a complement headed by a *noun* but with a value of marking different from *saturated* (e.g. *non-saturated*).

In Brazilian Portuguese, possessives can introduce an NP. In the corresponding phrases in European Portuguese the definite article must precede the possessive. A Brazilian example is in (52).

(52) Minha bicicleta tem um pneu furado.
my bicycle has a tire flat
My bicycle has a flat tire.

Since quantifier semantics is generally introduced in the predeterminer or determiner slots (Position I and Position II in Appendix A), the Brazilian possessives must have different lexical items from the possessives occurring with determiners, because these do not carry quantifier semantics, but the former must do so (see the next section for semantic representations of possessives). Also, they will present different constraints related to MARKING. More specifically, the feature MARK bears the value *saturated*.

Also note that NPs introduced by a possessive, like the one in (52), do not have readings characteristic of bare NPs — but bare NPs headed by a singular count noun are actually possible in Brazilian Portuguese (Munn and Schmitt, 1998; Müller, 2002)) —, so these NPs should not be considered to be bare NPs.

Prenominal possessives following a definite article or other determiner are also attested in Brazilian Portuguese.

The analysis just presented also covers sequences made up by a predeterminer *todo* (Section 4.5) followed immediately by a possessive, which is a possibility in Brazilian Portuguese. These sequences are derived by the lexical entry for *todo* that is specific to Brazilian Portuguese and a lexical entry for a possessive that is available to both varieties.

4.9.1 Possessives as Arguments of Nouns

Possessives can realize arguments of noun relations, which in Portuguese are in the unmarked case realized by postnominal material (53a). Consider (53b).

(53) a. o irmão da Ana
the brother of the Ana
Ana's brother

b. o seu irmão
the her brother
her brother

In both examples, *irmão* denotes a two-place predicate. In (53a) the second argument is realized by the PP *de Ana*, and in (53b) it surfaces as *seu*.

Possessives are implemented in LXGram as carrying personal pronoun semantics. Personal pronouns are associated with two relations *pronoun_q_rel* and *pronoun_n_rel*. The second one fills the restrictor of the quantifier relation *pronoun_q_rel*, so personal pronouns receive semantics similar to $\lambda P.pronoun\_q(x, pronoun\_n(x), P(x))$. This accounts for the deictic use of personal pronouns (their use as denoting an entity that is recovered from context, although LXGram does not recover that entity from the context, as it only parses isolated sentences).[4] This treatment of personal pronouns is a simplification. Consider a counter-example: *If anyone knew that*, **he** *wouldn't tell*. We could adopt an MRS representation inspired by the treatment of personal pronouns in Discourse Representation Theory (Kamp and Reyle, 1993) in order to address this issue. It would then be equivalent to $\lambda P.pronoun\_q(x, x = y)$, where y is free (to be recovered from the context). However, we assume the first representation shown, in order to conform to the other computational HPSGs.

When possessives do not fill a noun argument, an extra relation is included between the index of the personal pronoun and that of the head noun, called *possessive_a_rel*. An example is in Figure 4.11. The MRS in that example (*o seu cavalo* — *his/her/their horse*) corresponds to

$$\lambda P.\_o\_q(x2, pronoun\_q(x7, pronoun\_n(x7), possessive\_a(x2, x7) \wedge \_cavalo\_n(x2)), P(x2))$$

and to

$$\lambda P.pronoun\_q(x7, pronoun\_n(x7), \_o\_q(x2, possessive\_a(x2, x7) \wedge \_cavalo\_n(x2), P(x2)))$$

The two are equivalent under the assumption that $\lambda P.pronoun\_q(x, pronoun\_n(x), P(x))$ is equivalent $\lambda P.P(c)$ (both reduce to $\lambda P.\_o\_q(x2, possessive\_a(x2, c) \wedge \_cavalo\_n(x2)), P(x2)))$.

When possessives realize noun arguments, this relation is not present in the MRS. Instead, the index of the personal pronoun occurs as the second argument of the relation corresponding to the head noun. Figure 4.12 contains an MRS example of argumental possessives, for the NP *o seu irmão* (*his/her/their brother*). The MRS in that example is equivalent to

$$\lambda P.\_o\_q(x2, pronoun\_q(x7, pronoun\_n(x7), \_irmão\_n(x2, x7)), P(x2))$$

and to

$$\lambda P.pronoun\_q(x7, pronoun\_n(x7), \_o\_q(x2, \_irmão\_n(x2, x7)), P(x2))$$

Both are intended to mean $\lambda P.\_o\_q(x2, \_irmão\_n(x2, c), P(x2))$.

Because the number of elementary predications contributed to an MRS by these two sorts of elements (argumental vs. modifying possessives) is different, multiple lexical entries are required for possessives.

Argumental possessives and modifying possessives have the same syntactic distribution, though. This creates problems for the treatment of argumental possessives, since we have assumed in Section 4.8 that noun complements are saturated at a much lower level.

The first question to ask is whether projections of prenominal argumental possessives should be produced by some Head-Complement construction or by the *functor-head-phrase* discussed above.

[4]For instance, for a sentence like "he left", we can think of the semantics $leave(c)$, where c is a constant. We can say that "he" denotes c and "leave" denotes $\lambda x_{\in D_e}.leave(x)$, and that the denotation of the sentence is the denotation of the VP applied to the denotation of the NP subject — $(\lambda x.leave(x))c$. Equivalently, we can also say that "he" denotes $\lambda P_{\in D_{\langle e,t \rangle}}.P(c)$, with the same semantics for the verb as before. In this case, the denotation of the sentence is the denotation of the NP subject applied to the denotation of the VP — $(\lambda P.P(c))(\lambda x.leave(x))$. This is more convenient, because the denotation of sentences with an NP subject that has quantificational force, like "all men left", is also the denotation of the NP subject applied to the denotation of the VP — $(\lambda P.\forall x[man(x) \rightarrow P(x)])(\lambda x.leave(x))$. The expression $\lambda P.pronoun\_q(x, pronoun\_n(x), P(x))$ can be seen as meaning $\lambda P.P(c)$, as they are both of the semantic type $\langle\langle e,t\rangle,t\rangle$ (a set A of sets of entities, or a function from a set of entities to truth values that yields truth just in case its argument is in A).

In the MRS universe, the representation used here is employed in several other grammars. We chose to also use it in LXGram, in order to have representations similar to the other grammars.

The treatment of NPs with proper names is similar. Instead of saying that e.g. *Mary* denotes a constant m, we say it denotes $\lambda P_{inD_{\langle e,t \rangle}}.P(m)$, for the same reasons. In LXGram "a Maria" receives the MRS equivalent of $\lambda P.proper\_q(x, named(x, \text{“}Maria''), P(x))$.

$$
\left[\begin{array}{ll}
\textit{mrs} & \\
\text{LTOP} & \boxed{h1}\ h \\
\text{INDEX} & \boxed{x2}\ x \\
\text{RELS} & \left\langle \begin{array}{l}
\left[\begin{array}{ll}\textit{\_o\_q\_rel} & \\ \text{LBL} & \boxed{h1} \\ \text{ARG0} & \boxed{x2} \\ \text{RSTR} & \boxed{h4}\ h \\ \text{BODY} & \boxed{h3}\ h\end{array}\right],
\left[\begin{array}{ll}\textit{possessive\_a\_rel} & \\ \text{LBL} & \boxed{h5}\ h \\ \text{ARG0} & \boxed{e6}\ e \\ \text{ARG1} & \boxed{x2} \\ \text{ARG2} & \boxed{x7}\left[\begin{array}{ll} x & \\ \text{PNG.PERSON} & \textit{3rd}\end{array}\right]\end{array}\right], \\
\left[\begin{array}{ll}\textit{pronoun\_q\_rel} & \\ \text{LBL} & \boxed{h8}\ h \\ \text{ARG0} & \boxed{x7} \\ \text{RSTR} & \boxed{h9}\ h \\ \text{BODY} & \boxed{h10}\ h\end{array}\right],
\left[\begin{array}{ll}\textit{pronoun\_n\_rel} & \\ \text{LBL} & \boxed{h11}\ h \\ \text{ARG0} & \boxed{x7}\end{array}\right],
\left[\begin{array}{ll}\textit{\_cavalo\_n\_rel} & \\ \text{LBL} & \boxed{h5} \\ \text{ARG0} & \boxed{x2}\end{array}\right]
\end{array}\right\rangle \\
\text{HCONS} & \left\langle \left[\begin{array}{ll}\textit{qeq} & \\ \text{HARG} & \boxed{h4} \\ \text{LARG} & \boxed{h5}\end{array}\right], \left[\begin{array}{ll}\textit{qeq} & \\ \text{HARG} & \boxed{h9} \\ \text{LARG} & \boxed{h11}\end{array}\right] \right\rangle
\end{array}\right]
$$

Figure 4.11: MRS fragment corresponding to the NP *o seu cavalo* (*his/her/their horse*).

$$
\left[\begin{array}{ll}
\textit{mrs} & \\
\text{LTOP} & \boxed{h1}\ h \\
\text{INDEX} & \boxed{x2}\ x \\
\text{RELS} & \left\langle
\left[\begin{array}{ll}\textit{\_o\_q\_rel} & \\ \text{LBL} & \boxed{h1} \\ \text{ARG0} & \boxed{x2} \\ \text{RSTR} & \boxed{h4}\ h \\ \text{BODY} & \boxed{h3}\ h\end{array}\right],
\left[\begin{array}{ll}\textit{pronoun\_q\_rel} & \\ \text{LBL} & \boxed{h5}\ h \\ \text{ARG0} & \boxed{x6}\left[\begin{array}{ll} x & \\ \text{PNG.PERSON} & \textit{3rd}\end{array}\right] \\ \text{RSTR} & \boxed{h7}\ h \\ \text{BODY} & \boxed{h8}\ h\end{array}\right],
\left[\begin{array}{ll}\textit{pronoun\_n\_rel} & \\ \text{LBL} & \boxed{h9}\ h \\ \text{ARG0} & \boxed{x6}\end{array}\right],
\left[\begin{array}{ll}\textit{\_irmão\_n\_-de-\_rel} & \\ \text{LBL} & \boxed{h10}\ h \\ \text{ARG0} & \boxed{x2} \\ \text{ARG1} & \boxed{x6}\end{array}\right]
\right\rangle \\
\text{HCONS} & \left\langle \left[\begin{array}{ll}\textit{qeq} & \\ \text{HARG} & \boxed{h4} \\ \text{LARG} & \boxed{h10}\end{array}\right], \left[\begin{array}{ll}\textit{qeq} & \\ \text{HARG} & \boxed{h7} \\ \text{LARG} & \boxed{h9}\end{array}\right] \right\rangle
\end{array}\right]
$$

Figure 4.12: MRS fragment corresponding to the NP *o seu irmão* (*his/her/their brother*).

The motivation for considering argumental possessives to be complements is that they are in complementary distribution with PP complements (54a). The motivation for considering them functors is that they are also in complementary distribution with modifying possessives (54b).

(54) a. * o seu irmão da Ana
the her brother of the Ana

b. * o seu seu irmão
the her her brother

If they are treated as functors, then they are unusual in saturating an argument of the head they select.

If they are complements, then prenominal argumental possessives are unusual in preceding the head (in Portuguese this only occurs with clitics and fronted constituents).

In LXGram they are implemented as functors. This choice has the advantage of not requiring more syntactic machinery, but it results in untypical feature structures because, since argumental possessives are considered functors, they cannot discharge an element from the COMPS list of their head, in spite of realizing it themselves.

Since they can see the entire SYNSEM of their sister node via the SELECT attributes, they can unify the index of the personal pronoun relations they introduce with the index of an element in the COMPS attribute of the nominal projection they select for. This produces the right semantics, namely semantic representations exactly like the ones produced by Head-Complement constructions.

They place the same constraints on the values of *marking* as their modifier counterparts. Their non-iterability is in this way immediately predicted.

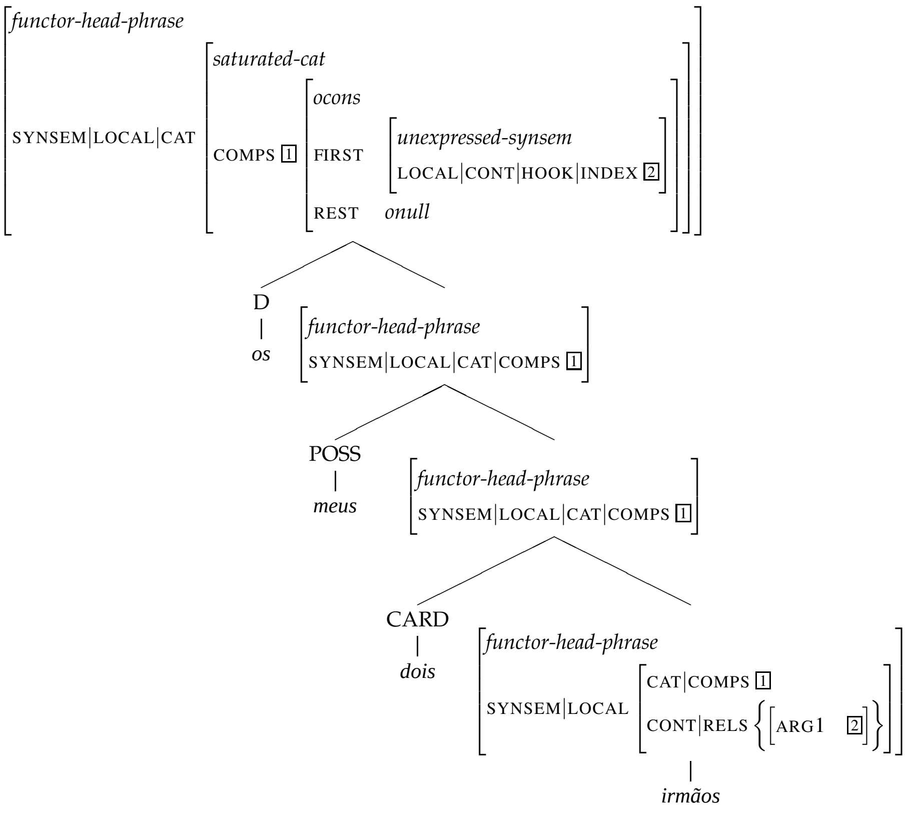
It is important to mention that what enables a non-empty COMPS to appear high enough in a tree in order to be visible by possessives in general is the choice of using type *olist* instead of *null* to constrain the COMPS of NPs, as explained in Section 4.3. Consider the following example:

(55) [$_{\text{NP}}$ os [meus [dois irmãos]]]
the my two brothers

my two brothers

All bracketed phrases in this example are instances of *functor-head-phrase* and as such have the same value for the feature COMPS. If NPs were constrained to have an empty COMPS, a unary rule would be needed to discharge the unexpressed complement of the noun (this rule could simply pass up the tail of the COMPS of its daughter). It would make sense to have this rule apply in the most embedded position (before the cardinal attaches), since that is where Head-Complement constructions occur. This way some of the constraints common to unary and binary Head-Complement constructions could be factored out in a single supertype. Discharging all complements in the same position is also less error-prone and makes the grammar easier to understand, to extend and to debug if needed. In this scenario, the sister node of the possessive would also have an empty COMPS.

By using the type *olist* (see Section 2.6) to constrain the COMPS of NPs instead (see Section 4.3), unrealized complements are visible at the point where possessives attach. For instance, the NP in (55) receives the following simplified analysis:



Here we are assuming that relational nouns, like *irmão* (*brother*) above, unify the INDEX of their complement with the ARG1 of the relation they introduce, so that the entry for *irmão* would contain these constraints, among others:

$$\begin{bmatrix}\text{ORTH ”irmão”} & \\ \text{SYNSEM}|\text{LOCAL} & \begin{bmatrix} \text{CAT}|\text{VAL}|\text{COMPS} \left\langle \begin{bmatrix}\text{LOCAL}|\text{CONT}|\text{HOOK}|\text{INDEX}\ \boxed{1}\end{bmatrix}\right\rangle \\ \text{CONT}|\text{RELS} \left\{\begin{bmatrix}\text{ARG1}\ \boxed{1}\end{bmatrix}\right\}\end{bmatrix}\end{bmatrix}$$

Constraints in the lexical types for argumental possessive can then be used to unify the index associated with the personal pronoun relations they introduce with the index of the first element in the head’s COMPS:

$$\begin{bmatrix}\text{SYNSEM}|\text{LOCAL} & \begin{bmatrix}\text{CAT}|\text{HEAD}|\text{MARKER}|\text{SELECT}|\text{LOCAL}|\text{CAT}|\text{VAL}|\text{COMPS}|\text{FIRST}|\text{LOCAL}|\text{CONT}|\text{HOOK}|\text{INDEX}\ \boxed{1} \\ \text{CONT}|\text{HOOK}|\text{INDEX}\ \boxed{1}\end{bmatrix}\end{bmatrix}$$

Crucially, the possessive cannot simply unify the entire SYNSEM of the head’s complement with its SYNSEM, for a number of reasons:(1) a cyclic structure would result, a situation that is not allowed by the systems used; (2) the noun selects for a PP, but a possessive is not a PP — the HEAD feature is different, for instance, and would not unify —; and (3) the first element of COMPS, which in examples like (55) is reduced to type *olist*, is an *unexpressed-synsem*, but the SYNSEM of the possessive ends up as a *canonical-synsem*, since it is realized, and these synsem types are incompatible (see Section 2.6).

The fact that the complement of a noun with a synsem of type *unexpressed-synsem* is actually realized makes this analysis rather uninteresting.

Since possessives can only realize PP complements of nouns (and not CPs for instance), argumental possessives must constrain the nominal projection they attach to to have a COMPS whose first element is a PP:

[SYNSEM|LOCAL|CAT|HEAD|MARKER|SELECT|LOCAL|CAT|VAL|COMPS|FIRST|LOCAL|CAT|HEAD *preposition*]

This constraint, like the constraint above to fix the semantics, is extremely non-local and against the spirit of HPSG.

Furthermore, it means that nouns do not necessarily have visibility over the entire SYNSEM of their complement: if nouns constrain it to be a PP, a possessive can detect this and realize it instead, but a possessive can have constrains on its SYNSEM drastically different from the constraints on the noun's complement. A consequence is that if constraints on noun complements must be added in the future to cover additional phenomena, it may be the case that the definitions for argumental possessives require modifications as well — the analysis is not extensible.

This is the analysis implemented in LXGram currently. An interesting alternative to the analysis of prenominal possessives is to treat them as elements extracted from a postnominal position. An analysis could be envisaged in a way similar to the treatment of long-distance dependencies, but possibly resorting to other features, so as to not interact with the analysis of unbounded dependencies. This would explain the paradox of arguments realized by possessives surfacing on the left of their head, and, under the assumption of a parallelism between sentence structure and NP structure, it would provide the NP counterpart for the left periphery of sentences.

Modifying possessives can occur with nouns with unsaturated arguments. There are some examples that are in fact ambiguous this way. For instance, if we consider that a noun like *livro*/*book* has an argument denoting the person who wrote it, the following NP can mean *the book that you have* (the interpretation that results from the possessive being a modifier) and *the book that you wrote* (the interpretation where the possessive realizes an argument):

(56) a.

o	teu	livro
the	your	painting

your book

Unfortunately, NPs like *o meu pai* (*my father*) also receive two analyses: one in which the possessive realizes the noun's argument, with semantics similar to the representation in Figure 4.12; and another in which the possessive has modifier semantics, similar to Figure 4.11 The last interpretation is not natural.

4.10 Cardinals, Ordinals and Markers of Indefinite Specifics

Position IV can be filled in by cardinals, ordinals or markers of indefinite specific NPs, like *certo* or *determinado* (*certain*).

They can co-occur with each other in almost any order (57), the exception being that ordinals cannot precede markers of indefinite specifics, as in (57d).

(57) a.

os	primeiros	dois	capítulos
the	first	two	chapters

the first two chapters

b.

os	dois	primeiros	capítulos
the	two	first	chapters

the first two chapters

c.

um	certo	primeiro	capítulo
a	certain	first	chapter

a certain first chapter

d. *

um	primeiro	certo	capítulo
a	first	certain	chapter

a certain first chapter

Certo is limited to indefinite NPs. Cardinals cannot co-occur with indefinite determiners,[5] so to test the word order possibilities between cardinals and *certo*, we have to look at NPs that begin with a cardinal or *certo*, as in (58). Such NPs are covered in Section 4.11, but (58) already shows that word order between cardinals and markers of indefinite specifics is in general also unconstrained.

(58) a.

dois	certos	capítulos
two	certain	chapters

two certain chapters

b.

certos	dois	capítulos
certain	two	chapters

two certain chapters

At most one item of each class can be present (59). They are not repeatable even when an item of a different sort intervenes (60).

(59) a. *

Os	dois	três	carros	avariaram.
the	two	three	cars	broke down

b. *

O	primeiro	segundo	lugar	está	ocupado.
the	first	second	seat	is	taken

c. *

Um	determinado	certo	carro	avariou.
a	certain	certain	car	broke down

(60) a. *

Os	dois	primeiros	três	lugares	estão	ocupados.
the	two	first	three	places	are	taken

b. *

Os	primeiros	dois	segundos	pratos	estão	atrasados.
the	first	two	second	dishes	are	late

c. *

Certos	dois	certos	carros	avariaram.
certain	two	certain	cars	broke down

A class of prenominals, "vague quantifiers" or "quantificational adjectives", has the exact distribution of cardinals (61).

(61)

Os	vários	participantes	passeiam	as	folhas	pela	sala.
the	various	participants	walk	the	paper sheets	through the	room

The various participants walk the paper sheets through the room.

They cannot co-occur with cardinals (62).

(62) a. *

os	vários	vinte	participantes
the	various	twenty	participants

b. *

os	vinte	vários	participantes
the	twenty	various	participants

[5] NPs like *some three cars* can be analyzed as involving an item *some* that is not a determiner but rather a modifier of the cardinal, since *some three* roughly means *around three*. This also applies to Portuguese expressions like *alguns três*, *uns três*, with the same meaning.

Vague quantifiers do occur with ordinals (63) and markers of indefinite specifics (64).

(63) a. os vários primeiros lugares
the various first seats
the various first seats

b. os primeiros vários lugares
the first various seats
the various first seats

(64) a. vários certos participantes
various certain participants
various certain participants

b. certos vários participantes
certain various participants
various certain participants

They cannot iterate (65).

(65) a. * os vários vários participantes
the various various participants

b. * os vários vinte vários participantes
the various vinte various participants

Vague quantifiers can thus be constrained exactly like cardinals. In the following discussion we will thus ignore them and only talk about cardinals.

Similarly, the class of ordinals can also be considered to include other elements with the same syntactic distribution. This is the case of items like *último* (*last*) and *próximo* (*next*). Consider:

(66) a. * os próximos primeiros capítulos
the next first chapters

b. * os primeiros próximos capítulos
the first next chapters

c. os três próximos capítulos
the three next chapters
the next three chapters

d. os próximos três capítulos
the next three chapters
the next three chapters

We will also have these elements in mind when we discuss ordinals, from now on. It should be mentioned that superlative forms of adjectives do not pattern with items like *último* or *próximo* or ordinals:[6]

(67) a. os dois melhores capítulos
the two best chapters
the two best chapters / the best two chapters

b. * os melhores dois capítulos
the best two chapters

[6]We mention superlatives because there are semantic and etymological similarities between superlative forms of adjectives and these items. There are also some syntactic similarities, like *o primeiro N de todos* (*the first N of all*), *o último N de todos* (*the last N of all*), *o melhor N de todos* (*the best N of all*).

We now turn to the discussion of implementing these three classes: ordinals, cardinals/vague quantifiers and markers of indefinite specific NPs.

A type hierarchy of *marking* alone cannot prevent these items from iterating — since the relative order between them is free, and they occupy the same slot as far as word order between them and the remaining NP elements is concerned, the way the features MARK and MARKING are constrained should be relatively similar for all of them.

Cardinals have the following type as the value of their HEAD feature:

$$\begin{bmatrix} \textit{cardinal} & \\ \textsc{marker} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \textsc{select}|\textsc{local}|\textsc{cat} & \begin{bmatrix} \textsc{head} & \textit{noun} \\ \textsc{marking} & \textit{no-poss-marking} \end{bmatrix} \\ \textsc{mark}\ \textit{no-poss-marking} & \end{bmatrix} \end{bmatrix}$$

Ordinals, and markers of indefinite specifics have similar constraints. These constraints make them attach closer to the noun than possessives.

As such, nothing prevents these items from iterating.

What is required is another way of recording which elements have already been saturated. This can be implemented by adding features under *marking*. Binary features can be used under *marking* types as in (Allegranza, 1998a,b). For instance, a feature ORDINAL can be used to denote whether a cardinal is present in a structure.

Although this distinction is binary, we do not use the type *bool*, since it is not immediately obvious whether a feature ORDINAL with the value + represents a structure where an ordinal is present and no other ordinal can combine with it or one where an ordinal is not present and it is possible for another ordinal to attach to it. A different set of values, in Figure 4.13, is used instead, in a hierarchy with a shape similar to that of boolean types.

$$\begin{tikzcd} & \textit{present-or-absent} & \\ \textit{present} \arrow[ur] & & \textit{absent} \arrow[ul] \end{tikzcd}$$

Figure 4.13: Type hierarchy under *present-or-absent* (version 1/2). Final version on p. 125.

Getting back to the example of cardinals, a syntactic constituent containing a cardinal will have the value *present* for the feature CARDINAL. Cardinals select a constituent with CARDINAL of type *absent* and produce a node with CARDINAL of type *present*. Because the CARDINAL attribute of the mother node is different from the CARDINAL of the head daughter, the natural place to put it is under MARKING.

Other features are needed to encode the presence of ordinals and markers of indefinite specific NPs: ORDINAL and INDEF-SPEC. In functors that are not ordinals, cardinals or markers of indefinite specifics, all of these features will have to be passed from their sister up to the mother node. This is achieved by unifying the new features under the attribute MARK and under the MARKING of the selected constituent. These functors may constrain these two features with different subtypes of *marking*, so unifying MARK and MARKING is not possible. We can therefore group these three attributes under another one, MK-VAL, which is appropriate for *marking*:

$$\begin{bmatrix} \textit{marking} & \\ \textsc{mk-val} & \textit{mk-val} \end{bmatrix}$$

$$\begin{bmatrix} \textit{mk-val} & \\ \textsc{cardinal} & \textit{present-or-absent} \\ \textsc{ordinal} & \textit{present-or-absent} \\ \textsc{indef-spec} & \textit{present-or-absent} \end{bmatrix}$$

Unifying MK-VAL thus unifies all these subfeatures. For instance, the head of adjectives now has the additional constraints:

$$\begin{bmatrix} \textit{adjective} & \\ \textsc{marker} & \begin{bmatrix} \textsc{mark}|\textsc{mk-val}\ \boxed{1} \\ \textsc{select}|\textsc{local}|\textsc{cat}|\textsc{marking}|\textsc{mk-val}\ \boxed{1} \end{bmatrix} \end{bmatrix}$$

Minimal types, as presented in Section 3.5.1 can also be used. This strategy has already been presented, so we explain it here briefly. The idea is to create a most general type with no features for MK-VAL, say *mk-val-min*, a subtype of *mk-val-min* for each feature that has to be used, where that feature is declared (*mkv-cardinal* for CARDINAL, *mkv-ordinal* for ORDINAL and *mkv-indef-spec* for INDEF-SPEC), and subtypes covering all combinations of features, resulting in a type hierarchy with 2^n types for n features. A possible hierarchy is:

$$\begin{tikzcd}
 & \textit{mk-val-min} & \\
\textit{mkv-cardinal} \arrow[ur] & \textit{mkv-ordinal} \arrow[u] & \textit{mkv-indef-spec} \arrow[ul] \\
\textit{mkv-cardinal-ordinal} \arrow[u] \arrow[ur] & \textit{mkv-cardinal-indef-spec} \arrow[ul] \arrow[ur] & \textit{mkv-ordinal-indef-spec} \arrow[ul] \arrow[u] \\
 & \textit{mk-val} \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

In *marking*, MK-VAL is declared to be of the type *mk-val-min*, instead of *mk-val*. Although MK-VAL is inherited by all subtypes of *marking*, the subfeatures will not be present if unconstrained.

For instance, prepositions constrain the constituent they select to have MARKING of type *basic-marking*. When they attach to a VP, the MARKING feature of that VP node will have a subfeature MK-VAL, but under it there will be no feature CARDINAL, ORDINAL and INDEF-SPEC, as they can be left unconstrained there (they do not make sense for verb headed constituents).

Cardinals now have the following constraints under their HEAD:

$$\begin{bmatrix} \textit{cardinal} & \\ \text{MARKER} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \text{MARK} & \begin{bmatrix} \textit{no-poss-marking} & \\ \text{MK-VAL} & \begin{bmatrix} \text{CARDINAL} & \textit{present} \\ \text{ORDINAL} & \boxed{1} \\ \text{INDEF-SPEC} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD}\ \textit{noun} & \\ \text{MARKING} & \begin{bmatrix} \textit{no-poss-marking} & \\ \text{MK-VAL} & \begin{bmatrix} \text{CARDINAL} & \textit{absent} \\ \text{ORDINAL} & \boxed{1} \\ \text{INDEF-SPEC} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The HEAD of ordinals is:

$$\begin{bmatrix} \textit{ordinal} & \\ \text{MARKER} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \text{MARK} & \begin{bmatrix} \textit{no-poss-marking} & \\ \text{MK-VAL} & \begin{bmatrix} \text{CARDINAL} & \boxed{1} \\ \text{ORDINAL} & \textit{present} \\ \text{INDEF-SPEC} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD}\ \textit{noun} & \\ \text{MARKING} & \begin{bmatrix} \textit{no-poss-marking} & \\ \text{MK-VAL} & \begin{bmatrix} \text{CARDINAL} & \boxed{1} \\ \text{ORDINAL} & \textit{absent} \\ \text{INDEF-SPEC} & \boxed{2}\ \textit{absent} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The constraint on the feature INDEF-SPEC to be *absent* is to prevent ordinals from preceding *certo* and *determinado*, thus blocking examples like (57d).

The HEAD of markers of indefinite specifics is very similar:

$$\left[\begin{array}{ll} \textit{indef-specific} & \\ \text{MARKER} & \left[\begin{array}{ll} \textit{pre-only-marker-min} & \\ \text{MARK} & \left[\begin{array}{ll} \textit{no-poss-marking} & \\ \text{MK-VAL} & \left[\begin{array}{ll} \text{CARDINAL} & \boxed{1} \\ \text{ORDINAL} & \boxed{2} \\ \text{INDEF-SPEC} & \textit{present} \end{array}\right] \end{array}\right] \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \left[\begin{array}{ll} \text{HEAD } \textit{noun} & \\ \text{MARKING} & \left[\begin{array}{ll} \textit{no-poss-marking} & \\ \text{MK-VAL} & \left[\begin{array}{ll} \text{CARDINAL} & \boxed{1} \\ \text{ORDINAL} & \boxed{2} \\ \text{INDEF-SPEC} & \textit{absent} \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Nouns come in the lexicon with MARKING of type *basic-marking*, as before, but additionally all features under MK-VAL have *absent* as their value:

$$\left[\begin{array}{ll} \text{SYNSEM}|\text{LOCAL}|\text{CAT} & \left[\begin{array}{ll} \text{HEAD} & \textit{noun} \\ \text{MARKING} & \left[\begin{array}{ll} \textit{basic-marking} & \\ \text{MK-VAL} & \left[\begin{array}{ll} \text{CARDINAL} & \textit{absent} \\ \text{ORDINAL} & \textit{absent} \\ \text{INDEF-SPEC} & \textit{absent} \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

4.10.1 Semantics of Markers of Indefinite Specifics

Prenominal items like *certo* and *determinado* (*certain*) as in the examples (26), repeated below, carry no semantic relations but instead simply restrict the set of available readings:

(68) a.

Todas	as	pessoas	leram	um	certo	livro.
all	the	people	have read	a	certain	book

All people have read a certain book.

$\exists y[\textit{book}(y) \wedge \forall x[\textit{person}(x) \rightarrow \textit{read}(x,y)]]$

b.

Todas	as	pessoas	leram	um	livro.
all	the	people	have read	a	book

All people have read a book.

$\forall x[\textit{person}(x) \rightarrow \exists y[\textit{book}(y) \wedge \textit{read}(x,y)]]$

$\exists y[\textit{book}(y) \wedge \forall x[\textit{person}(x) \rightarrow \textit{read}(x,y)]]$

The MRS assigned by LXGram to the sentence in (68a) is in Figure 4.14, and the MRS for the sentence in (68b) is in Figure 4.15.

The two MRSs have exactly the same relations and handle constraints. The only differences lie in the values of the features SCOPE in some of the handles (of type *h*) in these MRSs. The handles for which this feature is not displayed have it completely unconstrained (minimal types for handles are used to hide unconstrained SCOPE features).

Without these constraints for the SCOPE feature, these two MRSs can be scope resolved in the two following formulas:

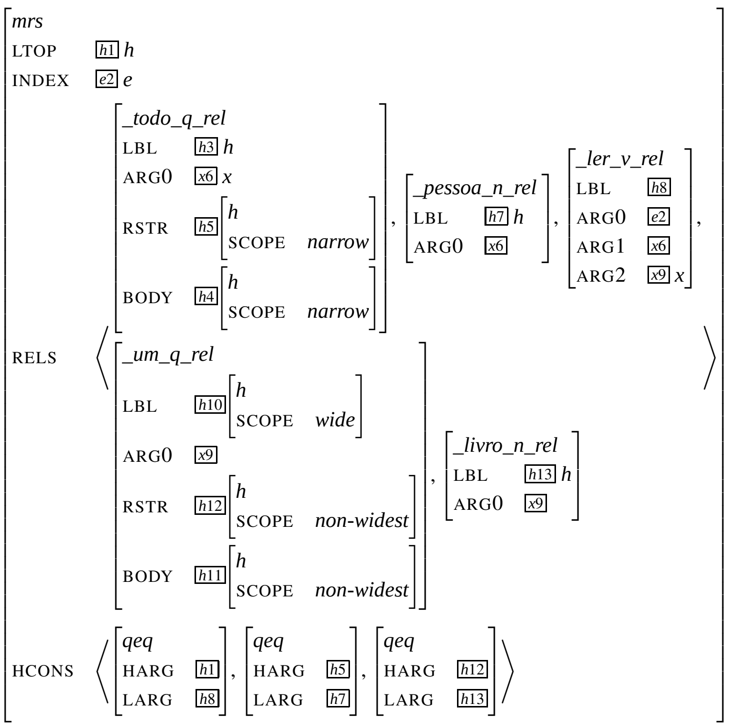
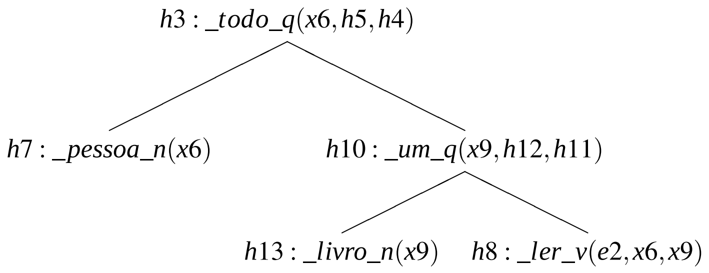


Figure 4.14: MRS with constrained quantifier scope. The sentence is *Todos as pessoas leram um certo livro* (*all people have read a certain book*).

- $\_um\_q(x9, \_livro\_n(x9), \_todo\_q(x6, \_pessoa\_n(x6), \_ler\_v(e2, x6, x9)))$
- $\_todo\_q(x6, \_pessoa\_n(x6), \_um\_q(x9, \_livro\_n(x9), \_ler\_v(e2, x6, x9)))$

These are in fact the two readings for the sentence in (68b). The constraints on SCOPE are intended to block the second reading for the example (68a), in Figure 4.14.

The idea is that, in the second reading, the handle tagged with $\boxed{h4}$ and the handle tagged with $\boxed{h10}$ in these MRSs correspond to the same node in the syntax tree for the scoped formula:



Since they represent the same node, we can assume that they must be compatible. The approach is then to make the constraints on these two handles incompatible in the MRS for the example (68a), but compatible in the MRS for the sentence in (68b).

The type hierarchy for the values that the feature SCOPE can take is in Figure 4.16.

The MRS in Figure 4.14 (for the example in (68a)) has $\boxed{h4}$ with its feature SCOPE with the value *narrow*, and the SCOPE feature of the handle $\boxed{h10}$ has the value *wide*. These types are incompatible according to the hierarchy in Figure 4.16.

This mechanism does not work in practice, because the LKB scope resolution algorithm does not perform unification operations on the handles that end up denoting the same node in the fully scoped

$$\begin{bmatrix} \textit{mrs} & \\ \text{LTOP} & \boxed{h1}\ h \\ \text{INDEX} & \boxed{e2}\ e \\ \text{RELS} & \left\langle \begin{array}{l} \begin{bmatrix} \textit{\_todo\_q\_rel} & \\ \text{LBL} & \boxed{h3}\ h \\ \text{ARG0} & \boxed{x6}\ x \\ \text{RSTR} & \boxed{h5}\begin{bmatrix} h & \\ \text{SCOPE} & \textit{narrow} \end{bmatrix} \\ \text{BODY} & \boxed{h4}\begin{bmatrix} h & \\ \text{SCOPE} & \textit{narrow} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textit{\_pessoa\_n\_rel} & \\ \text{LBL} & \boxed{h7}\ h \\ \text{ARG0} & \boxed{x6} \end{bmatrix}, \begin{bmatrix} \textit{\_ler\_v\_rel} & \\ \text{LBL} & \boxed{h8} \\ \text{ARG0} & \boxed{e2} \\ \text{ARG1} & \boxed{x6} \\ \text{ARG2} & \boxed{x9}\ x \end{bmatrix}, \\ \begin{bmatrix} \textit{\_um\_q\_rel} & \\ \text{LBL} & \boxed{h10}\ h \\ \text{ARG0} & \boxed{x9} \\ \text{RSTR} & \boxed{h12}\begin{bmatrix} h & \\ \text{SCOPE} & \textit{non-widest} \end{bmatrix} \\ \text{BODY} & \boxed{h11}\begin{bmatrix} h & \\ \text{SCOPE} & \textit{non-widest} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \textit{\_livro\_n\_rel} & \\ \text{LBL} & \boxed{h13}\ h \\ \text{ARG0} & \boxed{x9} \end{bmatrix} \end{array} \right\rangle \\ \text{HCONS} & \left\langle \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h1} \\ \text{LARG} & \boxed{h8} \end{bmatrix}, \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h5} \\ \text{LARG} & \boxed{h7} \end{bmatrix}, \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h12} \\ \text{LARG} & \boxed{h13} \end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 4.15: MRS allowing for quantifier scope ambiguity. The sentence is *Todos as pessoas leram um livro* (*all people have read a book*).

$$\begin{tikzcd} & \textit{scope} & & \\ \textit{widest} \arrow[ur] & & \textit{non-widest} \arrow[ul] & \\ & \textit{wide} \arrow[ur] & & \textit{narrow} \arrow[ul] \end{tikzcd}$$

Figure 4.16: Type hierarchy under *scope*

formulas. For this reason, these constraints on handles are still experimental in LXGram. It would of course be possible to resolve MRSs with an external component, that could take this information into account.

Markers of indefinite specifics contribute no semantics. Instead they simply constrain the features SCOPE of the associated quantifier relation. In LXGram, the quantifier relation of an NP is accessible in all noun headed phrases in the feature QUANT-REL under a feature KEYS (for key relations).[7]. Markers of indefinite specific NPs simply have the constraint:

$$\left[\text{SYNSEM}|\text{LOCAL}|\text{CAT}|\text{HEAD}|\text{MARKER}|\text{SELECT}|\text{LOCAL}|\text{CONT}|\text{KEYS}|\text{QUANT-REL}|\text{LBL}|\text{SCOPE}\ \textit{wide}\right]$$

If a marker of indefinite specifics is not present, this feature will simply have a more general type (allowing for more scope resolution possibilities, as desired).

The type *widest* in Figure 4.16 is used to constrain the LBL of the *proper_q_rel*, which is used with proper names. It is meant to ensure that proper names receive widest scope. The value *wide* is given to the LBL of quantifier relations associated with an NP where a marker of indefinite specifics is present.

[7]Items that introduce quantifier relations simply unify this relation with the value of this feature. The feature KEYS is unified between the mother node and the head daughter in all headed constructions.

The RSTR and BODY features of all quantifier relations are constrained with the SCOPE value *non-widest*, except in *proper_q_rel* relation, where they are not constrained. Quantifier relations of determiners and predeterminers that cannot occur with markers of indefinite specifics in the same NP (e.g. *todo* — *all*) have the SCOPE under these two features (RSTR and BODY) further constrained to be *narrow*. This set of items contains the predeterminer *todo* (*all*), the definite articles and the demonstratives:

(69)	a.	*	(Todos)	os	determinados	homens	leram	um	livro.
			all	the	certain	men	have read	a	book

	b.	*	Esses	determinados	homens	leram	um	livro.
			those	certain	men	have read	a	book

The scope ordering is thus the following: *widest* > *wide* > *narrow*.

Consider the following example:

(70)	Todos	os	filhos	da	Ana	leram	um	certo	livro.
	all	the	children	of the	Ana	have read	a	certain	book

All of Ana's children have read a certain book.

These constraints license only one quantifier scope possibility:

$$proper\_q(x8, named(x, \text{“}Ana''), \_um\_q(x16, \_livro\_n(x16), \_todo\_q(x4, \_filho\_n\_-de-(x4, x8), \_ler\_v(e2, x4, x16))))$$

The *proper_q_rel* relation cannot be embedded under any of the other quantifier relations, because its LBL has SCOPE of the type *widest*, but the handle arguments of the other quantifier relations have the value *non-widest* or *narrow* for their SCOPE features, and any of these types is incompatible with the type *widest*. The *_um_q_rel* relation in this example also cannot be under the scope of the *_todo_q_rel*, like in the previous example.

4.11 Cardinals and Markers of Indefinite Specifics as Determiners

As was mentioned in Section 4.10, cardinals and items like *certo* and *determinado* (*certain*) that mark indefinite specific NPs can themselves introduce an NP. Some examples from (58) are repeated below in (71).

(71)	a.	dois	certos	capítulos
		two	certain	chapters

two certain chapters

	b.	certos	dois	capítulos
		certain	two	chapters

two certain chapters

Ordinals cannot occur in NP initial position, though:

(72)	a.	um	DVD	com	dois	primeiros	episódios	dessa	série
		a	DVD	with	two	first	episodes	of that	series

a DVD with two first episodes of that series

	b.	*	um	DVD	com	primeiros	dois	episódios	dessa	série
			a	DVD	with	first	two	episodes	of that	series

When these elements are preceded by a determiner, it is the determiner that introduces quantifier semantics. Assuming that quantifier semantics is always introduced by a determiner or a bare NP construction, there are two ways of introducing quantifiers in these NPs: considering them instances of bare NPs or analyzing the first element as a determiner.

The first possibility is not very attractive for Portuguese, for a number of factors. Preverbal bare NP subjects have a very constrained distribution in European Portuguese. It is interesting to note that NPs introduced by a cardinal (73c) do not pattern with bare NPs (73a) in the following examples, but rather with NPs introduced by determiners (73b).

(73) a. */?? Cartas chegaram.
letters have arrived

b. Algumas cartas chegaram.
some letters arrived
Some letters arrived.

c. Duas cartas chegaram.
two letters arrived
Two letters arrived.

The example in (73c) sounds as good as the one in (73b), which is introduced by the item *alguns*, which can only occur in NP initial position and is thus not in a bare NP.

Second, bare NPs tend to have non-specific readings in Portuguese: they cannot scope over negation (74), universal quantifiers (75) or intensional verbs (76). These examples are Brazilian Portuguese, from (Munn and Schmitt, 1998) (we took the liberty of adding the corresponding logical formulas, for ease of exposition), but the same observations hold for European Portuguese. We also bracketed the relevant bare NPs in these examples.

(74) a. João não viu uma mancha no chão.
João not saw a spot on the floor
João didn't see a spot on the floor.

1. $\neg\exists x[spot\_on\_the\_floor(x) \wedge saw(João, x)]$
2. $\exists x[spot\_on\_the\_floor(x) \wedge \neg saw(João, x)]$

b. João não viu [manchas no chão.]
João not saw spots on the floor
João didn't see spots on the floor.

1. $\neg\exists x[spot\_on\_the\_floor(x) \wedge saw(João, x)]$

(75) a. Todo mundo leu um livro sobre girafas.
everyone read a book on giraffes
Everyone read a book on giraffes.

1. $\forall x[person(x) \rightarrow \exists y[book\_on\_giraffes(y) \wedge read(x, y)]]$
2. $\exists y[book\_on\_giraffes(y) \wedge \forall x[person(x) \rightarrow read(x, y)]]$

b. Todo mundo leu [livros sobre girafas.]
everyone read books on giraffes
Everyone read books on giraffes.

1. $\forall x[person(x) \rightarrow \exists y[book\_on\_giraffes(y) \wedge read(x, y)]]$

(76) a. Pedro quer encontrar um policial.
Pedro wants to meet a policeman
Pedro wants to meet a policeman.

1. $\exists x[policeman(x) \wedge want(Pedro, meet(Pedro, x))]$
2. $want(Pedro, \exists x[policeman(x) \wedge meet(Pedro, x)])$

b. Pedro quer encontrar [policiais.]
Pedro wants to meet policemen
Pedro wants to meet policemen.

1. $want(Pedro, \exists x[policeman(x) \wedge meet(Pedro, x)])$

NPs introduced by cardinals do not pattern with bare NPs in this respect and allow both readings. An example with negation is in (77), in which the semantics of the cardinal *duas/two* is represented by $\lambda P.\lambda Q.\exists x_1 \exists x_2[x_1 \neq x_2 \wedge P(x_1) \wedge P(x_2) \wedge Q(x_1) \wedge Q(x_2)]$. Ambiguity can be found in the other two contexts as well.

(77) João não viu duas manchas no chão.
João not saw two spots on the floor
João didn't see two spots on the floor.

1. $\neg\exists x_1 \exists x_2[x_1 \neq x_2 \wedge spot\_on\_the\_floor(x_1) \wedge spot\_on\_the\_floor(x_2)$
$\wedge saw(João, x_1) \wedge saw(João, x_2)]$
2. $\exists x_1 \exists x_2[x_1 \neq x_2 \wedge spot\_on\_the\_floor(x_1) \wedge spot\_on\_the\_floor(x_2)$
$\wedge \neg saw(João, x_1) \wedge \neg saw(João, x_2)]$

Furthermore, NPs introduced by markers of indefinite specifics should obviously not be analyzed as bare NPs if the latter are constrained to take non-specific readings:

(78) a. O João não viu certa mancha no chão.
the João not saw certain spot on the floor
João didn't see a certain spot on the floor.

1. $\exists x[spot\_on\_the\_floor(x) \wedge \neg saw(João, x)]$

b. Todas as pessoas leram certo livro sobre girafas.
all the people read certain book on giraffes
Everyone read a certain book on giraffes.

1. $\exists y[book\_on\_giraffes(y) \wedge \forall x[person(x) \rightarrow read(x,y)]]$

c. Pedro quer encontrar certo polícia.
Pedro wants to meet certain policeman
Pedro wants to meet a certain policeman.

1. $\exists x[policeman(x) \wedge want(Pedro, meet(Pedro, x))]$

Bare NPs do not co-occur with the *cada* (*each*) of (79), but NPs introduced by cardinals do (examples from (Müller, 2002)):

(79) a. Os países da UE mandaram um/dois/vários delegado(s) cada.
the countries of the EU sent a/two/various delegate(s) each
The EU countries sent a/two/various delegate(s) each.

b. * Os países da UE mandaram delegados cada.
the countries of the EU sent delegates each

We conclude that cardinals and markers of indefinite specifics at NP initial position are best treated as determiners. They introduce indefinite NPs and cannot co-occur with prenominal possessives. The constraints on the *marking* features must therefore be different from the ones on elements of Position IV. Therefore, the constraints on their HEAD must differ. NP initial cardinals also carry quantifier semantics, which the elements of Position IV arguably do not. We will be calling them cardinal determiners and indefinite specific determiners from now on.

The HEAD of an NP initial cardinal looks like this:

$$\left[\begin{array}{ll} \textit{cardinal-det} & \\ \text{MARKER} & \left[\begin{array}{ll} \textit{pre-only-marker-min} & \\ \text{MARK} & \left[\begin{array}{ll} \textit{saturated} & \\ \text{MK-VAL} & \left[\begin{array}{ll} \text{CARDINAL} & \textit{present} \\ \text{ORDINAL} & \boxed{1} \\ \text{INDEF-SPEC} & \boxed{2} \end{array}\right] \end{array}\right] \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \left[\begin{array}{ll} \text{HEAD } \textit{noun} & \\ \text{MARKING} & \left[\begin{array}{ll} \textit{no-poss-marking} & \\ \text{MK-VAL} & \left[\begin{array}{ll} \text{CARDINAL} & \textit{absent} \\ \text{ORDINAL} & \boxed{1} \\ \text{INDEF-SPEC} & \boxed{2} \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

The interesting point about this definition is that cardinal determiners actually saturate two slots: the determiner position (Position II) and the prenominal possessive position (Position III). This would not be possible to achieve if saturation were encoded via valence features, which must be discharged one at a time, unless a dedicated syntactic rule was used. The constraints using the different values of *marking* prevent cardinal determiners from iterating, and the constraint on MK-VAL|CARDINAL of the selected synsem prevents cardinal determiners from combining with the cardinal elements of Position IV. The remaining features under MK-VAL do not need to be constrained in the way just shown, since the values of *marking* already guarantee that nothing can combine to the left of a cardinal determiner, but we do so in order to ensure that every instance of them respects their semantics.

The constraints for indefinite specific determiners are similar, with the obvious differences regarding the features CARDINAL and INDEF-SPEC.

There are no determiner versions of ordinals, as they cannot initiate an NP.

There are two questions to address: the relation between these determiners and the items of Position IV, and preventing bare NPs from being formed from NPs starting with an element in Position IV.

There are two possibilities for the first issue: to produce the determiner version from the post-determiner one via a unary rule, or to have multiple lexical entries. In LXGram indefinite specifics receive multiple lexical entries, but cardinals do not. This is for reasons related to the composition of semantics and is explained in Section 4.11.1.

We assume that bare NPs are produced by a unary syntactic rule that adds quantifier semantics, imposes a value of marking on the mother node subsumed by *saturated-marking* and requires the daughter to be a noun headed sign with a MARKING subsumed by *no-det-marking*. In order to prevent bare NPs to be built from constituents that include a postdeterminer cardinal, ordinal or marker of

$$\begin{bmatrix} \textit{mrs} & \\ \text{LTOP} & \boxed{h1}\ h \\ \text{INDEX} & \boxed{e2}\ e \\ \text{RELS} & \left\langle \begin{array}{l} \begin{bmatrix} \textit{\_o\_q\_rel} & \\ \text{LBL} & \boxed{h3}\ h \\ \text{ARG0} & \boxed{x4}\ x \\ \text{RSTR} & \boxed{h6}\ h \\ \text{BODY} & \boxed{h5}\ h \end{bmatrix}, \begin{bmatrix} \textit{cardinal\_rel} & \\ \text{LBL} & \boxed{h7}\ h \\ \text{ARG0} & \boxed{e9}\ e \\ \text{ARG1} & \boxed{h8}\ h \\ \text{ARG2} & \boxed{j10}\ j \end{bmatrix}, \begin{bmatrix} \textit{greater-or-equal\_rel} & \\ \text{LBL} & \boxed{h7} \\ \text{ARG0} & \boxed{j10} \\ \text{ARG1} & \boxed{j11}\ j \end{bmatrix}, \\ \begin{bmatrix} \textit{int-equals\_rel} & \\ \text{LBL} & \boxed{h7} \\ \text{ARG0} & \boxed{j11} \\ \text{CARG} & \textit{2} \end{bmatrix}, \begin{bmatrix} \textit{\_carro\_n\_rel} & \\ \text{LBL} & \boxed{h8} \\ \text{ARG0} & \boxed{x4} \end{bmatrix}, \begin{bmatrix} \textit{\_avariar\_v\_rel} & \\ \text{LBL} & \boxed{h12} \\ \text{ARG0} & \boxed{e2} \\ \text{ARG1} & \boxed{x4} \end{bmatrix} \end{array} \right\rangle \\ \text{HCONS} & \left\langle \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h1} \\ \text{LARG} & \boxed{h12} \end{bmatrix}, \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h6} \\ \text{LARG} & \boxed{h7} \end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 4.17: MRS of a sentence with a postdeterminer cardinal. The sentence is *os dois carros avariaram* (*the two cars broke down*).

indefinite specifics, the daughter is also constrained to have the features ORDINAL, CARDINAL and INDEF-SPEC under MARKING|MK-VAL of type *absent*.

4.11.1 Cardinal Determiners and the Semantics of Cardinals

In LXGram we chose to relate cardinal determiners and cardinal postdeterminers via unary syntactic rules. In particular, the determiner versions are produced from the postdeterminer versions. This is tied to issues of composition of semantics.

For the postdeterminer cardinals, an example of the MRSs produced is in Figure 4.17. The cardinal corresponds to three relations in this MRS: the *cardinal_rel* relation, the *greater-or-equal_rel* relation, and the *int-equals_rel* relation.

In the literature, there are several approaches to the semantics of cardinals: they have been given the semantic types $\langle e,t\rangle$ (a set of entities),[8] $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ (a function from sets to sets) or $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$ (a determiner). We did not choose to give cardinals quantifier semantics, because they can occur after determiners, as in expressions like *all three*. When they do appear in NP initial position, quantifier semantics must be added, though. We opted for the $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ treatment (i.e. consider them modifiers), and do not commit to saying that cardinals are intersective modifiers.[9] Therefore the *cardinal_rel* scopes over the relation introduced by the head noun in MRSs. This is compatible with intersective semantics but does not enforce it. Our representation for cardinals is similar to $\lambda i_{\in I}.\lambda P_{\in D_{\langle e,t\rangle}}.\lambda x_{\in D_e}.cardinal(e,P(x),i)$, where the integer argument i is supplied in each lexical entry

[8]Or rather its characteristic function, a function from entities to truth values yielding true for all members of that set and for them only, i.e. a function of the form $\lambda x_{\in D_e}.P(x)$.

[9]We do not commit to saying that this function is $\lambda P_{\in D_{\langle e,t\rangle}}.\lambda x_{\in D_e}.P(x) \wedge Q(x)$, for some lexically given set Q. For example, if the denotation of *car* is $\lambda x_{\in D_e}.car'(x)$, the denotation of *two cars* would be $(\lambda P_{\in D_{\langle e,t\rangle}}.\lambda x_{\in D_e}.P(x) \wedge 2(x))(\lambda x_{\in D_e} car'(x)) = \lambda x_{\in D_e}.car'(x) \wedge 2(x)$ if intersective semantics is given to cardinals. This only makes sense if we consider the existence of plural (non-atomic) entities, whose atoms can be counted, in which case the set denoted by 2 in the above formula is the set of all plural entities with two atoms. There are many views on the semantics of cardinals, and we remain neutral with respect to the status of plural entities. In (Ionin and Matushansky, 2006) there are references to the main pieces of work in this field.

for cardinals. However, we do not define the meaning of the *cardinal* relation. It can be intersective if we posit that $\lambda i.\lambda P.\lambda x.cardinal(e,P(x),i) = \lambda i.\lambda P.\lambda x.P(x) \wedge count(P,i)$, where $\lambda i.\lambda x.count(x,i)$ is true if x is a plural entity with i atoms. Its meaning can however be defined differently, not necessarily in an intersective way.

Note that a definition that constrains the cardinality of the set denoted by the noun does not work. For instance a sentence like *three cars broke down* does not mean that the cardinality of the set of cars is three, but rather that the cardinality of the intersection of the set of cars and the set of things that broke down is three. Using plural entities, this sentence would be assigned a representation that says that there is a plural entity consisting of three cars that also belongs to the set of things that broke down, i.e. it is simple existential quantification, which is the semantics we will assume for cardinals occupying a determiner position (see below).

The other relations describe the integer argument of the *cardinal_rel* relation. It is widely assumed that an expression like *two children* means *at least two children* and not *exactly two children*. In the following example, the answer would be contradictory if *two children* meant *exactly two children*:

(80) — Do you have two children?
— Yes. In fact I have three.

The entire relevant piece of semantics is $cardinal(e9,\_carro\_n(x4),j10) \wedge greater\text{-}or\text{-}equal(j10,j11) \wedge int\text{-}equals(j11,2)$. Variables of the form jn, where $n > 0$ are supposed to be of type integer.[10] We can view these integer variables as existentially quantified by convention, so we do not explicitly include these quantifiers in the MRSs.

It would be more simple to produce $cardinal(e9,\_carro\_n(x4),2)$, assuming that the relation greater or equal is part of the meaning of *cardinal_rel*.

The motivation for introducing the *greater-or-equal_rel* relation is that in certain contexts we do not want it to appear in the MRSs. This is the case of expressions like *exactly two* or *at most two*. For an expression like *no máximo dois/at most two*, we can think of the semantics $\lambda P.\lambda x.cardinal(e,P(x4),j1) \wedge less\text{-}or\text{-}equal(j1,j2) \wedge int\text{-}equals\_rel(j2,2)$. In order to factor out the similarity with the representation for an unmodified *dois/two*, we explicitly introduce *greater-or-equal_rel* relation when a cardinal is not modified.[11]

The use of the *int-equals_rel* relation is a matter of convenience. It is not necessary, because, instead of the piece of semantics $cardinal(e9,\_carro\_n(x4),j10) \wedge greater\text{-}or\text{-}equal(j10,j11) \wedge int\text{-}equals(j11,2)$, we could simply use $cardinal(e9,\_carro\_n(x4),j10) \wedge greater\text{-}or\text{-}equal(j10,2)$.

It is more convenient for the generation algorithm in the LKB to associate at least one relation with every lexical item. If we did not include this relation in the lexical entry for cardinals, their only semantic content would be the integer constant that is an argument of relations like *greater-or-equal_rel* or *less-or-equal_rel*.

The implementation in LXGram associates to lexical items for cardinals only the *int-equals_rel* relations. All other relations are introduced in syntax, via rules that add semantics.

The first set of rules allows these expressions to combine with cardinal modifiers like *exactly*, *at most*, etc. Only one modifier is allowed, and if no modifier is present, a unary syntactic rule is used to add the *greater-or-equal_rel*. A cardinal modifier like *at most* introduces a *less-or-equal_rel*, a modifier like *exactly* introduces no relation.

[10]They can be created by manipulating the LKB configuration files, namely by redefining function **determine-variable-type** in the file mrsglobals.lisp.

[11]If we assumed that the *greater-or-equal_rel* relation is part of the meaning of *cardinal_rel* so that we would not have to include it in MRSs when a cardinal is not modified, expressions like *at most two* would not receive the correct semantics, or we could not use the *cardinal_rel* in these cases.

Immediately up the tree, a unary rule is used to add the *cardinal_rel* relation and producing a node with a HEAD of type *cardinal*. After this step the piece of semantics for *dois* (*two*) and for *pelo menos dois* (*at least two*) is like $\lambda P.\lambda x.cardinal(e,P(x),j_1) \wedge greater\text{-}or\text{-}equal(j_1,j_2) \wedge int\text{-}equals(j_2,2)$. The semantics for *no máximo dois* (*at most two*) is like $\lambda P.\lambda x.cardinal(e,P(x),j_1) \wedge less\text{-}or\text{-}equal(j_1,j_2) \wedge int\text{-}equals(j_2,2)$. The semantics for *exactamente dois* (*exactly two*) is like $\lambda P.\lambda x.cardinal(e,P(x),j_1) \wedge int\text{-}equals(j_1,2)$.

An optional rule can apply afterwards, changing the postdeterminer cardinal into a determiner (*cardinal-det* above) and adding quantifier semantics. So cardinal determiners are produced from cardinal postdeterminers via a syntactic rule.

We will not show the details of all these rules since they are relatively trivial. To control order of rule application LXGram uses different values of HEAD for these elements: many of these rules are non-headed. Only the two highest rules create nodes with values of *head* that inherit from *functor* and that can attach to nominal projections. These subtypes of *head* and their definitions (*cardinal* and *cardinal-det*) have already been presented.

This analysis is completely monotonic: we only add relations to an MRS, never remove or alter relations introduced elsewhere. This is a requirement of the LKB: composition of semantics has to be monotonic so that efficient algorithms can be used for generation. Also, every lexical entry for cardinals and every rule used in this process contributes at least one relation to the MRS.

Although a large number of dedicated rules is involved, they are used to build the semantics little by little and factor out the commonalities between the various pieces of MRS that are related to cardinals.

We assume that complex cardinal expressions like *vinte e um*/*twenty one* are recognized by a Named Entity Recognizer (NER) in a preprocessing step, and for the purposes of the grammar behave just like atomic cardinals like *vinte*/*twenty*. There is one NER developed in the University of Lisbon (Ferreira *et al.*, 2007), that can be integrated with LXGram. Since NERs are not necessarily bidirectional, we can parse these expressions but we cannot generate them, though.

4.12 PPs and AdvPs

PPs and some AdvPs can modify a noun on their left. Some examples are given in (81).

(81) a. pessoas [$_{PP}$ com mobilidade reduzida]
people with mobility reduced
people with reduced mobility

b. [$_{NP}$ Aquele carro ali$_{AdvP}$] [$_{VP}$ estava estacionado aqui ontem.]
that car there was parked here yesterday
That car over there was parked here yesterday.

The adverb *ali* (*there*) in (81b) cannot be analyzed as modifying the VP to its right, as a situation cannot happen simultaneously "here" and "there".

PPs and AdvPs cannot precede the noun, as shown in (82).

(82) a. * [$_{PP}$ com mobilidade reduzida] pessoas
with mobility reduced people

b. * Aquele ali$_{AdvP}$ carro estava estacionado aqui ontem.
that there car was parked here yesterday

Inside the NP, they have the syntactic distribution of postnominal adjectives (Position VIII in the table in Appendix A). In fact, because PPs and APs can be interspersed, ambiguity can arise concerning adjective attachment — consider (83).

(83)

carros	sem	assentos	vermelhos
cars	without	seats	red

red cars with no seats/cars without red seats

This example has two interpretations depending on the attachment site of the adjective: [[carros [$_{PP}$ sem assentos]] vermelhos$_{AP}$] (*red cars with no seats*) and [carros [$_{PP}$ sem assentos vermelhos]] (*cars without red seats*). This fact leads one to posit constraints on the head types of prepositions and adverbs similar to the ones on postnominal adjectives, as far as the feature POSTHEAD is concerned.

The constraints on the head of prepositions and adverbs that can modify nouns (as well as verbs) thus look like:

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{preposition-or-adnominal-adverb}} \\ \text{MARKER} & \left[\begin{array}{ll} \text{MARK} & \left[\begin{array}{l}\textit{basic-marking}\\ \text{MK-VAL} \ \boxed{1}\end{array}\right] \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \left[\begin{array}{ll} \text{HEAD} & \textit{noun-or-verb} \\ \text{MARKING} & \left[\begin{array}{l}\textit{basic-marking}\\ \text{MK-VAL} \ \boxed{1}\end{array}\right]\end{array}\right] \\ \multicolumn{2}{l}{\text{PREHEAD}|\text{SELECT}|\text{LOCAL}|\text{CAT}|\text{HEAD} \ \textit{verb}}\end{array}\right]\end{array}\right]$$

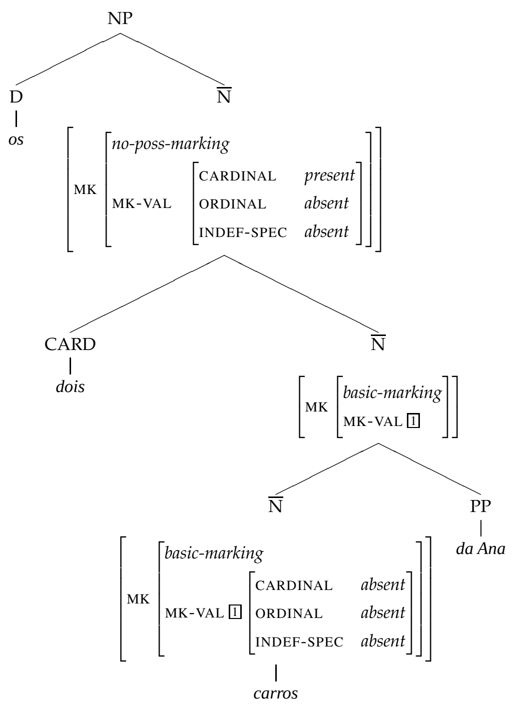
With the setup presented so far, the grammar overgenerates. Consider the following NP:

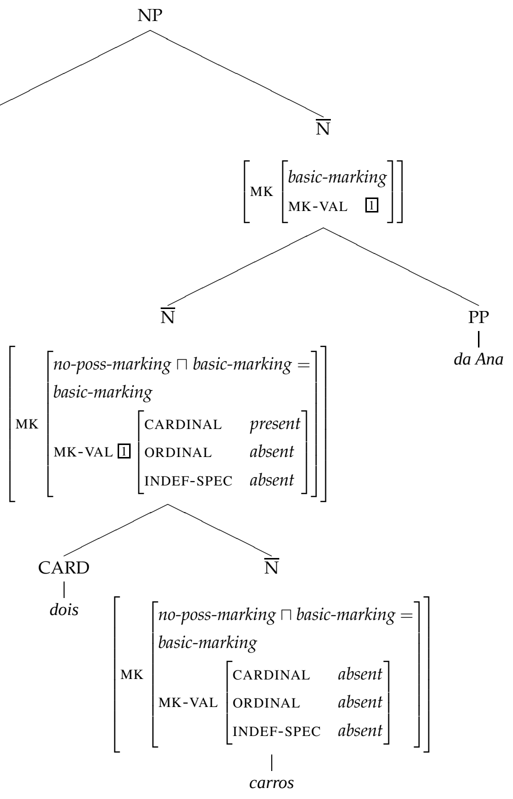
(84)

Os	dois	carros	da	Ana
the	two	cars	of the	Ana

Ana's two cars

There are two parses for this NP (with abridged feature paths, and MK standing for MARKING):





The parse tree on the right is allowed by the system of constraints presented so far, because the types *basic-marking* and *no-poss-marking* are compatible.

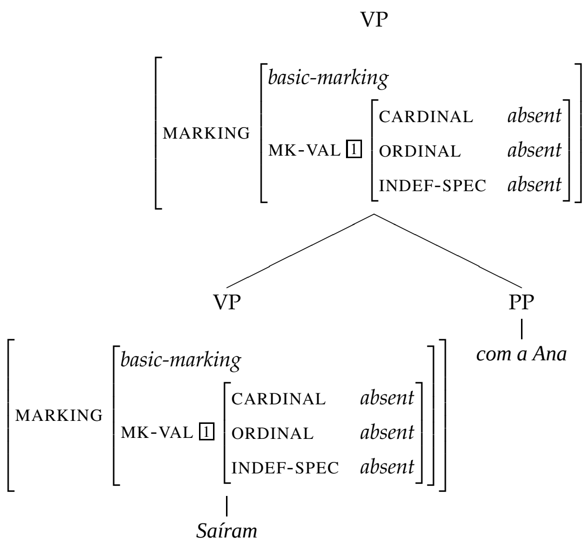
The parse tree on the left is preferred, because the syntactic scope among the various elements corresponds to their semantic scope: we want to restrict the cardinality of the set that is the intersection of the set of cars with the set of objects owned by Ana.

A way to block the parse on the right is to constrain prepositions to select for constituents with the value *absent* for the feature CARDINAL. The same problem arises with ordinals and markers of indefinite specific NPs, and the features ORDINAL and INDEF-SPEC should be similarly constrained. A preliminary solution to this problem involves refining the head type for prepositions and adverbs that can attach to nominal projections:

$$\begin{bmatrix} \textit{preposition-or-adnominal-adverb} & \\ \text{MARKER} & \begin{bmatrix} \text{MARK} & \begin{bmatrix}\textit{basic-marking} & \\ \text{MK-VAL} & \boxed{1}\end{bmatrix} \\ \text{SELECT}\mid\text{LOCAL}\mid\text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun-or-verb}\\ \text{MARKING} & \begin{bmatrix} \textit{basic-marking} & \\ \text{MK-VAL} & \boxed{1}\begin{bmatrix}\text{CARDINAL} & \textit{absent}\\ \text{ORDINAL} & \textit{absent}\\ \text{INDEF-SPEC} & \textit{absent}\end{bmatrix}\end{bmatrix}\end{bmatrix}\\ \multicolumn{2}{l}{\text{PREHEAD}\mid\text{SELECT}\mid\text{LOCAL}\mid\text{CAT}\mid\text{HEAD}\ \textit{verb}}\end{bmatrix}\end{bmatrix}$$

However, if we state these constraints directly in the head type for prepositions and adverbs that can attach to nominal projections, these features under MK-VAL will be present in the feature structures also when they attach to verbal projections. Consider the VP in (85) and its syntactic analysis below:

(85)	Saíram	com	a	Ana.
	they left	with	the	Ana
	They left with Ana.			



We can assume that verbs come in the lexicon also with MARKING of the type *basic-marking*, but MK-VAL of the type *mk-val-min* (see Section 4.10), so that the features CARDINAL, ORDINAL and INDEF-SPEC are not present. However, when a PP attaches to a verb, the constraints under the SELECT|LOCAL

|CAT|MARKING feature of the PP unify with the constraints under the SYNSEM|LOCAL|CAT|MARKING of the verb and these features are present:

$$\left[\begin{array}{ll}\textit{basic-marking} & \\ \text{MK-VAL} & \textit{mk-val-min}\end{array}\right] \sqcap \left[\begin{array}{ll}\textit{basic-marking} & \\ \text{MK-VAL} & \left[\begin{array}{ll}\textit{mk-val} & \\ \text{CARDINAL} & \textit{absent} \\ \text{ORDINAL} & \textit{absent} \\ \text{INDEF-SPEC} & \textit{absent}\end{array}\right]\end{array}\right] = \left[\begin{array}{ll}\textit{basic-marking} & \\ \text{MK-VAL} & \left[\begin{array}{ll}\textit{mk-val} & \\ \text{CARDINAL} & \textit{absent} \\ \text{ORDINAL} & \textit{absent} \\ \text{INDEF-SPEC} & \textit{absent}\end{array}\right]\end{array}\right]$$

This situation has no consequence for correctness. However, the presence of these features CARDINAL, ORDINAL and INDEF-SPEC in the feature structures for verb headed constituents are uninformative and redundant, since verbal projections cannot combine with any of these elements.

We can elaborate the type hierarchy under *marking* with the purpose of eliminating these features in verb headed elements, but still ensuring that they are present in noun headed items (as they are necessary to constrain the attachment possibilities of cardinals and the other elements in Position IV). A revised version of the hierarchy under *marking* is in Figure 4.18.

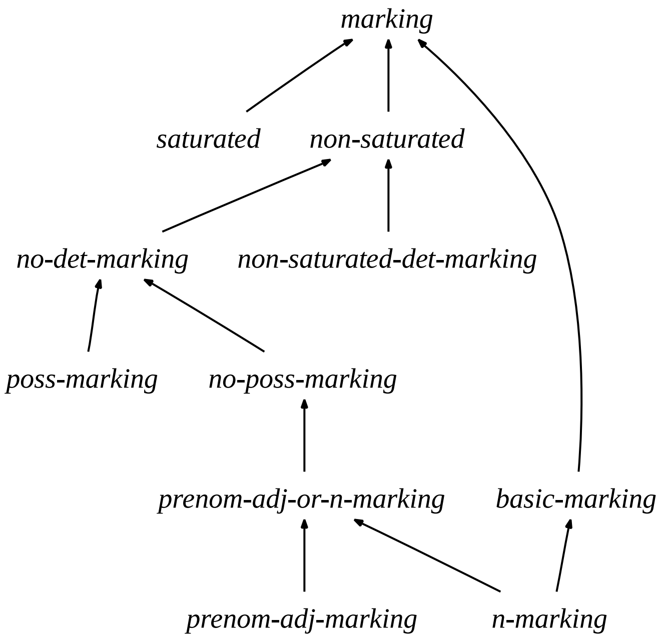


Figure 4.18: Type hierarchy under *marking* (version 5/6). Previous version on p. 91. Final version on p. 118.

In this hierarchy, the former type *prenom-adj-or-basic-marking* has been renamed to *prenom-adj-or-n-marking*. Syntactic constituents with a MARKING value subsumed by *prenom-adj-or-n-marking* can never contain a cardinal, an ordinal or a marker of indefinite specific NPs, since we are assuming that the elements in Position IV are more peripheral than any of the elements in Position V, Position VI, Position VII and Position VIII. Therefore, the type *prenom-adj-or-n-marking* can be constrained in the following way:

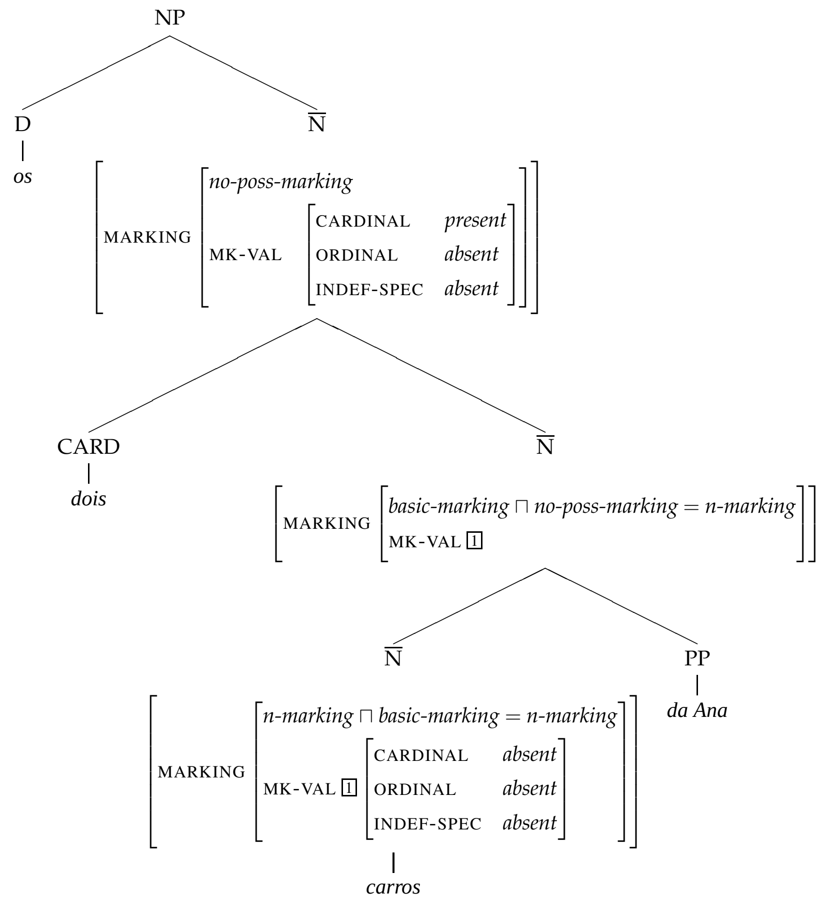
$$\left[\begin{array}{ll}\textit{prenom-adj-or-n-marking} & \\ \text{MK-VAL} & \left[\begin{array}{ll}\text{CARDINAL} & \textit{absent} \\ \text{ORDINAL} & \textit{absent} \\ \text{INDEF-SPEC} & \textit{absent}\end{array}\right]\end{array}\right]$$

These constraints are inherited by the new type *n-marking*. Nouns now come in the lexicon with the value *n-marking* for their feature MARKING.

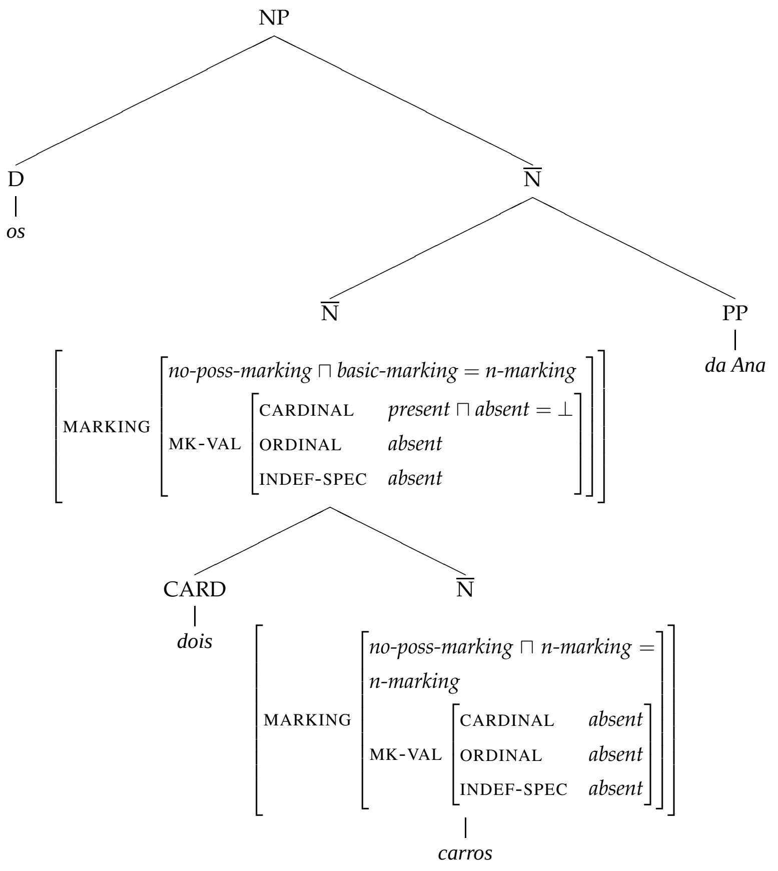
We can keep the original definition of *preposition-or-adnominal-adverb*:

$$\begin{bmatrix} \textit{preposition-or-adnominal-adverb} & \\ \text{MARKER} & \begin{bmatrix} \text{MARK} & \begin{bmatrix}\textit{basic-marking}\\ \text{MK-VAL}\ \boxed{1}\end{bmatrix} \\ \text{SELECT|LOCAL|CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun-or-verb} \\ \text{MARKING} & \begin{bmatrix}\textit{basic-marking}\\ \text{MK-VAL}\ \boxed{1}\end{bmatrix}\end{bmatrix} \\ \text{PREHEAD|SELECT|LOCAL|CAT|HEAD}\ \textit{verb} & \end{bmatrix}\end{bmatrix}$$

The analysis for the NP in (84) is now:

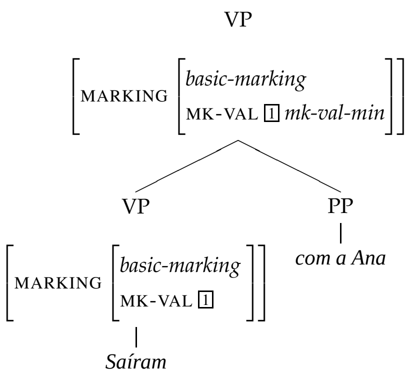

NP
D
os
N̄
MARKING [no-poss-marking, MK-VAL [CARDINAL present, ORDINAL absent, INDEF-SPEC absent]]
CARD
dois
N̄
MARKING [basic-marking ⊓ no-poss-marking = n-marking, MK-VAL 1]
N̄
PP
da Ana
MARKING [n-marking ⊓ basic-marking = n-marking, MK-VAL 1 [CARDINAL absent, ORDINAL absent, INDEF-SPEC absent]]
carros

The alternative analysis is blocked, as desired:



The sister node of the PP is constrained to be of the type *basic-marking* by the PP and to be of the type *no-poss-marking* by the cardinal. These two types unify to *n-marking*, which is constrained to bear the value *absent* under its MK-VAL|CARDINAL feature. However, the cardinal also constrains this feature to take the value *present*. Therefore, a unification failure is derived.

The types subsumed by *no-det-marking* are not used to constrain the MARKING of verbal projections, because the items where they are employed only attach to nouns. This means that when a PP attaches to a verbal constituent, the MARKING value of that constituent remains *basic-marking*. The result is never *n-marking*. This way, the features CARDINAL, ORDINAL and INDEF-SPEC are not present in verb headed elements. Consider the current analysis for the VP presented above in (85):



The remaining NP elements that have constraints on MARKING employing the type *basic-marking* can remain unchanged, since this value will end up being unified with a type subsumed by *no-det-marking* in every NP, producing the desired value *n-marking*.

The general type *basic-marking* helps in hiding the implementation of NP structure. The information about the exact position within the NP where PPs and AdvPs attach — namely the constraints on the absence of the items in Position IV — is encapsulated in a more specific type, *n-marking*. This more specific type is kept separate from the definitions of prepositions and adverbs.

4.13 Relative Clauses

We assume that relative clauses are not headed by a verb, and use a dedicated head type for them (*relative-comp*), i.e. head-filler constructions for relative clauses are assumed to be non-headed structures. This is compatible with the LinGO Grammar Matrix. We abstain from developing on the analysis of relative clauses here, as it would clearly lead us to far afield.

We add a new type to the hierarchy under *marking*, *rel-marking*, in order to model relative clause attachment. The resulting hierarchy is in Figure 4.19. This is the final version of the type hierarchy under *marking*.

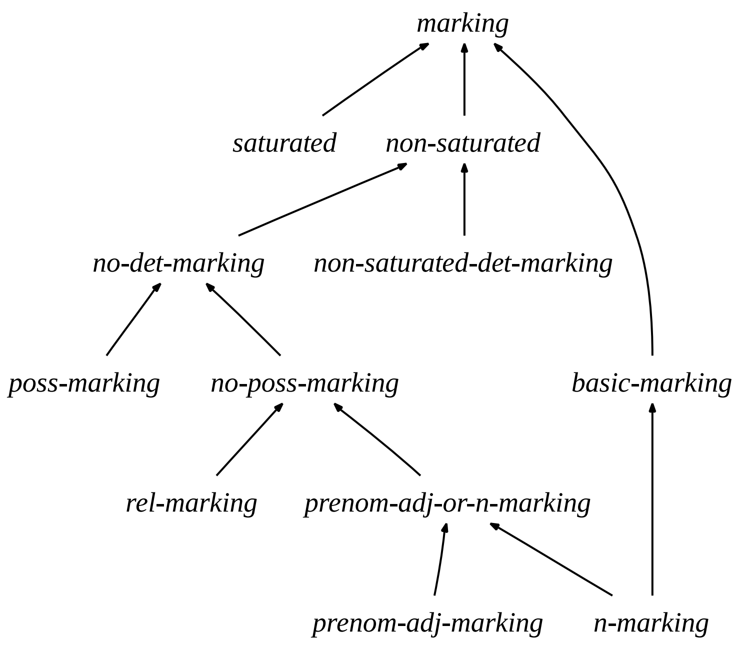


Figure 4.19: Type hierarchy under *marking* (final version — 6/6). Previous version on p. 115.

Restrictive relative clauses should be allowed to iterate, but they are more peripheral than APs and PPs inside an NP, and they always follow the noun:

$$\begin{bmatrix} \textit{relative-comp} & \\ \text{MARKER} & \begin{bmatrix} \textit{post-only-marker-min} & \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{MARKING} & \begin{bmatrix} \textit{no-poss-marking} \\ \text{MK-VAL } \boxed{1} \end{bmatrix} \end{bmatrix} \\ \text{MARK} & \begin{bmatrix} \textit{rel-marking} \\ \text{MK-VAL } \boxed{1} \end{bmatrix} & \end{bmatrix} \end{bmatrix}$$

The type hierarchy under *marking* and the constraints presented so far mean that restrictive relative clauses outscope prenominal adjectives. Semantically, this is borne out by the data. Consider the NP in (86) below. It describes an entity as being in fact Chinese. That is, the piece of semantics for that NP will be equivalent to $\lambda P.|((D_e - D) \cap C) \cap P| > 0$ (where D_e is the model's domain, D the set of doctors and C the set of Chinese entities, giving *false* the semantics $\lambda Q.D_e - Q$), with no mismatch between syntactic and semantic scope. It will not be $\lambda P.|(D_e - (D \cap C)) \cap P| > 0$, the semantics for the example in (34) (*um falso médico chinês — a false Chinese doctor*).

(86)

um	falso	médico	que	é	chinês
a	false	doctor	who	is	Chinese

a false doctor that is Chinese

Semantically, restrictive relative clauses are under the scope of cardinals. Consider the sentence in (87). If M is the set of movies, BM is the set of bad movies and S is the set of entities that "I saw there", the meaning of (87) is $M \cap S \subseteq BM \wedge |M \cap S| = 3$, not $M \cap S \subseteq BM \wedge |M| = 3$. In particular, any model with $M = \{m_1, m_2, m_3, m_4\}$, and $BM = S = \{m_1, m_2, m_3\}$ makes the first interpretation true and the second one false, and (87) is true in such models.

(87)

Todos	os	exactamente	três	filmes	que	lá	vi	eram	maus.
all	the	exactly	three	movies	that	there	I saw	were	bad

All exactly three movies I saw there were bad.

Continuing to espouse the assumption that, if there is no reason to assume the contrary, syntactic scope matches semantic scope, we therefore want relative clauses to attach lower than cardinals. Similar data can be envisaged for ordinals, but we will not present them for the sake of brevity. Semantic considerations cannot help us determine the relative scope between relative clauses and markers of indefinite specifics, because the latter contribute no relations to the resulting semantics. Since cardinals, ordinals and markers of indefinite specifics all occupy the same NP slot, we assume that relative clauses attach lower than all these elements. To force this attachment, we can simply add the following constraints to the type *rel-marking*:

$$\begin{bmatrix} \textit{rel-marking} & \\ \textsc{mk-val} & \begin{bmatrix} \textsc{cardinal} & \textit{absent} \\ \textsc{ordinal} & \textit{absent} \\ \textsc{indef-spec} & \textit{absent} \end{bmatrix} \end{bmatrix}$$

Recall that the feature MK-VAL is unified between MARKER|MARK and MARKER|SELECT|LOCAL|CAT |MARKING under the HEAD attribute of relative clauses, so the constraints just presented on *rel-marking* effectively make relative clauses select for constituents that lack all of these elements.

It is worth mentioning that there are data that this analysis does not contemplate:

(88)

a	sugestão$_N$	[$_{RelCl}$ que foi	mencionada]	[$_{COMP}$ de que	seria	impossível	proceder
the	suggestion	that was	mentioned	that	it would be	impossible	to act
de outro modo]							
in a different way							

the suggestion that was mentioned that it would be impossible to act in a different way

In (88) there is a relative clause, bracketed with RelCl, intervening between the head noun and its sentential complement, bracketed with COMP, for complement. However, according to our system of constraints, relative clauses must attach to projections with already saturated complements. This is because Head-Complement constructions constrain the head daughter to have MARKING of type *basic-marking*, as presented in Section 4.8. This constraint was necessary in order to force PP complements to precede relative clauses, as explained in that section.

We believe that this sort of situation only arises in specific cases (sentential complements) and that they should receive a special treatment, which we do not develop here.

Note that it is not possible to simply create a common subtype of *rel-marking* and *basic-marking* in order to overcome this limitation, because this would allow prenominal adjectives to attach higher than relative clauses, which we want to block (prenominal adjectives select for sister nodes with MARKING of type *prenom-adj-or-basic-marking*; the new type would provide a unifier for *rel-marking* and *prenom-adj-or-basic-marking*).

4.14 Postnominal Demonstratives and Possessives

In Position VIII we can find other elements besides adverbial PPs, AdvPs and APs and complements, which have been covered. These other elements are postnominal demonstratives, possessives and universal quantifiers:

(89) a.

A	bicicleta	essa	está	estragada.
the	bicycle	that	is	broken

That bicycle is broken.

b.

Chegaram	várias	cartas	tuas.
arrived	several	letters	yours

There arrived several letters of yours.

c.

Desapareceram	as	cartas	todas.
disappeared	the	letters	all

All the letters disappeared.

We will not address postnominal universal quantifiers. They motivate a more complicated composition of semantics, because they introduce a quantifier relation with scope over the rest of the semantic material of the NP they are in and yet occur in a position (Position VIII) that does not have syntactic scope over that material. Consider the following example, bracketed according to the syntactic structure that is assumed:

(90)

Desapareceram	[as	[[cartas	todas]	da	Ana.]]
disappeared	the	letters	all	of the	Ana

All of Ana's letters disappeared.

We leave this issue to future work.

4.14.1 Postnominal Demonstratives

Postnominal demonstratives are possible in some dialects of Portuguese. They are confined to NPs introduced by a definite article. They do not co-occur with prenominal demonstratives and do not iterate:

(91) a.

A	bicicleta	essa	está	estragada.
the	bicycle	that	is	broken

That bicycle is broken.

b. * Uma bicicleta essa está estragada.
a bicycle that is broken

c. * Essa bicicleta essa está estragada.
that bicycle that is broken

d. * Esta bicicleta essa está estragada.
this bicycle that is broken

e. * A bicicleta essa essa está estragada.
the bicycle that that is broken

The most simple way of blocking co-occurrence of prenominal and postnominal demonstratives and also preventing postnominal demonstratives from iterating is with an extra feature under MK-VAL for demonstratives:

$$\begin{bmatrix} \textit{mk-val} & \\ \text{CARDINAL} & \textit{present-or-absent} \\ \text{ORDINAL} & \textit{present-or-absent} \\ \text{INDEF-SPEC} & \textit{present-or-absent} \\ \text{DEMONSTRATIVE} & \textit{present-or-absent} \end{bmatrix}$$

This feature is used as it can be expected from the use of the other features of *mk-val* as presented before: prenominal and postnominal demonstratives select for sisters with MARKING|MK-VAL |DEMONSTRATIVE of type *absent* and have MARK|MK-VAL|DEMONSTRATIVE with the value *present*, the remaining functors unify the DEMONSTRATIVE attributes under the paths SELECT|LOCAL|CAT|MARKING|MK-VAL and MARK|MK-VAL. In order to block the co-occurrence of indefinite determiners with postnominal demonstratives, indefinite determiners also select for sisters with an *absent* DEMONSTRATIVE.

Prenominal demonstratives and postnominal demonstratives must come in the lexicon in different entries, or related by lexical rules, because the prenominal ones are determiners and carry quantifier semantics. The constraints on MARKING and MARK are also different between these two sets of items. Prenominal demonstratives have a HEAD of type *determiner*, while postnominal ones must have *marking* constraints almost identical to the other elements in the same slot (postnominal adjectives, etc.). The word order between functor and head is also different. The head of postnominal demonstratives looks like:

$$\begin{bmatrix} \textit{postnominal-demonstrative} & \\ \text{MARKER} & \begin{bmatrix} \textit{post-only-marker-min} & \\ \text{MARK} & \begin{bmatrix} \textit{basic-marking} & \\ \text{MK-VAL} & \begin{bmatrix} \text{CARDINAL} & \boxed{1} \\ \text{ORDINAL} & \boxed{2} \\ \text{INDEF-SPEC} & \boxed{3} \\ \text{DEMONSTRATIVE} & \textit{present} \end{bmatrix} \end{bmatrix} \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD } \textit{noun} & \\ \text{MARKING} & \begin{bmatrix} \textit{basic-marking} & \\ \text{MK-VAL} & \begin{bmatrix} \text{CARDINAL} & \boxed{1} \\ \text{ORDINAL} & \boxed{2} \\ \text{INDEF-SPEC} & \boxed{3} \\ \text{DEMONSTRATIVE} & \textit{absent} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

To make the composition of semantics easier with demonstratives, we view determiner demonstratives as carrying two semantic relations: a quantifier relation and an intersective relation in the restrictor of the quantifier. When a demonstrative is used deictically, the second relation can be semantically considered to be roughly similar to the relation of adverbs like *here* or *there*. In this case a noun phrase like *that car* is considered semantically close to a noun phrase like *the car there*, and *this car* to *the car here*. There is some empirical support to this analysis, as the demonstrative and the adverb must agree with respect to deixis:

(92) a. Esta bicicleta aqui está estragada.
this bicycle here is broken
This bicycle here is broken. (the bicycle near me/us)

b. Essa bicicleta aí está estragada.
that bicycle there is broken
That bicycle there is broken. (the bicycle near you)

c. Aquela bicicleta ali está estragada.
that bicycle there is broken
That bicycle there is broken. (the bicycle away from me and you)

d. * Esta bicicleta aí está estragada.

e. * Esta bicicleta ali está estragada.

f. * Essa bicicleta aqui está estragada.

g. * Essa bicicleta ali está estragada.

h. * Aquela bicicleta aqui está estragada.

i. * Aquela bicicleta aí está estragada.

The names of these predicates in the restrictor of the quantifier are the lemma of the demonstrative — we do not commit to saying that they are identical to adverbial relations. They are obviously not so when demonstratives are employed anaphorically, in which case these predicates are assumed to take a different meaning. We cannot detect automatically with the grammar whether a demonstrative is being used anaphorically or deictically, so the relations visible in the MRSs are meant to be underspecifications.

Postnominal demonstratives introduce a single, plain intersective, relation in the MRS, and the quantifier relation comes from the definite article. Prenominal demonstratives introduce two relations in the MRS representation, a similar intersective one and a quantifier relation. We give them semantics similar to that of a postnominal demonstrative co-occurring with a definite article. That is, the quantifier relation of prenominal demonstratives is the same as that of definite articles, thus treating the following examples as paraphrases of one another:

(93) a. o carro esse
the car that
that car

b. esse carro
that car
that car

MRSs for these two cases are shown in Figure 4.20. The LBL of the *_esse_a_rel* relation is the LTOP of the demonstrative determiner's sister: demonstrative determiners unify this LBL with the path SELECT|LOCAL|CONT|HOOK|LTOP under their HEAD.

$$\begin{bmatrix} \textit{mrs} & \\ \text{LTOP} & \boxed{h1}\ h \\ \text{INDEX} & \boxed{e2}\ e \\ \text{RELS} & \left\langle \begin{bmatrix} \textit{\_o\_q\_rel} & \\ \text{LBL} & \boxed{h3}\ h \\ \text{ARG0} & \boxed{x4}\ x \\ \text{RSTR} & \boxed{h6}\ h \\ \text{BODY} & \boxed{h5}\ h \end{bmatrix}, \begin{bmatrix} \textit{\_esse\_a\_rel} & \\ \text{LBL} & \boxed{h7}\ h \\ \text{ARG0} & \boxed{e8}\ e \\ \text{ARG1} & \boxed{x4} \end{bmatrix}, \begin{bmatrix} \textit{\_carro\_n\_rel} & \\ \text{LBL} & \boxed{h7} \\ \text{ARG0} & \boxed{x4} \end{bmatrix}, \begin{bmatrix} \textit{\_avariar\_v\_rel} & \\ \text{LBL} & \boxed{h9} \\ \text{ARG0} & \boxed{e2} \\ \text{ARG1} & \boxed{x4} \end{bmatrix} \right\rangle \\ \text{HCONS} & \left\langle \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h1} \\ \text{LARG} & \boxed{h9} \end{bmatrix}, \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h6} \\ \text{LARG} & \boxed{h7} \end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 4.20: MRS of a sentence with a demonstrative. The corresponding sentences are *esse carro avariou* and *o carro esse avariou* (*that car broke down*).

The attribution of two relations to prenominal demonstratives is also useful in the context of a predeterminer, as in the following example:

(94)	[$_{\text{NP}}$ Todos	esses carros]	avariaram.
	all	those cars	broke down

All those cars broke down.

In Section 4.5 it was stated that determiners co-occurring with predeterminers contribute no quantifier semantics. The fact that demonstratives introduce two relations in the MRS gives us a simple way to distinguish between semantic representations of sentences with a predeterminer and a demonstrative determiner from semantic representations of sentences with a predeterminer and a definite article. More specifically, there have to be versions of prenominal demonstratives that do not introduce quantifier semantics as well, with *marking* constraints similar to the ones on the vacuous definite articles presented in Section 4.5. These determiners are however not semantically vacuous, they still introduce the special predicate in the restrictor of the quantifier. The quantifier relation is introduced by the predeterminer, as before.

In the presence of a predeterminer, NPs with postnominal demonstratives and NPs with prenominal demonstratives have similar MRSs, too. An example MRS is in Figure 4.21. It is the MRS for the sentences *todos esses carros avariaram* and *todos os carros esses avariaram* (*all those cars broke down*).

To control the co-occurrence restrictions between the set of determiners and postnominal demonstratives in (91), the relevant determiners are constrained to select for a sister with DEMONSTRATIVE of type *absent*.

4.14.2 Postnominal Possessives

Postnominal possessives are constrained to occur in indefinite NPs or NPs introduced by a demonstrative (Section 4.9).

We expand the features under MK-VAL with the attribute POSSESSIVE. The new attribute records the presence of a possessive.

The HEAD of postnominal possessives has constraints similar to the other elements in the same NP slot (Position VIII):

$$\begin{bmatrix} \textit{mrs} & \\ \text{LTOP} & \boxed{h1}\ h \\ \text{INDEX} & \boxed{e2}\ e \\ \text{RELS} & \left\langle \begin{bmatrix} \textit{\_todo\_q\_rel} & \\ \text{LBL} & \boxed{h3}\ h \\ \text{ARG0} & \boxed{x4}\ x \\ \text{RSTR} & \boxed{h6}\ h \\ \text{BODY} & \boxed{h5}\ h \end{bmatrix}, \begin{bmatrix} \textit{\_esse\_a\_rel} & \\ \text{LBL} & \boxed{h7}\ h \\ \text{ARG0} & \boxed{e8}\ e \\ \text{ARG1} & \boxed{x4} \end{bmatrix}, \begin{bmatrix} \textit{\_carro\_n\_rel} & \\ \text{LBL} & \boxed{h7} \\ \text{ARG0} & \boxed{x4} \end{bmatrix}, \begin{bmatrix} \textit{\_avariar\_v\_rel} & \\ \text{LBL} & \boxed{h9} \\ \text{ARG0} & \boxed{e2} \\ \text{ARG1} & \boxed{x4} \end{bmatrix} \right\rangle \\ \text{HCONS} & \left\langle \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h1} \\ \text{LARG} & \boxed{h9} \end{bmatrix}, \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h6} \\ \text{LARG} & \boxed{h7} \end{bmatrix} \right\rangle \end{bmatrix}$$

Figure 4.21: MRS of a sentence with predeterminer and a demonstrative. The sentence is *todos esses carros avariaram* (*all those cars broke down*).

$$\begin{bmatrix} \textit{postnominal-possessive} & \\ \text{MARKER} & \begin{bmatrix} \textit{post-only-marker-min} & \\ \text{MARK} & \begin{bmatrix} \textit{basic-marking} & \\ \text{MK-VAL} & \begin{bmatrix} \text{POSSESSIVE} & \textit{present} \\ \text{CARDINAL} & \boxed{1} \\ \text{ORDINAL} & \boxed{2} \\ \text{INDEF-SPEC} & \boxed{3} \\ \text{DEMONSTRATIVE} & \boxed{4} \end{bmatrix} \end{bmatrix} \\ \text{SELECT}|\text{LOCAL}|\text{CAT} & \begin{bmatrix} \text{HEAD}\ \textit{noun} & \\ \text{MARKING} & \begin{bmatrix} \textit{basic-marking} & \\ \text{MK-VAL} & \begin{bmatrix} \text{POSSESSIVE} & \textit{absent} \\ \text{CARDINAL} & \boxed{1} \\ \text{ORDINAL} & \boxed{2} \\ \text{INDEF-SPEC} & \boxed{3} \\ \text{DEMONSTRATIVE} & \boxed{4} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Just like the feature DEMONSTRATIVE is percolated in postnominal possessives, so must the feature POSSESSIVE be percolated in postnominal demonstratives and other functors, so the constraints presented in the previous sections must be expanded in order to accommodate the new feature POSSESSIVE.

The co-occurrence of prenominal and postnominal possessives can be prevented by constraining prenominal possessives to also select for a sister with POSSESSIVE *absent*.

The same attributes (the ones under MK-VAL) can also be used to constrain the different distribution of prenominal and postnominal possessives, by expanding the type hierarchy of types appropriate for these features in order to include information relating to realization and also word order. The new type hierarchy is in Figure 4.22, and these features (CARDINAL, ORDINAL, etc.) are still declared to be of type *present-or-absent*, as presented above.

With this hierarchy, postnominal possessives have *posthead-present* under HEAD|MARKER|MARK|MK-VAL|POSSESSIVE instead of *present*, while prenominal possessives have this feature with the value *prehead-present*. Definite articles, which cannot co-occur with postnominal possessives, select for sisters with SYNSEM|LOCAL|CAT|MARKING|MK-VAL|POSSESSIVE *absent-or-prehead-present*. As usual, all functors that are not possessives must percolate this feature by unifying their MARKER|MARK|MK-VAL

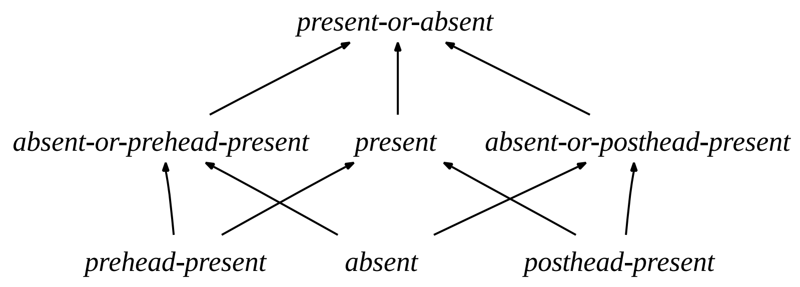


Figure 4.22: Type hierarchy under *present-or-absent* (final version — 2/2). Previous version on p. 100.

|POSSESSIVE with their MARKER|SELECT|LOCAL|CAT|MARKING|MK-VAL|POSSESSIVE.

Indefinite determiners can directly select a sister with MARKING of type *no-poss-marking*, since they cannot co-occur with prenominal possessives, in a way similar to the way cardinal determiners are defined in Section 4.11.

Prenominal demonstratives, which can co-occur with both prenominal and postnominal possessives, do not need to constrain the feature POSSESSIVE.

The constraints just presented are completely stipulative. If we cannot identify a correlation between relative word order between noun and possessive and another property, be it semantic or syntactic, there is no way to escape a stipulative constraint. Indeed, the partition of syntactic and semantic contexts induced by relative order between possessive and noun does not have easily identifiable correlates: it is not a matter of definiteness, because demonstratives pattern differently from definite articles, and, unlike English, where *this* can introduce indefinite specific NPs, demonstratives never have indefinite readings in Portuguese.

4.15 Summary

In this chapter we presented the analysis of NP structure and semantics for Portuguese that has been implemented in LXGram.

We focused on several NP constituents: predeterminers, determiners, prenominal possessives, cardinals, ordinals, markers of indefinite specific NPs, prenominal adjectives modifying the head noun, postnominal adjectives that realize an argument of the noun's semantic relation, postnominal modifying adjectives, postnominal possessives, postnominal demonstratives, PP and AdvP modifiers of nouns, and relative clauses.

We described their place within the noun phrase. We modeled co-occurrence restrictions among these elements, word order constraints, and their role in the composition of the meaning of the NP. We illustrated the implementation of these phenomena in LXGram with examples, granting empirical support to the system of constraints developed to model NP syntax and semantics.

5 Noun Ellipsis and Missing Noun Generics

5.1 Overview

In this chapter, we present an analysis of noun phrases where the head noun is not present.

The most widespread treatment of such constructions is to consider the existence of an empty noun, with an orthographic form that is the empty string. Here we develop an approach that does not resort to empty categories. Empty categories present non-trivial computational problems for parsers, and the systems where LXGram runs do not support them.

We present relevant data to illustrate the properties of such constructions. We also discuss prominent analyses in the literature. A solution that does not involve null elements is then developed. Several aspects are discussed — syntax, semantics, contextual information —, how they can be integrated in HPSG, and to a large extent, implemented in the LKB. Implementation details are also provided.

5.2 Subject Matter

In some NPs there is no overt head noun. Some examples are in (95).

(95) a. [My parents] were here on holidays with [John's -].
  b. [The rich -] always abuse [the poor -].

The sentence in (95a) is however very different from the one in (95b) with respect to the semantics of the missing nouns. In (95a) the noun form *parents* is recoverable from context, but in (95b) the missing noun (something close to *people*) is independent of context and its semantics can be described as generic.

For this reason, the phenomenon in (95b) has been referred to as *people* deletion (Pullum, 1975)) or null-N generics (Nerbonne and Mullen, 2000). Here we will adopt the designation *missing-N generics*, in order to remain neutral to the status of non-realized elements.

On the other hand, the phenomenon in (95a) is known as noun ellipsis.

English employs anaphoric *one* (also known as *one* anaphor) in several contexts where languages that lack this item, e.g. Portuguese, exhibit a missing noun. *One* anaphor has the same semantics as noun ellipsis, and never the semantics of missing-N generics as far as one can tell from the literature. The cases where *one* anaphora are employed are also relevant for our discussion, since we will be focusing on Portuguese, which resorts to not realizing the noun in these circumstances. There is not a one-to-one correspondence between *one* anaphora and noun ellipses: in (96) there is a Portuguese example corresponding to anaphoric *one* and one example where both languages lack the head noun.

(96) a. a casa azul e [a - verde]
the house blue and the green
the blue house and the green one

b. algumas crianças com chapéu e [algumas - com boné]
some children with hat and some with cap
some children in hats and some in caps

As will be apparent in the following discussion, the syntax of noun ellipsis and null-N generics is essentially the same. Their semantics, however, are different, since missing-N generics do not have an antecedent, typically denote humans and carry kind readings, whereas noun ellipsis always has an antecedent and is not semantically restricted in these ways.

In this connection, it is worth noting that the difference between the two constructions (NP ellipsis and missing-N generics) also involves lexical idiosyncrasies. For example, the Portuguese NP in (97a), when taken in isolation, is ambiguous between the noun ellipsis and the missing-N generic reading, as its two English correlates indicate. The English correlate with anaphoric *one* corresponds to the ellipsis reading, and the English correlate with a missing noun corresponds to the missing-N generic reading.

(97) a. [os pobres -]
the poor
the poor (missing-N generic reading)
the poor ones (noun ellipsis reading)

b. [os dois -]
the two
the two (noun ellipsis reading)

The NP in (97b) lacks the missing-N generic reading, and, accordingly, only has one English correlate. But in this case, English surprisingly uses the missing noun strategy, although one would expect noun ellipsis readings to correspond to anaphoric *one* here, too.

We will not go very deeply in the distribution of the English anaphoric *one* here, though, as our focus is on Portuguese.

It is worth pointing out that missing-N generics do not imply the presence of an adjective. The fact that a sentence like *Some like it hot* is the title of a motion picture indicates that the subject NP must be headed by a generic because it cannot be headed by an ellipsis: since this sentence can be uttered out of the blue, there does not need to be antecedent for it to be interpreted. More examples illustrate this point:

(98) a. os sem abrigo
the without shelter
the homeless

b. Os que podem ajudar nunca ajudam.
the who can help never help
The people who can help never do so.

5.3 Data

The following is a list of typical properties of NP ellipsis and missing-N generics that have been reported in the literature.

As these constructions are to be viewed as a phenomenon different from null arguments, at least one specifier, one complement or one modifier is present in the elliptical NP.

In some languages, like German, ellipsis cannot be NP initial (Netter, 1996), but others, like Portuguese, allow it:

(99) a. Alte Männer mit Hut haben [junge - mit Mütze] getroffen.
old men with hat have young with cap met
Old men in hats met young ones in caps.

b. * Alte Männer mit Hut haben [- mit Mütze] getroffen.

c. *Discussion about a purple and pink jellyfish found on a beach:*
Nunca tinha visto [- com estas cores].
never I had seen with these colors
I had never seen ones with these colors.

In some languages, some determiners, like the English definite articles, cannot alone form an NP (English example from (Lobeck, 1995) in (100a)), while other determiners can (cf. (100b) from the same source). It must be noted that we are assuming, like much of the literature on noun ellipsis, that if an item can appear in an NP which is restrictively modified, it is not a pronoun but a determiner, since we also assume that restrictive modifiers attach to $\overline{\mathrm{N}}$ and pronouns are full NPs. Therefore, *some* in (100b) cannot be a pronoun, because it can appear in NPs like *some that we tasted*.

(100) a. * A single protester attended the rally because [the -] apparently felt it was important.

b. We tasted many wines, and I thought that [some -] were extremely dry.

In some languages that have pre-head and post-head adjectives, like Portuguese and Spanish, pre-head ones cannot appear in this construction (Spanish example in (101a) from (Ticio, 2005)), although postnominal adjectives can (Portuguese example in (101b)).

(101) a. * Ayer vi a la verdadera terrorista y a [la supuesta -].
yesterday I saw the true terrorist and the alleged
Yesterday I saw the real terrorist and the alleged one.

b. a terrorista real e [a - imaginada]
the terrorist real and the imagined
the real terrorist and the imagined one

This has sometimes been correlated with intensionality, since at least some prenominal adjectives are intensional. However, as pointed out by Abeillé and Godard (1999), word order and intensionality are at best weakly related.

In addition, the elliptical NP relies on an antecedent to be interpreted, from which it inherits gender as well as subcategorization and count or mass properties (Netter, 1996; Masullo, 1999), exemplified in (102), but not necessarily number (103):

(102) a. die starke Konzentration auf die Wirtschaft und [die weniger grosse - auf den Umweltschutz]
the strong concentration on the economy and the less large on the environment
the strong concentration on the economy and the less large on the environment

b. * Juan visitó a [sus tíos $_{MASC}$] y Pedro visitó a [la - suya $_{FEM}$].
Juan visited his uncles/aunts and Pedro visited the his
(intended:) *Juan visited his aunt and uncle and Pedro visited his aunt.*

(103)	a.	Juan visitó a [sus tíos$_{PLURAL}$] y Pedro visitó a[l - suyo$_{SINGULAR}$]
		Juan visited his uncles/aunts and Pedro visited the his
		Juan visited his aunt and uncle and Pedro visited his uncle.

In this dissertation, we will not describe how the various co-occurrence restrictions mentioned in this section can be controlled. This was done in (Branco and Costa, 2006), based on an analysis of missing noun constructions quite similar to the one presented in this chapter. We will not repeat here the solutions developed there.

5.4 Previous Accounts

Many previous analyses of NP ellipsis, either in the HPSG framework (e.g. (Netter, 1996; Nerbonne and Mullen, 2000)) or under other theoretical persuasions (e.g. (Lobeck, 1995; Ticio, 2005)), assume an empty category approach where the missing noun is assumed to be an actual, though phonetically null, lexical item.

In line with a view of grammar free of reified empty categories, alternatives to this approach have been advanced as well. One of such alternatives was put forward in (Winhart, 1997) and consists in analyzing adjectives in elliptical NPs as the result of a nominalization lexical rule. A major problem for this account, pointed out in (Netter, 1996), is that it cannot derive an elliptical NP where the adjective has modifiers (104).

(104)	die ziemlich alten Männer und [die [besonders jungen] -]
	the quite old men and the particularly young
	the quite old men and the particularly young ones

The same problem arises with missing-N generics — consider cases like *the desperately poor*. A lexical rule is not appropriate because these modifiers attach to adjectives in syntax, but no syntactic category will be adjectival, since the adjective has been converted to a noun in the lexicon.

A similar analysis, based on explaining away the data via some category change of the elements occurring in elliptical NPs, might be envisaged for determiners: when items from these categories appear in elliptical NPs, they could be taken as pronouns, either as a result of some lexical rule, or even as homonymous items included in the lexicon from the start. Such an approach has also found appropriate appreciation and criticism in (Nerbonne and Mullen, 2000), the main argument against it being the possibility of restrictive modification.

Another line of research has been to propose the underspecification of adjectives and other NP elements so that they can function as nouns as well. A crucial problem here concerns how the semantics of the NP is composed given that modifying adjectives (of semantic type $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$) and nouns (of semantic type $\langle e,t\rangle$) , for instance, make different contributions to its semantic content. This is the approach explored in (Beavers, 2003b) for nouns and determiners. That work is limited in its range because it only covers elliptical NPs with a single determiner.

Another option to be explored for an analysis that does not resort to empty categories is to use a unary syntactic rule, which can operate in tandem with the usual Head-Specifier or Head-Adjunct schemata. This possibility is appreciated in (Netter, 1996), to be dismissed as being theoretically uninteresting. Taking into account, however, how the use of unary schemata has been enhanced since then,[1] this is clearly an option worth considering, and it is the approach that will be explored in the next Sections.

[1]Ginzburg and Sag (2000) make heavy use of them, also in the analyses of constructions related to ellipsis, like sluicing, and Sag (2000) employs a syntactic rule to handle VP ellipsis that in some cases may be unary.

Two computational HPSGs for German (Müller and Kasper, 2000; Müller, 1996) indeed use unary syntactic rules that apply to noun adjuncts and produce a noun-headed projection.

The analysis proposed in the following chapters presents a unified treatment of noun adjuncts and determiners in noun ellipsis constructions.

5.5 A Unary Syntactic Rule

In the approach developed in the previous chapters, both for specifiers and adjuncts, the information about their head can be found in a single place in their grammatical representation (viz. the SELECT feature), and the same holds for the information on the nature of the constituents they yield when they are attached to their head (under the MARK feature). This account of NPs in general brings two important advantages: (1) specifiers and modifiers receive a uniform treatment; (2) since most syntactic properties of the constituent resulting from the attachment of a functor with its head are present in the functor, they will be known even if the head is missing. Therefore, a single schema for missing noun constructions can be implemented for both specifiers and adjuncts ensuring syntactic structures that replicate to the widest extent possible the ones obtained when the nominal head is not missing.

Against this background, NPs with a missing head and no complements can be easily accounted for with the help of a single syntactic schema *basic-missing-n-phrase*, which is a straightforward unary version of the functor phrases presented in the previous chapters for NPs, but now without the HEAD-DTR. Some properties of this schema are:

- the MARKING value of the mother node is given by its functor's MARK value;

- the SYNSEM of the mother node is partly shared with the SYNSEM of the functor's SELECT value: it is shared at least for the features HEAD and VAL.[2] As for the remainder features, note that, on the one hand, the SYNSEM|LOCAL|CONT|RELS of the mother node must be the union of the functor's RELS with a multi-set of relations corresponding to the denotation of the missing noun; on the other hand, the MARKING values (i.e. the MARKING feature of the mother node and the MARKING feature of the synsem in the SELECT attribute of the functor) may be incompatible and should not be shared at all;

- the HEAD of the mother is constrained to be a noun (functors not selecting nouns via the SELECT feature will thus not be eligible to feed this syntactic rule), and its COMPS should be inherited from the antecedent.

- As with binary Head-Functor constructions, the functor daughter is required to be a saturated phrase. The type *saturated-cat*, already employed before in Head-Functor constructions, can also be used here. This ensures that e.g. a PP can feed this rule, but not a preposition.

Hence, given an NP with a non-realized head, this schema will directly apply to the functor with the most specific marking type. The other functors will be combined as expected, following the usual schemata in place also for NPs with an overt head.

Figure 5.1 depicts the syntactic constraints associated with the missing noun schema. The semantic properties of this construction are discussed in Section 5.8.

[2]These are the same features that are shared between the mother and the head daughter in a Head-Functor phrase.

$$\left[\begin{array}{l} \textit{basic-missing-n-phrase} \\ \text{SYNSEM}|\text{LOCAL}|\text{CAT} \left[\begin{array}{ll}\text{HEAD} & \boxed{1}\ \textit{noun}\\ \text{VAL} & \boxed{2} \\ \text{MARKING} & \boxed{3}\end{array}\right] \\ \text{ARGS} \left\langle \left[\begin{array}{l} \text{SYNSEM}|\text{LOCAL}|\text{CAT} \left[\begin{array}{l} \textit{saturated-cat} \\ \text{HEAD}|\text{MARKER} \left[\begin{array}{l} \text{SELECT}|\text{LOCAL}|\text{CAT} \left[\begin{array}{ll}\text{HEAD} & \boxed{1} \\ \text{VAL} & \boxed{2}\end{array}\right] \\ \text{MARK}\ \boxed{3}\end{array}\right]\end{array}\right]\end{array}\right]\right\rangle \end{array}\right]$$

Figure 5.1: Outline of the schema for missing noun constructions.

5.5.1 Example

We present the parse tree for the NP *estes dois* (*these two*) in Figure 5.2. The numeral *dois* (*two*) feeds the *basic-missing-n-phrase* rule and yields a node with [HEAD *noun*] and [MARKING *no-poss-marking*]. The determiner attaches as expected, via *functor-head-phrase*, giving rise to a node with [MARKING *saturated*], a full (saturated) NP. The resulting structure is completely parallel to the one of an NP like *estes dois carros* (*these two cars*), except for the missing N node and the branch connecting it.

In general, NPs with missing nouns are derived by an application of the missing N rule to the most embedded constituent as defined by the marking hierarchy and the constraints on the features MK-VAL for the various functors involved. The other functors that are present combine as expected, via the Head-Functor rules presented in Chapter 2. In (105) we show the structures produced by the present analysis for the Portuguese examples *alguns* (*some*), *os seus dois* (*his two*), *a verde* (*the green one*) and *alguns jovens com chapéu* (*some young ones in hats*). The derivation of the last example relies on the fact that prenominal adjectives cannot feed the *basic-missing-n-phrase* (when the noun is realized, the adjective *jovem* can attach to either side, as in *jovens músicos*/*músicos jovens* — *young musicians*), the analysis of which is formalized below.

(105) a. [$_{\textit{saturated}}$ [$_{\text{D}}$ alguns]]

b. [$_{\textit{saturated}}$ [$_{\text{D}}$ os] [$_{\textit{poss-marking}}$ [$_{\text{Poss}}$ seus] [$_{\textit{no-poss-marking}}$ [$_{\text{Num}}$ dois]]]]

c. [$_{\textit{saturated}}$ [$_{\text{D}}$ a] [$_{\textit{n-marking}}$ [$_{\text{A}}$ verde]]]

d. [$_{\textit{saturated}}$ [$_{\text{D}}$ alguns] [$_{\textit{n-marking}}$ [$_{\textit{n-marking}}$ [$_{\text{A}}$ jovens]] [$_{\text{PP}}$ com chapéu]]]

5.6 Transformation of CFGs with Epsilons

Any context free grammar (CFG) with epsilon productions (null constituents) can be transformed into a context free grammar without any epsilons that is weakly equivalent, i.e. it generates the same strings (Bar-Hillel *et al.*, 1961). Consider the fragment in Figure 5.3. Its epsilon-free counterpart is in Figure 5.4.

These two grammar fragments cover exactly the same strings (if terminal symbols are not added, they cover none), but the parse trees are slightly different.

The analysis just presented is essentially the result of applying the same transformation, this time to an HPSG. In fact, the parse trees produced with the missing noun constructions are very similar to the ones yielded by the epsilon free CFG. For instance, an NP consisting of a single determiner

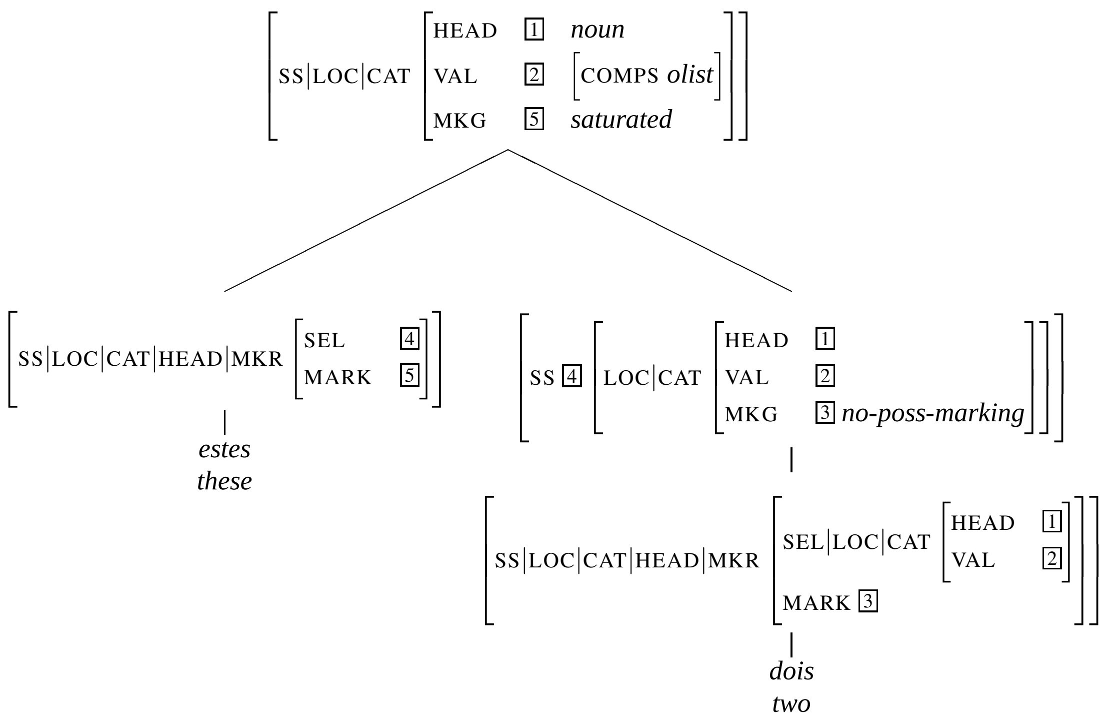


Figure 5.2: Parse for the NP *estes dois* - (*these two*). SS abbreviates SYNSEM, LOC abbreviates LOCAL, MKG abbreviates MARKING, MKR abbreviates MARKER, and SEL abbreviates SELECT.

$$\begin{array}{lll}
\text{NP} & \rightarrow & \text{D} \quad \overline{\text{N}} \\
\overline{\text{N}} & \rightarrow & \overline{\text{N}} \quad \text{PP}_{adjunct} \\
\overline{\text{N}} & \rightarrow & \text{N} \\
\overline{\text{N}} & \rightarrow & \text{N}_{pp\_comp} \quad \text{PP}_{complement} \\
 & & \\
\text{N} & \rightarrow & \varepsilon \\
\text{N}_{pp\_comp} & \rightarrow & \varepsilon
\end{array}$$

Figure 5.3: CFG fragment with epsilons

followed by a PP adjunct receives the structure $[_{\text{NP}}$ D $[_{\overline{\text{N}}}$ PP]] both with the CFG in Figure 5.4 and with the analysis presented before.

A crucial point to note is that the parse trees with the missing noun support the construction of the same semantic representations as the ones supported by trees with reified orthographically null constituents.

5.7 Antecedent Resolution with Noun Ellipsis

In this section, we focus on antecedent resolution for the cases where the missing noun is a noun ellipsis. As mentioned above, missing-N generics do not have an antecedent, and therefore the following considerations do not apply to them.

In many cases, it is not possible to distinguish between noun ellipsis and missing noun generics, so a grammar that tries to recover the antecedent of noun ellipsis will often generate multiple parses for NPs containing a noun ellipsis, as it will also produce a parse corresponding to a missing noun generic.

$$\begin{array}{lll}
\text{NP} & \rightarrow & \text{D} \quad \overline{\text{N}} \\
\overline{\text{N}} & \rightarrow & \overline{\text{N}} \quad \text{PP}_{adjunct} \\
\overline{\text{N}} & \rightarrow & \text{N} \\
\overline{\text{N}} & \rightarrow & \text{N}_{pp\_comp} \quad \text{PP}_{complement} \\
 & & \\
\text{NP} & \rightarrow & \text{D} \\
\overline{\text{N}} & \rightarrow & \text{PP}_{adjunct} \\
\overline{\text{N}} & \rightarrow & \text{PP}_{complement}
\end{array}$$

Figure 5.4: CFG fragment without epsilons

5.7.1 Data and Generalizations

The relation between an NP with an elided noun and its antecedent has been reported in the literature to have properties in common with the kind of anaphoric binding ruled by Principle B (Hankamer and Sag, 1976; Lobeck, 1995, among others; the following examples are theirs). In fact, the antecedent can be given pragmatically, as in (106a), or be in a different sentence (106b).

(106) a. *At a food vendor's*: I'll take [two -].

b. - John caught a big fish.
- Yes, but [Mary's -] was bigger.

This is parallel to anaphoric binding according to Principle B (for personal pronouns). Consider the following examples:

(107) a. *At a food vendor's*: I'll take **them**.

b. - John caught a big fish.
- Yes, **it** was.

The way to determine the antecedent may thus be dependent on how anaphoric binding is analyzed in general, which will not be discussed here. But it is worth noting that, whereas in binding there is an anaphoric relation between NPs, here there is a semantic dependency relation between predicators. Sentence (99a), repeated below in (108a), illustrates this point clearly: there is no anaphoric relation between the subject and the object of that sentence. This is true for the German example in (108a) as well as for its English counterpart in (108b).

(108) a.

Alte	Männer	mit	Hut	haben	[junge	-	mit	Mütze]	getroffen.
old	men	with	hat	have	young		with	cap	met

Old men in hats met young ones in caps.

b. Old men in hats met young ones in caps.

The relation between the non-realized noun and its antecedent is that the predicate that they contribute to the semantics is identical. Note also that the arguments of that predicate are however different. That is, the semantics for the German sentence could be like:

$$\begin{aligned}
& exists(x_1, exists(x_2, hut(x_2), \\
& \qquad alt(x_1) \wedge mann(x_1) \wedge mit(x_1, x_2)), \\
& \quad exists(y_1, exists(y_2, muetze(y_2), \\
& \qquad jung(y_1) \wedge mann(y_1) \wedge mit(y_1, y_2)), \\
& \qquad treffen(x_1, y_1)))
\end{aligned}$$

where *exists* represents an existential generalized quantifier.[3] The piece of semantics associated to the missing noun is $mann(y_1)$ and the one associated to its antecedent is $mann(x_1)$. Although it is the same relation, it is instantiated differently. There is no anaphoric relation between the variables x_1 and y_1 in that formula.

Note that there can be an anaphoric relation between noun ellipsis or anaphoric *one* and their antecedent, as in (109a) for noun ellipsis and (109b) for anaphoric *one*, although it is not co-referentiality. In this example, anaphoric *one* stands for *apples that Bill bought*, a behavior similar to that of e-type pronouns.[4] As mentioned above, this is not always the case.

(109) a. O Bill comprou algumas maçãs. [Todas as vermelhas -] estavam podres.
the Bill bought some apples all the red were rotten
Bill bought some apples. All the red ones were rotten.

b. Bill bought some apples. [All the red ones] were rotten.

c. Bill bought some fruit. [All the red apples] were rotten.

However, anaphoric relations can hold between NPs without noun ellipsis or anaphoric *one*, as in (109c). In this example, *apples* stands for *apples that Bill bought*, thus entering an indirect anaphoric relation. So we conclude that anaphoric behavior is not a defining characteristic of noun ellipsis or of *one* — they are independent.

Although the relation between noun ellipsis or *one* and their antecedent is not necessarily anaphoric, we will continue using the expressions *one anaphor* and *anaphoric one* in this text.

In any case, it is not entirely clear that this kind of semantic dependency must be captured by anaphoric relations or is rather simply a matter of pragmatic restriction on the quantifiers involved. In fact, similar data is invoked in (Nerbonne *et al.*, 1989) in order to claim that the antecedent of noun ellipsis can be a non-atomic expression (110a). The authors also mention that sometimes this dependency is blocked (110b).

(110) a. Ten students attended the meeting. [Three -] spoke.

b. Most deliveries were on time, but [some -] weren't.

The sentence in (110a) contrasts with the one in (110b). In the first the information that is missing is *students that attended the meeting*, whereas in the latter it is only *deliveries* that is understood, not *deliveries that were on time*. Nerbonne *et al.* (1989) point out that the second kind of reading is not restricted to cases where a contradiction would be produced ((110b) would be contradictory with the first kind of reading), and present examples which we will omit here.

Once again, this does not seem to be directly related to an absent noun. One still finds the same contrast with realized nouns:

(111) a. Ten students attended the meeting. Three students spoke.

b. Most deliveries were on time, but some deliveries weren't.

The second sentence in (111a) is also interpreted as *Three students that attended the meeting spoke*, whereas the second sentence of (111b) is not contradictory. For detailed discussions, see studies like the one of Moxey and Sanford (1987) on so-called complement anaphora.

[3]Its first order counterpart is $\exists x_1 \exists x_2 \exists y_1 \exists y_2 [alt(x_1) \wedge mann(x_1) \wedge hut(x_2) \wedge mit(x_1,x_2) \wedge jung(y_1) \wedge mann(y_1) \wedge muetze(y_2) \wedge mit(y_1,y_2) \wedge treffen(x_1,y_1)]$.

[4]A popular example in the literature is *Every host bought just one bottle of wine and served it with dessert.* Here, the pronoun **it** stands for *the bottle of wine that x bought*, where *x* is the variable bound by the universal quantifier.

We thus assume that these effects are also a (possibly pragmatic) phenomenon that affects both overt and covert nouns, about which we have nothing to say here. It seems safe to conclude that it is sufficient to recover the predicate of the antecedent and let the (pragmatic) mechanism responsible for the contrast in (111) also produce the contrast in (110), i.e. pragmatics (or whatever other mechanism underlies these interpretations) must operate on the output of resolution of noun ellipsis antecedents.

Sensitivity to Word Sense

Nerbonne *et al.* (1989) also present the datum in (112).

(112) ? Bill broke his leg falling over a log, and Lois entered it in her daily one.

The point here is that recovery of information from the antecedent is sensitive to word sense. The word *log* is used with the sense *part of tree trunk*, and *one* is intended to take it as its antecedent, but with the sense *record*. The result is infelicitous.

We can assume that word sense is visible in the semantics, e.g. the two word senses of *log* could give rise to two different predicates in the semantic representations. In practice, computational HPSGs cannot perform word sense disambiguation in general though,[5] so it is often the case that the semantic representations that they produce in parsing do not specify word sense.

There my be many ways to integrate word sense with HPSGs. We can assume for simplicity that word sense can be modeled via the type hierarchy, since relation names are also types. In the case of the word *log*, we could have the type *_log_n_rel* with the two subtypes *_log_n_1_rel* and *_log_n_2_rel*. If the relation *_log_n_1_rel* denotes the set of tree trunks and *_log_n_2_rel* the set of records, *_log_n_rel* denotes the union of these two sets (i.e. something that is a tree trunk or a record). This would also be the relation presented in the MRSs produced by a computational grammar.

If antecedent resolution is implemented via unification of predicate names, and if a grammar could enforce semantic restrictions between these predicates and contextual information (i.e. with lexical semantics) the above example could be blocked based on the co-occurrence of conflicting constraints.

The other elements in the first clause would force a reading with *_log_n_1_rel*, and the elements in the second clause would force one with *_log_n_2_rel*. So this sentence would be blocked because of unification failure between *_log_n_1_rel* and *_log_n_2_rel* (i.e. we would be saying that nothing can simultaneously be a tree trunk and a record). But computational grammar use constraints on lexical semantics in a very limited way, if at all, for technical reasons, so we cannot resort to an implementation using lexical semantics.

The point here is that the previous considerations may lead one to think that antecedent resolution must operate on the output of word sense disambiguation (and that we cannot resolve antecedents without having first disambiguated word sense), but this is not true: a grammar can still produce semantic representations underspecified with respect to word sense and resolve antecedents; subsequent word sense disambiguation would produce conflicting constraints that would block sentences like (112).

Under an approach where word sense is represented by typing relation names, all that is required is that the relation names of antecedent and (resolved) missing noun be unified. This note is important, because, we will be resolving antecedents but not word sense.

[5]It can only be done when word sense differences correlate with syntactic differences and these are present in the input. In this case a different lexical entry is required for each sense. This strategy can also be used when there are no syntactic differences between word senses, but it will just multiply parses. Word sense disambiguation will still be required, disguised as parse selection.

Similarities with Principle B

It has been stated that antecedent resolution of noun ellipses or anaphoric *one* has much in common with binding according to Principle B (Lobeck, 1995). There is a difficulty in interpreting this claim in that binding principles describe relations between full NPs, while the relation between noun ellipses/anaphoric *one* and their antecedents is one between nouns. It appears that what is meant is that if the problem is cast in terms of the NPs of which these nouns are heads then a parallelism can be made with binding principles. Under this interpretation, noun ellipses and *one* anaphora share with pronominals (the elements that bind according to Principle B) several aspects:

- The antecedent can be given pragmatically or in a different sentence, as illustrated above in (106).
- They are subject to the Backwards Anaphora Constraint (they cannot precede their antecedent unless they are in a subordinate clause). Example (113a) is from (Lobeck, 1995).

(113) a. Because **she** doesn't like meat, **Sue** ate fish.
b. Because **some -** were rotten, Sue threw away **all the apples** in the fridge.

- They can violate the Complex NP Constraint (the ellipsis can be inside a sentence inside another NP). Example (114a) is also from (Lobeck, 1995).

(114) a. Bill really likes **his new car**. I think that [$_{\text{NP}}$ the fact that **it** is an antique] was a big selling point.
b. Bill really likes his new **cars**. I think that [$_{\text{NP}}$ the fact that **some -** are antiques] was a big selling point.

- They can can have split antecedents:

(115) a. **Jack** went fishing with **Jill**. **They** caught a lot of fish.
b. Jack bought **apples** and **oranges**. **The better looking ones** had a bad taste.

Even so, example (99a), repeated below in (116a), and its Portuguese counterpart in (116b) present a case parallel to binding according to Principle A.

(116) a.

Alte	Männer	mit	Hut	haben	[junge	-	mit	Mütze]	getroffen.
old	men	with	hat	have	young		with	cap	met

Old men in hats met young ones in caps.

b.

Homens	velhos	de	chapéu	encontraram	[-	novos	de	boné.]
men	old	of	hat	met		young	of	cap

Old men in hats met young ones in caps.

From this discussion, we can draw some conclusions, and sketch an HPSG analysis, which is unfortunately not implementable in the LKB— see the next Section.

5.7.2 Towards an Analysis of Antecedent Resolution of Noun Ellipses

An account of antecedent resolution for noun ellipsis/anaphoric *one* cannot resort to unification of noun relations, because the arguments of these relations would be instantiated with the same values, as they would also be unified. Only their names can be unified (the feature PRED). However, the arity of the relation contributed by the noun that serves as the antecedent of a noun ellipsis must be copied somehow, along with its name.

We can think of this as a function *clone* that returns a copy of its argument. Crucially, the feature structure that is returned is not reentrant with the feature structure that is the input to that function.[6]

Information about number also needs to be erased, since a noun ellipsis and its antecedent can have different number values. We assume a function *erase_num* that does precisely this. Function *erase_num* is a function from *local* objects to *local* objects.

Lexical units for nouns could have a constraint like

$$\begin{bmatrix} \textsc{synsem}|\textsc{local}\ \boxed{1} \\ \textsc{introduced-n-antecedents}\ \left\{\boxed{1}\right\} \end{bmatrix}$$

where INTRODUCED-N-ANTECEDENTS contains the possible antecedents for ellipsis or anaphoric *one* that are introduced by this item, and will be empty for all lexical items that are not nouns (the way the value of this feature is calculated is presented below).

Anaphoric *one* and noun ellipsis[7] could then have constraints like the following, where the two functions mentioned above are employed:

$$\begin{bmatrix} \textsc{synsem}|\textsc{local}\ \mathit{erase\_num}(\mathit{clone}(\boxed{1})) \\ \textsc{introduced-n-antecedents}\ \{\} \\ \textsc{possible-n-antecedents}\ \left\{\cdots, \boxed{1}, \cdots\right\} \end{bmatrix}$$

i.e. the LOCAL value of noun ellipsis is type-identical to an element of POSSIBLE-N-ANTECEDENTS, with number information removed. This is its antecedent. The feature POSSIBLE-N-ANTECEDENTS is instantiated elsewhere and propagated to the noun ellipsis sign (see below).

Noun ellipsis also seems to add its antecedent to INTRODUCED-N-ANTECEDENTS (which is not patent in the constraints presented), but this is not entirely clear. Consider the examples in (117).

(117) a. Although we tasted many wines, I thought some were extremely dry. [A few -] were very good, though.

b. Although we tasted many wines, I thought some whiskeys were better. [A few -] were very good, though.

In the second example it is difficult to interpret the ellipsis as having the form *wines* as its antecedent. The form *whiskeys* would seem a better candidate, were it not for the item *though*. So the first example, where *wines* is the antecedent for both ellipses, may indicate that ellipsis makes its antecedent available for further ellipses.

However, it is not clear whether it is a syntactic effect — a syntactic explanation could claim that *wines* would not be available as an antecedent outside its sentence because it is in an embedded clause (the analysis presented here will not predict this however)— or rather a matter of performance (recency, short term memory).

[6]It can be seen as producing a "deep clone", along the practice of some object oriented programming languages (Arnold *et al.*, 2005). That is, all subfeatures are recursively cloned.

Sometimes the expression *type identity* is used within HPSG to refer to this sort of relation, with unification being referred to as *token identity*, by contrast.

[7]There is no lexical entry for the null noun in our analysis, which uses unary syntactic rules for noun ellipsis constructions, as presented so far. But in Section 5.9.1 we will use a lexical type for the missing noun, in spite of maintaining the unary rule approach.

INTRODUCED-N-ANTECEDENTS and POSSIBLE-N-ANTECEDENTS have to be sets of LOCAL objects (of type *local*), since the information taken from the antecedent includes semantic as well as syntactic information (e.g. subcategorization).

The interesting question is then how the values of the features POSSIBLE-N-ANTECEDENTS and INTRODUCED-N-ANTECEDENTS are calculated in phrases. The basic idea is that INTRODUCED-N-ANTECEDENTS can be propagated up a syntactic tree via set union until the root node is reached, at which point it is converted into POSSIBLE-N-ANTECEDENTS with the possible addition of pragmatic or discourse antecedents. POSSIBLE-N-ANTECEDENTS is then propagated down the tree in a way that is possibly sensitive to linear precedence and subordination.

More precisely, the value of INTRODUCED-N-ANTECEDENTS in the mother node of phrases is the union of the values of this feature in all the daughters. Binary phrases thus have a constraint like:

$$\begin{bmatrix} \text{INTRODUCED-N-ANTECEDENTS } \boxed{A} \cup \boxed{B} \\ \text{ARGS } \left\langle \begin{bmatrix} \text{INTRODUCED-N-ANTECEDENTS } \boxed{A} \\ \text{INTRODUCED-N-ANTECEDENTS } \boxed{B} \end{bmatrix} \right\rangle \end{bmatrix}$$

Unary phrases simply pass up INTRODUCED-N-ANTECEDENTS:

$$\begin{bmatrix} \text{INTRODUCED-N-ANTECEDENTS } \boxed{A} \\ \text{ARGS } \left\langle \begin{bmatrix} \text{INTRODUCED-N-ANTECEDENTS } \boxed{A} \end{bmatrix} \right\rangle \end{bmatrix}$$

In root nodes, which can be produced by rules specific to root nodes, we can make use of another feature, CONTEXT-N-ANTECEDENTS, to contain the antecedents that are given pragmatically or in a preceding sentence. Root nodes must have the constraints:

$$\begin{bmatrix} \text{CONTEXT-N-ANTECEDENTS } \boxed{A} \\ \text{INTRODUCED-N-ANTECEDENTS } \boxed{B} \\ \text{POSSIBLE-N-ANTECEDENTS } \boxed{A} \cup \boxed{B} \end{bmatrix}$$

The value of the feature CONTEXT-N-ANTECEDENTS can be produced from several sources. It can be considered empty, in which case the grammar cannot pick up antecedents outside the sentence where a noun ellipsis occurs (but sentences with a noun ellipsis will not receive a parse if no noun precedes the noun ellipsis, because the constraint on the noun ellipsis sign according to which its LOCAL is in POSSIBLE-N-ANTECEDENTS will fail as this feature will have an empty set as its value).

If a grammar is prepared to parse multi-sentence strings, the feature CONTEXT-N-ANTECEDENTS can be instantiated from the value of POSSIBLE-N-ANTECEDENTS of the preceding sentence or sentences. Alternatively, it can be filled by components external to the grammar.

Assuming at most binary branching, most binary phrases can instantiate the value of POSSIBLE-N-ANTECEDENTS in their daughters with the constraints:

$$\begin{bmatrix} \text{POSSIBLE-N-ANTECEDENTS } \boxed{A} \\ \text{ARGS } \left\langle \begin{bmatrix} \text{POSSIBLE-N-ANTECEDENTS } \boxed{A} - \boxed{B} \end{bmatrix}, \begin{bmatrix} \text{POSSIBLE-N-ANTECEDENTS } \boxed{A} \\ \text{INTRODUCED-N-ANTECEDENTS } \boxed{B} \end{bmatrix} \right\rangle \end{bmatrix}$$

where the first element of ARGS denotes the left daughter and the second one the right daughter (see Section 3.4.1). The possible antecedents of the left daughter do not contain the ones that are introduced in the right daughter, since in most cases a noun ellipsis cannot precede its antecedent. Unary constructions simply identify their POSSIBLE-N-ANTECEDENTS with that of its sole daughter.

Constructions responsible for subordination must allow the nouns introduced in the right daughter (the main clause) to be possible antecedents in the left daughter (the subordinated clause — cf. (113b)). Adverbial subordinate clauses must then be introduced by specific constructions, with the constraints:[8]

$$\begin{bmatrix} \text{POSSIBLE-N-ANTECEDENTS } \boxed{A} \\ \text{ARGS } \left\langle \begin{array}{l} \begin{bmatrix}\text{POSSIBLE-N-ANTECEDENTS } \boxed{A}\end{bmatrix}, \\ \begin{bmatrix}\text{POSSIBLE-N-ANTECEDENTS } \boxed{A}\end{bmatrix} \end{array} \right\rangle \end{bmatrix}$$

That is, unlike the constraints for the remaining binary phrases, the constraints on phrases for head-final adverbial subordination do not subtract the set of noun antecedents introduced in the right daughter from the set of possible antecedents of the left daughter.

Finally, we can make an adjustment in lexical items. As explained so far, for every noun its LOCAL will be a member of its POSSIBLE-N-ANTECEDENTS. Although this is innocuous, since overt nouns do not need to be resolved with respect to an antecedent, it is also spurious. We can add a constraint to the syntactic rules that prevents the LOCAL element of a daughter from being a member of its POSSIBLE-N-ANTECEDENTS. The constraints on binary phrases look like this after this adjustment (the constraints on phrases for adverbial subordinate clauses may remain unchanged as none of their daughters will be a noun):

$$\begin{bmatrix} \text{POSSIBLE-N-ANTECEDENTS } \boxed{A} \\ \text{ARGS } \left\langle \begin{array}{l} \begin{bmatrix}\text{SYNSEM}|\text{LOCAL } \boxed{1} \\ \text{POSSIBLE-N-ANTECEDENTS } \boxed{A} - \boxed{B} - \left\{\boxed{1}\right\}\end{bmatrix}, \\ \begin{bmatrix}\text{SYNSEM}|\text{LOCAL } \boxed{2} \\ \text{POSSIBLE-N-ANTECEDENTS } \boxed{A} - \left\{\boxed{2}\right\} \\ \text{INTRODUCED-N-ANTECEDENTS } \boxed{B}\end{bmatrix} \end{array} \right\rangle \end{bmatrix}$$

The features INTRODUCED-N-ANTECEDENTS, POSSIBLE-N-ANTECEDENTS and CONTEXT-N-ANTECEDENTS do not need to be under SYNSEM, since there do not appear to be any items that subcategorize for a constituent with a particular value in one of these attributes. Therefore, we define them to be appropriate for *sign*.

This mechanism was not implemented in the LKB. The LKB does not allow for the definition of arbitrary functions, like *clone* above, or support basic operations on collections, like testing for set membership or computing set difference, which are necessary for this analysis.

The analysis sketched here is intended to be an initial approximation to antecedent resolution of noun ellipsis, on which more interesting ones could be developed in the future. We do not claim that it is a fully satisfying solution. It does not allow for split antecedents and it does not allow for noun ellipses in a complex NP subject (or in any subordinate clause that is not adverbial) to have an antecedent in the main clause, as in (118).

[8]At this point the INTRODUCED-N-ANTECEDENTS of the subordinate clause could be put in a different set, not presented here, so that these elements could be subtracted from the CONTEXT-N-ANTECEDENTS of subsequent sentences, if what is claimed above about the examples in (117) is true.

(118) The fact that [some -] were extremely dry did not prevent us from tasting many wines.

It might also be interesting to consider using lists instead of sets, so that element order could represent recency, which might be a useful heuristic for choosing a single antecedent when several are possible.

We present an example in Figure 5.5, for the sentence *men in hats met ones in caps*. The presents analysis predicts that *hat* is also a possible antecedent for the anaphoric *one* in that example. This is a correct possibility (from the syntactic perspective), in view of sentences like *some men with old hats bought new ones*. The example also illustrates the fact that uniquely identifying antecedents of ellipses requires domain knowledge (men can be in caps, whereas it is difficult for hats to be), and in this example recency does not help. A final comment is that there is still some superfluous information, with the feature POSSIBLE-N-ANTECEDENTS containing several elements in nodes that dominate no ellipsis or anaphoric *one*. This could be amended, but we do not pursue it here.

It is worth pointing out that the mechanism proposed here for resolving the antecedent of noun ellipsis has many similarities with more general algorithms proposed for anaphora resolution, for instance the one of Branco (1999, 2002), which also manipulates collections of antecedent candidates along parse trees. It will be important to work out the present proposal in connection to those more general mechanisms of anaphora resolution, whose insights could be incorporated here.

5.8 Semantics

We sketch a first approximation to the semantics of constructions with missing nouns, ignoring the antecedent if the missing noun is a noun ellipsis, since, as discussed previously, antecedent resolution is not straightforwardly implementable in the LKB and is not fully determinable from syntactic form or configuration alone. More details on the composition of semantics in missing noun constructions will be presented in the subsequent sections.

Figure 5.6 shows the semantic constraints on *basic-missing-n-phrase*.

Its main properties are the following:

- the SYNSEM|LOCAL|CONT|RELS of the mother node is the union of the functor's RELS with a multi-set with a nominal object in it. In the case of noun ellipsis, because we are not recovering the antecedent, this relation can be called *ellipsis_n_rel*, which stands for a relation that has yet to be determined. For missing-N generics we also assume a relation *generic_n_rel*, the extension of which can be assumed to be the set of human beings. A noun phrase like *the poor* is thus analyzed as meaning $\lambda P.the(x, poor(x) \wedge generic(x), P(x))$, i.e. its restrictor is the intersection of the set of poor entities with the set of humans. Because we cannot distinguish between the two constructions in general, the relation that is added is called *ellipsis-or-generic_n_rel*, which is intended to underspecify the two. The added relation is useful for the cases where a single determiner makes up the entire NP, since if a relation were not added, there would be no material in the restrictor of the determiner.[9] We encode the added semantics under the feature C-CONT, where semantics added by specific rules is described.

- Since no handle constraints should be associated with the missing noun, the HCONS feature of the mother node is simply the HCONS of the daughter.

[9]Put differently, if determiners have the semantic type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$, they need to combine with two elements of type $\langle e,t\rangle$ (unary predicates). The second is the denotation of the VP in the same sentence, and the first is the semantic representation of the remaining material in the NP. In the cases where a single determiner makes up the NP, there would be no predicate that could fill the first argument of the quantifier relation introduced by the determiner if this *ellipsis-or-generic_n_rel* were not added.

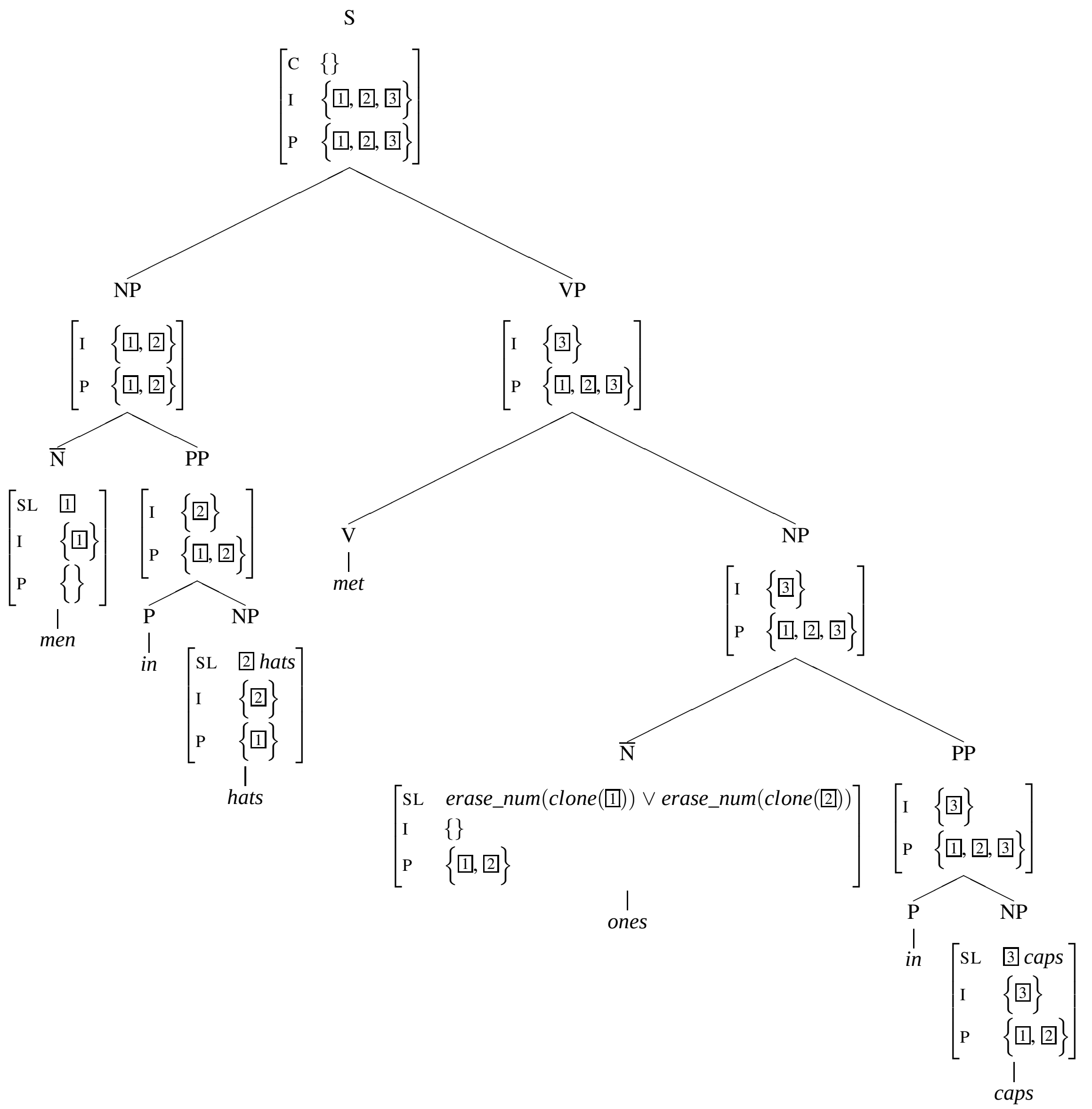


Figure 5.5: Example antecedent resolution in a parse tree. C abbreviates CONTEXT-N-ANTECEDENTS, I abbreviates INTRODUCED-N-ANTECEDENTS, P abbreviatesPOSSIBLE-N-ANTECEDENTS, SL abbreviates the path SYNSEM|LOCAL and italicized noun forms abbreviate feature structures of type *local*. We also omit non-branching rules responsible for bare NPs, as well as feature structures for verb and prepositions, because of space.

$$\left[\begin{array}{l} \textit{basic-missing-n-phrase} \\ \text{SYNSEM}|\text{LOCAL}|\text{CONT} \left[\begin{array}{ll} \text{HOOK} & \left[\begin{array}{ll}\text{LTOP} & \boxed{1} \\ \text{INDEX} & \boxed{3}\end{array}\right] \\ \text{RELS} & \boxed{A} \cup \boxed{B} \\ \text{HCONS} & \boxed{C} \cup \boxed{D} = \boxed{C} \end{array}\right] \\ \text{ARGS} \left\langle \left[\text{SYNSEM}|\text{LOCAL} \left[\begin{array}{l} \text{CAT}|\text{HEAD}|\text{MARKER}|\text{SELECT}|\text{LOCAL}|\text{CONT}|\text{HOOK} \left[\begin{array}{ll}\text{LTOP} & \boxed{2}\\ \text{INDEX} & \boxed{3}\end{array}\right] \\ \text{CONT} \left[\begin{array}{ll} \text{HOOK}|\text{LTOP} & \boxed{1} \\ \text{RELS} & \boxed{A} \\ \text{HCONS} & \boxed{C}\end{array}\right]\end{array}\right]\right]\right\rangle \\ \text{C-CONT} \left[\begin{array}{ll} \text{RELS} & \boxed{B} \left\{ \left[\begin{array}{ll} \text{LBL} & \boxed{2} \\ \text{PRED} & \textit{ellipsis-or-generic\_n\_rel} \\ \text{ARG0} & \boxed{3}\end{array}\right]\right\} \\ \text{HCONS} & \boxed{D}\{\} \end{array}\right] \end{array}\right]$$

Figure 5.6: Semantic constraints of the missing noun schema.

- In general, the INDEX of a nominal projection is the INDEX of the head noun, which is structure-shared with the ARG0 of the noun's relation in the lexical entry for the noun. In the absence of this lexical unit, this unification must be performed here by directly identifying the INDEX of the mother node with the ARG0 of the *ellipsis-or-generic_n_rel* relation.

- The functor must be allowed to see the LTOP and the INDEX of the node it selects because they can be arguments of the relation or relations the functor contributes to the semantics. Since a noun would equate its LTOP with the LBL feature of its relation and its INDEX with the ARG0 feature there, these are unified with the LTOP and INDEX under the SELECT attribute of the functor.

- To simplify our presentation, we ignore Kasper's problem once again (see Section 3.4.2) in this analysis and, as before for Head-Functor phrases, (1) unify the LTOP of the mother node with the LTOP of the (non-head) daughter, and (2) assume in what follows that, in the lexicon, intersective modifiers identify their LTOP with the LTOP of what they select.

5.8.1 Example

We present an example parse for the NP *alguns em Lisboa* (*some in Lisbon*), decorated with LTOP and INDEX features, in Figure 5.7. In that figure, it is assumed that the features SYNSEM|LOCAL|CAT|HEAD|MARKER|SELECT|LOCAL|CONT|HOOK|LTOP and SYNSEM|LOCAL|CONT|HOOK|LTOP are unified in the lexical entry for the preposition.

The resulting MRS is presented in Figure 5.8.

5.9 Missing Daughters in Phrase Types

In Section 3.4.1 it was stated that the order of the daughters of phrasal constituents is denoted in the LKB by the order of elements in the list-valued feature ARGS. Furthermore, attributes like HEAD-DTR

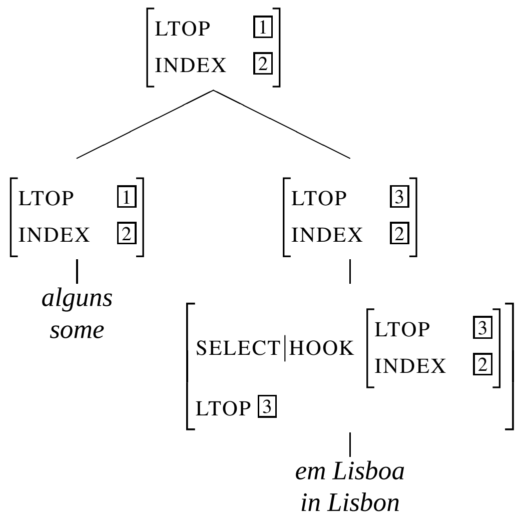


Figure 5.7: Parse for the example NP *alguns - em Lisboa* (*some in Lisbon*). Feature paths are abbreviated.

$$
\begin{bmatrix}
\text{LTOP} & \boxed{h7}\ h \\
\text{INDEX} & \boxed{x6}\ x \\
\text{RELS} & \left\langle
\begin{bmatrix} \textit{algum\_rel} & \\ \text{LBL} & \boxed{h7} \\ \text{ARG0} & \boxed{x6} \\ \text{RSTR} & \boxed{h9}\ h \\ \text{BODY} & \boxed{h8}\ h \end{bmatrix},
\begin{bmatrix} \textit{ellipsis-or-generic\_n\_rel} & \\ \text{LBL} & \boxed{h10}\ h \\ \text{ARG0} & \boxed{x6} \end{bmatrix},
\begin{bmatrix} \textit{em\_rel} & \\ \text{LBL} & \boxed{h10} \\ \text{ARG1} & \boxed{x6} \\ \text{ARG2} & \boxed{x11}\ x \end{bmatrix},
\begin{bmatrix} \textit{proper\_rel} & \\ \text{LBL} & \boxed{h13}\ h \\ \text{ARG0} & \boxed{x11} \\ \text{RSTR} & \boxed{h15}\ h \\ \text{BODY} & \boxed{h14}\ h \end{bmatrix},
\begin{bmatrix} \textit{named\_rel} & \\ \text{LBL} & \boxed{h16}\ h \\ \text{ARG0} & \boxed{x11} \\ \text{CARG} & \textit{Lisboa} \end{bmatrix}
\right\rangle \\
\text{HCONS} & \left\langle
\begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h9} \\ \text{LARG} & \boxed{h10} \end{bmatrix},
\begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{h15} \\ \text{LARG} & \boxed{h16} \end{bmatrix}
\right\rangle
\end{bmatrix}
$$

Figure 5.8: MRS for the NP *alguns - em Lisboa* (*some in Lisbon*).

and NON-HEAD-DTR are merely pointers to these elements, useful when one wants to abstract from word order.

Nothing requires that these daughter features point to an existing element of ARGS, though. That is, it is possible to have constructions with the two features, HEAD-DTR and NON-HEAD-DTR, but with an ARGS list of less than two elements.

This would be a way to model a class of missing syntactic constituents. Assuming that phrases are binary at most (this is enforced in LXGram and several other computational HPSGs), these constructions are prototypically unary, but have semantic or syntactic properties of some other binary constructions.

The difference between ARGS and daughter features (HEAD-DTR and NON-HEAD-DTR) has no theoretical status in HPSG, and the attribute ARGS is specific to the LKB. But we can make a conceptual distinction between them, and give them a theoretical status. The feature ARGS denotes the realized daughters of a phrase, whereas the daughter features (like HEAD-DTR and NON-HEAD-DTR) include them as well as elements that correspond to empty constituents.

There are thus two dimensions: the HEAD-DTR and NON-HEAD-DTR level, which abstracts from the possibility of non-realized constituents, and the ARGS level, which is more superficial in this respect. ARGS is also the best place where the Principle of Canonicality can be enforced (all elements of ARGS are required to have SYNSEMs of type *canonical-synsem*). Therefore, ARGS can be declared to be of the generic type *list(*[SYNSEM *canonical-synsem*]*)*, at least for phrase types (ARGS is also relevant for lexical rules in the systems considered).

Because the LKB and PET do not support parameterized types, it is necessary to create subtypes of *list*: *list-of-signs-with-canonical-synsem*, *cons-of-signs-with-canonical-synsem* (a non empty list of such elements), *null-of-signs-with-canonical-synsems* (an empty list of such signs). The relevant part of the type hierarchy is:

$$\begin{tikzcd}
 & & \textit{list} & & \\
\textit{cons} \arrow[urr] & & \textit{list-of-signs-with-canonical-synsem} \arrow[u] & & \textit{null} \arrow[ull] \\
 & \textit{cons-of-signs-with-canonical-synsem} \arrow[ul] \arrow[ur] & & \textit{null-of-signs-with-canonical-synsem} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

The type *cons-of-signs-with-canonical-synsem* constrains the features FIRST (the head of the list) and REST (its tail), both inherited from *cons*:

$$\begin{bmatrix} \textit{cons-of-signs-with-canonical-synsem} \\ \text{FIRST}|\text{SYNSEM } \textit{canonical-synsem} \\ \text{REST } \textit{list-of-signs-with-canonical-synsem} \end{bmatrix}$$

Note that the most general type for which the feature SYNSEM is appropriate is *sign*, so type inference determines that FIRST is of this type.

The feature ARGS of the type *phrase* is constrained to be of the type *list-of-signs-with-canonical-synsem*. In the subtypes where its size is constrained (it will never be empty), the type for ARGS will be inferred to be *cons-of-signs-with-canonical-synsem*, as this type is the most general unifier of *list-of-signs-with-canonical-synsem*, a constraint inherited from *phrase*, and *cons*, the most general type for which FIRST and REST are appropriate.

For instance, the constraints defined in the supertype of phrases with two daughters can be very simple:

$$\begin{bmatrix} \textit{basic-binary-phrase} \\ \textsc{args} \left\langle \textit{*top*}, \textit{*top*} \right\rangle \end{bmatrix}$$

Since *basic-binary-phrase* inherits from *phrase*, where ARGS is declared to be of the type *list-of-signs-with-canonical-synsem*, the full constraints on *basic-binary-phrase* will be as desired (the definition just presented is notationally equivalent to the left operand):

$$\begin{bmatrix} \textit{basic-binary-phrase} & \\ \textsc{args} & \begin{bmatrix} \textit{cons} & \\ \textsc{first} & \textit{*top*} \\ \textsc{rest} & \begin{bmatrix} \textit{cons} & \\ \textsc{first} & \textit{*top*} \\ \textsc{rest} & \textit{null} \end{bmatrix} \end{bmatrix} \end{bmatrix} \sqcap \begin{bmatrix} \textit{phrase} & \\ \textsc{args} & \textit{list-of-signs-with-canonical-synsem} \end{bmatrix} =$$

$$\begin{bmatrix} \textit{basic-binary-phrase} & \\ \textsc{args} & \begin{bmatrix} \textit{cons-of-signs-with-canonical-synsem} & \\ \textsc{first} & \begin{bmatrix} \textit{sign} \\ \textsc{synsem}\ \textit{canonical-synsem} \end{bmatrix} \\ \textsc{rest} & \begin{bmatrix} \textit{cons-of-signs-with-canonical-synsem} & \\ \textsc{first} & \begin{bmatrix} \textit{sign} \\ \textsc{synsem}\ \textit{canonical-synsem} \end{bmatrix} \\ \textsc{rest} & \textit{null-of-signs-with-canonical-synsem} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The Principle of Canonicality is then simply a constraint on this feature ARGS in a very general type (*phrase*).

The implementation of the missing noun phrases is thus an interesting case to justify these two levels. It is explained in Section 5.9.1.

In the remainder of this section, we focus on the technical issues that need to be addressed in order to dissociate ARGS and the daughter features.

In the LinGO Grammar Matrix, the feature NON-HEAD-DTR is declared for the type *basic-binary-headed-phrase*, which also constrains its ARGS to have two elements (this constraint is inherited from the supertype *basic-binary-phrase*). The attribute HEAD-DTR is in one of its supertypes (*headed-phrase*). For the proposed design to work, this has to be changed, since we want to allow HEAD-DTR and NON-HEAD-DTR to be present also when ARGS is a singleton list.

The attribute NON-HEAD-DTR can be declared in a new type *basic-phrase-with-non-head-dtr*. This type serves the only purpose of introducing the feature NON-HEAD-DTR, and accordingly is the supertype of all phrasal types where this feature is present, no matter what the size of their feature ARGS is. ARGS is not constrained in *basic-phrase-with-non-head-dtr*:

$$\begin{bmatrix} \textit{basic-phrase-with-non-head-dtr} \\ \textsc{non-head-dtr}\ \textit{sign} \end{bmatrix}$$

The relevant part of the hierarchy involving these phrase types can be as presented in Figure 5.9.

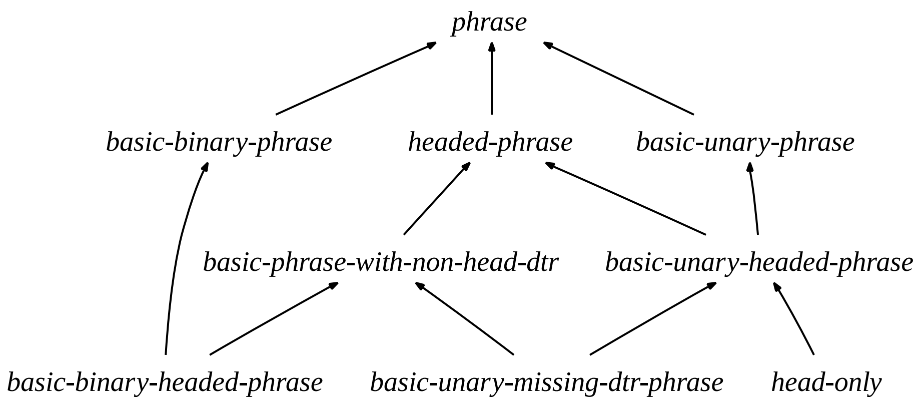

phrase
basic-binary-phrase
headed-phrase
basic-unary-phrase
basic-phrase-with-non-head-dtr
basic-unary-headed-phrase
basic-binary-headed-phrase
basic-unary-missing-dtr-phrase
head-only

Figure 5.9: Type hierarchy of Head-Functor constructions (version 1/3). Final version on p. 149.

In this figure, the types *basic-unary-phrase* and *basic-binary-phrase* constrain the length of ARGS to be one and two respectively:

$$\begin{bmatrix} \textit{basic-unary-phrase} \\ \text{ARGS } \left\langle \textit{*top*} \right\rangle \end{bmatrix} \qquad \begin{bmatrix} \textit{basic-binary-phrase} \\ \text{ARGS } \left\langle \textit{*top*}, \textit{*top*} \right\rangle \end{bmatrix}$$

The type *headed-phrase* is where HEAD-DTR is introduced. It is the supertype of all headed constructions. It contains these constraints, among others:

$$\begin{bmatrix} \textit{headed-phrase} & \\ \text{SYNSEM|LOCAL|CAT|HEAD } \boxed{1} & \\ \text{HEAD-DTR} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM|LOCAL|CAT|HEAD } \boxed{1} \end{bmatrix} \end{bmatrix}$$

The type *basic-phrase-with-non-head-dtr* is where the attribute NON-HEAD-DTR is declared. It is the supertype of *basic-binary-headed-phrase*, from which all binary and headed constructions inherit, and also of *basic-unary-missing-dtr-phrase*, which is the type of construction we are discussing, with a singleton ARGS but both HEAD-DTR and NON-HEAD-DTR. It introduces no constraints of its own, but it inherits several constraints from its supertypes:

$$\begin{bmatrix} \textit{basic-unary-missing-dtr-phrase} & \\ \text{SYNSEM} & \begin{bmatrix} \textit{phrase-synsem} \\ \text{LOCAL|CAT|HEAD } \boxed{1} \end{bmatrix} \\ \text{HEAD-DTR} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM|LOCAL|CAT|HEAD } \boxed{1} \end{bmatrix} \\ \text{NON-HEAD-DTR } \textit{sign} & \\ \text{ARGS} & \begin{bmatrix} \textit{cons-of-signs-with-canonical-synsem} & \\ \text{FIRST} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM } \textit{canonical-synsem} \end{bmatrix} \\ \text{REST } \textit{null-of-signs-with-canonical-synsem} & \end{bmatrix} \end{bmatrix}$$

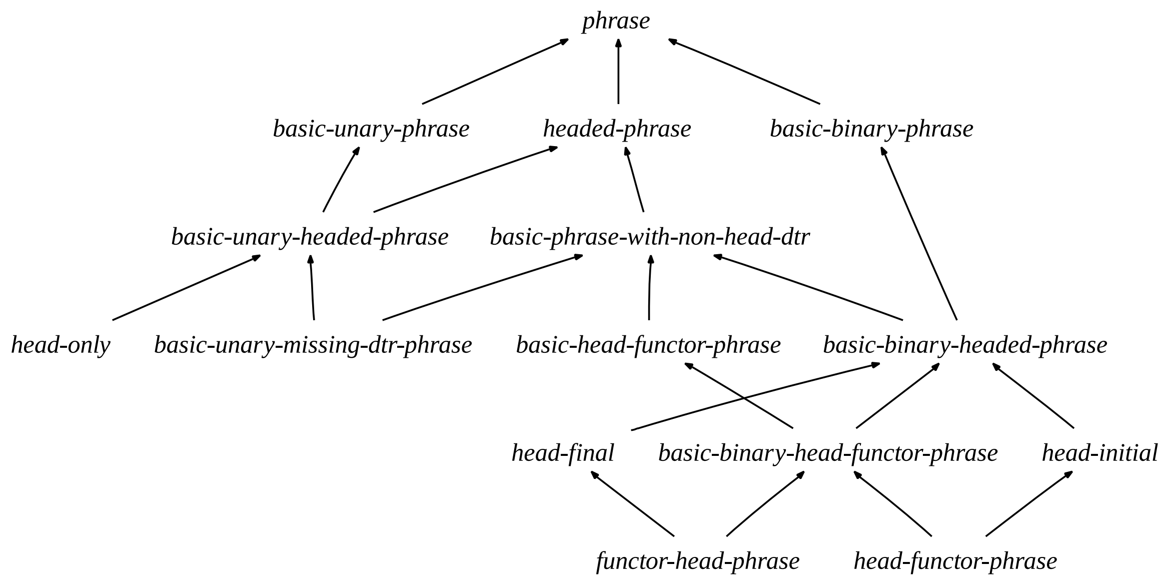

phrase
basic-unary-phrase
headed-phrase
basic-binary-phrase
basic-unary-headed-phrase
basic-phrase-with-non-head-dtr
head-only
basic-unary-missing-dtr-phrase
basic-head-functor-phrase
basic-binary-headed-phrase
head-final
basic-binary-head-functor-phrase
head-initial
functor-head-phrase
head-functor-phrase

Figure 5.10: Type hierarchy of Head-Functor constructions (version 2/3). Previous version on p. 147. Final version on p. 149.

We leave open the possibility for headed unary phrases without the NON-HEAD-DTR feature. Their supertype is *head-only*.

The basic shape of the hierarchy is like in the LinGO Grammar Matrix. The new types are *basic-phrase-with-non-head-dtr*, *basic-unary-missing-dtr-phrase* and *basic-unary-headed-phrase*.[10]

In Section 3.4.1 the types *head-initial* and *head-final* were presented. They define the relation between HEAD-DTR and NON-HEAD-DTR and the elements of ARGS, thus constraining word order between the daughters of a phrase. The result of incorporating these abstract types, as well as the functor phrases, into the revised hierarchy for headedness and arity is in Figure 5.10.

The main point is that the type *basic-head-functor-phrase*, presented in Section 3.4 and the supertype of all constructions involving a functor daughter, does not inherit from *basic-binary-headed-phrase*, because the former can also be a supertype of missing noun constructions (not shown yet). The extra type *basic-binary-head-functor-phrase* is a glbtype. It would be created automatically by the system in order to guarantee a single unifier for *basic-head-functor-phrase* and *basic-binary-headed-phrase* (it could be *functor-head-phrase* or *head-functor-phrase*).

5.9.1 HEAD-DTR in Missing Noun Phrases

With the setup of Figure 5.10, the most intuitive place to put missing noun constructions is as a descendant of *basic-head-functor-phrase* and *basic-unary-missing-dtr-phrase*. This new type is *basic-missing-n-phrase*. The hierarchy for functor phrase types expanded with *basic-missing-n-phrase* is in Figure 5.11.

[10]In the LinGO Grammar Matrix, there are some more intermediate types which are ignored here.

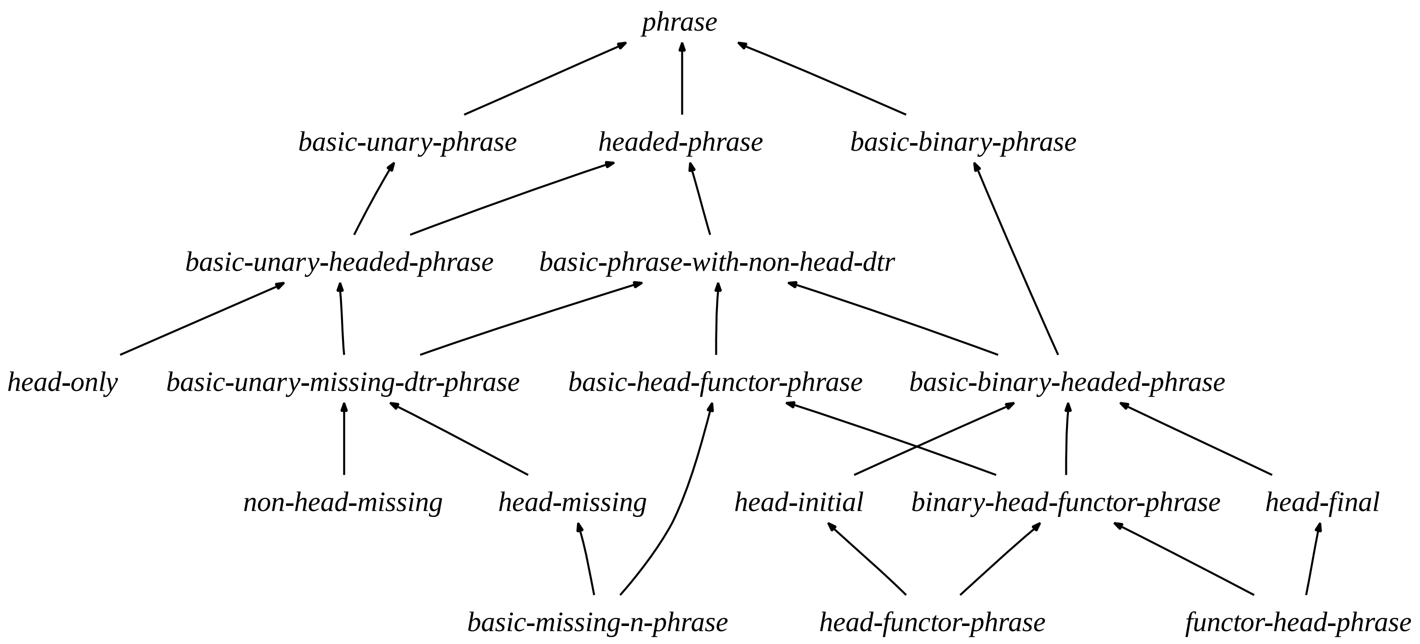


Figure 5.11: Type hierarchy of Head-Functor constructions (final version — 3/3). Previous version on p. 148.

The type *basic-missing-n-phrase* must be further constrained in that it must be specified which daughter is realized. We chose to do it in a supertype, *head-missing*, assuming that there can be other constructions with this property, which could be defined to also inherit from *head-missing*:

$$\begin{bmatrix} \textit{head-missing} \\ \textsc{head-dtr}|\textsc{synsem}\ \textit{non-canonical-synsem} \\ \textsc{non-head-dtr}\ \boxed{1} \\ \textsc{args}\ \left\langle \boxed{1} \right\rangle \end{bmatrix}$$

The attribute SYNSEM of the missing daughter is constrained to be of the type *non-canonical-synsem*, in view of the fact that it is not realized.

Its counterpart type, *non-head-missing*, is also part of the hierarchy in Figure 5.11 and should be specified to have the expected constraints, namely:

$$\begin{bmatrix} \textit{non-head-missing} \\ \textsc{head-dtr}\ \boxed{1} \\ \textsc{non-head-dtr}|\textsc{synsem}\ \textit{non-canonical-synsem} \\ \textsc{args}\ \left\langle \boxed{1} \right\rangle \end{bmatrix}$$

The interesting part of this design is that, in order to add noun semantics and constrain the type of the HEAD and MARKING features (to be *noun* and *n-marking* respectively) that the functor feeding the *basic-missing-n-phrase* will see under its SELECT feature, one can put this information under the HEAD-DTR attribute. The basic machinery put in place to percolate syntactic information from the daughters in headed phrases and Head-Functor schemata fills the appropriate values in the mother node — it is completely inherited from supertypes. Furthermore, the composition of semantics is exactly as in regular binary phrases. All that is required is that the supertypes never constrain ARGS, and use HEAD-DTR and NON-HEAD-DTR instead.

This approach can be taken even further. Since the constraints on the HEAD-DTR feature effectively consist of the definition of a noun, HEAD-DTR can simply be constrained to be of a type that is a supertype of lexical items for nouns.

Suppose the most general supertype of lexical types for nouns is *noun-sign*. The exact constraints on this type are dependent on the grammar and we do not want to restate them in the syntactic rules for noun ellipsis. Under the present analysis, *noun-sign* should include constraints that determine the SYNSEM|LOCAL|CAT|HEAD feature to be of type *noun* and the attribute SYNSEM|LOCAL|CAT|MARKING to be *n-marking*. Under SYNSEM|LOCAL|CONT, HCONS is empty and RELS includes a single relation with an ARG0 of type *ref-index* (the real type name of the variables that show up in MRSs with type *x*) structure-shared with HOOK|INDEX and HOOK|LTOP is unified with the LBL of that relation. We also assume that the feature SUBJ is empty for all nouns and that this is a constraint on *noun-sign*:

$$\begin{bmatrix} \textit{noun-sign} & \\ \textsc{synsem}|\textsc{local} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textsc{head} & \textit{noun} \\ \textsc{val}|\textsc{subj} & \langle\rangle \\ \textsc{marking} & \begin{bmatrix} \textit{n-marking} & \\ \textsc{demonstrative} & \textit{absent} \\ \textsc{possessive} & \textit{absent} \end{bmatrix} \end{bmatrix} \\ \textsc{cont} & \begin{bmatrix} \textsc{hook} & \begin{bmatrix} \textsc{ltop} & \boxed{1} \\ \textsc{index} & \boxed{2} \end{bmatrix} \\ \textsc{rels} & \left\{ \begin{bmatrix} \textsc{lbl} & \boxed{1}\ \textit{handle} \\ \textsc{arg0} & \boxed{2}\ \textit{ref-index} \end{bmatrix} \right\} \\ \textsc{hcons} & \{\} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

A descendant of *noun-sign* is *covert-noun-sign*, representing a noun that has no phonetic realization. The type *covert-noun-sign* has an empty list as the value of its COMPS feature, since we are not performing antecedent resolution. The semantics specific to noun ellipsis constructions (under the approach of not resolving the antecedent) and missing-N generics can also be specified here:

$$\begin{bmatrix} \textit{covert-noun-sign} & \\ \textsc{synsem} & \begin{bmatrix} \textit{unexpressed-synsem} & \\ \textsc{local} & \begin{bmatrix} \textsc{cat}|\textsc{val}|\textsc{comps}\ \langle\rangle \\ \textsc{cont}|\textsc{rels} \left\{ \begin{bmatrix} \textsc{pred}\ \textit{ellipsis-or-generic\_n\_rel} \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The constraint on the type of its SYNSEM attribute (type *unexpressed-synsem*) denotes the fact that this noun is not realized.

This lexical type is not used in lexical entries, since our analysis does not resort to null constituents, but it can be used in the definition of the constructions with missing nouns. These constructions specify their head daughter to be a *covert-noun-sign*:

$$\begin{bmatrix} \textit{basic-missing-n-phrase} \\ \textsc{head-dtr}\ \textit{covert-noun-sign} \end{bmatrix}$$

All the properties specific to *basic-missing-n-phrase* follow immediately from the constraints inherited from its supertypes and the constraints on *covert-noun-sign*, and do not have to be stated as specific constraints in the *basic-missing-n-phrase* type. The full constraints of *basic-missing-n-phrase* — the ones inherited from supertypes and the ones on *covert-noun-sign* — are in Figure 5.12. We do not show the constraints on the NON-LOCAL features, as long distance dependencies fall outside the scope of this dissertation.

$$\begin{bmatrix}
\textit{basic-missing-n-phrase} & \\
\text{SYNSEM}|\text{LOCAL} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
\text{HEAD} & \boxed{1}\ \textit{noun} \\
\text{VAL} & \boxed{2}\begin{bmatrix}\text{SUBJ} & \langle\rangle \\ \text{COMPS} & \langle\rangle\end{bmatrix} \\
\text{MARKING} & \boxed{3}
\end{bmatrix} \\
\text{CONT} & \begin{bmatrix}
\text{HOOK} & \begin{bmatrix}\text{LTOP} & \boxed{4} \\ \text{INDEX} & \boxed{5}\end{bmatrix} \\
\text{RELS} & \boxed{A}\cup\boxed{B}\cup\boxed{C}=\boxed{A}\cup\boxed{B} \\
\text{HCONS} & \boxed{D}\cup\boxed{E}\cup\boxed{F}=\boxed{E}
\end{bmatrix}
\end{bmatrix} \\
\text{HEAD-DTR} & \begin{bmatrix}
\textit{covert-noun-sign} & \\
\text{SYNSEM}\ \boxed{6} & \begin{bmatrix}
\textit{unexpressed-synsem} & \\
\text{LOCAL} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
\text{HEAD} & \boxed{1} \\
\text{VAL} & \boxed{2} \\
\text{MARKING} & \begin{bmatrix}
\textit{n-marking} & \\
\text{CARDINAL} & \textit{absent} \\
\text{ORDINAL} & \textit{absent} \\
\text{INDEF-SPEC} & \textit{absent} \\
\text{DEMONSTRATIVE} & \textit{absent} \\
\text{POSSESSIVE} & \textit{absent}
\end{bmatrix}
\end{bmatrix} \\
\text{CONT} & \begin{bmatrix}
\text{HOOK} & \begin{bmatrix}\text{LTOP} & \boxed{7} \\ \text{INDEX} & \boxed{5}\end{bmatrix} \\
\text{RELS} & \boxed{A}\ \left\{\begin{bmatrix}\text{LBL} & \boxed{7}\ \textit{handle} \\ \text{PRED} & \textit{ellipsis-or-generic\_n\_rel} \\ \text{ARG0} & \boxed{5}\ \textit{ref-ind}\end{bmatrix}\right\} \\
\text{HCONS} & \boxed{D}\ \{\}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
\text{NON-HEAD-DTR}\ \boxed{8} & \begin{bmatrix}
\text{SYNSEM}|\text{LOCAL} & \begin{bmatrix}
\text{CAT} & \begin{bmatrix}
\textit{saturated-cat} & \\
\text{HEAD}|\text{MARKER} & \begin{bmatrix}\text{SELECT} & \boxed{6} \\ \text{MARK} & \boxed{3}\end{bmatrix} \\
\text{VAL}|\text{COMPS} & \textit{olist} \\
\text{MARKING} & \textit{saturated}
\end{bmatrix} \\
\text{CONT} & \begin{bmatrix}
\text{HOOK}|\text{LTOP} & \boxed{4} \\
\text{RELS} & \boxed{B} \\
\text{HCONS} & \boxed{E}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
\text{ARGS} & \left\langle\boxed{8}\right\rangle \\
\text{C-CONT} & \begin{bmatrix}\text{RELS} & \boxed{C}\ \{\} \\ \text{RELS} & \boxed{F}\ \{\}\end{bmatrix}
\end{bmatrix}$$

Figure 5.12: General type for missing noun phrases after type expansion. NON-LOCAL features are ignored.

The advantages of this implementation are:

- No *ad hoc* constraints on missing noun phrases are needed to add noun semantics or to constrain the value of the HEAD feature of the mother node or of the SELECT feature of the functor daughter. The constraints necessary to compose semantics are also inherited from very general supertypes.

- The constraints common to overt nouns and the missing head are stated in a single place. Furthermore, these constraints basically define what a noun is. This makes it easier to change the implementation. For instance, changes in the type hierarchy of *marking* that require changes in the value of MARKING of nouns do not require changes both in the lexical types of nouns and in the definition of missing noun phrases.

- The constraints that define what a noun is are encapsulated in the type used to constrain the HEAD-DTR feature and not directly stated in the type for missing noun phrases.

The main disadvantage is that the feature structures for missing noun phrases will be substantially larger, since the feature structure for an entire lexical item will be present under HEAD-DTR. Note however that it does not imply more unification operations at run time, since no node will be unified with the entire HEAD-DTR attribute, as it is not an element of ARGS.

This approach opens the way for similar analyses to other constructions. In Section 5.9.2 some of these are briefly mentioned.

5.9.2 Other Constructions with Missing Daughters

Other constructions are good candidates for receiving a similar treatment. One example is bare NPs. Consider the following example involving a bare NP:

(119)	Compraram	[$_{\text{NP}}$ maçãs.]
	they bought	apples

They bought apples.

Items that select one NP complement, like the verb in this example, constrain it to be saturated (see Section 4.3). Therefore the constituent bracketed with NP in (119) cannot be the noun directly: nouns have the value *n-marking* for their attribute MARKING, which is incompatible with the type of MARKING of saturated phrases (*saturated*). We assume a unary rule that can account for bare NPs. This rule must produce a node with saturated MARKING and must also add quantifier semantics (*udef_q_rel* is the name given in several computational HPSGs for the quantifier relation of bare NPs). It should contain constraints like the following:

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{bare-np-phrase}} \\
\text{SYNSEM|LOCAL} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll} \text{HEAD} & \boxed{1}\ \textit{noun} \\ \text{VAL} & \boxed{2} \\ \text{MARKING} & \textit{saturated} \end{array}\right] \\
\text{CONT} & \left[\begin{array}{ll} \text{HOOK} & \left[\begin{array}{ll} \text{LTOP} & \boxed{3} \\ \text{INDEX} & \boxed{4} \end{array}\right] \\ \text{RELS} & \boxed{\textit{A}} \cup \boxed{\textit{B}} \\ \text{HCONS} & \boxed{\textit{C}} \cup \boxed{\textit{D}} \end{array}\right]
\end{array}\right] \\
\text{ARGS} & \left\langle \left[\begin{array}{ll} \text{SYNSEM|LOCAL} & \left[\begin{array}{ll}
\text{CAT} & \left[\begin{array}{ll} \text{HEAD} & \boxed{1} \\ \text{VAL} & \boxed{2} \\ \text{MARKING} & \textit{no-det-marking} \end{array}\right] \\
\text{CONT} & \left[\begin{array}{ll} \text{HOOK} & \left[\begin{array}{ll} \text{LTOP} & \boxed{5} \\ \text{INDEX} & \boxed{4} \end{array}\right] \\ \text{RELS} & \boxed{\textit{A}} \\ \text{HCONS} & \boxed{\textit{C}} \end{array}\right]
\end{array}\right] \end{array}\right] \right\rangle \\
\text{C-CONT} & \left[\begin{array}{lll}
\text{RELS} & \boxed{\textit{B}} & \left\{ \left[\begin{array}{ll} \text{LBL} & \boxed{3} \\ \text{PRED} & \textit{udef\_q\_rel} \\ \text{ARG0} & \boxed{4} \\ \text{RSTR} & \boxed{6} \\ \text{BODY} & \textit{handle} \end{array}\right] \right\} \\
\text{HCONS} & \boxed{\textit{D}} & \left\{ \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{qeq}} \\ \text{HARG} & \boxed{6} \\ \text{LARG} & \boxed{5} \end{array}\right] \right\}
\end{array}\right]
\end{array}\right]$$

Missing noun phrases are instances of Functor-Head constructions with the head daughter missing, where the missing daughter is constrained to be a noun. Bare NPs can be considered Functor-Head constructions with the functor daughter missing, where the missing daughter is a determiner. Bare NP constructions would thus inherit from *non-head-missing* and *basic-head-functor-phrase*, presented above.

A solution for bare-NPs similar to the one for missing noun constructions can be envisaged, using the type hierarchy for lexical types of determiners to factor out the behavior common to overt determiners and the missing daughter of bare NPs. Suppose there is a type *determiner-with-semantics-sign* that is a supertype of all lexical types for determiners that carry quantifier semantics (the ones that appear at NP initial position, not a supertype of the lexical types for the determiners that follow predeterminers). This type, *determiner-with-semantics-sign*, contains the constraints that are common to realized determiners and the constraints required in bare NP constructions as well. Among these, one finds quantifier semantics, the constraint on HEAD to be of the type *determiner*, etc:

$$\begin{bmatrix} \textit{determiner-with-semantics-sign} & \\ \textsc{synsem}|\textsc{local} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textsc{head} & \begin{bmatrix} \textit{determiner} & \\ \textsc{marker} & \begin{bmatrix} \textit{pre-only-marker-min} & \\ \textsc{select}|\textsc{local} & \begin{bmatrix} \textsc{cat} & \begin{bmatrix} \textsc{head}\ \textit{noun} \\ \textsc{marking}\ \textit{no-det-marking} \end{bmatrix} \\ \textsc{cont}|\textsc{hook} & \begin{bmatrix} \textsc{ltop} & \boxed{4} \\ \textsc{index} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \textsc{mark}\ \textit{saturated} & \end{bmatrix} \end{bmatrix} \\ \textsc{val} & \begin{bmatrix} \textsc{subj} & \langle\rangle \\ \textsc{comps} & \langle\rangle \end{bmatrix} \end{bmatrix} \\ \textsc{cont} & \begin{bmatrix} \textsc{hook}|\textsc{ltop}\ \boxed{1} & \\ \textsc{rels} & \left\{ \begin{bmatrix} \textsc{lbl} & \boxed{1} \\ \textsc{arg0} & \boxed{2} \\ \textsc{rstr} & \boxed{3} \\ \textsc{body} & \textit{handle} \end{bmatrix} \right\} \\ \textsc{hcons} & \left\{ \begin{bmatrix} \textit{qeq} & \\ \textsc{harg} & \boxed{3} \\ \textsc{larg} & \boxed{4} \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

There would also be a subtype of *determiner-with-semantics-sign*, *covert-determiner-sign*, representing a null determiner:

$$\begin{bmatrix} \textit{covert-determiner-sign} & \\ \textsc{synsem} & \begin{bmatrix} \textit{unexpressed-synsem} & \\ \textsc{local}|\textsc{cont}|\textsc{rels} & \left\{ \begin{bmatrix} \textsc{pred}\ \textit{udef\_q\_rel} \end{bmatrix} \right\} \end{bmatrix} \end{bmatrix}$$

The constraints on bare NP phrases could thus be very simple, too:

$$\begin{bmatrix} \textit{bare-np-phrase} \\ \textsc{non-head-dtr}\ \textit{covert-determiner-sign} \end{bmatrix}$$

Since *bare-np-phrase* inherits from *non-head-missing* and *basic-head-functor-phrase*, the full constraints on *bare-np-phrase* would be as presented in Figure 5.13. Most of the constraints that are necessary in bare NP phrases are thus inherited from the types that were already defined for Head-Functor constructions and from the definitions of the lexical types for determiners. We achieve the desired effect by exploiting the type hierarchy.

It is also interesting to examine if all constructions that involve a non-empty C-CONT feature (where relations and handle constraints that do not correspond to lexical items but are added by constructions are stated) could be cast in terms of this architecture, thus effectively making the feature C-CONT

$$
\begin{bmatrix}
\textit{bare-np-phrase} \\
\text{SYNSEM}|\text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \boxed{1}\ \textit{noun} \\ \text{VAL} & \boxed{2} \\ \text{MARKING} & \boxed{3}\ \textit{saturated}\end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{HOOK} & \begin{bmatrix}\text{LTOP} & \boxed{4} \\ \text{INDEX} & \boxed{5}\end{bmatrix} \\ \text{RELS} & \boxed{A} \cup \boxed{B} \cup \boxed{C} = \boxed{A}\cup\boxed{B} \\ \text{HCONS} & \boxed{D}\cup\boxed{E}\cup\boxed{F} = \boxed{D}\cup\boxed{E}\end{bmatrix}\end{bmatrix} \\
\text{HEAD-DTR}\ \boxed{6} & \begin{bmatrix}\text{SYNSEM}\ \boxed{7} & \begin{bmatrix}\textit{canonical-synsem} \\ \text{LOCAL} & \begin{bmatrix}\text{CAT} & \begin{bmatrix}\text{HEAD} & \boxed{1} \\ \text{VAL} & \boxed{2} \\ \text{MARKING} & \textit{no-det-marking}\end{bmatrix} \\ \text{CONT} & \begin{bmatrix}\text{HOOK} & \begin{bmatrix}\text{LTOP} & \boxed{8} \\ \text{INDEX} & \boxed{5}\end{bmatrix} \\ \text{RELS} & \boxed{A} \\ \text{HCONS} & \boxed{D}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\
\text{NON-HEAD-DTR} & \begin{bmatrix}\textit{covert-determiner-sign} \\ \text{SYNSEM} & \begin{bmatrix}\textit{unexpressed-synsem} \\ \text{LOCAL} & \begin{bmatrix}\text{CAT} & \begin{bmatrix}\textit{saturated-cat} \\ \text{HEAD} & \begin{bmatrix}\textit{determiner} \\ \text{MARKER} & \begin{bmatrix}\textit{pre-only-marker-min} \\ \text{SELECT} & \boxed{7} \\ \text{MARK} & \boxed{3}\end{bmatrix}\end{bmatrix} \\ \text{VAL} & \begin{bmatrix}\text{SUBJ} & \langle\rangle \\ \text{COMPS} & \textit{onull}\end{bmatrix} \\ \text{MARKING} & \textit{saturated}\end{bmatrix} \\ \text{CONT} & \begin{bmatrix}\text{HOOK}|\text{LTOP}\ \boxed{4} \\ \text{RELS} & \boxed{B}\left\{\begin{bmatrix}\text{LBL} & \boxed{4} \\ \text{PRED} & \textit{udef\_q\_rel} \\ \text{ARG0} & \boxed{5} \\ \text{RSTR} & \boxed{9} \\ \text{BODY} & \textit{handle}\end{bmatrix}\right\} \\ \text{HCONS} & \boxed{E}\left\{\begin{bmatrix}\textit{qeq} \\ \text{HARG} & \boxed{9} \\ \text{LARG} & \boxed{8}\end{bmatrix}\right\}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\
\text{ARGS} & \left\langle\boxed{6}\right\rangle \\
\text{C-CONT} & \begin{bmatrix}\text{RELS} & \boxed{C}\ \{\} \\ \text{HCONS} & \boxed{F}\ \{\}\end{bmatrix}
\end{bmatrix}
$$

Figure 5.13: Full constraints on bare NP phrases

unnecessary. In our example, the first version of the *basic-missing-n-phrase*, presented in Section 5.5 and Section 5.8, constrained the C-CONT in order to add the semantics for the missing noun. In the final version, however, the semantics for the missing noun comes from the HEAD-DTR and is combined with the semantics of the functor daughter via the constraints for the composition of semantics that are inherited from *basic-head-functor-phrase*. We no longer needed C-CONT to contain the semantic relation that must be added. A similar situation happened with the phrasal type for bare NP constructions. So our designs dispenses with the attribute C-CONT. We believe that our solution is much more general than a solution involving C-CONT, since we can factor out many of the commonalities between Head-Functor constructions and constructions for missing nouns via supertypes.

5.10 Predeterminers in Missing Noun Constructions

The account presented for NPs with a missing head is general enough to accommodate practically all the NP elements presented in Chapter 4. There is however an exception: the predeterminer *todos* (*all*). This element can appear in noun ellipsis (120a) or missing-N generic constructions (120b).

(120) a.

O	João	comprou	maçãs	e	todas	estavam	podres.
the	João	bought	apples	and	all	were	rotten

João bought apples and all (of them) were rotten.

b.

Todos	são	livres.
all	are	free

All (people) are free.

When this element appears in NPs with an overt head, a determiner must also be present in the case of European Portuguese, as reported in Section 4.5. The analysis presented in that section resorted to two lexical entries for this item, one of them common to European and Brazilian Portuguese, where the presence of a determiner is required, and another specific to Brazilian Portuguese, where it is not.

For the first entry, a lexical specification was employed in predeterminers according to which they select for a nominal projection with a value of MARKING subsumed by *non-saturated-det-marking*. This constraint makes it incompatible with the missing-N construction developed in the present chapter, since the daughter of this rule is constrained to be a functor selecting for a constituent that bears the value *n-marking* for the attribute MARKING.

The second entry, specific to Brazilian Portuguese, is allowed in the missing noun constructions, but sentences like the ones in (120) are possible in European Portuguese, too.

Therefore, we would like the less specific item (i.e. the one common to European and Brazilian Portuguese) to be allowed in this construction, and the one exclusive to Brazilian Portuguese to be blocked, since that would just multiply parses.

Underspecifying the value of MARKING corresponding to the missing noun (e.g. constraining it to be simply *non-saturated* in the type *covert-noun-sign*) will allow both elements to feed the rules for the missing nouns. To this end, it is necessary to change the constraints on *noun-sign*. Namely, the constraint on the MARKING value (*n-marking*) should not be stated in *noun-sign* (if it were, it would be inherited by *covert-noun-sign*), but rather in the subtypes of *noun-sign* that are used in the lexical entries of nouns. The updated constraints on *noun-sign* are:

$$\begin{bmatrix} \textit{noun-sign} & \\ \text{SYNSEM}|\text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \textit{noun} \\ \text{VAL}|\text{SUBJ} & \langle\rangle \\ \text{MARKING} & \begin{bmatrix} \textit{non-saturated} & \\ \text{CARDINAL} & \textit{absent} \\ \text{ORDINAL} & \textit{absent} \\ \text{INDEF-SPEC} & \textit{absent} \\ \text{DEMONSTRATIVE} & \textit{absent} \\ \text{POSSESSIVE} & \textit{absent} \end{bmatrix} \end{bmatrix} \\ \text{CONT} & \begin{bmatrix} \text{HOOK} & \begin{bmatrix} \text{LTOP} & \boxed{1} \\ \text{INDEX} & \boxed{2} \end{bmatrix} \\ \text{RELS} & \left\{ \begin{bmatrix} \text{LBL} & \boxed{1} & \textit{handle} \\ \text{ARG0} & \boxed{2} & \textit{ref-index} \end{bmatrix} \right\} \\ \text{HCONS} & \{\} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The lexical types for realized nouns inherit these constraints and their MARKING attribute is further constrained to be of the type *n-marking*.

The item exclusive to Brazilian Portuguese can be constrained to select for a constituent with a synsem of type *canonical-synsem*. This will prevent it from feeding the rules for missing nouns, since the HEAD-DTR of these constructions has a SYNSEM of type *unexpressed-synsem*, that is incompatible with *canonical-synsem*.

5.11 Summary

In this chapter we presented the data concerning noun phrases with the head missing and a model of these data that has been implemented in LXGram.

We began by reporting on the solutions that have been presented in the literature concerning these noun phrases. These solutions involve the postulation of a noun with no phonetic or orthographic manifestation.

We proceeded to present an analysis that does not resort to empty categories and can cover the same data. We modeled these constructions via a special version of the Head-Functor constructions that were presented in the previous chapters. This special version is a unary phrase whose only daughter is the functor. It is further constrained in order to allow as valid daughters only the functors that can attach to a noun. The semantics of a missing noun is also added by this rule to the semantic representation that is built for the sentence where such noun phrases occur.

Furthermore, the analysis developed in this chapter does not require a large number of specific stipulations. We took advantage of several levels of abstraction: the concept of functors as an abstraction over specifiers and adjuncts, ARGS as a separate level of representation from the level of features like HEAD-DTR and NON-HEAD-DTR, the level of the *sign* and the level of the *synsem*, the type hierarchy for phrases and the hierarchy of lexical types. With these tools, we can account for the various peculiarities of these constructions with a very small set of stipulations: most of the information is inherited from general types or results from very general combinations, like constraining the missing daughter of the relevant constructions with a type in the hierarchy of noun lexemes.

We also discussed several limitations faced by a computational treatment of this phenomenon. They mostly surface due to the difficulty of integrating contextual information, which is still a difficult task for grammars for the deep linguistic processing of sentences.

6 Conclusions

In this dissertation we presented a formal description of Portuguese NP syntax and semantics implemented in a deep computational grammar for Portuguese. We also considered the case of NPs where the head noun is not realized.

We used the framework of Head-Driven Phrase Structure Grammar (HPSG) to model these linguistic phenomena. We modeled the syntax of the Portuguese NP, with constraints that describe the possible word order pattern of the elements internal to the NP and the restrictions on their co-occurrence. We also described their semantics. To that end, we used Minimal Recursion Semantics (MRS) as the format of semantic descriptions. MRS allows avoiding some calculations (like determining the relative scope among the various quantifiers in a logical formula that is determined by a computational grammar to be the translation of a natural language sentence) that are sometimes not required for applications.

The system of constraints that was developed to account for these linguistic phenomena was implemented in a computational grammar, LXGram, using the Linguistic Knowledge Builder (LKB), a platform that includes parsing and generation algorithms and debug tools. LXGram can also be used with the PET parser, an efficient parser for computational HPSGs. LXGram was implemented using the set of predefined types in the LinGO Grammar Matrix and expanding upon it. The LinGO Grammar Matrix comes with the core set of the types and the attributes commonly used in HPSG already defined.

The analyses presented are new and to a large extent cover phenomena for which there are not many HPSG analyses. As far as we know, there is little HPSG work on the internal syntax of the NP beyond the work of Abeillé and Godard (1999); Allegranza (1998a,b); Beavers (2003a,b); Nerbonne *et al.* (1989); Nerbonne and Mullen (2000); Netter (1996); Van Eynde (2003a,b); Winhart (1997). Also, other existing computational HPSGs tend to focus on other aspects, like sentence structure and verbal subcategorization, so a systematic implementation of the phenomena covered is also new. A detailed and deep computational implementation of the structure of the Portuguese noun phrase is new as well.

6.1 Summary and Discussion

The system of constraints presented in this dissertation covers a non-trivial subset of NP level phenomena.

In the first two chapters we presented the formalism employed to address the task at hand. We described the most important properties of this formalism, that resorts to a type system supporting multiple inheritance, *has-a* relations between objects, and unification of types and data structures. We presented the framework used to tackle the problem at hand, HPSG, and some of the types and attributes that are used in HPSG. We demonstrated the functionality of a computational grammar, which translates between natural language sentences and logical formulas. The format of semantic representations used in LXGram, MRS, was also explained.

In Chapter 3, we presented a set of features and types that were used in the remaining chapters to model Portuguese NP syntax and semantics. We resorted to the concept of functor, which merges the two linguistic concepts of specifier and adjunct, heavily inspired by the work of Allegranza (1998a,b) and Van Eynde (2003a,b). We presented the implementation of the functors organization in LXGram. The set of attributes that was used allows describing arbitrary levels of saturation of syntactic constituents (their combinatorial potential with other elements) via a type system. It also allows constraining word order possibilities.

In Chapter 4, we modeled the interaction of several NP constituents:

- **Determiners and Predeterminers** We considered NPs starting with determiners (articles, prenominal demonstratives, etc.) or with elements like *todo* (*all*) followed by a determiner:

  (121) a. as$_{\text{DET}}$ pessoas
  the people
  the people

  b. aquelas$_{\text{DET}}$ pessoas
  those people
  those people

  c. todas$_{\text{PREDET}}$ as$_{\text{DET}}$ pessoas
  all the people
  all (the) people

  d. todas$_{\text{PREDET}}$ aquelas$_{\text{DET}}$ pessoas
  all those people
  all those people

  We discussed semantic issues (which elements correspond to quantifiers), and controlled word order between predeterminers and determiners. Our approach was to consider that a subset of determiners (the ones that can be preceded by a predeterminer) are ambiguous between a version carrying quantifier semantics and giving rise to full NPs (the ones that occur at NP-initial position) and a version carrying no semantics and not producing a full NP immediately (the ones that occur after a predeterminer). This kind of ambiguity is fully determined by syntactic context and is not a cause of overgeneration.

- **Possessives** We noted that possessives can occur before or after the noun in Portuguese, depending on syntactic context:

  (122) a. a minha$_{\text{POSS}}$ irmã
  the my sister
  my sister

  b. uma irmã minha$_{\text{POSS}}$
  a sister my/mine
  a sister of mine

  We identified the syntactic contexts responsible for the two word order possibilities and modeled these restrictions accordingly. Our treatment also blocks the presence of more than one possessive in the same NP. We observed that sometimes possessives express ownership (as in (123a) below),[1] and sometimes their semantic content corresponds to an argument of the relation denoted by the noun (as in (123b) below, where the relational noun *irmã* denotes a two-place predicate relating a person with their sisters):

[1]It is really not necessarily ownership, but we use this description here as a simplification.

(123) a. a minha$_{\text{POSS}}$ bicicleta
the my bicycle
my bicycle

b. a minha$_{\text{POSS}}$ irmã
the my sister
my sister

We implemented a way to account for both kinds of semantics. We analyzed possessives as ambiguous (denoting ownership or not).

- **Cardinal numerals** We discussed the semantics of cardinal numbers and their syntax in the presence vs. absence of material preceding them in the same NP:

(124) a. [$_{\text{NP}}$ Os dois$_{\text{CARD}}$ carros] avariaram.
the two cars broke down
The two cars broke down.

b. [$_{\text{NP}}$ Dois$_{\text{CARD}}$ carros] avariaram.
two cars broke down
Two cars broke down.

Our treatment covers the fact that they can appear in NP-initial position or following a determiner. It does not allow them to occur after indefinite determiners, like *algum* (*some*) or to appear repeated.

In view of the fact that they can co-occur with items that carry quantificational semantics (*todas aquelas quatro pessoas — all those four people*) that is not existential, the classical semantic treatment mentioned in the literature (Barwise and Cooper, 1981, that associates them with existential quantification) could not be adopted. When cardinals occur at the beginning of an NP they must introduce quantifier semantics but in other contexts they do not carry this piece of semantics. Accordingly, we analyzed them as ambiguous elements in this respect, but once again the semantic content correlates perfectly with the position at which they occur, so this local ambiguity does not multiply parses. We decomposed the semantics of cardinals in several semantic relations. We did not resort to multiple lexical entries as a way to treat that kind of ambiguity, as one of the versions can be produced from the other by monotonically adding information, and some of the steps in deriving the full semantics are required in both versions.

- **Ordinal numerals** We observed that in Portuguese word order between cardinals and ordinals is free after a determiner:

(125) a. os dois$_{\text{CARD}}$ primeiros$_{\text{ORD}}$ capítulos
the two first chapters
the first two chapters

b. os primeiros$_{\text{ORD}}$ dois$_{\text{CARD}}$ capítulos
the first two chapters
the first two chapters

The analysis that was developed accounts for this and the following facts: ordinals cannot be repeated, even when other elements intervene; they always precede the noun and prenominal adjectives; they follow determiners and prenominal possessives; they never occur at the beginning of an NP.

- **Elements like prenominal *certo* and *determinado* (*certain*) that mark NPs with indefinite specific readings** We discussed and modeled the word order constraints between these elements and the other prenominal items:

(126) a. Certos dois carros avariaram.
certain two cars broke down
Two certain cars broke down.

b. Dois certos carros avariaram.
two certain cars broke down
Two certain cars broke down.

We concluded that they occupy the same position as cardinals and ordinals with respect to word order constraints between them and the remaining NP elements. The analysis that was implemented accounts for this as well as the fact that these elements are not repeatable inside the same NP.

We also described the implementation of their semantic content in LXGram. Their presence merely affects scope between the existential quantifier in the NP they belong to and other scope bearing elements (quantifiers or other elements, like negation), as the following examples illustrate:

(127) a. Todos os homens leram um livro.
all the men read a book
All men read a book.
1. $\exists y[book(y) \wedge \forall x[man(x) \rightarrow read(x,y)]]$
2. $\forall x[man(x) \rightarrow \exists y[book(y) \wedge read(x,y)]]$

b. Todos os homens leram um certo livro.
all the men read a certain book
All men read a certain book.
1. $\exists y[book(y) \wedge \forall x[man(x) \rightarrow read(x,y)]]$

In these examples the first reading corresponds to the case of there being at least one book that all men have read, whereas the second reading asserts that for all men there is at least one book that they have read, but it can be a different book for each of them. When the item *certo* is present, the first reading is enforced (but note that the second reading is a logical consequence of the first, so it will be true if the first reading is true; but if the sentence in the second example is false, we know nothing about the truth value of the second formula).

In this dissertation we explained a way to block unacceptable scope possibilities between quantifiers. However, our approach assumes an extension of the MRS formalism, which is not implemented in the systems in which LXGram runs.

- **Adjectives** We developed an analysis for adjectival attachment both for prenominal and postnominal adjective phrases (APs) that takes into account semantic interactions with other elements, like relative clauses. In the case of postnominal adjectives, the analysis that was implemented also captured the fact that adjectives following the head noun of an NP can be interspersed with several other NP elements, like PP complements of the noun:

(128) a. a irmã [$_{AP}$ mais velha] do Rui
the sister most old of the Rui
Rui's eldest sister

b. a irmã do Rui $[_{AP}$ mais velha]
the sister of the Rui most old
Rui's eldest sister

- **PP and AdvP modification of nouns** The attachment site of these elements was investigated in the context of the other NP constituents. An example follows.

(129) uma casa $[_{PP}$ com janelas azuis]
a house with windows blue
a house with blue windows

They cluster with elements like postnominal adjectives, PP complements and postnominal demonstratives in terms of word order.

The analysis that was implemented allows for the non-multiplication of lexical entries for prepositions, at the same time accounting for their different behavior when attaching to verbal projections. In particular, PPs and AdvP can attach to the left or right of VPs in Portuguese, but only to the right of nouns:

(130) a. * uma $[_{PP}$ com janelas azuis] casa$_{\overline{N}}$
a with windows blue house

b. Você $[_{VP}$ pode criar um Web site completo] $[_{PP}$ com um Mac.]
you can create a Web site complete with a Mac
You can create a complete Web site with a Mac.

c. Você $[_{PP}$ com um Mac] $[_{VP}$ pode criar um Web site completo.]
you with a Mac can create a Web site complete
You can create a complete Web site with a Mac.

The constraints on prepositions and adverbs that we used allow for a single lexical entry for all the uses of the preposition *com* (*with*) in these examples.

- **PP complements of nouns** We considered PP complements of relational nouns like the following:

(131) a irmã $[_{PP}$ do Rui]
the sister of the Rui
Rui's sister

We presented a system of constraints that allows several other postnominal constituents to intervene between the head noun and the PP complement in Portuguese or to surface after the PP. These elements include PP and AdvP adjuncts, postnominal demonstratives and adjectives.

- **Relative clause attachment** We used semantic considerations to propose that relative clauses (RCs) attach higher than prenominal adjectives but lower than cardinals and all other prenominals that precede cardinals, giving rise to structures like the following one:

(132) $[_{NP}$ as $[_{\overline{N}}$ duas $[_{\overline{N}}$ grandes guerras] $[_{RC}$ que abalaram o mundo]]]
the two great wars that shook the world
the two great wars that shook the world

The implementation of relative clauses and the other NP elements that we described produces such structures.

- **Postnominal demonstratives** We also considered postnominal demonstratives (although they are confined to some dialects of Portuguese), as in this example:

  (133) a. o filme esse$_{\text{DEM}}$
  the movie that
  that movie

  Our analysis predicts that word order is free between postnominal demonstratives and several other elements, like PP complements, PP and AdvP adjuncts and adjectives.

We provided a precise account of word order phenomena and constituency. Furthermore, co-occurrence restrictions between elements of these classes were also accounted for. For instance, prenominal possessives cannot appear in an NP with an indefinite article occupying the determiner slot, although it is acceptable if this position is filled by a definite article:

(134) a. a minha bicicleta
the my bicycle
my bicycle

b. * uma minha bicicleta
a my bicycle

We also described the way that several NP elements combine semantically with other elements.

We used a very general design to account for most of these phenomena (all except complements of nouns), presented in Chapter 3. This design is very clean, because arguments of predicates in the semantics are always selected syntactically by the head corresponding to these predicates, either via the valence features SUBJ and COMPS, where co-occurrence restrictions on a head's subject and complements are stated, or via the SELECT feature of functors. The only exception was our treatment of possessives realizing a noun's argument, which were implemented as functors selecting for the noun (instead of the other way around).

In Chapter 5 we focused on NPs lacking an overt head, like these examples (the first one is adapted from an English one in (Lobeck, 1995)):

(135) a. Provámos muitos vinhos, e achei que [$_{\text{NP}}$ alguns$_{\text{DET}}$] eram extremamente secos.
we tasted many wines, and i thought that some were extremely dry
We tasted many wines, and I thought that some were extremely dry.

b. [$_{\text{NP}}$ Os$_{\text{DET}}$ [$_{\text{AP}}$ muito ricos]] sempre abusaram d' [$_{\text{NP}}$ os$_{\text{DET}}$ [$_{\text{AP}}$ muito pobres.]]
the very rich always abused of the very poor
The very rich always abused the very poor.

We mentioned their syntactic peculiarities, as reported in the literature. For instance, although in (135a) a single determiner is present in the highlighted NP, not all determiners can be used in this context, viz. the definite articles of English and Portuguese cannot form an NP containing no other visible element.

A typology of phrases was developed that relates missing noun constructions to their counterparts that display no absent elements, taking advantage of multiple inheritance. We also sketched a very simple treatment of antecedent resolution, i.e. a method to recover the missing noun from contextual information, based on syntactic structures. An interesting result is that it does not seem possible in general to determine a unique possible antecedent without the help of non-grammatical knowledge (world knowledge seems to be required) — syntax can nevertheless reduce the number of candidates. Although this simple treatment is not implementable exactly as it was described using the systems under consideration (LKB and PET), it is nonetheless clearly computationally implementable.

The approach we followed to handle these NPs lacking a realized noun resorted to dedicated rules instead of positing an ε production, a nominal lexical entry with the empty string as its surface form (a null or empty noun). Even in the HPSG literature that is not directly linked to computational implementations, there is a line of research that seeks to eliminate empty categories altogether. It has been very successful in the treatment of unbounded dependencies, culminating in the work of Bouma *et al.* (2001). The solution developed in Chapter 5 also dispenses with null categories, and is equivalent to an analysis positing them. Although empty elements were not used, the analysis resorted to the empty category metaphor and accordingly shows many similarities with approaches employing these elements. In fact, we ended up defining a type that could be used in the lexical entry for the null noun, if we used it. This type was then employed in the phrases responsible for producing noun-headed projections lacking a realized head noun. Essentially, this means that an implementation avoiding null nouns does not have to be less modular than an implementation employing them: the interaction between general syntax rules properties and the characteristics specific to the empty noun can still be separated and factored out, as was done.

The analyses are implemented in a computational grammar for Portuguese currently in development at the University of Lisbon. Some of the analyses presented here are more detailed than the corresponding analyses of other, larger-scale computational HPSGs. As far as we know, no other DELPH-IN grammar currently contemplates the possibility of possessives realizing arguments of nouns or prevents multiple ordinal numerals from modifying the same noun, among other things. Our implementation of quantifier scope restrictions with indefinite specific NPs is also completely original, and no other computational grammar that we know of restricts the set of possible readings as much as was done here.

6.2 Evaluation

We present here the results obtained with [incr tsdb()] for a test suite containing NPs with the phenomena covered in this dissertation. The sentences that make up this test suite are presented in Appendix B. They are based on the examples mentioned throughout this dissertation. This test suite is composed of 186 items, 149 of which are grammatical examples.

For the experiment reported here, we used the LKB parser on a P4 3GHz machine with 1GB of RAM. We used several of the parser options concerning efficiency:

- **pre-unification filtering** (Malouf *et al.*, 2000)
  A list of the feature paths most likely to fail unification can be used, so that they can be tried first and unification failures can be detected early.

- **bidirectional parsing** (Oepen and Carroll, 2000)
  We can specify which daughter of a rule should be unified first, typically the one that is the most constrained, so that unification failures are detected early.

- **active parsing** (Oepen and Carroll, 2000)
  Dynamic programming techniques are used in order to avoid recomputations — the feature structures for all phrases that are produced are stored in memory —; with active parsing, feature structures for active edges — e.g. edges of binary rules where only one of the daughters has been instantiated — are also stored in memory.

We asked for all solutions (exhaustive search).

The following tables were produced automatically with [incr tsdb()]. They concern coverage, overgeneration and performance:

Coverage Profile

Phenomenon	**total items** ♯	**positive items** ♯	**word string** ϕ	**lexical items** ϕ	**distinct analyses** ϕ	**total results** ♯	**overall coverage** %
Total	**186**	**149**	**5.44**	**48.47**	**2.69**	**149**	**100.0**

(generated by [incr tsdb()] at 15-nov-2007 (16:09 h))

Overgeneration Profile

Phenomenon	**total items** ♯	**negative items** ♯	**word string** ϕ	**lexical items** ϕ	**distinct analyses** ϕ	**total results** ♯	**overall coverage** %
Total	**186**	**37**	**5.95**	**52.76**	**0.00**	**0**	**0.0**

(generated by [incr tsdb()] at 15-nov-2007 (16:10 h))

Performance Profile

Phenomenon	**items** ♯	**etasks** ϕ	**filter** %	**edges** ϕ	**first** ϕ (s)	**total** ϕ (s)	**space** ϕ (kb)
Total	**1860**	**729**	**95.9**	**214**	**0.22**	**0.21**	**20950**

(generated by [incr tsdb()] at 15-nov-2007 (16:10 h))

All the positive examples in this test suite received at least one parse (149 "total results" out of 149 "positive items" in the Coverage table). The average sentence received 2.69 parses (the column "distinct analyses ϕ" in the same table). The terminal symbols were highly ambiguous: there were around 50 possibilities (column "lexical items ϕ" in the tables Coverage and Overgeneration) for an average sentence length of 5 or 6 words (columns "word string ϕ" in the same two tables). None of the examples marked as ungrammatical was parsable, as desired (0 "total results" out of 37 "negative items" in the Overgeneration table).

The results in the Performance table are for the same test suite, but with every sentence repeated 10 times (there can be small fluctuations in the performance values between different runs with the same test suite and the same grammar, so we wanted to approximate the average values for 10 runs). The average space required to analyze each of these examples was a little over 20MB ("space ϕ" in the Performance table). On average, each sentence required 0.21 seconds to be processed (the column "total ϕ (s)" in the same table). This value is smaller than the one for obtaining the first reading (the column "first ϕ (s)" in the same table), because almost 20% of the examples in this test suite are ungrammatical and receive no parse.

6.3 Future Work

The analysis developed in this dissertation does not exhaust the topics of NP syntax and semantics. There are several other elements with particular syntax that are ignored here. For instance, an item like *outro* (*other*) cannot be treated as a regular adjective even when it follows a determiner, because, unlike adjectives, it can precede cardinals — cf. (29b) and (136) —, and it cannot iterate (137).

(136) Os adeptos entusiasmaram-se depois de [$_{NP}$ outras duas vitórias do clube.]
the fans got excited after other two victories of the club
The fans got excited after other two victories of their club.

(137) a. [Os grandes, grandes filmes desse realizador] eram assim também
the great great films by that director were like that too
The great great films by that director were like that, too.

b. * [$_{NP}$ Os outros outros filmes desse realizador] eram assim também.
the other other filmes by that director were like that too

It cannot be classified in any of the other categories in the table in Appendix A either, because it displays behavior different from all of them. We ignore this item *outro* as well as some other elements with particular syntax, like *cada* (*each*), *tal* (*such*). Instead we focused on the classes that were presented in the table in Appendix A.

We did not say anything about proper names and their combinatorial potential either, as they are often mentioned in the literature. Similarly, there are other issues that are analyzed in the literature that we did not discuss here. For instance, prenominal adjectives can surface after the noun when they are modified or coordinated, as reported and analyzed in (Abeillé and Godard, 1999)): cf. *un grand avantage* vs. *un avantage plus grand*.

Appositive modification was not taken into account, either. It is often assumed that appositive modifiers occupy a position more peripheral than any of the positions under consideration, so it should not interact with these data. It is claimed that appositive modifiers are more peripheral than restrictive modifiers, because appositive modifiers modify entire NPs, whereas restrictive modifiers are NP internal. For instance, appositive relative clauses can attach to personal pronouns (138b) — which are assumed to be full NPs —, whereas restrictive relative clauses cannot (138d). Therefore we consider appositive modifiers to attach to full NPs, unlike all other elements that were discussed in this dissertation. The bracketing in the following examples reflects such an analysis.

(138) a. Ali podiam olhar para [$_{NP}$ [$_{NP}$ os barcos,] [$_{RelClause}$ que tinham mastros verdes.]]
there they could look at the boats, which had masts green
There they could look at the boats, which had green masts.

b. Ali podiam olhar para [$_{NP}$ [$_{NP}$ eles,] [$_{RelClause}$ que tinham mastros verdes.]]
there they could look at them, which had masts green
There they could look at them, which had green masts.

c. Ali podiam olhar para [$_{NP}$ os [$_{\overline{N}}$ [$_{\overline{N}}$ barcos] [$_{RelClause}$ que tinham mastros verdes.]]] Os outros barcos tinham mastros azuis.
there they could look at the boats that had masts green. the other boats had masts blue
There they could look at the boats that had green masts. The other boats had blue masts.

d. * Ali podiam olhar para eles que tinham mastros verdes. Os outros barcos tinham mastros azuis.
there they could look at them that had masts green, the other boats had masts blue

Our treatment of the semantics of possessives is not exhaustive, as sometimes possessives realize arguments of adjectives (the following example is adapted from (Partee, 1983)):

(139) o meu filme favorito
the my movie favorite
my favorite movie

An NP like this could receive a representation not very different from the one given to *the movie favored by me*.

Also, there are scopal interactions with non-intersective adjectives. For instance, an NP like *John's former car* or *his former car* can refer to a car that was once John's or to something that is John's but is no longer a car (e.g. it is scrap now): roughly the sole entity in the set $FORMER(HIS) \cap CAR$ or in $HIS \cap FORMER(CAR)$. The semantics $FORMER(HIS \cap CAR)$ does not capture both possibilities because the NP can describe scrap that John acquired when it was no longer a car, so it was never "his car".

Note that the analyses we presented produce $HIS \cap FORMER(CAR)$ with prenominal possessives and $FORMER(HIS \cap CAR)$ with postnominal ones, but word order does not seem to have any consequence for the semantics here.

Throughout this text, we mentioned some other phenomena that also fall outside the scope of this dissertation. We did not contemplate the possibility of extraposed complements of prenominal adjectives (see (19) on p. 49). We ignored the problem of the interaction between scopal and intersective modifiers and instead chose to present a simplified mechanism for the composition of semantics (see the discussion in Section 3.4.2). We also chose not to address the composition of semantics with postnominal universal quantifiers (see the example in (90) on p. 120). In the section about complements of nouns, we focused only on nouns that subcategorize for a single PP complement (Section 4.8). There are many other subcategorization frames for nouns, and we could not cover them all here. The analysis presented for modifying and argumental possessives overgenerates for expressions like *seu pai* (*his father*), as we do not block the modifier version of this possessive from adjoining to a relational noun like *pai* (*father*): see Section 4.9.1. We also do not allow relative clauses to precede sentential complements of nouns (example (88) in Section 4.13).

We leave these issues for future work.

Appendices

A Positions within the Noun Phrase

	I	II	III	IV	V	VI	VII	VIII	IX
(a)		a	minha	primeira		bicicleta		com pedais amarelos	
		the	*my*	*first*		*bicycle*		*with yellow pedals*	
	my first bicycle with yellow pedals								
(b)	todas	aquelas		três mil		pessoas		ali	
	all	*those*		*three thousand*		*people*		*over there*	
	all those three thousand people over there								
(c)		um		certo	grande	espírito			que criou o mundo
		a		*certain*	*great*	*spirit*			*that created the world*
	a certain great spirit that created the world								
(d)		a				invasão	americana	do Iraque	
		the				*invasion*	*American*	*of the Iraq*	
	the American invasion of Iraq								
(e)				quatro		colegas		teus	
				four		*collegues*		*of yours*	
	four collegues of yours								
(f)		a				pesca	baleeira	intensa	
		the				*fishing*	*whale-like*	*intense*	
	the intense whale fishing								
(g)		aquelas	suas	muitas		queixas			
		those	*their*	*many*		*complaints*			
	those many complaints of theirs								
(h)		o				papa		esse	que é tão snob
		the				*pope*		*that*	*who is such a snob*
	that pope who is such a snob								

Positions within the Noun Phrase:

- I — Predeterminers;
- II — Determiners;
- III — Prenominal Possessives;
- IV — Cardinals (b) (e), Ordinals (a), Vague Quantifiers (g), Markers of Indefinite Specifics (c);
- V — Prenominal Adjective Phrases;
- VI — Head Noun;
- VII — Adjectival Arguments;
- VIII — Adjective Phrase Adjuncts (f), Prepositional Phrase Arguments (d), Prepositional Phrase Adjuncts (a), Adverbial Phrase Adjuncts (b), Postnominal Possessives (e), Postnominal Demonstratives (h);
- IX — Restrictive Relative Clauses.

B Test Suite

Negative (i.e. ungrammatical) examples are preceded by a star ("*").

1

Eles	avariaram.
they	broke down

They broke down.

2

Esses	carros	avariaram.
those	cars	broke down

Those cars broke down.

3

Os	meus	carros	avariaram.
the	my	cars	broke down

My cars broke down.

4

Aqueles	meus	dois	carros	avariaram.
those	my	two	cars	broke down

Those two cars of mine broke down.

5

Chegou	um	falso	médico	chinês.
arrived	a	fake	doctor	Chinese

A fake Chinese doctor arrived.

6

Chegou	um	falso	médico	que	é	chinês.
arrived	a	fake	doctor	who	is	Chinese

A fake doctor who is Chinese arrived.

7

O	meu	carro	está	na	oficina.
the	my	car	is	at the	mechanic's

My car is at the mechanic's.

8

Todos	os	homens	são	mortais.
all	the	men	are	mortal

All men are mortal.

9

Chegou	a	tua	encomenda.
has arrived	the	your	order

Your order has arrived.

10

Chegou	uma	encomenda	tua.
has arrived	a	order	your/yours

An order of yours has arrived.

11

Chegaram	as	duas	encomendas.
have arrived	the	two	orders

The two orders have arrived.

12	Chegaram	duas	encomendas.
	have arrived	two	orders

Two orders have arrived.

13	O	primeiro	lugar	está	vago.
	the	first	seat	is	free

The first seat is free.

14	Todos	os	homens	leram	um	certo	livro.
	all	the	men	have read	a	certain	book

All men have read a certain book.

15	Todos	os	homens	batem	num	pobre	burro.
	all	the	men	beat up	on a	poor	donkey

All men beat up a poor donkey.

16	Todos	os	homens	batem	num	burro	cinzento.
	all	the	men	beat up	on a	donkey	gray

All men beat up a gray donkey.

17	O	pai	do	Rui	chegou	ontem.
	the	father	of the	Rui	arrived	yesterday

Rui’s father arrived yesterday.

18	Era	um	cão	com	três	pernas.
	it was	a	dog	with	three	legs

It was a dog with three legs.

19	Os	que	chegarem	primeiro	esperam.
	the	who	arrive	first	wait

The ones who arrive first wait.

20	Não	existem	com	cinco.
	not	they exist	with	five

There are none with five.

21	Isso	sai	com	benzina.
	that	goes away	with	benzine

That goes away with benzine.

22	Isso	com	benzina	sai.
	that	with	benzine	goes away

That goes away with benzine.

23	Era	um	chapéu	com	uma	antena.
	it was	a	hat	with	a	antenna

It was a hat with an antenna.

24 *

Isso	era	um	com	uma	antena	chapéu
that	was	a	with	a	antenna	hat

25

São	os	quatro	naipes.
they are	the	four	suites

It's the four suites.

26

Muitas	espécies	de	sapos	da	Amazónia	já	estão	extintas.
many	species	of	frogs	of the	Amazon Rainforest	already	are	extinct

Many species of frogs of the Amazon Rainforest are already extinct.

27

Bastantes	espécies	de	sapos	da	Amazónia	já	estão	extintas.
several	species	of	frogs	of the	Amazon Rainforest	already	are	extinct

Several species of frogs of the Amazon Rainforest are already extinct.

28

As	muitas	espécies	de	sapos	da	Amazónia	já	estão	extintas.
the	many	species	of	frogs	of the	Amazon Rainforest	already	are	extinct

The many species of frogs of the Amazon Rainforest are already extinct.

29 *

As	bastantes	espécies	de	sapos	da	Amazónia	já	estão	extintas.
the	several	species	of	frogs	of the	Amazon Rainforest	already	are	extinct

30

Todas	as	pessoas	leram	um	certo	livro.
all	the	people	have read	a	certain	book

All people have read a certain book.

31

Todas	as	pessoas	leram	um	livro.
all	the	people	have read	a	book

All people have read a book.

32

Foi	a	invasão	americana	do	Iraque.
it was	the	invasion	American	of the	Iraq

It was the American invasion of Iraq.

33

Os	primeiros	dois	filmes	passaram	aqui.
the	first	two	movies	were shown	here

The first two movies were shown here.

34

Os	dois	primeiros	filmes	passaram	aqui.
the	two	first	movies	were shown	here

The two first movies were shown here.

35

Os	adeptos	sentiram	entusiasmo	depois	de	duas	grandes	vitórias	do	clube.
the	fans	felt	enthusiasm	after	of	two	great	victories	of the	club

The fans were excited after two great victories of their club.

36 *

Os	adeptos	sentiram	entusiasmo	depois	de	grandes	duas	vitórias	do	clube.
the	fans	felt	enthusiasm	after	of	great	two	victories	of the	club

37		Os	seres	humanos	são	livres.
		the	human	beings	are	free

Human beings are free.

38		Todos	os	seres	humanos	são	livres.
		all	the	human	beings	are	free

All human beings are free.

39		Todas	as	pessoas	são	livres.
		all	the	people	are	free

All people are free.

40		Todas	aquelas	pessoas	são	livres.
		all	those	people	are	free

All those people are free.

41		Todas	pessoas	são	livres.
		all	people	are	free

All people are free.

42		As	pessoas	todas	são	livres.
		the	people	all	are	free

All people are free.

43		Chegou	um	falso	médico	chinês.
		arrived	a	fake	doctor	Chinese

A fake Chinese doctor arrived.

44		Atacaram	um	mero	inspector.
		they attacked	a	mere	inspector

They attacked a mere inspector.

45	*	Atacaram	um	inspector	mero.
		they attacked	a	inspector	mere

46	*	Atacaram	um	japonês	inspector.
		they attacked	a	Japanese	inspector

47		Atacaram	um	inspector	japonês.
		they attacked	a	inspector	Japanese

They attacked a Japanese inspector.

48		Atacaram	um	falso	inspector.
		they attacked	a	fake	inspector

They attacked a fake inspector.

49		Atacaram	um	inspector	falso.
		they attacked	a	inspector	fake

They attacked a fake inspector.

50		Era	um	grande,	grande	filme.
		it was	a	great	great	movie

It was a great, great movie.

51		Era	um	filme	chato,	chato.
		it was	a	movie	boring	boring

It was a boring, boring movie.

52		Viram	a	alunagem	americana	na	televisão.
		they saw	the	moon landing	American	on the	television

They saw the American moon landing on television.

53		Viram	a	invasão	americana	do	Iraque.
		they saw	the	invasion	American	of the	Iraq

They saw the American invasion of Iraq.

54	*	Viram	a	invasão	do	Iraque	americana.
		they saw	the	invasion	of the	Iraq	American

55		Viram	a	alunagem	americana	de	1969.
		they saw	the	moon landing	American	of	1969

They saw the American moon landing of 1969.

56	*	Viram	a	alunagem	de	1969	americana.
		they saw	the	moon landing	of	1969	American

57	*	Viram	a	invasão	americana	iraquiana.
		they saw	the	invasion	American	Iraqi

58		Viram	o	consumo	galopante	de	petróleo.
		they saw	the	consumption	ever increasing	of	oil

They saw the ever increasing consumption of oil.

59		Viram	o	consumo	de	petróleo	galopante.
		they saw	the	consumption	of	oil	ever increasing

They saw the ever increasing consumption of oil.

60		Viram	o	consumo	de	petróleo	que	continua	a	crescer.
		they saw	the	consumption	of	oil	that	keeps	on	growing

They saw the consumption of oil that keeps increasing.

61		A	minha	bicicleta	tem	um	pneu	vermelho.
		the	my	bicycle	has	a	tire	red

My bicycle has a red tire.

62		Uma	bicicleta	minha	tem	um	pneu	vermelho.
		a	bicycle	my/mine	has	a	tire	red

A bicycle of mine has a red tire.

63		Aquela	tua	bicicleta	tem	um	pneu	vermelho.
		that	your	bicycle	has	a	tire	red

That bicycle of yours has a red tire.

64		Aquela	bicicleta	tua	tem	um	pneu	vermelho.
		that	bicycle	your/yours	has	a	tire	red

That bicycle of yours has a red tire.

65		Ele	é	teu	irmão?
		he	is	your	brother

Is he your brother?

66		As	minhas	duas	bicicletas	estão	aqui.
		the	my	two	bicycles	are	here

My two bicycles are here.

67	*	Minhas	as	duas	bicicletas	estão	aqui.
		my	the	two	bicycles	are	here

68	*	As	duas	minhas	bicicletas	estão	aqui.
		the	two	my	bicycles	are	here

69		Ele	é	pianista.
		he	is	pianist

He is a pianist.

70	*	Ele	viu	pianista.
		he	saw	pianist

71		Minha	bicicleta	tem	um	pneu	vermelho.
		my	bicycle	has	a	tire	red

My bicycle has a red tire.

72		O	irmão	da	Ana	está	aqui.
		the	brother	of the	Ana	is	here

Ana's brother is here.

73		O	seu	irmão	está	aqui.
		the	her	brother	is	here

Her brother is here.

74	*	O	seu	seu	irmão	está	aqui.
		the	her	her	brother	is	here

75		Os	meus	dois	irmãos	estão	aqui.
		the	my	two	brothers	are	here

My two brothers are here.

76		O	teu	livro	está	aqui.		
		the	your	book	is	here		
		Your book is here.						
77		Os	primeiros	dois	capítulos	são	cómicos.	
		the	first	two	chapters	are	funny	
		The first two chapters are funny.						
78		Os	dois	primeiros	capítulos	são	cómicos.	
		the	two	first	chapters	are	funny	
		The two first chapters are funny.						
79		Um	certo	primeiro	capítulo	é	cómico.	
		a	certain	first	chapter	is	funny	
		A certain first chapter is funny.						
80	*	Um	primeiro	certo	capítulo	é	cómico.	
		a	first	certain	chapter	is	funny	
81		Dois	certos	capítulos	são	cómicos.		
		two	certain	chapters	are	funny		
		Two certain chapters are funny.						
82		Certos	dois	capítulos	são	cómicos.		
		certain	two	chapters	are	funny		
		Two certain chapters are funny.						
83	*	Os	dois	três	carros	avariaram.		
		the	two	three	cars	broke down		
84	*	O	segundo	primeiro	lugar	está	vago.	
		the	second	first	seat	is	free	
85		Um	certo	carro	avariou.			
		a	certain	car	broke down			
		A certain car broke down.						
86		Um	determinado	carro	avariou.			
		a	certain	car	broke down			
		A certain car broke down.						
87	*	Um	determinado	certo	carro	avariou.		
		a	certain	certain	car	broke down		
88	*	Os	dois	primeiros	três	lugares	estão	vagos.
		the	two	first	three	seats	are	free
89	*	Os	primeiros	dois	segundos	pratos	estão	aqui.
		the	first	two	second	dishes	are	here

90	*	Certos	dois	certos	carros	avariaram.
		certain	two	certain	cars	broke down

91		Os	vários	participantes	passeiam	as	folhas	pela	sala.
		the	several	participants	walk	the	paper sheets	around the	room

The several participants walk the paper sheets around the room.

92	*	Os	vários	vinte	participantes	estão	aqui.
		the	several	twenty	participants	are	here

93	*	Os	vinte	vários	participantes	estão	aqui.
		the	twenty	several	participants	are	here

94		Os	vários	primeiros	lugares	estão	aqui.
		the	several	first	seats	are	here

The several first seats are here.

95		Os	primeiros	vários	lugares	estão	aqui.
		the	first	several	seats	are	here

The first several seats are here.

96		Viu	vários	certos	participantes.
		he saw	several	certain	participants

He saw certain several participants.

97		Viu	certos	vários	participantes.
		he saw	certain	several	participants

He saw certain several participants.

98	*	Viu	os	vários	vários	participantes.
		he saw	the	several	several	participants

99	*	Viu	os	vários	vinte	vários	participantes.
		he saw	the	several	twenty	several	participants

100	*	Verá	os	próximos	primeiros	capítulos.
		he wil see	the	next	first	chapters

101	*	Verá	os	primeiros	próximos	capítulos.
		he wil see	the	first	next	chapters

102		Verá	os	três	próximos	capítulos.
		he wil see	the	three	next	chapters

He'll see the three next chapters.

103		Verá	os	próximos	três	capítulos.
		he wil see	the	next	three	chapters

He'll see the next three chapters.

104		Verá	os	dois	melhores	capítulos.
		he wil see	the	two	best	chapters

He'll see the two best chapters.

105	*	Verá	os	melhores	dois	capítulos.
		he wil see	the	best	two	chapters

106		Todas	as	pessoas	leram	um	certo	livro.
		all	the	people	have read	a	certain	book

All people have read a certain book.

107		Todas	as	pessoas	leram	um	livro.
		all	the	people	have read	a	book

All people have read a book.

108	*	Todos	os	determinados	homens	leram	um	livro.
		all	the	certain	men	have read	a	book

109	*	Os	determinados	homens	leram	um	livro.
		the	certain	men	have read	a	book

110	*	Esses	determinados	homens	leram	um	livro.
		those	certain	men	have read	a	book

111		Todos	os	filhos	da	Ana	leram	um	certo	livro.
		all	the	children	of the	Ana	have read	a	certain	book

All of Ana's children have read a certain book.

112		Estão	aqui	dois	certos	capítulos.
		are	here	two	certain	chapters

Two certain chapters are here.

113		Estão	aqui	certos	dois	capítulos.
		are	here	certain	two	chapters

Two certain chapters are here.

114		Está	aqui	um	DVD	com	dois	primeiros	episódios	dessa	série.
		is	here	a	DVD	with	two	first	episodes	of that	show

A DVD with two first episodes of that show is here.

115	*	Está	aqui	um	DVD	com	primeiros	dois	episódios	dessa	série.
		is	here	a	DVD	with	first	two	episodes	of that	show

116		Algumas	cartas	chegaram.
		some	letters	have arrived

Some letters have arrived.

117

Duas	cartas	chegaram.
two	letters	have arrived

Two letters have arrived.

118

João	não	viu	uma	mancha	no	chão.
João	not	saw	a	spot	on the	floor

João didn't see a spot on the floor.

119

João	não	viu	manchas	no	chão.
João	not	saw	spots	on the	floor

João didn't see spots on the floor.

120

Todas	as	pessoas	leram	um	livro	sobre	girafas.
all	the	people	have read	a	book	on	giraffes

All people have read a book on giraffes.

121

Todas	as	pessoas	leram	livros	sobre	girafas.
all	the	people	have read	books	on	giraffes

All people have read books on giraffes.

122

Pedro	quer	encontrar	um	policial.
Pedro	wants	to find	a	police officer

Pedro wants to find a police officer.

123

Pedro	quer	encontrar	policiais.
Pedro	wants	to find	police officers

Pedro wants to find police officers.

124

João	não	viu	duas	manchas	no	chão.
João	not	saw	two	spots	on the	floor

João didn't see two spots on the floor.

125

O	João	não	viu	certa	mancha	no	chão.
the	João	not	saw	certain	spot	on the	floor

João didn't see a certain spot on the floor.

126

Todas	as	pessoas	leram	certo	livro	sobre	girafas.
all	the	people	have read	certain	book	on	giraffes

All people have read a certain book on giraffes.

127

Pedro	quer	encontrar	certo	polícia.
Pedro	wants	to find	certain	police officer

Pedro wants to find a certain police officer.

128

Aquele	carro	ali	estava	aqui	ontem.
that	car	there	was	here	yesterday

That car over there was here yesterday.

129	*	Aquele	ali	carro	estava	aqui	ontem.
		that	there	car	was	here	yesterday

130	Vi	carros	sem	assentos	vermelhos.	
	I saw	cars	without	seats	red	

I saw cars with no red seats / I saw red cars with no seats.

131	Vi	os	dois	carros	da	Ana.
	I saw	the	two	cars	of the	Ana

I saw Ana's two cars.

132	Saíram	com	a	Ana.
	they left	with	the	Ana

They left with Ana.

133	Chegou	um	falso	médico	que	é	Chinês.
	has arrived	a	fake	doctor	that	is	Chinese

A fake doctor that is Chinese has arrived.

134	Todos	os	exactamente	três	filmes	que	lá	vi	eram	maus.
	all	the	exactly	three	movies	that	there	I saw	were	bad

All the exactly three movies that I saw there were bad.

135	A	bicicleta	essa	é	verde.
	the	bicycle	that	is	green

That bicycle is green.

136	Chegaram	várias	cartas	tuas.
	have arrived	several	letters	your/yours

Several letters of yours have arrived.

137	Desapareceram	as	cartas	todas.
	have disappeared	the	letters	all

All the letters have disappeared.

138	*	Uma	bicicleta	essa	é	verde.
		a	bicycle	that	is	green

139	*	Essa	bicicleta	essa	é	verde.
		that	bicycle	that	is	green

140	*	Esta	bicicleta	essa	é	verde.
		this	bicycle	that	is	green

141	*	A	bicicleta	essa	essa	é	verde.
		the	bicycle	that	that	is	green

142

Esta	bicicleta	aqui	é	verde.
this	bicycle	here	is	green

This bicycle over here is green.

143

Essa	bicicleta	aí	é	verde.
that	bicycle	there	is	green

That bicycle over there is green.

144

Aquela	bicicleta	ali	é	verde.
that	bicycle	there	is	green

That bicycle over there is green.

145

O	carro	esse	é	verde.
the	car	that	is	green

That car is green.

146

Esse	carro	é	verde.
that	car	is	green

That car is green.

147

Todos	esses	carros	avariaram.
all	those	cars	broke down

All those cars broke down.

148

Vi	a	casa	azul	e	a	verde.
I saw	the	house	blue	and	the	green

I saw the blue house and the green one.

149

Vi	algumas	crianças	com	chapéus	e	algumas	com	bonés.
I saw	some	children	with	hats	and	some	with	caps

I saw some children in hats and some in caps.

150

Vi	os	pobres.
I saw	the	poor

I saw the poor / I saw the poor ones.

151

Vi	os	dois.
I saw	the	two

I saw the two.

152

Vi	os	sem	abrigo.
I saw	the	without	shelter

I saw the homeless.

153

Os	que	podem	ajudar	nunca	ajudam.
the	who	can	help	never	help

The ones who can help never do so.

154	Vi	os	homens	bastante	velhos	e	os	especialmente	novos.
	I saw	the	men	quite	old	and	the	specially	young

I saw the quite old men and the specially young ones.

155	Vi	alguns.
	I saw	some

I saw some.

156	Vi	os	seus	dois.
	I saw	the	his	two

I saw his two.

157	Vi	a	verde.
	I saw	the	green

I saw the green one.

158	Vi	alguns	jovens	com	chapéus.
	I saw	some	young	with	hats

I saw some young ones in hats/ I saw some young people in hats.

159	Comprei	maçãs.
	I bought	apples

I bought apples.

160	Todas	estavam	podres.
	all	were	rotten

All were rotten.

161	Todos	são	livres.
	all	are	free

All are free.

162	As	pessoas	chegaram.
	the	people	have arrived

The people have arrived.

163	Aquelas	pessoas	chegaram.
	those	people	have arrived

Those people have arrived.

164	Todas	as	pessoas	chegaram.
	all	the	people	have arrived

All the people have arrived.

165	Todas	aquelas	pessoas	chegaram.
	all	those	people	have arrived

All those people have arrived.

166 A minha irmã está aqui.
the my sister is here
My sister is here.

167 Uma irmã minha está aqui.
a sister my/mine is here
A sister of mine is here.

168 A minha bicicleta está aqui.
the my bicycle is here
My bicycle is here.

169 Os dois carros avariaram.
the two cars broke down
The two cars broke down.

170 Dois carros avariaram.
two cars broke down
Two cars broke down.

171 Os dois primeiros capítulos estão aqui.
the two first chapters are here
The two first chapters are here.

172 Os primeiros dois capítulos estão aqui.
the first two chapters are here
The first two chapters are here.

173 Certos dois carros avariaram.
certain two cars broke down
Two certain cars broke down.

174 Dois certos carros avariaram.
two certain cars broke down
Two certain cars broke down.

175 Todos os homens leram um livro.
all the men have read a book
All men have read a book.

176 Todos os homens leram um certo livro.
all the men have read a certain book
All men have read a certain book.

177 A irmã mais velha do Rui chegou.
the sister most old of the Rui has arrived
Rui's eldest sister has arrived.

178

A	irmã	do	Rui	mais	velha	chegou.
the	sister	of the	Rui	most	old	has arrived

Rui's eldest sister has arrived.

179

Mora	numa	casa	com	janelas	azuis.
he lives	in a	house	with	blue	windows

He lives in a house with blue windows.

180 *

Mora	numa	com	janelas	azuis	casa.
he lives	in a	with	blue	windows	house

181

As	duas	grandes	guerras	que	abalaram	o	mundo	foram	más.
the	two	great	wars	that	shook	the	world	were	bad

The two great wars that shook the world were bad.

182

O	filme	esse	é	mau.
the	movie	that	is	bad

That movie is bad.

183

A	minha	bicicleta	é	azul.
the	my	bicycle	is	blue

My bicycle is blue.

184 *

Uma	minha	bicicleta	é	azul.
a	my	bicycle	is	blue

185

Alguns	eram	extremamente	secos.
some	were	extremely	dry

Some were extremely dry.

186

Os	muito	ricos	sempre	abusaram	dos	muito	pobres.
the	very	rich	always	have abused	of the	very	poor

The very rich have always abused the very poor.

Bibliography

ABEILLÉ, ANNE AND DANIÈLE GODARD, 1999. La Position de l'Adjectif Épithète en Français : le Poids des Mots. In *Recherches linguistiques de Vincennes*, volume 28, pages 9–31.

ALLEGRANZA, VALERIO, 1998a. Determination and Quantification. In FRANK VAN EYNDE AND PAUL SCHMIDT, editors, *Linguistic Specifications for Typed Feature Structure Formalisms*, pages 281–314. Office for Official Publications of the European Communities, Luxembourg.

ALLEGRANZA, VALERIO, 1998b. Determiners as Functors: NP Structure in Italian. In SERGIO BALARI AND LUCA DINI, editors, *Romance in Head-driven Phrase Structure Grammar*, volume 75 of *CSLI Lecture Notes*, pages 55–108. CSLI Publications, Stanford.

ALSHAWI, HIYAN AND RICHARD S. CROUCH, 1992. Monotonic Semantic Interpretation. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL-92)*, pages 32–39. Newark, NJ.

ANTÓNIO, BRANCO AND COSTA FRANCISCO, 2007. Self- or Pre-Tuning? Deep Linguistic Processing of Language Variants. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 57–64. Association for Computational Linguistics, Prague, Czech Republic.

ARNOLD, KEN, JAMES GOSLING AND DAVID HOLMES, 2005. *The Java Programming Language*. Addison-Wesley Professional.

BAR-HILLEL, YEHOSHUA, M. PERLES AND E. SHAMIR, 1961. On Formal Properties of Simple Phrase Structure Grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172.

BARWISE, JON AND ROBIN COOPER, 1981. Generalized Quantifiers and Natural Language. *Linguistics and Philosophy*, 4(1):159–219.

BEAVERS, JOHN, 2003a. Heads and Categories: Finding the Nominal Chimera. Manuscript.

BEAVERS, JOHN, 2003b. More Heads and Less Categories: A New Look at Noun Phrase Structure. In STEFAN MÜLLER, editor, *Proceedings of the HPSG-2003 Conference, Michigan State University, East Lansing*, pages 47–67. CSLI Publications, Stanford.

BENDER, EMILY M., DAN FLICKINGER AND STEPHAN OEPEN, 2002. The Grammar Matrix: An Open-Source Starter-Kit for the Development of Cross-Linguistically Consistent Broad-Coverage Precision Grammars. In JOHN CARROLL, NELLEKE OOSTDIJK AND RICHARD SUTCLIFFE, editors, *Procedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14. Taipei, Taiwan.

BOND, FRANCIS, STEPHAN OEPEN, MELANIE SIEGEL, ANN COPESTAKE AND DAN FLICKINGER, 2005. Open Source Machine Translation with DELPH-IN. In *Proceedings of the Open-Source Machine Translation Workshop at the 10th Machine Translation Summit*, pages 15–22. Phuket, Thailand.

BOS, JOHAN, 1996. Predicate Logic Unplugged. In *Proceedings of the 10th Amsterdam Colloquium*, pages 133–143.

BOUMA, GOSSE, ROB MALOUF AND IVAN SAG, 2001. Satisfying Constraints on Extraction and Adjunction. *Natural Language and Linguistic Theory*, 1(19).

BRANCO, ANTÓNIO, 1999. *Reference Processing and its Universal Constraints*. Ph.D. thesis, Universidade de Lisboa, Lisbon.

BRANCO, ANTÓNIO, 2002. Binding Machines. *Computational Linguistics*, 28:1–18.

BRANCO, ANTÓNIO AND FRANCISCO COSTA, 2006. Noun Ellipsis without Empty Categories. In STEFAN MÜLLER, editor, *The Proceedings of the 13th International Conference on Head-Driven Phrase Structure Grammar*, pages 81–101. CSLI Publications, Stanford.

BRANCO, ANTÓNIO AND FRANCISCO COSTA, 2007a. Accommodating Language Variation in Deep Processing. In TRACY HOLLOWAY KING AND EMILY M. BENDER, editors, *Proceedings of the GEAF07 Workshop*, pages 67–86. CSLI, Stanford, CA.

BRANCO, ANTÓNIO AND FRANCISCO COSTA, 2007b. Identification and Handling of Dialectal Variation with a Single Grammar. In PETER DIRIX, INEKE SCHUURMAN, VINCENT VANDEGHINSTE AND FRANK VAN EYNDE, editors, *Proceedings of the 17th Meeting of Computational Linguistics in the Netherlands (CLIN17)*, pages 5–19. LOT, Utrecht.

CALLMEIER, ULRICH, 2000. PET — A Platform for Experimentation with Efficient HPSG Processing Techniques. *Natural Language Engineering*, 6(1):99–108. (Special Issue on Efficient Processing with HPSG).

CARPENTER, BOB, 1992. *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science 32. Cambridge University Press, Cambridge, Massachusetts, USA.

CARROLL, JOHN, ANN COPESTAKE, DAN FLICKINGER AND VICTOR POZNAŃSKI, 1999. An Efficient Chart Generator for (Semi-)Lexicalist Grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*, pages 86–95. Toulouse.

CARROLL, JOHN AND STEPHAN OEPEN, 2005. High Efficiency Realization for a Wide-Coverage Unification Grammar. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP05)*. Springer Verlag.

CHIERCHIA, GENNARO AND SALLY MCCONNELL-GINET, 1990. *Meaning and Grammar: An Introduction to Semantics*. MIT Press, Cambridge, MA.

CHOMSKY, NOAM, 1957. *Syntactic Structures*. Mouton, The Hague, The Netherlands.

COPESTAKE, ANN, 2000. Appendix: Definitions of Typed Feature Structures. *Natural Language Engineering*, 6(1):109–112. (Special Issue on Efficient Processing with HPSG).

COPESTAKE, ANN, 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford, California.

COPESTAKE, ANN AND DAN FLICKINGER, 2000. An Open-Source Grammar Development Environment and Broad-Coverage English Grammar Using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*. Athens, Greece.

COPESTAKE, ANN, DAN FLICKINGER, IVAN A. SAG AND CARL POLLARD, 2005. Minimal Recursion Semantics: An Introduction. *Journal of Research on Language and Computation*, 3(2–3):281–332.

COVINGTON, MICHAEL A., 1994. *Natural Language Processing for Prolog Programmers*. Prentice-Hall, Englewood Cliffs, New Jersey.

DAVIDSON, DONALD, 1980. *Essays on Actions and Events*. Oxford University Press, New York.

DE SWART, HENRIËTTE, 1998. *Introduction to Natural Language Semantics*. CSLI Publications, Stanford.

DOWTY, DAVID, ROBERT WALL AND STANLEY PETERS, 1981. *Introduction to Montague Semantics*. D. Reidel, Dordrecht.

EGG, MARKUS, ALEXANDER KOLLER AND JOACHIM NIEHREN, 2001. The Constraint Language for Lambda Structures. *Journal of Logic, Language and Information*, 10:457–485.

FERREIRA, EDUARDO, JOÃO BALSA AND ANTÓNIO BRANCO, 2007. Combining Rule-based and Statistical Methods for Named Entity Recognition in Portuguese. In *Actas do V Workshop em Tecnologia da Informação e da Linguagem Humana TIL*.

FLICKINGER, DAN, 2000. On Building a More Efficient Grammar by Exploiting Types. *Natural Language Engineering*, 6(1):15–28. (Special Issue on Efficient Processing with HPSG).

GINZBURG, JONATHAN AND IVAN A. SAG, 2000. *Interrogative Investigations: the Form, Meaning, and Use of English Interrogatives*. CSLI Publications, Stanford, California.

HANKAMER, JORGE AND IVAN SAG, 1976. Deep and Surface Anaphora. *Linguistic Inquiry*, 7(3):391–426.

IONIN, TANIA AND ORA MATUSHANSKY, 2006. The Composition of Complex Cardinals. *Journal of Semantics*, 23:315–360.

KAMP, HAND AND UWE REYLE, 1993. *From Discourse to Logic: An Introduction to Modeltheoretic Semantics, Formal Logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht, Germany.

KASPER, ROBERT T., 1996. The Semantics of Recursive Modification. Manuscript.

KIEFER, BERND, HANS-ULRICH KRIEGER, JOHN CARROL AND ROB MALOUF, 1999. A Bag of Useful Techniques for Efficient and Robust Parsing. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics*, pages 473–480. College Park, MD.

KRIEGER, HANS-ULRICH AND ULRICH SCHÄFER, 1994. $\mathcal{TDL}$ — A Type Description Language for Constraint-Based Grammars. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 893–899. Kyoto, Japan.

LOBECK, ANNE, 1995. *Ellipsis – Functional Heads, Licensing, and Identification*. Oxford University Press, New York, Oxford.

MALOUF, ROBERT, JOHN CARROL AND ANN COPESTAKE, 2000. Efficient Feature Structure Operations without Compilation. *Natural Language Engineering*, 6(1):29–46. (Special Issue on Efficient Processing with HPSG).

MASULLO, JOSÉ PASCUAL, 1999. Variable vs. Intrinsic Features in Spanish Nominal Ellipsis.

MELNIK, NURIT, 2005. From "hand-written" to Computationally Implemented HPSG Theories. In STEFAN MÜLLER, editor, *The Proceedings of the 12th International Conference on Head-Driven Phrase Structure Grammar, Department of Informatics, University of Lisbon*, pages 311–321. CSLI Publications, Stanford.

MONTAGUE, RICHARD, 1974. The Proper Treatment of Quantification in Ordinary English. *Formal Philosophy*, pages 247–270.

MOXEY, LINDA AND ANTHONY SANFORD, 1987. Quantifiers and Focus. *Journal of Semantics*, 5:189–206.

MÜLLER, ANA, 2002. The Semantics of Generic Quantification in Brazilian Portuguese. *Probus: International Journal of Latin and Romance Linguistics*, 14(2):279–298.

MÜLLER, STEFAN, 1996. The Babel-System—An HPSG Prolog Implementation. In *Proceedings of the Fourth International Conference on the Practical Application of Prolog*, pages 263–277. London.

MÜLLER, STEFAN AND WALTER KASPER, 2000. HPSG Analysis of German. In WOLFGANG WAHLSTER, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 238–253. Springer-Verlag, Berlin Heidelberg New York, Artificial Intelligence edition.

MUNN, ALAN AND CRISTINA SCHMITT, 1998. Against the Nominal Mapping Parameter: Bare nouns in Brazilian Portuguese. In *Proceedings of NELS 29*, pages 339–353. GLSA, The University of Massachusetts, Amherst, MA.

MUSKENS, REINHARD, 1995. Order-Independence and Underspecification. In J. GROENENDIJK, editor, *Ellipsis, Underspecification, Events and More in Dynamic Semantics*.

NERBONNE, JOHN, MASAYO IIDA AND WILLIAM LADUSAW, 1989. Running on Empty: Null Heads in Head-Driven Grammar. In JANE FEE AND KATHERINE HUNT, editors, *Proceedings of the Eightth West Coast Conference on Formal Linguistics*, volume 8, pages 276–288. CSLI Publications/SLA.

NERBONNE, JOHN AND TONY MULLEN, 2000. Null-Headed Nominals in German and English. In FRANK VAN EYNDE, INEKE SCHUURMAN AND NESS SCHELKENS, editors, *Proc. of Computational Linguistics in the Netherlands 1998*, pages 143–64.

NETTER, KLAUS, 1996. *Functional Categories in a an HPSG for German*, volume 3 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology*.

OEPEN, STEPHAN, 2001. [incr tsdb()] — Competence and Performance Laboratory. User Manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany. In preparation.

OEPEN, STEPHAN AND JOHN CARROLL, 2000. Parser Engineering and Performance Profiling. *Natural Language Engineering*, 6(1):81–98. (Special Issue on Efficient Processing with HPSG).

PARTEE, B., A. TER MEULEN AND R. E. WALL, 1990. *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, Dordrecht, Germany.

PARTEE, BARBARA, 1983. Uniformity vs. Versatility: the Genitive, a Case Study. In JOHAN VAN BENTHEM AND ALICE TER MEULEN, editors, *The Handbook of Logic and Language*, pages 464–470. Elsevier, Amsterdam.

PENN, GERALD, 2004. Balancing Clarity and Efficiency in Typed Feature Logic Through Delaying. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 239–246. Barcelona, Spain.

POESIO, MASSIMO, 1994. Ambiguity, Underspecification and Discourse Interpretation. In H. BUNT, R. A. MUSKENS AND G. RENTIER, editors, *Proceedings of the International Workshop on Computational Semantics*, pages 151–160. Tilburg.

POLLARD, CARL AND IVAN SAG, 1994. *Head-Driven Phrase Structure Grammar*. Chicago University Press and CSLI Publications.

POLLARD, CARL J. AND IVAN A. SAG, 1987. *Information-based Syntax and Semantics, Vol. 1*. Number 13 in CSLI Lecture Notes. CSLI Publications, Stanford University. Distributed by University of Chicago Press.

PULLUM, GEOFFREY K., 1975. *People* Deletion in English. In *Working Papers in Linguistics*, volume 14, pages 95–101. Ohio State University.

REYLE, UWE, 1993. Dealing with Ambiguities by Underspecification: Construction, Representation and Deduction. *Journal of Semantics*, 10:123–179.

SAG, IVAN A., 2000. Rules and Exceptions in the English Auxiliary System. Manuscript.

SAG, IVAN A., THOMAS WASOW AND EMILY M. BENDER, 2003. *Syntactic Theory – A Formal Introduction*. CSLI Publications, Stanford, California, 2nd edition.

SCHIEBER, S. M., 1993. The Problem of Logical Form Equivalence. *Computational Linguistics*, 19(1):179–190.

SIEGEL, MELANIE AND EMILY M. BENDER, 2002. Efficient Deep Processing of Japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and International Standardization. Coling 2002 Post-Conference Workshop*, pages 31–38. Taipei, Taiwan.

TICIO, M. EMMA, 2005. NP-Ellipsis in Spanish. In DAVID EDDINGTON, editor, *Selected Proceedings of the 7th Hispanic Linguistics Symposium*. Somerville, MA.

VAN EYNDE, FRANK, 2003a. On the Notion 'Determiner'. In STEFAN MÜLLER, editor, *Proceedings of the HPSG-2003 Conference, Michigan State University, East Lansing*, pages 391–396. CSLI Publications, Stanford.

VAN EYNDE, FRANK, 2003b. Prenominals in Dutch. In JONG-BOK KIM AND STEPHEN WECHSLER, editors, *The Proceedings of the 9th International Conference on HPSG*. CSLI Publications, Stanford University.

WINHART, HEIKE, 1997. Die Nominalphrase in einem HPSG-Fragment des Deutschen. In ERHARD HINRICHS, WALT DETMAR MEURERS, FRANK RICHTER, MANFRED SAILER AND HEIKE WINHART, editors, *Ein HPSG-Fragment des Deutschen. Teil 1: Theorie*, chapter 5, pages 319–384. Universität Tübingen, Tübingen.