Technical Report

UCAM-CL-TR-533
ISSN 1476-2986

Number 533

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

The acquisition of a unification-based generalised categorial grammar

Aline Villavicencio

April 2002

JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted September 2001 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Hughes Hall.

The thesis was partially sponsored by a doctoral studentship from CAPES/Brazil.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/TechReports/

Series editor: Markus Kuhn

ISSN 1476-2986

To my family

Abstract

The purpose of this work is to investigate the process of grammatical acquisition from data. In order to do that, a computational learning system is used, composed of a Universal Grammar with associated parameters, and a learning algorithm, following the Principles and Parameters Theory. The Universal Grammar is implemented as a Unification-Based Generalised Categorial Grammar, embedded in a default inheritance network of lexical types. The learning algorithm receives input from a corpus of spontaneous child-directed transcribed speech annotated with logical forms and sets the parameters based on this input. This framework is used as a basis to investigate several aspects of language acquisition. In this thesis I concentrate on the acquisition of subcategorisation frames and word order information, from data. The data to which the learner is exposed can be noisy and ambiguous, and I investigate how these factors affect the learning process. The results obtained show a robust learner converging towards the target grammar given the input data available. They also show how the amount of noise present in the input data affects the speed of convergence of the learner towards the target grammar. Future work is suggested for investigating the developmental stages of language acquisition as predicted by the learning model, with a thorough comparison with the developmental stages of a child. This is primarily a cognitive computational model of language learning that can be used to investigate and gain a better understanding of human language acquisition, and can potentially be relevant to the development of more adaptive NLP technology.

Acknowledgements

First of all, I want to express my gratitude to my supervisor, Ted Briscoe, for all his support, for making this whole project more enjoyable with his excellent discussions, and for his patience during stressful times. I'm grateful to Ann Copestake for her friendly support, and for innumerable suggestions and solutions to the most varied problems. Many thanks to Karen Sparck Jones and Stephen Pulman for broadening my knowledge of natural language processing. I'm also indebted to Ben Waldron for allowing me to use his system. I also want to thank all the people that at one stage or another contributed to this work, especially Julia Hockenmaier, Judita Preiss, Claire Taylor and James Thomas for their comments on the thesis.

Thanks to all the staff at the Computer Laboratory who were always so friendly, and made my life easier during the PhD. I am also grateful for the generous travel grants given to me by both the Computer Laboratory and Hughes Hall. The research reported on this thesis was supported by a doctoral studentship from CAPES/Brazil. Thanks to everyone from CAPES, especially to Vanda Lucena.

My time in Cambridge was made happier with the presence of all the friends that I made here, among them: Martin Choquette, Pierre Jourlin, Sylvia Knight, Olivia Kwong, Naila Mimouni, Miles Osborne and Donnla NicGearailt, Tetsuya Sakai, Advaith Siddharthan, Jana Sukkarieh and James Thomas. Anna Korhonen was a constant source of friendship, since we started the MPhil and throughout our PhDs. To all the many friends that entered my life during this period, thank you all for your friendship and encouragement.

All the support I received from my family in Brazil was a constant motivation for completing the PhD. Ricardo, Ingrid, Hilda, Fabio, Ricardo Neto and Bianca with their love and wisdom made me see how beautiful life can be. Finally, I'd like to thank my husband Fabio Nemetz, without whose love, support, patience and friendship, this research would never have been finished, and to whom this thesis is dedicated.

Contents

List of Acronyms and Abbreviations

AB-CG	AB-Categorial Grammar
AVS	Attribute-Value Specification
BIPS	Bayesian Incremental Parameter Setting
CCG	Combinatory Categorial Grammar
CG	Categorial Grammar
CUG	Categorial Unification Grammar
FS	Feature Structure
HPSG	Head-Driven Phrase Structure Grammar
MDL	Minimum Description Length
MLE	Maximum Likelihood Estimation
NLP	Natural Language Processing
N	Noun
NP	Noun Phrase
PAC-learning	Probably Approximately Correct Learning
PP	Prepositional Phrase
PRT	Particle
PPT	Principles and Parameters Theory
OVS	Object-Verb-Subject
RNR	Right Node Raising
S	Sentence
SCG	Stochastic Categorial Grammar
SOV	Subject-Object-Verb
STL	Structural Triggers Learner
SVO	Subject-Verb-Object
TDFS	Typed Default Feature Structure

TFS	Typed Feature Structure
TLA	Triggering Learning Algorithm
UB-GCG	Unification-Based Generalised Categorial Grammar
UCG	Unification Categorial Grammar
UG	Universal Grammar
VCA	Valid Category Assignment
VSO	Verb-Subject-Object

List of Figures

List of Tables

Chapter 1

Introduction

Over the last four decades, researchers in the Natural Language Processing (NLP) community have been developing techniques to enable computers to understand human language. Traditionally, NLP systems use manually constructed resources containing linguistic information. The construction of these resources is labour intensive and time consuming, and results in static and inflexible resources.

Due to the open-ended and constantly evolving nature of natural languages, researchers have started to investigate how to equip natural language processing systems with learning models for automatically acquiring linguistic information. Such systems would have the capacity to constantly evolve with their linguistic environments. In this way, a system would be prepared to handle novel uses of linguistic constructions, which can be introduced in a given language, for instance, by communities of speakers, either in the more traditional sense of regional communities, but also communities such as mobile phone users, computer users, business people, and so on. Moreover, such systems should be robust to noise in the linguistic environment. The need for systems that can dynamically adapt to their linguistic environment has increased with the necessity of processing huge quantities of unrestricted data that are currently available, thanks to advances in technology.

In trying to solve the question of how to get a machine to automatically learn linguistic information from data, we can look at the way people do it. The acquisition of language from data is a task that we humans do when we acquire our mother language. Children acquire a grammar that corresponds to their language just by being exposed to a linguistic environment. This environment includes noisy and ungrammatical sentences, the potential influence of other languages, and many other linguistic phenomena. In spite of that, most children are successful in the acquisition of a grammar in a relatively short time, acquiring a sophisticated mechanism for expressing their ideas, based on data that is noisy and said to be too impoverished to generate such a complex capacity. One approach to explain the acquisition of languages proposes that children must have some innate knowledge about language, a Universal Grammar (UG), to help them

overcome the problem of the poverty of the stimulus and acquire a grammar on the basis of positive evidence only [Chomsky 1965]. According to Chomsky's Principles and Parameters Theory [Chomsky 1981], the UG is composed of principles and parameters, and the process of learning a language is regarded as the setting of values of a number of parameters, given exposure to this particular language.

These ideas about human language acquisition are employed, in this work, in the construction of a computational learning system that can learn from its linguistic environment, which may contain noise and ambiguities. Studies like this can also be used to help us understand better the process of human language learning, but if that is to be achieved, we need to concentrate only on algorithms and resources that a human learner could employ. Thus, there are significant constraints on the assumptions that can be made in the learning system implemented. In this way, the learner cannot have access to negative information, which is a source of information used by a number of learning algorithms. The learner also cannot start with information specific to a particular language, and can only assume information that is general among human languages. Another aspect is that learning has to be on-line and incremental, with the system only processing one sentence at a time, without the possibility of storing sentences and reprocessing previously seen sentences, or doing multiple passes through the corpus. Moreover, the kind of data given to the learner must be compatible with the linguistic environment of a child. In this work the linguistic environment of the learner is recreated using spontaneous child-directed speech in English, which was phonemically transcribed, taken from the Sachs corpus [MacWhinney 1995] and [Sachs 1983]. Children can use semantic and contextual information when processing sentences, and semantic information is introduced in the corpus by annotating the sentences with logical forms. In many psychological studies children were observed to be sensitive to statistical patterns in the data to which they are exposed; in this work the learning system is also sensitive to statistical patterns occurring in the input data.

The wider goal of this project is to investigate the process of grammatical acquisition, and the focus of this work is on particular aspects, namely the acquisition of subcategorisation frames and word order information from data, and the effect of noise and ambiguity in the learning process. This work involved building a learning system, determining an appropriate form for the UG with principles and parameters, studying the relations between the parameters and the data that sets each of them, and implementing an appropriate learning algorithm. The UG is represented as a Unification-Based Generalised Categorial Grammar (UB-GCG), and is embedded in a default inheritance hierarchy. The UG provides the core knowledge about grammars that the learner has, and a learning algorithm fixes the parameters to a particular language based on exposure to it. Moreover, a UB-GCG for English is implemented, and this grammar is used to annotate the parents' sentences from the Sachs corpus with the appropri-

ate logical forms. This corpus, which we refer to as the annotated Sachs corpus, is then given as input to the implemented learning system, and it simulates some of the environment in which a child acquires her language.
The learning environment contains noise and ambiguity and the learner has to be able to deal with these problems if it is to converge to the target. In this work the ambiguity is in the form of locative Prepositional Phrases that can occur either as arguments to a verb or as adjuncts, and the noise is caused by sentences where the thematic roles in the associated logical form are incorrectly specified, corresponding to the cases where the learner is uncertain about the thematic roles. Faced with the input data provided by the annotated Sachs corpus, the learner has to set the parameters of the UG accordingly to be able to successfully process this data. The learning algorithm implements a Bayesian Incremental Parameter Setting (BIPS) learner [Briscoe 1999], and when setting the parameters it uses a Minimum Description Length (MDL) style bias to choose the most probable grammar that describes the data well. Several experiments are conducted for testing the performance of the learner in different environments, and the results obtained are analysed to determine the success of the learner in the language learning tasks. In this evaluation, the English UB-GCG serves as the target grammar, to which the learner has to converge in the learning tasks.

1.1 Structure of the Thesis

This thesis is organised into eight chapters, where chapters 2 and 3 provide some background concepts that are important for the investigation conducted in the thesis. Chapter 2 provides an overview of the relevant research in Linguistics and Psychology about human language acquisition, and in Computer Science about language learning systems.

Chapter 3 presents a discussion of the methods and techniques for encoding grammatical knowledge that have been successfully employed in natural language processing. Given that we are interested in efficient ways of representing linguistic information, this discussion provides the necessary background from Linguistics and NLP for this topic.

This is followed, in chapter 4, by a description of the Unification-Based Generalised Categorial Grammar implemented for English as part of this thesis. This English UB-GCG implements several ideas and insights from linguistics and computational linguistics, and represents them using defaults. In this way, the grammar joins together many proposals about the use of defaults, and also implements new ideas for the application of defaults proposed in the scope of this thesis. This chapter also proposes a model of the UG in terms of UB-GCGs. The UG and its principles and parameters are embedded in a default inheritance hierarchy, which is a structure that is likely to be used in human processing for representing and classifying several types of concepts. A description of the different types of

parameters defined is provided. Finally, chapter 4 also contains a description of the annotated Sachs corpus, discussing the annotation undertaken and the characteristics of the constructions found in the corpus.

In chapter 5 we describe two systems developed by Waldron [1999] that are used to pre-process each sentence as it is given as input to the learning system. These systems assign candidate logical forms and syntactic categories to the words in a sentence. We present a detailed description of each of the systems, as well as an evaluation of their performance in relation to the annotated Sachs corpus. As discussed in the evaluation, these systems are prone to errors, being able to correctly assign semantic and syntactic categories to only a small portion of the corpus.

This is followed by a description, in chapter 6, of the learning system developed. The learning system implements a Bayesian Incremental Parameter Setting algorithm [Briscoe 1999], which uses an MDL-style bias to set the parameters of the grammar. The original algorithm proposed by Briscoe is adapted for learning within a unification-based framework. The learning system needs to be able to learn new and more complex categories, and include them in the currently used grammar. It also needs to set the parameters according to the linguistic input it receives. During the learning task, the learner collects statistical information about the data and uses this information to help in obtaining the most concise encoding for the grammar, according to the MDL-style bias.

Chapter 7 describes the learning tasks that the learner has to perform for setting its parameters according to the language of the environment. Initially, the learner starts with a small set of categories that can be used to process the sentences, and, as learning progresses, more complex categories are also learned. During the learning process, some of the sentences that the learner receives are ambiguous, and the learner has to solve this ambiguity before learning can proceed. We concentrate on investigating the ambiguity caused by locative Prepositional Phrases, which can occur as arguments of the verb or as adjuncts. Once the ambiguity is solved, the learner can proceed as usual and set the parameters of the UG. Moreover, there are several possible starting points for the learner in the learning tasks: the learner can start with its parameters unset, but it can also have some parameters that are initially set to certain values, as defaults. This is an important point, which in some learning systems described in the literature ([Gibson and Wexler 1994], [Berwick and Niyogi 1996] and [Kohl 1999]) was shown to determine whether the learner would converge to the target or not. Since the starting point can affect the performance of the learner, we investigate several possible starting points and analyse the different results obtained. Furthermore, during the learning process, some sentences may contain noise, and this noise interferes with the convergence of the learner to the target grammar. We investigate how the learner performs when facing different levels of noise provided by different environments.

Finally, we present some conclusions, with an analysis of the contributions of

this work:

- the integration of several ideas on the use of defaults in linguistic description,
- the proposal of a plausible model of the Universal Grammar based on typological and linguistic studies, implemented as a UB-GCG, which allows featural variation, being suited to capture natural languages,
- the development of a model of the UG embedded in a default inheritance network of types that reduces of the amount of information that needs to be learned because the information is passed by default from supertypes to subtypes, so that even if a parameter has not been set, it already has the inherited value,
- the development of a computational system where learning is based on principles such as the Categorial Principles developed for independent reasons, and that uses the default inheritance network and the MDL Principle for learning. Such a system can successfully learn from a corpus of real child-directed data, containing noise and ambiguity, in a computational account of parameter setting that is compatible with several characteristics of human language acquisition.

We conclude with a discussion of future work that can be developed given the framework implemented in this research.

Chapter 2

Literature Review

In this work we are interested in investigating the learning of grammatical knowledge from data. A learning situation usually involves five elements [from Bertolo 2001]:

- the concept being learned,
- the kind of hypotheses that the learner can formulate,
- the environment providing the data to the learner and defining the order of presentation of this data,
- the learning framework which defines the restrictions for the updating of the hypotheses in relation to the presentation of data, and
- the conditions for success of the learner.

A learner can be seen as an individual equipped with a learning procedure which can map a set of input examples into hypotheses that can classify the examples. The **concept being learned** in the scope of this study is a language from the set of natural languages.

The **hypotheses** in relation to the target concept are expressed by means of grammars that can classify the input examples as generated or not by the grammar of the target language. In this work, the grammar space is composed of grammars that are allowed by a model of a Universal Grammar [Chomsky 1965], defined in terms of principles and parameters. Following the Principles and Parameters Theory (PPT), the parameters of the UG are set to a particular grammar that provides the linguistic environment to which the learner is exposed, as is described in section 2.1.

The **environment** provides the data given to the learner, which consists of sentences produced by the target language. Besides providing the examples, a learning environment can also correctly inform whether a given example is in the target or not, it can incorrectly inform, or it can remain silent. Certain

environments can provide infinitely many examples to the learner while others can provide only a finite number of examples. Some environments may present the examples according to a certain rule known to the learner, and this makes the learning task easier. One widely investigated notion of environment is that of a *text* that is defined by Gold [1967] as an infinite sequence of sentences for a given language L such that every sentence of L appears at least once in it and no sentence that is not in L appears in it. As far as this work is concerned, the environment is a finite subset of a text that provides sentences annotated with logical forms, extracted from a corpus of parents' sentences to a child, which presents the sentences in the order in which they were produced.

Learners can follow different strategies. For instance, some use past evidence to decide on which hypothesis to select next, having a perfect memory. Others are memory-less and do not use any past data to choose their next hypothesis, possibly revisiting past hypotheses. There are also those that only modify their current hypothesis when it is incompatible with the current example, among other possible learners. This is discussed in section 2.2.

The last element is a **criterion of success**, which is used to evaluate whether a learning strategy is successful or not. Two widely used learning paradigms are Identification in the Limit and Probably Approximately Correct (PAC) learning. These two paradigms are considered to be too strict to be used in practical situations [Bertolo 2001]. Nonetheless, as demonstrated in [Bertolo 2001], if a class of languages is generated by a class of grammars consistent with the Principles and Parameters Theory, it is learnable under each of the criteria. An alternative to the first two paradigms is the Minimum Description Length Principle (MDL) that is an algorithmic paradigm for defining learners, which has been successfully used in several computational learning systems. These learning paradigms are discussed in section 2.3.

2.1 Universal Grammar

One of the remarkable facts about languages is that, although they are complex systems, children learn them reliably and in a short period of time. As Hyams [1986] observes, children have to learn the form and meaning of individual lexical items used in their languages; they must develop a processing system to generate and comprehend sentences; they must learn pragmatic and social skills to be able to use language appropriately in different contexts; and they must learn the grammar that corresponds to their language. However, the data to which children are exposed is so limited that it cannot provide an explanation of how it is that children achieve all the aspects of the mature state [Lightfoot 1991]. This problem, known as the *poverty of the stimulus*, arises because, in general, the learner is exposed to simple grammatical input, which for the most part consists of sentences without much embedding, and, as observed by Gleitman and Wanner

[1982], that are "*propositionally simple, limited in vocabulary, slowly and carefully enunciated, repetitive, deictic, and usually referring to the here and now*". So how can children based on this data arrive at a mature state that is so sophisticated? According to Lightfoot [1991], the input to which children are exposed is poor in three ways [from Lightfoot 1991, p. 3]:

- "*The child's experience is finite, but the capacity eventually attained ranges over an infinite domain, and therefore must incorporate some recursive property not demanded by experience*". Children do not simply imitate what they hear, or memorise the list of all sentences to which they are exposed, but they create new sentences, which suggests that they have an internal system of rules that allows them to creatively produce new sentences.

- "*The experience consists partly of degenerate data which have no effect on the emerging capacity*". The linguistic environments to which children are exposed may contain noise and interruptions, may be influenced by another language, and so on. Nevertheless, children successfully converge to their target language.

- "*Most importantly, it fails to provide the data needed to induce many principles and generalisations manifested by the mature capacity*". For instance, the relation between a declarative sentence such as *The book on the shelf is expensive*, and a question such as *Is the book on the shelf expensive?* is learned, even though this is not explicitly stated when each of these examples is presented to the learner. Moreover, such a relation can be applied to other sentences, not being exclusive to the particular words used in these two sentences, which suggests that the learner must be making generalisations from specific examples.

A child has to learn a language just by being exposed to it and without, in general, receiving any feedback about possible errors it makes. Furthermore, the linguistic environments to which children are exposed have a variety of different dialects and possibly more than one language. However, children are robust to this variety of influences and successfully learn the languages and dialects to which they are consistently exposed. The question that arises is: 'How does this work?'.

Chomsky's proposed explanation [Chomsky 1965] is that humans must have some innate knowledge about languages, and, as a consequence, they do not need to learn some of the complex aspects of languages. Thus the hypothesis is that children must have some innate knowledge about language that guides them and helps them overcome the poverty of the stimulus, to acquire a grammar based only on positive evidence and in a finite amount of time.

According to Chomsky's [1981] Principles and Parameters Theory, this core common knowledge, or Universal Grammar, is composed of a set of **principles** and **parameters**. The principles constitute the child's prior knowledge of

languages, representing characteristics that are common across languages. The parameters represent the points of variation across them and are fixed to a particular language based on exposure to a given linguistic environment. This theory suggests that human languages follow a common set of principles and differ among one another only in finitely many respects, represented by the parameters. As there is a finite number of parameters, that can take a finite number of values, there is a finite number of languages that the learner needs to consider, instead of an infinite space of possibilities. This constrains the class of languages that can be learned and makes language acquisition possible in principle. By setting the parameters of the UG to the appropriate values, based on exposure to a particular linguistic environment, the result is the selection of the grammar that can account for the input sentences, among all the various possible grammars allowed by the UG. The sentences that activate the setting of the parameters are called triggers, as discussed in section 2.1.1.

The form of the UG is still an unresolved question. While it has to be restrictive enough to allow language acquisition to happen on the basis of poor stimulus in a finite amount of time, it has to be general enough to allow for the range of possible human languages.[1]

One possible source of additional information consists of the statistical properties of the data to which children are exposed. This data presents statistical properties that can potentially be used during the learning process. It may be the case that children ignore this source of information, but if they do use it, it could help them minimise several of the difficulties that they face. Moreover, noise resistance may be explained in terms of sensitivity to statistical properties of the environment. As Clark [2001] observed, recent work in psycholinguistics has indicated that children are indeed sensitive to these statistical properties ([Redington and Chater 1998] and [MacDonald 1999]).[2] Furthermore, children may also gather additional information from the context in which they hear sentences. As Pinker [1995] remarked, children do not hear sentences in isolation but in a context. Thus, when a child interacts with other speakers, these tend to talk more about the here and now, and the child can observe the context and guess what the speaker might have meant. In fact, many models of language acquisition assume that the input the child receives consists not only of the sentence, but also of a representation of the meaning of the sentence ([Wexler and Culicover 1980], [Pinker 1984], [Berwick 1985], [Briscoe 1999], among others). How much of the context surrounding the hearing of a sentence is used by children is an open question. Besides, as Landau and Gleitman [1985] observe, blind children have a severely limited access to non-linguistic aspects of the world, but they succeed in learning languages without many problems. However, it is reasonable to as-

[1]For a discussion of possible parametric systems see Hyams [1986], Clark [1992], Culicover [1997], Briscoe [1997 to 2000], among others.

[2]Some computational systems of language acquisition that use statistical information are described in [Kapur and Clark 1996], [Briscoe 1999] and [Yang 1999].

sume that children can use their knowledge about the current situation and the meaning of familiar words in a sentence to try to understand (at least part of) the sentences they hear.

As a child is exposed to its linguistic environment, a learning procedure sets the values of the parameters such that the resulting grammar is usually equivalent to that of the environment. The process of acquiring the target grammar can be understood through Chomsky's model of a child searching through a restricted hypothesis space of grammars in order to select the correct one [Chomsky 1965]. Constraints help the learner get closer to the target hypothesis by excluding many incorrect hypotheses that do not conform to these constraints [Berwick 1985]. The UG can then be considered a restriction in the hypothesis space, such that it only contains those grammars that conform to the principles and to the possible assignments of values of the parameters. In this way, a child does not need to consider every possible grammar, but only the ones consistent with the UG. However, even in this restricted hypothesis space, if we assume that parameters are binary-valued[3] and that there are between 20 and 30 independent parameters, these give rise to between 2^{20} (1,048,576) and 2^{30} (1,073,741,824) estimated possible grammars. Consequently an efficient search needs to be conducted.

2.1.1 Triggers

As already stated, the observed simplicity of the input to children has several implications for learning, such as increasing the need for some sort of innate learning device which would help the learner overcome the poverty of the stimulus. Another implication is that, if children learn a language based on this simplified input, it must be the case that all the data needed for setting the parameters of the UG to the target language must be present in this simple input. The process of setting the syntactic parameters is known as **triggering**, and the input that provides evidence for the learner to set its parameters is known as a **trigger**.

Triggering is a particular type of learning where languages are perceived as differing in terms of parameter settings. This is an alternative to another type of learning referred to as hypothesis formation, where grammars are regarded as rule systems that considerably vary from one language to another. As Sakas and Fodor [2001] observe, such a type of learning has a much heavier workload and does not seem to be suited to model human language learning, which is a uniform and gradual process. Parameter setting, on the other hand, seems to be less costly and much more uniform.

Precise definitions of triggers are hard to find and variable, but a commonly used concept is that of a sentence of the target language that the learner detects and uses to set the values of parameters to the target. Lightfoot [1991] defines triggers as consisting of set of utterances haphazardly appearing in a given context

[3]Throughout the document, unless otherwise stated, parameters are binary-valued.

of a type that any child hears very frequently, and not including any kind of negative evidence.

But which sentences are triggers, and what form do these sentences have? Clark [1992] defines the concept of a fair text, where triggers must be expressed in sentences that are simple. He formalises the notion of **parameter expression**, where a sentence expresses a certain parameter if it requires the grammar to have that parameter set to a particular value for the sentence to be grammatical. Any input sentence can express parameters, and a sentence is a trigger for a parameter if it expresses that parameter. Thus, for a system of parameters to be learnable, there must exist triggers for all the values of its parameters in the input data, and the learner must be able to detect them. Furthermore, Clark also defines the notion of **frequency of parameter expression**, which states that triggers for each of the parameters must occur above some minimal frequency in the data. Then, given that the learner needs to be able to resolve possible ambiguities, it must be the case that the target settings must be expressed with a higher frequency than alternative non-target settings. Besides that, it must be guaranteed that the parameters can be expressed in simple structures that are likely to occur in the learner's environment, and this guarantee is formalised as the **boundedness of parameter expression**.

Some computational learning algorithms, like the Triggering Learning Algorithm (TLA) [Gibson and Wexler 1994], need triggers to activate them, and, based on these sentences, they change the parameters of the UG to account for a particular language. In an error-driven algorithm like the TLA, a trigger is recognised by parsing failure, and it is used to signal a need for changing the parameter settings. The learner then attempts to change the parameters in constrained ways, so that it can successfully parse the sentence.

The precise form of triggers is also varied. Are triggers expressed only in the words in the sentences? Or are triggers also expressed in other sources of information? For instance, Briscoe [Briscoe 1999] uses a notion of triggers that consist of a sentence annotated with an appropriate logical form, from which evidence for certain parameter values are obtained, making use of the meaning of the sentence. Fodor's Structural Triggers Learner (STL) [1998] uses treelets obtained from input sentences as triggers. Treelets consist of subtrees containing information about a derivation tree produced by a parser for a given sentence, and they are used to set the parameters of the UG .

One important question regarding the use of triggers for parameter setting is that it presupposes that the learner can identify the relevant triggers in the input data for each of the parameters. However, how does the learner detect triggers automatically and reliably from the input sentences? How does it know which parameter and which value is being triggered? Dresher and Kaye [1990] propose that the learner is innately provided with the knowledge of the cues (triggers) associated with every parameter and, when the input provides cues, the learner sets the values corresponding to the relevant parameters. Kapur and Clark [1994]

investigate the problem of trigger detection using some statistical properties of the data. They define a learner that analyses the patterns in the occurrence of certain phenomena and uses these patterns to set the parameters of the UG.

Thus, learners detect triggers in the data and use them to set the parameters of the UG. But do all triggers provide reliable evidence for a particular value? Clark [1992] discusses the problem of **indeterminacy of parameter expression**, where a given trigger may be consistent with different parameter values. One such case is that of word order ambiguity, where the surface order of constituents in an input sentence may not reflect the underlying word order. For example, German, whose canonical order is Subject-Object-Verb (SOV), also accepts Subject-Verb-Object (SVO) orders due to the verb second (V2) phenomenon, which moves the verb to the second position. In this way, when an SVO sentence occurs, it might have been generated by an SVO language such as English, but it might also have been produced by an SOV-V2 language like German. The problem for the learner is that not all triggers are going to be giving evidence for only one parameter value, and the learner has to decide how to proceed when faced with ambiguous triggers. Sakas [2000] examines the problem of ambiguous triggers in relation to Fodor's Structural Trigger Learner, where the variant of the STL employed only learns from triggers that are unambiguous. From the results he obtained, assuming that sentences contain a fixed number of triggers, such a learning strategy would need an unreasonable number of input sentences to set the parameters of the UG to the target language, since it needs to wait for completely unambiguous triggers. The results obtained by Sakas are specific to the STL model, but they highlight the importance of more investigation into the relation between triggers and learning models, since this may clarify whether a given model can present a reasonable account of language acquisition.

2.1.2 Negative evidence

As Pinker [1994] notes, children seem to acquire their native languages using only positive evidence, in the form of sentences of their languages. However, to clarify this issue some researchers investigated the possibility of the availability of negative evidence too. The availability of negative information changes considerably the nature of the language learning problem, avoiding several of the difficulties that arise when only positive evidence is available. Furthermore, if negative evidence were available to the learner, it would have to occur in a systematic manner in the environment, in the same way as positive evidence does. Indeed, many of the learnability proofs rely on such evidence being systematically available in the environment. Gold [1967], in his learnability study, showed that when only positive evidence was available in the form of a text, not even the class of regular languages is identifiable in the limit, whereas if negative evidence is provided by an informant, the class of languages that can be identified in the limit includes that of context sensitive languages. Thus, the possibility of both

positive and negative information available for learning alters considerably the acquisition problem.

Parents' correction is considered by many to be a source of negative evidence. However, even though some parents do correct their children, it is not true of all parents, and as such it is not available to all children. Consequently the occasional correction of grammatical errors cannot be regarded as systematically available. Furthermore, in certain cultures not even the occasional correction is available to children, which suggests that the possibility that children rely on negative information for learning is extremely remote [Marcus 1993]. Moreover when there is parental correction available, as Ingram [1992] observes, it is mostly concerned with semantic appropriateness and in the cases where it is related to syntax, besides the fact that in some cases it is difficult to know what is wrong, children seem to be oblivious to this kind of correction. Thus, it does not seem that the occasional correction that occurs so infrequently and unsystematically can be used by children as explicit negative evidence to help them acquire their languages.

2.2 Learners

The focus of this thesis is the acquisition of grammatical knowledge, in terms of Categorial Grammars (CGs), within the Principles and Parameters framework. Thus, this section provides an overview of some of the work done on the computational learning of CGs and on learning within the Principles and Parameters Theory.

2.2.1 Categorial Grammar Learning

A considerable body of work has been developed for CG learning. Given a set of input sentences generated by a particular language, these grammar learning systems define the learning task as the search for a grammar to recognise and/or generate all grammatical sentences in that language. In these systems, the learner is given sentences from the target language as input, along with negative evidence in some cases, from which they need to learn a grammar. In what follows, we describe some of these systems, noting that a proper discussion of CG is provided in chapter 4.

In terms of learnability, the basic AB-Categorial Grammars (AB-CG) [Ajdukiewicz 1935], [Bar Hillel 1953], which employ a small set of categories and only the rules of functional application, have good learning algorithms from positive examples, when these examples are structured. Buszkowski and Penn ([Buszkowski 1987], and [Buszkowski and Penn 1990]) investigate the so-called "discovery-procedures" for AB-CGs, which are algorithms for learning CGs from functor-argument structures, with a resulting lexicon being generated where the

words are assigned the corresponding categories. These functor-argument structures are derivation tree structures, where each branch is annotated not with category labels, but instead with an indication of the functor daughter. Such results can be obtained for some classes of AB-CGs, such as the class of Rigid Grammars, where each word in the lexicon has exactly one possible category. Kanazawa [1998] further investigates the algorithms developed by Buszkowski and Penn, exploring not only learning from functor-argument structures, but also learning from flat strings of words. Kanazawa also discusses how the learnability results obtained can be adapted to variants of CG, such as Combinatory Categorial Grammars (CCG) [Steedman 1985 to 2000], which employ larger sets of categories and/or rules than AB-CG.

Among the computational systems developed for learning CGs, we now look at three systems: Adriaans' EMILE [1992], Watkinson and Manandhar's [2000] unsupervised learning system, and Osborne and Briscoe's [1997] Stochastic Categorial Grammar learner.

Adriaans ([Adriaans 1992] and [Adriaans and de Haas 2000]) defines EMILE as a system that performs CG induction within the PAC-learning framework, which is discussed in section 2.3.2. The learning algorithm has access to an oracle that provides correct positive examples and also answers questions, providing explicit negative evidence. Thus, the learner is allowed to ask questions about the grammatical validity of new sentences. In the grammar learning task, the learner assumes that each construction in the grammar can be exemplified by means of a small and simple sentence. It also assumes that there is a higher probability of the oracle using simpler sentences, and complex sentences have a lower probability of occurring. Using the oracle, the system tests the hypothesis generated about the categories in the grammar. Even though the system performs well, it is in a highly supervised environment, receiving both positive and negative input, so it is arguable whether it is a plausible approach to the study of human language learning.

Watkinson and Manandhar [2000] developed a system for learning CGs from unsupervised texts and without negative evidence. Their learner assumes that it knows from the start the set of all the possible categories, and it is allowed to start with a lexicon of closed class categories, such as prepositions and determiners. Then, for each word in a sentence the system tries to infer categories, from the set of possible categories, which would provide successful derivations, using the rules of functional application. During this process the system collects the frequency of occurrence of each word with a particular category and stores this in the lexicon too. The learner also maintains for each sentence processed, the sentence and the N most likely parses. During learning, each time the lexicon is modified, the system reparses the examples that were parsed before and that can be affected by the changes in the lexicon, and it recomputes the frequencies of the words, in case this reparsing finds new most likely parses. As a result of the process, the system acquires a lexicon containing the categories assigned to the

words and their corresponding relative frequencies, induced from a corpus of raw sentences. Furthermore, by storing the sentences with the N most likely parses found, an annotation for these sentences is also obtained, and this can be used as an annotated corpus. When the system is tested in an artificially generated corpus, which contains 39 words and one case of verb-noun ambiguity, the system performs well, achieving 100% lexicon accuracy, in relation to the categories assigned to the words, and 100% parsing accuracy. However, testing on a more realistic corpus, which contains 554 sentences, was not as successful, achieving 73.2% lexicon accuracy and 28.5% parsing accuracy. The task attempted by the system is very ambitious, and the system needs to rely on some sources of information: the set of possible categories, the rules, and the closed class lexicon. However, it is arguable whether this system is a plausible model of human language acquisition given its assumptions, especially about its access to all the sentences previously parsed and the possibility of reparsing them when needed.

Osborne and Briscoe [1997] define a system that learns Stochastic Categorial Grammar (SCG) from data, in a greedy, bottom-up, incremental manner. Initially, the system has access to a list of possible nodes. Given an input sequence of part-of-speech tags, the system analyses each two adjacent tags, and hypothesises a node, from the list of possible nodes, spanning these two tags. Then, all the possible trees consisting of a putative node and its two daughters are evaluated using Bayes' Rule. The putative tree with the highest posterior probability is selected and placed in the lexicon. These operations are executed recursively until a single tree spanning through the complete tag sequence is generated. These putative trees are evaluated with Bayes' Rule, and the authors compare the use of two different approaches to compute the prior probability of Bayes' Rule. The first one uses Maximum Likelihood Estimation (MLE), where the posterior probability is estimated using an uninformative prior. The second uses the Minimum Description Length (MDL) Principle, where the posterior probability is estimated using an informative prior defined in terms of the compactness of the hypothesis. The results obtained indicate that using the MDL Principle for the estimation of SCGs generated slightly better grammars than those generated using MLE. In spite of the good results obtained, such a learner is provided with a list of possible nodes, and it starts from a tagged corpus, learning categories for the tags rather than for the words, thus avoiding potential ambiguities.

Although these works investigate the problem of learning CGs with considerable success, they employ the basic AB-CG, that is not considered to be adequate to capture natural languages. In this work a more complex variant of CG which uses a unification-based approach is employed, and it has different learnability properties than the works described above. Moreover, learning is within the Principles and Parameters framework, with the use of a Universal Grammar. As is discussed in section 2.3, learning within the PPT was shown to have positive learnability results for several criteria such as Identification in the Limit and PAC-learning. Thus, in what follows we discuss some learning systems that

operate within PPT, although not necessarily using CGs.

2.2.2 Parameter Setting Learning

Even though the idea of a UG with principles and parameters is attractive, workable and accurate models of acquisition are difficult to design, and only recently have some models been developed to study the process of parameter setting. The problem of setting the parameters of the UG within the PPT has been addressed by many researchers, proposing algorithms and studying the conditions under which these would work satisfactorily, setting the parameters correctly. The goal is to arrive at the target grammar, which is the one for which all the parameters are appropriately set, given the evidence provided by the environment. In what follows a brief outline of some of the learning algorithms developed is given.

One such learning algorithm is the **Triggering Learning Algorithm** (TLA) [Gibson and Wexler 1994], which is error-driven, so that if it succeeds in parsing a sentence with the current setting of parameters, then it does not do anything. However, if it fails to parse a sentence, the TLA tries to change some of the values of the current parameter settings, and in case this change generates a grammar that can successfully parse the current sentence, it is adopted. Initially, the learner chooses randomly a possible parameter setting as the starting point and waits for triggers to come to take it from the starting point to a final state that hopefully corresponds to the target grammar. When parse failure of a sentence is detected, it signals for a change in parameters that is obtained by randomly selecting parameters to (re)set, and, according to the Single Value Constraint, the learner can select only one parameter to set. The learner then selects randomly an alternative value for the parameter and, if this change results in a successful parsing of the current sentence, then the change is adopted; otherwise it is ignored, following the Greediness Constraint.

The TLA was studied in a word order learning task, defined by Gibson and Wexler [1994], where the parameter space contained three parameters:

- the order of the S(ubject) in relation to the V(erb) (SV or VS),
- the order of the O(bject) in relation to the V(erb) (OV or VO) and
- the possibility of verb-second movement (+V2 or -V2).

These parameters give rise to 8 possible languages. The learner is exposed to triggers from the target that are assumed to contain only canonical orders of constituents (not receiving interrogative sentences, for example). Moreover, it is also assumed that, in the triggers, objects do not appear without overt subjects (e.g. the learner does not receive sentences with null subjects), and that the input data is pre-analysed and is in a form that represents word order and grammatical functions (e.g. the learner receives as input 'Subject-Verb-Object'). Gibson and

Wexler then analyse the conditions in which the learner would converge to the target. One of the problems faced by the TLA is that there are certain starting points from which it does not converge to the target, but instead reaches a local maximum, which is a point from which there are no triggers that can make the learner reach the target. For example, in this 3-parameter space there are 56 possible routes defined in terms of possible starting points and targets, and, from these, Gibson and Wexler found 6 local maxima routes for which no triggers would take the learner to the target. Moreover, as Berwick and Niyogi [1996] discovered, there are further 6 states that are connected to local maxima, from which the learner could either converge to the target or end up in a local maximum. Thus, depending on the starting point, which is randomly selected, the learner would not be able to converge to the target. Kohl [1999] investigated the model in a more linguistically realistic parameter space, containing 12 parameters: from the 4,096 possible grammars, 2,336 are not learnable even using the best starting state, and, in the worst starting state, 3,892 grammars are unlearnable, with an overall average of 3,348 unlearnable grammars.

Another problem for an error-driven algorithm like the TLA occurs when it generates a superset of the target grammar. An error-driven algorithm only changes parameters when a sentence that is not generated by the current grammar is encountered. However, as a superset of the target grammar generates all the sentences that the target generates, the algorithm will never be able to escape this situation and reach the target grammar.

Berwick and Niyogi [1996] formalised the TLA in terms of a Markov Chain, providing a formal description for the conditions for learnability in the hypothesis space. They were then able to compute the probabilities of each state and to quantify the amount of input data that a leaner needs to receive to converge to a certain target, which can be thought of as characterising quantitatively the convergence times required by the TLA. Frank and Kapur [1996] provide formalisation for different concepts of trigger and determine the implications of the use of each of these concepts in terms of learnability.

The TLA is a memoryless learner in the sense that it only uses the current input and the current grammar to decide on the hypothesis to adopt next, and it never stores any sentence previously encountered. One advantage of not storing this kind of information is that it seems to be psychologically plausible, because it does not impose an excessive memory burden, which would certainly be the case if the learner were required to memorise every input received. On the other hand, this characteristic allows rejected hypotheses to be revisited over and over again, increasing the amount of input data needed in order to converge to the target. As Brent [1996] notes, this does not seem to be plausible, since there is no evidence that children randomly revisit neighbouring grammars. One solution for this problem is to assume that parameters can only be reset once, which means that, once triggering data sets a parameter, it is permanent and cannot be reset again. The problem then is that a single incorrect trigger is enough for the learner

to incorrectly set its parameters, being unable to converge to the target.

In the **Bayesian Incremental Parameter Setting** (BIPS) learner defined by Briscoe [1999 and in press], the parameters have probabilities associated with each of the possible values. The defined BIPS learner is equipped with a UG and associated parameters (represented in terms of a Generalised Categorial Grammar) embedded in a default inheritance hierarchy. It also has a parser and a learning algorithm that updates parameter values and their associated probabilities according to the triggers provided in the data. The parameters are binary-valued and inheritance defines a partial ordering on them. Moreover, parameters can be initialised with absolute, default or unset values. Thus, when a triggering sentence that is successfully parsed provides evidence for certain parameter values, these values are reinforced by incrementing the corresponding probabilities. If a sentence cannot be parsed, the learner changes some of the parameter values in constrained ways, and, if the changes allow the sentence to be successfully parsed, the values used are reinforced. Such a learner is conservative, waiting for enough evidence confirming a certain value before setting a parameter to that value. It is also robust to a certain amount of noise in the triggers. Due to these characteristics, the learner is also able to avoid some of the shortcomings of the TLA, such as the problem of local maxima.

Fodor [1998] defines the **Structural Triggers Learner** (STL). It is a learning system that employs the notion of structural triggers, where triggers are in the form of tree-like structures called treelets. These treelets are obtained from the derivation tree for a given sentence, and parameter values are also in the form of treelets. The UG initially provides the learner with all the possible treelets, one for each parameter value. When the learner is given an input sentence, it parses the sentence and the treelets found in the derivation tree provide evidence for the corresponding parameter values. When a sentence cannot be parsed, the learner examines the partial derivation obtained and searches among the possible treelets allowed by the UG for some that may permit the sentence to be parsed. These are then adopted as values for the corresponding parameters. In one version of STL only those treelets in a completely unambiguous sentence are employed for learning. This requirement for unambiguous triggers may result in an unreasonably large amount of sentences required for the learner to converge, as discussed in section 2.1.1.

In the **Variational Model** [Yang 1999], language acquisition is viewed as the task in which a population of grammars compete in a Darwinian selectionist process. The learning model is equipped with a population of grammars, corresponding to the possible variations among languages, and where each grammar is associated with a weight. The learner is given sentences as input, and the learning algorithm proceeds in a selectionist manner, rewarding those grammars that can successfully analyse the input and penalising those that cannot. Language acquisition is guided by evidence that disambiguates the target grammar from competing ones, and the learner converges to the target when the composition

and distribution of the grammar population stabilise.

2.3 Learning Paradigms

In this section we discuss three learning paradigms starting with Identification in the Limit and PAC-learning, which are two criteria for the learner's success, and finishing with Minimum Description Length which is an algorithmic way for defining learners.

2.3.1 Identification in the Limit

Gold, in his influential paper [Gold 1967], defines the problem of **identification in the limit**, where the learner is asked to infer a particular target concept, given an enumeration of hypotheses and a sequence of positive and negative examples which may contain repetitions.

The learner then performs the learning task using **induction by enumeration**, where all the hypotheses in the enumeration that are not consistent with the set of examples seen so far are eliminated, from left to right, up to the first consistent hypothesis. At each new example received, the learner repeats this process, until a hypothesis in the enumeration is found that is identical to the target and that will be consistent with any new example. At this stage, the process is said to have converged.

In general terms, the idea is that the learner is considered successful when it *eventually* stabilises on a hypothesis that corresponds *exactly* to the correct one, regardless of the *order* of presentation of the examples.

As Bertolo [2001] remarks, in one sense the definition of identification in the limit is too idealised for human language learning, since it assumes that the information provided for the target language is complete and perfectly reliable and does not consider noise, ambiguity and incomplete information. Moreover, this definition is also too liberal, because it does not impose any restrictions about the number of examples and the time that the learner needs to make the identification. It only requires that this identification occurs *eventually*. Li and Vitanyi [1995] observe that, if significant time restrictions are added to the inductive model, almost all concepts are made not learnable, except some trivial ones, under these requirements. In any case, the class of grammars consistent with the PPT is learnable under this criterion, as discussed by Bertolo [2001].

2.3.2 PAC-Learning

Probably Approximately Correct (PAC) learning [Valiant 1984], [Li and Vitanyi 1993], [Li and Vitanyi 1995] is an alternative criterion for the success of the learner that is more adequate than identification in the limit.

It requires the learner to identify a hypothesis with high probability that is very close to the target concept, based on a reasonable number of examples. In this framework every language has a particular probability distribution according to which the examples are given to the learner. The idea is that the learner is successful if it can produce a hypothesis that would misclassify only the sentences that are unlikely to be given as part of its learning environment. The speed of convergence depends on how small the chance of error is required to be, so that the smaller the chance of error, the larger the amount of examples needed and the longer the convergence time is going to be.

In PAC-learning not all the examples will be presented to the learner and, as a consequence, the target concept cannot be precisely inferred and can only be approximated. Furthermore, the difference between the inferred hypothesis and the target can be expressed in terms of the probability of the set of examples on which these two disagree.

However, even though PAC-learning is not as stringent in its requirements as identification in the limit, the application of the PAC-learning model led to more negative than positive results. It seems that its requirements are still too strict for the learner. Nonetheless, as Li and Vitanyi [1995] observe, in practical learning situations, what happens is that the class of probability distributions can be restricted, the learners can usually ask questions, or the teachers can be helpful, among other things that can make the learning task less difficult for the learner. Once again, any finite class of languages consistent with the Principles and Parameters Theory is PAC-learnable [Blumer et al. 1987].

2.3.3 Minimum Description Length

The **Minimum Description Length** (MDL) Principle ([Rissanen 1989] and [Li and Vitanyi 1995]) is a method for designing learners, thus differing from identification in the limit and PAC-learning that are formal criteria of success to determine whether a given learner has performed well. In MDL the task of the learner is, given a sequence of examples, to choose a hypothesis that can be described concisely and at the same time does not make too many errors in relation to these examples. Therefore, given an enumeration of hypotheses and a set of examples, the MDL Principle states that the best hypothesis to infer from the data (the set of examples) is the one whose description is compact in length and that explains the examples well. MDL incorporates the idea that by only considering hypotheses that model the data well, one gets a theory with a more complex description where the number of misclassified examples decreases. However, it also means that this theory will probably overfit and will not predict well any unseen data. On the other hand, by only considering compact hypotheses, the description of a theory may be so simple that it will probably model the data badly. Thus, when evaluating hypotheses it is often the case that greater accuracy is achieved in the description of new data with a small but imperfect theory, than

with a theory that perfectly describes the known data [Quinlan and Rivest 1989].

The MDL Principle can be incorporated into Bayesian Learning, which is employed as the basis of some language learning systems. Bayes' theorem is defined as:

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

where:

- the term P(H) is the **prior probability**, which represents the learner's belief in the hypothesis H. If the prior is accurate, it will give higher probabilities to hypotheses that are close to the target than to the ones that are further away,
- P(D|H) is the **likelihood probability** and it represents how well the hypothesis H encodes the data D,
- P(D) is the **probability of the data**, and
- P(H|D) is the **posterior probability**. It can be seen as the combination of the prior and likelihood probabilities, so that one tries to find a hypothesis that maximises these two terms given the data.

The prior probability can be thought of as a bias or background knowledge the learner has, that helps searching for the target hypothesis. When the learning process starts, the prior probability is the only source of information for the learner and, as such, dominates the value of the posterior probability. However, as the learner receives more and more input data, it also relies on the data to perform the search. In the limit, it is the data (the likelihood probability) that dominates the posterior probability regardless of the initial belief of the learner (the prior probability). By helping to guide the search, an informative prior can make the convergence faster, which is especially important because there is usually only a limited amount of data available to the learner. However, it is not always easy to find an informative prior. One commonly used approach is to assume an uninformative prior, which results in a **Maximum Likelihood Estimator** (MLE). However, this approach leads to the problem of overfitting, where the best hypothesis is the one which only accounts for the data seen so far, predicting poorly any unseen data. A better approach is to use an MDL-style informative prior, using the notion of complexity of hypotheses, such that each hypothesis is evaluated in terms of its length and smaller hypothesis are preferred. Thus, the longer the length, the more complex the hypothesis is. This prior favours smaller (more compact) hypotheses by assigning them higher probabilities. The likelihood probability, on the other hand, prefers hypotheses

that model the data well. Given that a Bayesian learner is searching for a theory with high posterior probability, it needs to find the most compact hypothesis that describes the data well, balancing the prior with the likelihood probability. In addition, as Quinlan and Rissanen [1989] note, the great contribution of MDL is that it provides the learner with the conceptually simpler problem of computing the lengths of the hypotheses.

MDL is a more practical paradigm than either Identification in the Limit and PAC-Learning, and it has been successfully applied in a number of learning systems, like those in [Chen 1996], [Osborne and Briscoe 1997], [Stolcke 1994], [Briscoe 1999]).

2.4 Summary

In this chapter an overview of some relevant work on human language acquisition and on computational learning systems was presented. The UG is a useful concept that provides the core knowledge that the learner needs in order to acquire a particular language. By doing this, the UG restricts the hypothesis space that the learner needs to consider in the search for the target grammar. It is the responsibility of the learning algorithm to set the values of the parameters appropriately in order to converge to the target grammar based only on positive evidence about a particular language. A subset of this positive evidence consists of triggers which are the linguistic data that give evidence for the parameter values. In terms of learning, an efficient search strategy needs to be conducted, because even though the space allowed by the UG is more restricted, it is still too vast for an enumerative search to find the target grammar in a reasonable amount of time. Thus, in this work, the UG is going to be used in conjunction with an MDL interpretation of Bayes' Rule to direct the search for the target grammar: while the UG restricts the hypothesis space to only those grammars that are consistent with the principles and parameters defined, the MDL Principle guides the search in this hypothesis space by leading the learner to look for a compact hypothesis that models the data well. The Bayesian Incremental Parameter Setting learner defined by Briscoe [1999] is the learning model adopted for this work. It is adapted to learn a Unification-Based Generalised Categorial Grammar from child directed sentences annotated with logical forms. Such a model provides the testing grounds for linguistic and psychological theories of language acquisition and can potentially provide several insights into language acquisition.

Chapter 3

Grammar Encoding

In a lexicalised linguistic formalism, like Categorial Grammar (CG) or Head-Driven Phrase Structure Grammar (HPSG), the lexicon is the focal point, with most of the information being located there and with the syntactic component being drastically reduced. In such theories different kinds of linguistic information are described in the lexicon. Thus, a lexical entry can contain, for example, information about the orthography, syntax and semantics of a word. These theories are said to be 'sign-based', since they adopt Saussure's idea that a sign is composed by the association of sound, form and meaning ([de Saussure 1916] and [Pollard and Sag 1987]). The issue of how to organise lexical information is essential, since the burden of linguistic description is concentrated in the lexicon. Thus, if lexical entries are organised as unrelated lists, there is a significant loss of generalisation and increase in redundancy.

In this chapter we discuss some grammar encoding techniques to organise lexical information. The first one is the use of inheritance hierarchies, in section 3.1. It is followed by a discussion of the use of type systems in section 3.2. The use of default specifications in the representation of linguistic knowledge is discussed in section 3.3 and the use of lexical rules in section 3.4. Finally, the last two sections provide brief descriptions of the specific versions of default unification operations that are employed in this research.

3.1 Inheritance Hierarchies

When building a lexicon, a lot of the information described in one lexical sign will also be contained in other signs. For example, many of the properties of the verb '*like*' are also shared by the verb '*love*': both are transitive verbs, both have past form ending in '*ed*', both have third person singular form ending in '*s*', and so on. These properties are also shared by a great number of other verbs. As a consequence, a sensible alternative is to define the information common to a group of signs just once, rather than repeat it in each of these signs, and

use inheritance hierarchies to propagate this information. Inheritance hierarchies allow us to do that by providing representations that are able to capture linguistic regularities about classes of items that behave similarly.

The organisation of these objects into class and subclass relations results in a hierarchical structure where linguistic regularities are encoded near the top of the hierarchy; nodes further down the hierarchy are used to represent sub-regularities. As a consequence, all the properties associated with a given class are inherited by all its subclasses and added to the properties defined as idiosyncratic in each subclass. If each subclass in an inheritance hierarchy has only one parent class from where all the properties are inherited, it is classified as a **single inheritance hierarchy**. On the other hand, if a given subclass inherits properties from more than one parent class, it is in a **multiple inheritance hierarchy**. In this case, care must be taken because it is possible for a class to inherit contradictory information from different parents. Furthermore, if all the properties associated with a parent class are inherited by all its subclasses without exceptions or possibility of change, a hierarchy has **monotonic inheritance**. If, however, the properties specified at a given class take precedence over those inherited from its parent classes, the class has **nonmonotonic inheritance**. This is also known as **default inheritance**, since the value for a particular property in a subclass is inherited by default from its parent classes, unless it is specified in the subclass itself, in which case its specification will then take precedence over the inherited value. Monotonic single inheritance hierarchies are easy to build and to understand, but they are not really suited for the description of natural languages, since subregularities and exceptions cannot be easily or straightforwardly defined [Daelemans et al. 1992]. More suited for the description of natural languages are multiple inheritance hierarchies, where different hierarchies can be defined for different properties, and nonmonotonic inheritance hierarchies, where the overriding mechanism can be used to allow for exceptions.

In terms of applications, inheritance in a unification-based grammar processing system can be found in PATR, where, according to Shieber [1986], the use of lexical templates corresponds to a language for defining multiple inheritance hierarchies. One of the theories that has made use of inheritance networks is HPSG. Pollard and Sag [1987] treat the lexicon as a monotonic multiple inheritance hierarchy, where information common to a class of items is inherited by its subclasses, shown in figure 3.1 [from Pollard and Sag 1987, p. 206]. Flickinger et al. [1985] define the lexicon in terms of multiple inheritance hierarchies. They distinguish between two modes of inheritance:

- in the **normal mode** locally declared values override more general values inherited from classes higher up in the hierarchy,
- in the **complete mode** of inheritance the more general values defined in the hierarchy take precedence over the more specific ones.

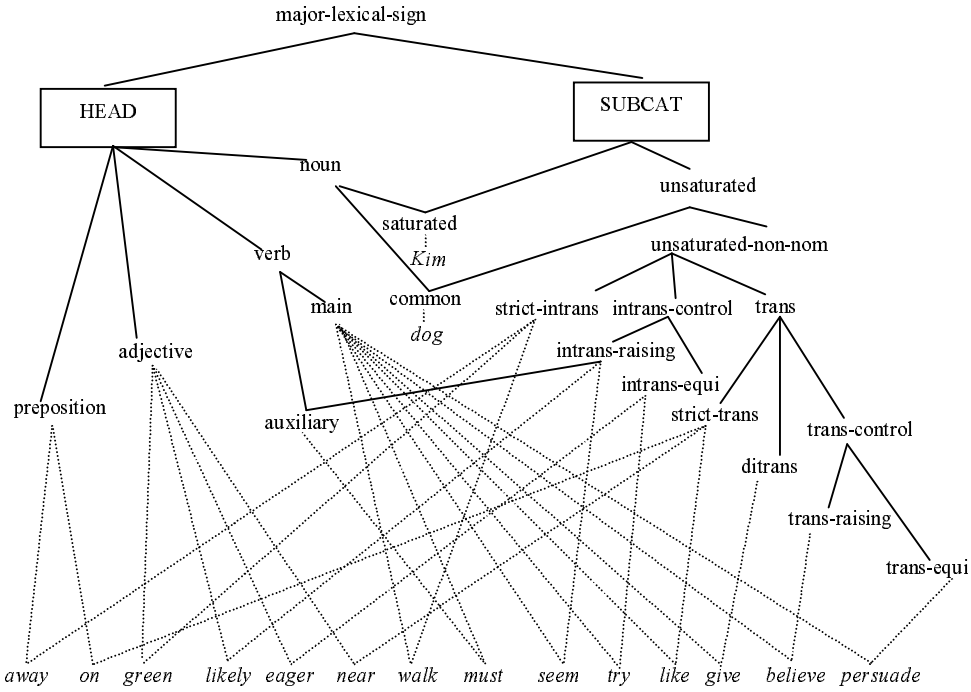

Figure 3.1: Pollard and Sag's Hierarchy

Flickinger and Nerbonne [1992], in their analysis of *easy* adjectives, also highlight the advantages of using inheritance hierarchies for lexical representation, concentrating on its ease of maintenance and modification. Other theories that use multiple inheritance networks are Unification Categorial Grammar [Moens et al 1989] and Word Grammar [Hudson 1990]. DATR is an implemented theory of lexical knowledge representation incorporating multiple default inheritance and, although it requires orthogonality, Evans et al [Evans et al. 1993] show how to encode some kinds of prioritised inheritance.

3.2 Types

In addition to inheritance hierarchies, a typing system can also be used to structure information [Aït-Kaci 1984], [Carpenter 1990], [Carpenter 1992]. Thus, information is typed and structured according to a partial ordering on the types defined. Types place appropriateness conditions on classes, and each element of a class must follow these conditions. The typing system determines which structures are mutually compatible and which attributes can occur for a given class. It can be used to define an inheritance hierarchy, where it imposes appropriateness conditions that are monotonically passed to all the subtypes of the type that

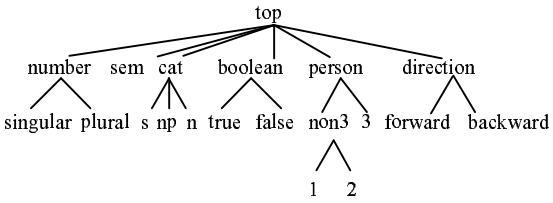

Figure 3.2: Fragment of a Type Hierarchy

$$\begin{bmatrix} \textbf{nominal-m-feats} \\ \text{PRON} : \textbf{boolean} \\ \text{WH} : \textbf{boolean} \\ \text{LOC} : \textbf{boolean} \\ \text{AGR} : \begin{bmatrix} \textbf{agr} \\ \text{NUMBER} : \textbf{number} \\ \text{PERSON} : \textbf{person} \\ \text{GENDER} : \textbf{gender} \end{bmatrix} \end{bmatrix}$$

Figure 3.3: Specification of a Complex Feature Structure

introduced those conditions.

Figure 3.2 shows a type system defining a hierarchy. In the examples that follow, attributes are displayed in uppercase letters and values in lowercase bold face, e.g. CAT and **s**, respectively. The value of an attribute needs to be a type in the hierarchy, and it can also be marked as reentrant with another attribute. **top** is the most basic type defined and is the common supertype of all the types and, inside feature structures it is represented as $\top$. Boxed alphanumeric symbols (e.g. $\boxed{1}$) are used to indicate *reentrancies*, or structure sharing between substructures, with the coindexation being used for representing substructures that are reached by different sequences of attributes. Throughout this document, only the relevant information is shown in the figures for reasons of clarity.

The type hierarchy defines a partial order on the types. The more specific the type, the lower in the hierarchy it is defined. A more specific type has all the information contained in a more general supertype, with possibly some additional information.

Each type in a hierarchy has an associated feature structure (FS) that is appropriate to it. If an FS is basic, it has an atomic value that corresponds to one of the types defined, as can be seen in figure 3.3 for the attributes PRON, WH and LOC, with value **boolean**. If the FS is complex, it provides values for one or more attributes, which in their turn, can be either basic or complex, as in the attribute AGR (figure 3.3). The FS associated with a type acts as a constraint that defines what is appropriate for that type, and it is inherited by all its subtypes. Thus for the type **agr**, the attributes PERSON, NUMBER and GENDER are appropriate attributes.

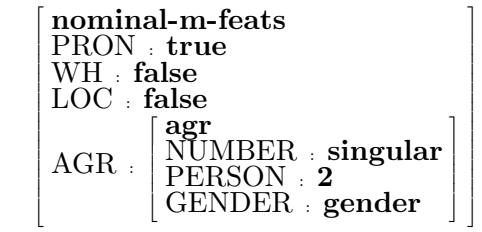

Figure 3.4: Specification of Another Complex Feature Structure

Typed feature structures (TFSs) can be regarded as being ordered by specificity, so that the FS shown in figure 3.4 is more specific than that shown in figure 3.3. The **unification** of two typed feature structures results in the most general feature structure that retains all the information contained in each one of them. There is also a **subsumption** relation between TFSs: a more general one **subsumes** a more specific one. For example, the FS shown in figure 3.3 subsumes that in figure 3.4, since the former is contained in the latter.

In terms of applications of types, Sanfilippo [Sanfilippo 1993] defines a system of verb types for English with a hierarchy of semantic types, a hierarchy of syntactic types, and a hierarchy of signs. The signs combine information from the syntactic and semantic types by coindexing the subcategorised elements and the semantic arguments. The theory of lexical organisation outlined by Pollard and Sag [1987], shown in figure 3.1, can also be defined using a hierarchy of types, where all the types in the **subcat** hierarchy have an attribute SUBCAT encoding a list of subcategorised elements. The analysis of *easy* adjectives proposed by Flickinger and Nerbonne [Flickinger and Nerbonne 1992] can also be described in terms of types organised in a nonmonotonic inheritance hierarchy.

3.3 Defaults

The use of inheritance hierarchies to structure linguistic information, allows generalisations about classes of items to be economically expressed. Moreover, inheritance based on typing provides a way to define appropriateness conditions on classes of linguistic objects. However, if the inheritance hierarchies are defined as monotonic and absolute systems, like Pollard and Sag's [1987], they fail to make use of defaults which would significantly reduce redundancy in lexical specifications and would enable them to straightforwardly express sub-regularities.

As several authors have highlighted, defaults are important in the representation of linguistic knowledge, providing linguistically adequate descriptions for natural language phenomena [Gazdar 1987], [Daelemans et al. 1992], [Bouma 1992], [Krieger and Nerbonne 1993], [Briscoe 1993].

Among the characteristics that make the use of multiple default inheritance

networks so appealing, it is possible to cite [from Daelemans et al. 1992]:

- parsimony - inheritance lexicons should be smaller than the full-entry counterparts,
- ease of maintenance - changes or corrections are localised to a few nodes,
- uniformity - different levels of linguistic descriptions are encoded in the same manner and are subject to the same rules of inference,
- modularity - different taxonomies can apply for different levels of description,
- interaction - properties at different levels of description interact with one another,
- the ability to capture linguistic generalisations,
- the ability to capture linguistic subregularities and exceptions with the overriding mechanism,
- the ability to provide non-redundant representations.

Defaults can be viewed as a restricted form of nonmonotonicity that may be used only for lexical generalisations and the defaults can be configured to act as monotonic constraints outside the lexicon [Bouma 1990], [Krieger and Nerbonne 1993]. Default networks allow properties to be incrementally true, so that things can be considered true by default until further evidence is discovered that indicates that they are false (and they are then overridden). Moreover, the use of default inheritance hierarchies is not only motivated for considerations of parsimony and conciseness, but also by psychological considerations, since speakers recognise systematic relationships among words, as pointed out by Sag and Wasow [1999].

An example of the use of defaults for linguistic description is found in [Lascarides and Copestake 1999] in a treatment of modals that uses defaults to allow *ought* to be an exception to the general class of modals. Modals can be inverted ('*can*' in '*Can I use the car?*'), can be negated without requiring the presence of '*do*' ('*will*' in '*I will not go to the theatre tomorrow*'), and can have the contracted negated form ('*shouldn't*' in '*He shouldn't stay there longer than five days*'). The specification of the **modal** type is shown in figure 3.5 [from Lascarides and Copestake 1999]. In this figure, the non-default (or indefeasible) value of an attribute is separated with '/' from the default (or defeasible) value, so that they are represented as *Indefeasible/Defeasible*. This specification can be abbreviated to *Indefeasible*, if *Indefeasible* = *Defeasible* and abbreviated to */Defeasible* if $\top$*/Defeasible*, with $\top$ (**top**), being the most general type. This notation follows Lascarides et al. [1996] and is adopted throughout this document.

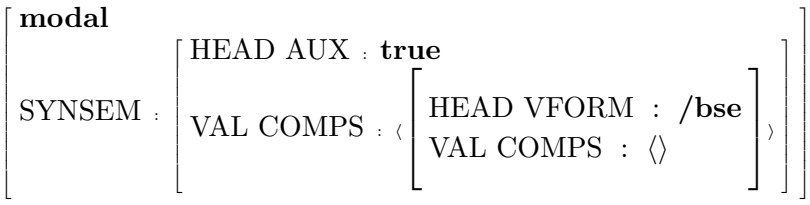

Figure 3.5: Constraint on Type **modal**

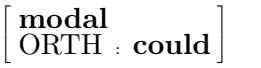

Figure 3.6: Lexical Description for *could*

Figure 3.5 shows the definition of some syntactic characteristics of a modal verb in the SYNSEM attribute. Firstly, it is a verbal category which has the behaviour of auxiliary verbs, encoded in SYNSEM:HEAD:AUX:**true**. Moreover, in terms of valence, encoded in the VAL attribute, a modal verb is specified as subcategorising for a verb phrase that is in the base form (SYNSEM:VAL COMPS:<HEAD VFORM:/**bse**>) and has an empty complements list (SYNSEM:VAL COMPS:<VAL COMPS:<>>). This is the type associated with most modals, such as '*could*', which is encoded as shown in figure 3.6. Since '*could*' does not introduce any other constraint, it inherits all the information encoded in the **modal** type, without exception, as can be seen in figure 3.7, where '*could*' is shown expanded. *Ought*, on the other hand, has the syntactic behaviour of modals in most respects, but unlike most modals it requires a *to*-infinitive verb as a complement, being an exception to the general case. Thus, in its lexical entry, *ought* defines the requirement for an infinitival verb phrase (SYNSEM:VAL COMPS:<HEAD VFORM:**inf**>), as shown in figure 3.8. Since the **modal** type specifies the form of the subcategorised verb phrase as default information (SYNSEM:VAL COMPS:<HEAD VFORM:/**bse**>) (figure 3.5), even though '*ought*' inherits most of the information from this type, it also overrides this default information, being expanded as shown in figure 3.9.

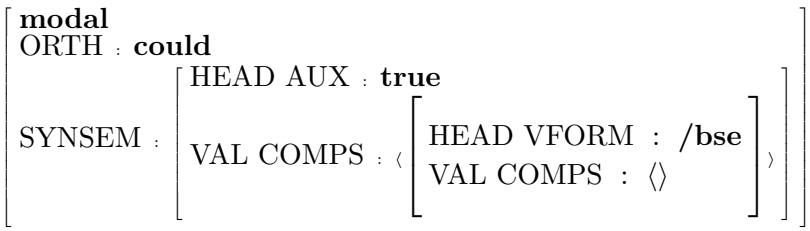

Figure 3.7: Expanded Feature Structure for *could*

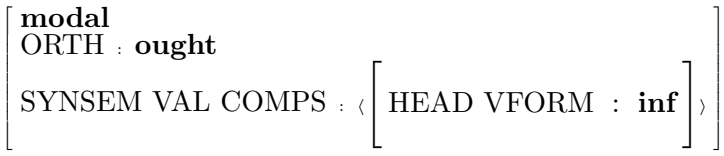

Figure 3.8: Lexical Description for *ought*

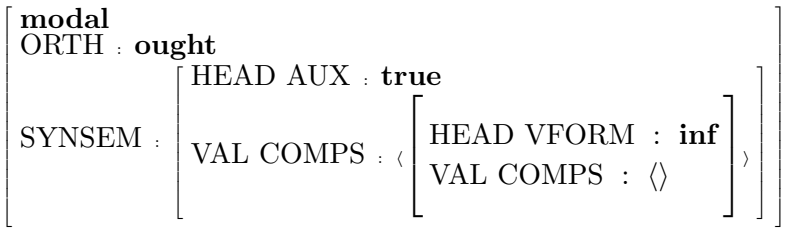

Figure 3.9: Expanded Feature Structure for *ought*

In the case of modal verbs, the use of default specifications provides a way of capturing the generalisation that most modals take a base form complement. At the same time, it licenses the integration of exceptions, such as *ought* that takes an infinitival complement.

This is only one of the possible applications for defaults in the definition of linguistic phenomena. Defaults can also be used for encoding inflectional morphology [de Smedt 1984], [Flickinger et al. 1985], [Briscoe and Copestake 1999], and they can be used in the description of rules, schemata or constructions [Gazdar 1987], [Lascarides and Copestake 1999], among other applications. Throughout these and other works[1], defaults have been successfully used to express the idea that natural languages have general constraints, but that there are usually exceptions to these constraints.

3.4 Lexical Rules

Another mechanism that can be used to further reduce redundancy in lexical specification is that of the lexical rule. Lexical rules are used to generate a lexical entry based on information from another entry. The lexicon can then be divided into two groups of entries: basic and derived entries, and lexical rules can be used to construct a predictably related derived entry from a basic or another derived entry. In a lexicalist approach to grammar, they have a major role of capturing lexical and morphological generalisations, such as plural formation and

[1]For an overview of the use of defaults see [Gazdar 1987], [Daelemans et al. 1992], [Krieger and Nerbonne 1993], [Lascarides et al. 1996b], [Malouf 1998], and [Sag and Wasow 1999].

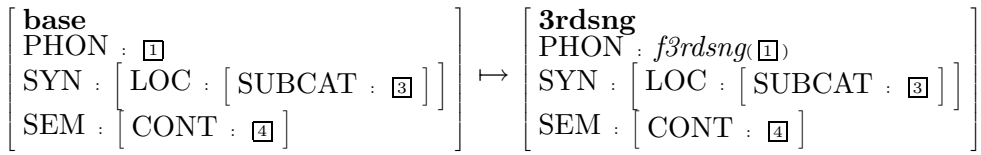

Figure 3.10: HPSG Third Singular Verb Formation Lexical Rule

Figure 3.11: Abbreviated Form of the Lexical Rule

verb alternations.

A common interpretation of lexical rules is as conditional relationships between lexical entries. Thus, a lexical rule is applied to an input lexical entry to generate a new entry that is related to the input in terms of form, syntactic category and semantics. For instance, lexical rules can add morphological affixes to lexical entries, e.g. by generating the third person singular form of a verb from its base form. In this way, with an input and an output specification for lexical rules, when a lexical entry unifies with the input or antecedent description of a rule, a new derived entry is created by copying specified elements to the output or consequent description. The output will be defined in terms of idiosyncratic information that is copied or computed from the particular input lexical entry and predictable information that is available via inheritance links. In these rules, coindexations are usually interpreted as copying operators - as opposed to their usage as reentrancies within a sign - copying information from the input to the output of a rule. As an example, the rule of third singular verb formation is shown in figure 3.10 [from Pollard and Sag 1987]. This rule specifies that the input structure, in the left hand sign, should be a verbal lexical entry in the base form, whose syntactic and semantic characteristics are copied to the output structure (SYN:LOC:SUBCAT: 3, and SEM:CONT: 4), in the right hand side. This rule also establishes that the orthography of the output structure is determined by a function (*f3rdsng*) that takes the orthographic form of the input and returns an inflected word corresponding to the third person singular form (PHON:*f3rdsng*(1)). However, in terms of notation, it is common in the literature for this rule to be abbreviated to the one shown in figure 3.11, with the elements that are to be copied unchanged from input to output left unstated.

Lexical rules are a widely used mechanism in lexical representation. For instance, Flickinger et al [Flickinger et al. 1985] use lexical rules as one of the mechanisms for eliminating redundancy in lexical representation. They also combine the use of inheritance hierarchies which, among other things, allow them to specify the domain to which a lexical rule is to apply. These two mechanisms are

Figure 3.12: Reinterpreted Third Singular Verb Formation Lexical Rule

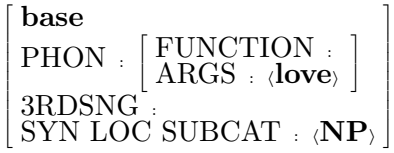

Figure 3.13: Lexical Description for the Base Form of *love*

also used for **blocking**, where the application of a lexical rule to create a regular form for a particular word is prevented by the existence of an irregular form of that word. This is achieved by allowing a lexical entry, in its specification, to explicitly define its appropriate irregular form. Sanfilippo [Sanfilippo 1993] also employs lexical rules, but constrains their applications by specifying in the lexical entries a list containing the rules that can be applied to them. In this way, a lexical rule will only be applied to an entry that contains that rule name in the list of possible rules. Lexical rules are also widely used by Sag and Wasow [1999], which, among other rules, define those for capturing inflectional morphology.

As pointed out by Briscoe and Copestake [1999], the approach of treating lexical rules as a homogeneous class introduces a series of problems. For instance, the use of lexical rules to perform arbitrary manipulations of lists results in the possibility of generating any recursively enumerable language [Carpenter 1991], posing no limits on expressivity.

An alternative approach suggested by Briscoe and Copestake [1999] is to define lexical rules using asymmetric default unification. By using asymmetric default unification the output structure is identical to the input, except where otherwise specified in the output. This operation allows the carrying over of consistent information from the input to the output structures. This can be seen in figure 3.12 [from Briscoe and Copestake 1999] where the third person singular verb formation rule is reformulated in terms of asymmetric default unification.

This rule specifies that the third person singular form of a verb is obtained from a base form input (**base**, in the left hand side of the rule). In the right hand side of the rule, a function applied to the orthography of the word generates the required form, in the attribute 3RDSNG. When this rule is applied to a lexical item, like the word '*love*' shown in figure 3.13, the input, shown in the right hand side, is treated as defeasible and the output, shown in the left hand side, as indefeasible, figure 3.14.

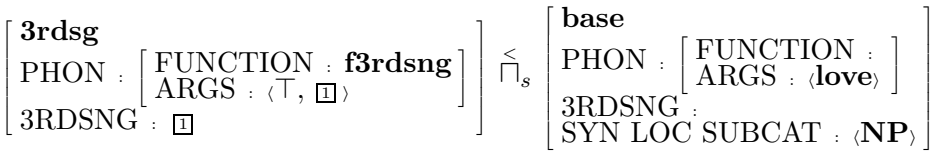

Figure 3.14: Asymmetric Default Unification of Output and Input Structures

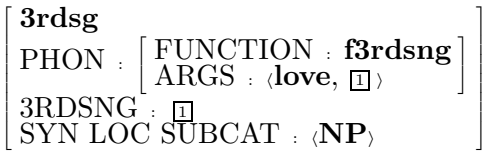

Figure 3.15: Application of the Third Singular Verb Formation Lexical Rule

The result shown in figure 3.15 contains all the information from the indefeasible output specification and also all the consistent information from the defeasible input specification. Thus, if the input contains information that is inconsistent with the output, then this information is lost. However, if the output contains information that is incompatible with the input, this information survives. In terms of linguistic applications, lexical rules based on asymmetric unification are used in the treatment of dative constructions proposed by Briscoe and Copestake [1999]. This treatment captures the idea of a family of dative constructions [Goldberg 1995] with the same syntactic and related semantic properties.

By generating a lexical entry based on information from another entry, lexical rules help to reduce redundancy in lexical specification. Moreover, asymmetric default unification successfully allows the encoding of lexical rules by only defining what is to be changed and by automatically carrying over what is to remain unchanged from the input to the output form. The use of asymmetric unification in the specification of lexical rules results in a more restricted mechanism than that used in conventional lexical rules, since it cannot arbitrarily 'rearrange' material between the input and the output structures [Briscoe and Copestake 1999].

3.5 Asymmetric Default Unification

In this section, we describe the particular version of asymmetric default unification used in this work to implement lexical rules.

In the asymmetric default unification of two FSs the first one is treated as an indefeasible structure and the second one as defeasible. The result of the asymmetric unification is equal to the indefeasible FS but also incorporates compatible information from the defeasible FS. This operation is non-deterministic and may generate more than one possible resulting structure. When this happens, the

result is taken to be the generalisation of the unified structures, implementing **Skeptical Default Unification** as defined, for example, by Carpenter [1993]. In this way, the skeptical default unification of two FSs only maintains default information that does not conflict in any way with the non-default information, and this is the information that can be found in every possible resulting structure. The skeptical unification, $\overset{<}{\sqcap}_s$, of two structures is shown in figure 3.16. In this figure, **true** and **false** are types in the hierarchy shown in figure 3.2, being both subtypes of **boolean** and incompatible for unification. As the generalisation of two types is their lowest upper bound in the hierarchy, these two types generalise to their supertype **boolean**. The asymmetric unification of the two FSs in figure 3.16.a generates two possible structures, the first one ignoring the reentrancies and the second one maintaining them. Then these need to be generalised for a skeptical unification, as in figure 3.16.b, where both the value for F, which is **true** in the two FSs, and the value for H, which is **false** in both of them, are kept in the generalisation of the two FSs. However, as the value of G is **false** in one FS and **true** in the other, in their generalisation G has value **boolean** which is the common supertype of **true** and **false**.

$$\text{a.} \begin{bmatrix} \text{F} : \textbf{true} \\ \text{G} : \textbf{boolean} \\ \text{H} : \textbf{false} \end{bmatrix} \overset{<}{\sqcap}_S \begin{bmatrix} \text{F} : \boxed{1}\ \textbf{false} \\ \text{G} : \boxed{1} \end{bmatrix} = \left\{ \begin{bmatrix} \text{F} : \textbf{true} \\ \text{G} : \textbf{false} \\ \text{H} : \textbf{false} \end{bmatrix}, \begin{bmatrix} \text{F} : \boxed{1}\ \textbf{true} \\ \text{G} : \boxed{1} \\ \text{H} : \textbf{false} \end{bmatrix} \right\}$$

$$\text{b.} \qquad = \begin{bmatrix} \text{F} : \textbf{true} \\ \text{G} : \textbf{boolean} \\ \text{H} : \textbf{false} \end{bmatrix}$$

Figure 3.16: Skeptical Asymmetric Default Unification

This version of skeptical unification is extended by Briscoe and Copestake [1999] for typed feature structures. Thus, the unification of two TFSs is only allowed if the resulting TFS has only attributes that are appropriate for that particular type.[2]

3.6 Symmetric Default Unification

In this section we describe the version of symmetric default unification used to implement the default multiple inheritance network of types that represents linguistic information throughout this work.

YADU [Lascarides et al. 1996b], [Lascarides and Copestake 1999] is an order independent default unification operation on typed feature structures. YADU can be regarded as a default operation that tries to incorporate the maximal amount

[2] A more detailed description of asymmetric default unification can be found in [Carpenter 1993] and [Briscoe and Copestake 1999].

of information given its priority. The information is encoded in features that have both non-default and default values. In order to distinguish non-default from default information, YADU uses an extended definition of TFSs called typed default feature structures (TDFSs). A TDFS is composed of:

- an indefeasible part (I), which contains the non-default information,
- a defeasible part (D), which contains the default information, and
- a tail T, which is composed of a set of pairs where the first element is a path or path equivalence and the second is the associated type, recording unification history.

As a consequence, during default unification, non-default information can always be preserved and only consistent default information is incorporated into the defeasible TFS. Another important point is that default unification of two FSs is deterministic, always returning a single value.

There are basically three operations: $\overset{<>}{\sqcap}$, *DefFS* and *DefFill*. $\overset{<>}{\sqcap}$ can be informally defined as an operation that takes two TDFSs and produces a new one where:

- the indefeasible part is the result of unifying the indefeasible information defined in the input TDFSs, and
- the tail results from the union of the tails, with all the elements that are incompatible with the resulting indefeasible part I removed.

The defeasible part of the TDFS is calculated using the operation *DefFS*, which defines the defeasible part as the result of unifying the indefeasible TFS with the maximal set of compatible values of the tail, in order of specificity (given by the position of the associated types in the hierarchy). As tails may contain conflicting information, this operation is used in order to know which elements in the tail 'win'. To compute the result of the default unification of two FSs, it is first necessary to apply $\overset{<>}{\sqcap}$ to compute I, the indefeasible FS, and T, the tail, and then to apply *DefFS* to compute D, the defeasible FS, as shown in the following example.

Given that $\mathbf{t}' \sqsubset \mathbf{t}$ ($\mathbf{t}'$ is more specific than $\mathbf{t}$) and that $\top$ represents the most general type in the hierarchy, the symmetric default unification of two FSs:

1. $$\begin{bmatrix} \mathbf{t}' \\ \mathrm{F} : \top/\mathbf{a} \\ \mathrm{G} : \top/\mathbf{b} \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top/\boxed{1} \\ \mathrm{G} : \top/\boxed{1} \end{bmatrix}$$

   is computed by unifying the indefeasible parts of the two FSs:

   - $$I = \begin{bmatrix} \mathbf{t}' \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \sqcap \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} = \begin{bmatrix} \mathbf{t}' \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix}$$

and by taking the set union of the two tails:

- $T = \{\langle \left[\,\mathrm{F} : \mathbf{a}\,\right], \mathbf{t}' \rangle, \langle \left[\,\mathrm{G} : \mathbf{b}\,\right], \mathbf{t}' \rangle, \langle \left[\begin{smallmatrix}\mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1}\end{smallmatrix}\right], \mathbf{t} \rangle\}$

Then it is possible to calculate the resulting defeasible structure:

$$
\begin{aligned}
2.\quad D_{12} &= \sqcup\left(\left(\begin{bmatrix} \mathbf{t}' \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \overset{<}{\sqcap}_s \left\{ \left[\,\mathrm{F} : \mathbf{a}\,\right], \left[\,\mathrm{G} : \mathbf{b}\,\right] \right\} \right) \overset{<}{\sqcap}_s \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix} \right) \\
&= \sqcup\left(\begin{bmatrix} \mathbf{t}' \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix} \overset{<}{\sqcap}_s \begin{bmatrix} \mathrm{F} : \boxed{1} \\ \mathrm{G} : \boxed{1} \end{bmatrix} \right) \\
&= \begin{bmatrix} \mathbf{t}' \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix}
\end{aligned}
$$

where the defaults are added in order of specificity of types, with the constraints introduced by $\mathbf{t}'$ (F:**a**, G:**b**) being added first and then the constraints introduced by $\mathbf{t}$ (F:$\boxed{1}$, G:$\boxed{1}$). As the latter are incompatible with the resulting FS, they are ignored.

The third operation, *DefFill* turns default specifications into non-default information. It has a TDFS as input and returns a TFS by simply taking the defeasible TFS constructed by *DefFS*. Taking the previous result we have that:

$$
\begin{aligned}
&\mathit{DefFill}\left(\begin{bmatrix} \mathbf{t}' \\ \mathrm{F} : \top/\mathbf{a} \\ \mathrm{G} : \top/\mathbf{b} \end{bmatrix} / \{\langle \left[\,\mathrm{F} : \mathbf{a}\,\right], \mathbf{t}' \rangle, \langle \left[\,\mathrm{G} : \mathbf{b}\,\right], \mathbf{t}' \rangle, \langle \left[\,\mathrm{F} : \boxed{1}\,\right], \mathbf{t} \rangle, \langle \left[\,\mathrm{G} : \boxed{1}\,\right], \mathbf{t} \rangle\} \right) \\
&= \begin{bmatrix} \mathbf{t}' \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix}
\end{aligned}
$$

The *DefFill* operation can be executed, for example, at the interface between the lexicon and the rest of the system. This operation has to be applied to all the defaults that are only used in the grammar to capture lexical generalisations, and they have to be made indefeasible after the lexicon; otherwise they may be overridden during later processing. These defaults are considered to be non-persistent defaults. YADU also provides the possibility of defining defaults that are going to contribute to the output of the grammar: the persistent defaults. These defaults are marked to persist outside the lexicon with the p operator [Lascarides et al. 1996b]. This operator has already been shown to be important for keeping lexically encoded semantic defaults until discourse processing, where they can be overridden by discourse information when appropriate [Lascarides et al. 1996a].

Furthermore, YADU supports the definition of inequalities. The inequality relation defined between two nodes is a restricted form of stating negative information in terms of FSs, as described in [Carpenter 1992]. Thus, inequalities are a way of defining that two FSs are not equivalent, so that they never become identical after an inequality is encountered. Inequalities can be used to override

default reentrancies in the unification of two types when no conflicting values are defined in the types involved [Lascarides and Copestake 1999].

By using YADU, defaults become an integrated part of the typed feature structure system implemented. It achieves the perspicuity, declarativity and expressivity that are familiar from unification-based approaches to non-default inheritance.[3]

3.7 Summary

In this chapter several mechanisms for grammatical implementation were discussed. They all contribute to the construction of a structured lexicon that is succinct and avoids redundancy in the specification of linguistic information. They allow the representation of generalisations present in natural languages, while also describing subregularities and exceptions that do not fully follow the general case, but nonetheless have many properties in common. Finally, the two versions of default unification used throughout this work were presented.

[3] A more detailed description of YADU can be found in [Lascarides and Copestake 1999].

Chapter 4

Unification-Based Generalised Categorial Grammar

This chapter provides a description of the grammar implemented in the scope of this research. The grammar is formalised as a Unification-Based Generalised Categorial Grammar, but the description of types provided is meant to be compatible with several theoretical approaches. By using a radically lexicalised theory of grammar, the lexical types defined contain a great amount of information. Therefore, the use of a different framework would only require the removal or translation of the linguistic information from the lexical types to another component of the grammar.

This chapter starts with an introduction to Categorial Grammars. Then, in section 4.2, Unification-Based Generalised Categorial Grammars, the particular version of CGs being used in this work is presented, followed by a description of the set of rules, in section 4.3, and by an analysis of the coverage obtained by such a grammar, in section 4.4. Section 4.5 provides a detailed account of how the implemented grammar is used to formalise a theory of the UG and section 4.6 describes the corpus used to set the parameters of the UG.

4.1 Categorial Grammars

Categorial Grammar [Ajdukiewicz 1935], [Bar Hillel 1953] and [Lambek 1958] is a lexicalised grammatical formalism. It is characterised by regarding language in terms of functions and arguments and by allowing the representation of semantics directly in syntax with each rule of syntax being a rule of semantics.

As a lexicalised formalism, the syntactic behaviour of any item is directly encoded in the lexicon, as the item's lexical category specification. In this way, practically all the information about how words can be combined into phrases is present in their lexical entries: a linguistic entity is described in terms of categories with which it can combine and the result of this combination. For

example, the attributive adjective *big* has its syntactic behaviour encoded as N/N, meaning that it needs an N to its right, denoted by '/N', to form a noun 'N'. As a consequence, there is no need for a separate grammar rule component that describes the constituents that need to be combined and the constituent resulting from the combination. Instead, CGs have a small set of rules that are applied to the categories corresponding to the words of a given sentence to check the legality of the sentence and to build its semantic interpretation.

A CG is composed of a lexicon and a small set of rules. Many versions of CGs have been defined, and they differ in terms of the set of categories defined and/or in terms of the set of rules used (e.g. [Lambek 1958], [van Benthem 1988], [Morrill 1987], [Moortgat 1988], [Carpenter 1998] and [Steedman 2000]). This discussion starts with a description of AB-CG, the basic CG, and is followed by a description of other configurations that extend either the set of categories, or the set of rules, or both.

4.1.1 AB-Categorial Grammar

The basic version of CG, known as AB-Categorial Grammar (AB-CG), or classical Categorial Grammar, is defined by Ajdukiewicz [Ajdukiewicz 1935] and Bar-Hillel [Bar Hillel 1953]. In CGs categories can be defined as either basic or complex. AB-CG has a set of basic categories that contains S (for sentences) and N (for nouns). Complex categories, are composed of more than one category separated by forward or backward slashes, which are operators over categories. Given that X and Y are variables over categories, the directional slash operators can be described as follows:

- the **forward slash** is used with rightward combining categories, represented as X/Y, denoting that this category needs to combine with a category Y to its right to yield X, and
- the **backward slash** operates over leftward combining categories, X\Y, indicating that the category needs to combine with a category Y to its left to yield X.

Complex categories can be seen as incomplete units: a category X/Y, for example, needs an element of category Y to give as a result category X. In terms of function applications category X/Y is a function from the **argument** Y (the subcategorised category that is needed) to a complete expression X which is known as the **result** or **value**. In this way, an adjective that has category N/N is identified as a function from N (the argument) into N (the result), indicating that it is a category which needs an N to form another N. Throughout this document, the notation used specifies a complex category with the argument always to the right of the slash and the result to the left.

The set of categories is the inductive closure of the set of basic categories under the slash operators, such that it contains the basic categories and the categories that can be defined using the slash operators. As categories can be defined recursively, if X and Y are categories, X/Y and X\Y are also categories. As a consequence, categories can have more than one argument and mix these two kinds of slashes (e.g. (X\Y)/Z), being able to combine with different arguments in different directions. However, in languages in general the number of arguments of a complex category is small. For instance, for English the maximum number seems to be five, for a small number of verbs such as *bet* (*I bet you five pounds for Red Rum to win* [from Briscoe 1999]).

These categories are combined by means of the **Functional Application Rule**. This rule simply combines a function with its argument. In this way, a complex category that needs an argument is applied to an adjacent category that is equal to the argument needed. This rule has two versions:

- the **Forward Application** rule allows a rightward-combining category (X/Y) to combine with an argument (Y) to its right. This rule, which in a derivation is represented as '>', can be defined as:

  X/Y Y → X

  Figure 4.1 shows the use of the forward application rule to combine the adjective *big*, which has category N/N, with the noun *dog*, which has category N.

  Throughout this document, in a derivation, the combination of categories is indicated by a solid line underlining the categories involved, ending with the symbol of the rule used and with the resulting category written underneath.

- the **Backward Application** rule (<) is used to combine a leftward-combining category (X\Y) with its argument (Y) to the left:

  Y X\Y → X

Although AB-CG, with two atomic categories (S and N) and two rules (forward and backward application), can capture some linguistic constructions, it is considered to be inadequate to handle natural languages even by its authors.

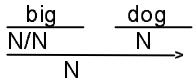

Figure 4.1: Using Forward Application

4.1.2 Extensions to AB-Categorial Grammar

Most of the work in the area extends AB-CG in terms of the basic categories defined and/or in terms of the rules used. The basic category set is extended by including categories like NP (for noun phrases) PP (for prepositional phrases), VP (for verb phrases), ADJP (for adjectival phrases), ADVP (adverbial phrases), among others. In relation to the set of rules, the following are some of the rules that are used in different configurations of CGs.

The rule of **Associativity** (A) states that a function with two arguments, one on either side, can combine with them in either order:

- (Y\X)/Z → (Y/Z)\X
- (Y/Z)\X → (Y\X)/Z

Associativity can be used to combine, for example, a transitive verb syntactically with either the object or the subject first, as can be seen in figures 4.2 and 4.3, in the derivations of the sentence *John likes Mary*. This rule guarantees that the semantics of the verb is still intact, with the subject and the object being assigned the correct roles in the semantics, even if the order in which they are combined has changed.

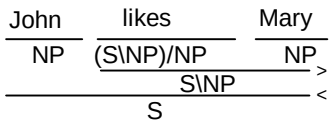

Figure 4.2: First Derivation of Sentence *John likes Mary*

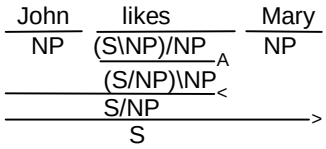

Figure 4.3: Second Derivation of Sentence *John likes Mary*

The rule of **Functional Composition** allows a functor category missing an argument to compose with an adjacent function that outputs that argument as its result. For example, the word *will* is an auxiliary verb with category (S\NP)/(S\NP) and the word *buy* is a transitive verb with category (S\NP)/NP.

Thus *will* and *buy* can be combined by composition to form the constituent *will buy* with category (S\NP)/NP. There are two versions of this rule:

- **Forward Composition** (>**B**) allows the combination of an item (X/Y) with another item (Y/Z) to the right:

  X/Y Y/Z → X/Z

  This rule is used in the derivation of the sentence *Mary will buy the car*, as shown in figure 4.4.

- **Backward Composition** (<**B**) allows the combination of the argument of a complex category (X\Y) with the result of the category to its left (Y\Z):

  Y\Z X\Y → X\Z

While application must absorb a complete constituent, composition is used to combine contiguous strings that do not form what is traditionally known as a constituent. In this way, it is possible to obtain items like *will buy* and *Mary will buy* by composing their categories. CGs, unlike other formalisms, consider *will buy* and *Mary will buy* genuine constituents because they form prosodic units and they have a coherent semantics [Steedman 1990].

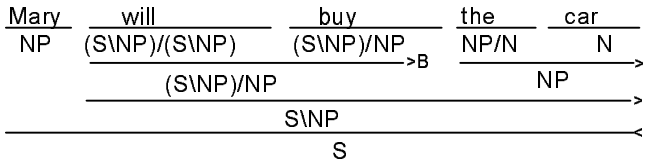

Figure 4.4: Using Forward Composition

Type Raising rules turn arguments into functions over functions over such arguments. For example, given that S\NP is a function over argument NP, through type raising the category NP becomes the category S/(S\NP), i.e. a function (S/) over a function over NP (S\NP). By allowing arguments to turn into functions, it is, then, possible to compose them with other functions. There are two versions of this rule:

- **Forward Type Raising** (>T) allows the transformation of category X into a higher order category function with the purpose of combining it with

a constituent to its right:

$X \rightarrow Y/(Y\backslash X)$

This rule is used to transform subjects from NPs into S/(S\NP). It is used in the derivation of sentence *Mary smells and I paint the flower*, as shown in figure 4.5.

- **Backward Type Raising** (<T) allows a category X to be raised into a higher order category function in order to be combined with a constituent to its left:

  $X \rightarrow Y\backslash(Y/X)$

  It is used to transform indirect objects into functions from ditransitive verbs into transitive verbs and direct objects into functions from transitive to intransitive verbs. This rule can be used to capture, among other constructions, coordinations such as the one in the sentence *Jane shows Mary the pictures and John the paintings*, whose derivation is shown in figure 4.6.

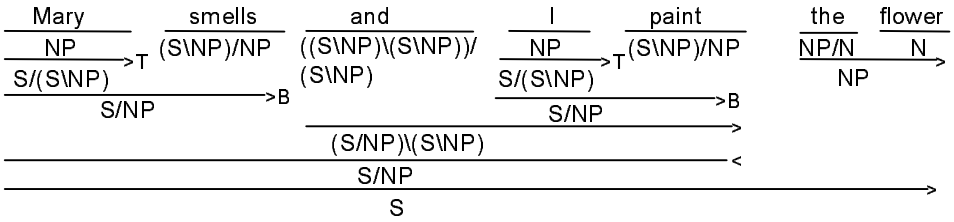

Figure 4.5: Using Forward Type Raising

Raising extends the weak generative power of the grammar in the description of non-standard constituent structures. Since raising is a recursive rule, an argument can be raised over its functor, which can be raised over the new category of the original argument and so on *ad infinitum*. However, this recursion can be restricted, for example, by constraining the raised categories to those defined in the lexicon or by applying the rule only when there is no other way to obtain a derivation.

The **Division** (D) rules are unary rules that are applied to both sides of the main slash operator in a functor category:

- $X/Y \rightarrow (X/Z)/(Y/Z)$
- $X\backslash Y \rightarrow (Z\backslash X)\backslash(Z\backslash Y)$

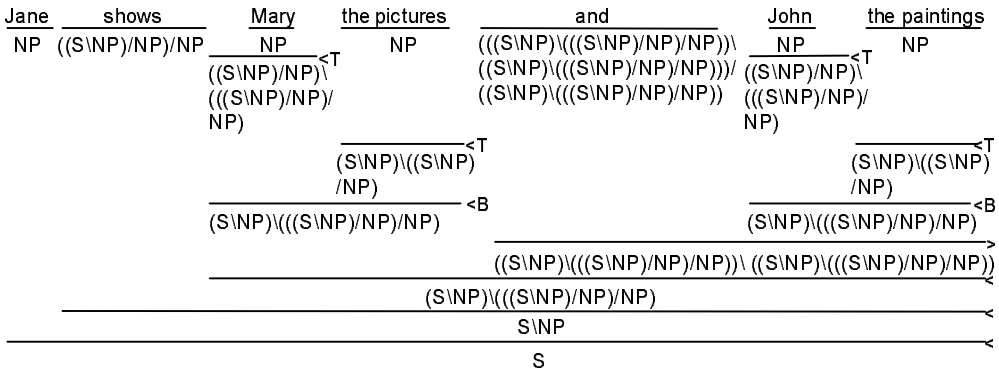

Figure 4.6: Using Backward Type Raising and Backward Composition

Lambek [1958] used this rule to show that a sentence-modifying adverb is also a predicate-modifying adverb: S\S → (S\NP)\(S\NP), being able to generate alternative analyses for the sentence *Bill runs here*, figures 4.7 and 4.8.

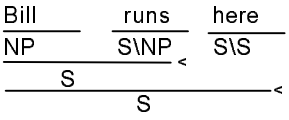

Figure 4.7: First Derivation of Sentence *Bill runs here*

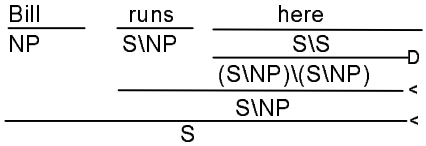

Figure 4.8: Second Derivation of Sentence *Bill runs here*

Division preserves word order and logical implication, but, due to the fact that it is recursive, it causes the extension of the power of the system to structural completeness. This means that any substring can be a constituent and any item in the substring can be its head [Wood 1993]. It leads to the extreme case of spurious ambiguity, where different derivations are generated with the same semantic interpretation.

The **Backward Crossing Substitution** is one of the Substitution rules and it is used in the treatment of parasitic gaps such as the one in the sentence *Mary*

will copy and file without reading any articles longer than ten thousand words, from Steedman [2000]:

- **Backward Crossing Substitution** (<Sx): $Y/Z\ (X\backslash Y)/Z \rightarrow X/Z$

The rule of **Generalised Weak Permutation** (GWP) [Briscoe 1997] rotates the arguments of a complex category, allowing a functor category to combine with its arguments in any order, while keeping the original direction associated with each argument:

- **Generalised Weak Permutation** (P): $((X|Y_1)\ldots|Y_n) \rightarrow (((X|Y_n)|Y_1)\ldots)$

where '|' is a variable over '/' and '\'. If the arguments of a complex category are ordered in a way that does not allow the category to be combined, the GWP rule rotates the arguments until they are in an appropriate order for combining with an adjacent category. However, the number of possible permutation operations over a category is finite and bounded by the number of arguments of the category in question. Thus, for example, a functor category with three arguments has three different argument sequences, as can be seen in the case of an oblique transitive verb like *donates*:

- $((S\backslash NP_1)/PP)/NP_2$,
- $((S/NP_2)\backslash NP_1)/PP$ and
- $((S/PP)/NP_2)\backslash NP_1$

In practical terms the application of GWP can be controlled, for example, by maintaining a count of the number of permutations performed with respect to the possible permutations for a given category.

While the associativity rule is restricted to be applied to functor categories with two arguments, the permutation rule can be applied to a functor category with any number of arguments, which can be combined in any order.

Since GWP can generate all the different orderings of arguments for a given complex category, this rule allows the grammar to capture more flexible constituent orders. For instance, the oblique transitive verb *donate*, whose initial category is $((S\backslash NP_1)/NP_2)/PP$, captures not only sentences where the subcategorised constituents follow this initial ordering as in *She donated to the school those computers*, but also the ones where the ordering is different as in *She donated those computers to the school*. The derivations for these sentences are shown in figures 4.9 and 4.10 respectively. As a consequence, there is no need to define extra categories for capturing more flexible constituent orders if they differ in the order in which constituents are combined, because these are already produced by GWP. This rule is also used to capture, among other constructions, unbounded

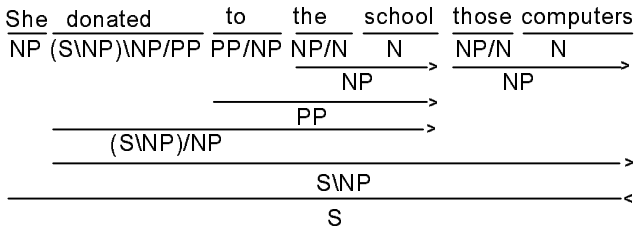

Figure 4.9: Derivation of Sentence *She donated to the school those computers*

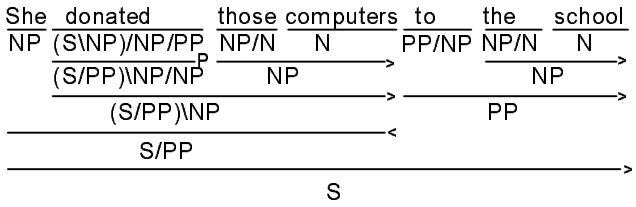

Figure 4.10: Derivation of Sentence *She donated those computers to the school*

dependencies, such as the one in the sentence *Who does Goldie love?*, which are discussed in section 4.4.

Associativity, Composition, Division and GWP introduce so-called "spurious ambiguity", which is the property where alternative constituent structures for a sentence have the same semantic interpretation. Figures 4.11 and 4.12 show two alternative semantically equivalent derivations for the sentence *Jane eats the cake*. Wood [1993] notes that in most cases spurious ambiguity will not affect the grammaticality judgements for particular sentences. It rather causes a processing problem, since for a given sentence, it allows many derivations to be generated that have equivalent semantic interpretations. Steedman [1991] also notes that the alternative derivations generated, far from being spurious, are genuine ambiguities at the level of Information Structure. Moreover, as the spurious ambiguity caused by these rules is not considered to be a problem in the scope of this research, this issue is not going to be addressed here.

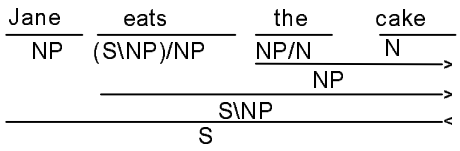

Figure 4.11: First Derivation of Sentence *Jane eats the cake*

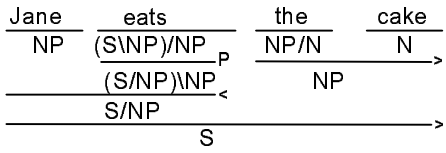

Figure 4.12: Second Derivation of Sentence *Jane eats the cake*

The rules described in this section are some of those that can be used to extend the basic AB-CGs. Apart from them, some proposals also extend the set of operators used to build complex categories. In this case, besides the two slash operators, a third one known as the **product operator** ($*$) [Lambek 1958] can be used. The product operator is used to form an ordered pair from two adjacent categories. Thus a complex category that needs to combine with two arguments (e.g. (X/Y)/Z), one at a time, by two operations of functional application can combine with the product of these arguments (Z*Y) in a single operation. For instance, the verb 'gave', with category ((S\NP)/NP)/NP, can also be assigned the equivalent category (S\NP)/(NP*NP). Wood [1989] uses this equivalence to capture coordinations such as the one in the sentence *He gave Bill a guitar and Helen a piano* (figure 4.13).

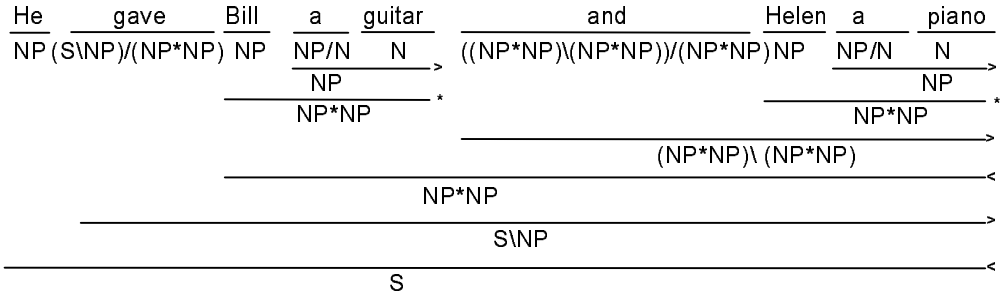

Figure 4.13: Derivation of Sentence *He gave Bill a guitar and Helen a piano*

Another possible extension of the basic CGs is obtained by using **combinators**. Combinators "*bear a striking resemblance to the 'combinators' which Curry and Feys [1958] use to define the foundations of the lambda calculus and all applicative systems - that is, systems expressing the operation of functional application and abstraction*' [Steedman 1988]. The combinators used in Steedman's **Combinatory Categorial Grammars** (CCGs) are three: **B** (functional composition), **T** (type raising), and **S** (substitution).

1. **B** is the combinator for functional composition. Given the functions F and G, the use of this combinator results in the composition of these two

functions, whose semantics is given by the following identity:

- **B**FG=λx F(Gx)

where F and G represent functions, and x and y represent their arguments. Thus, the rule of Forward Composition can be rewritten as:

- X/Y:F Y/Z:G $\Rightarrow$ X/Z: λx F(Gx)

indicating a category X/Y with interpretation F which is forward composed with a category Y/Z with interpretation G, giving as a result a category X/Z with interpretation **B**FG. Given the example in figure 4.4, the application of this rule to *will* and *buy* results in the appropriate interpretation: λx.λy.will(buy(x,y))

2. **T** is the combinator for type raising, which is defined as:

   - **T**x=λF Fx.

   A function of two arguments is mapped onto a function of one argument, that identifies the two arguments.

3. **S**, the functional substitution combinator, has semantics defined according to the following equivalence:

   - **S**FG=λx Fx(Gx)

   This rule allows a two-argument function (X\Y)/Z with interpretation F to be combined with a one-argument function Y/Z with interpretation G, yielding a function of category X/Z, which has interpretation **S**FG.

Steedman's proposal of CCGs respects formal and linguistic considerations, with these combinators being used to express, in a natural way, almost all rules needed for natural language description.

It is also possible to define a CG as a **unification-based grammar**, where objects are described in terms of attribute-value pairs. Using such an approach, specific attributes can be defined, for example, to account for agreement. In this case, the information related to number and person, for example, is specified in the appropriate attributes and can be added to a category like N. Thus, a noun like *dogs* has the value 'plural' for the *number* attribute and the value 'third' for the *person* attribute. Furthermore, attributes can have variables as values. For example, in the case of a determiner like *the*, which can be used with singular as well as with plural words, the value for the *number* attribute is a variable, allowing it to be used with both singular and plural words. Different ways of

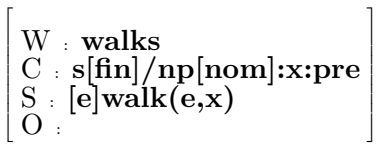

Figure 4.14: UCG's Encoding of *walks*

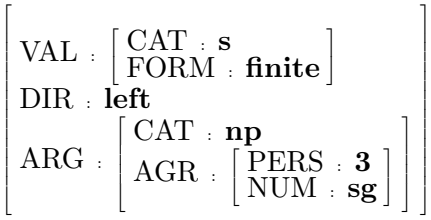

Figure 4.15: CUG's Encoding of *runs*

combining the principles of CGs and those of unification-based formalisms were proposed. In what follows, some of these proposals are briefly discussed.

Unification Categorial Grammar (UCG) [Zeevat et al 1987], [Zeevat 1988] is a proposal of a CG enriched by some aspects of unification. In UCG, a verb such as *walks* is encoded as shown in figure 4.14 where:

- W specifies that the orthography of the word is *walks*,
- C that it is an intransitive verb which needs an argument NP in the nominative case preceding the verb (pre) to form a finite sentence (s[fin]), and the NP has its semantics represented by the variable 'x',
- S specifies its semantics, and
- O is left unspecified.

In CCGs, Steedman [1996] regards the basic categories like S and NP as complex objects that include the use of attribute-value pairs. Thus, a third person singular word such as *loves* is represented as $(S\backslash NP_{3s})/NP$, where NP_{3s} specifies that the subject should be a third person singular NP. However, he does not discuss the attributes used or how they should be encoded in the categories.

Categorial Unification Grammar (CUG) [Uszkoreit 1986] encodes the essential properties of both formalisms: unification and Categorial Grammars. CUG describes an intransitive verb such as *runs* as shown in figure 4.15, where **VAL** describes the functor, **DIR** gives the direction of the argument and **ARG** has the specifications of the argument.

The advantage is that CUG is a general framework, while UCG is a specific individual theory of CG that adopts some insights from unification-based formalisms. As a consequence UCG is not as general as CUG.

These are some of the possible extensions to the basic AB-CG that have been proposed. Different versions of CG use these extensions in different configurations. For example, Lambek Calculus [Lambek 1958] uses the rules of application, composition, associativity, raising and division, and the product operator.[1] Steedman's CCG [Steedman 1988 to 2000] includes S, N, NP, PP and CONJ in the category set and uses the combinators and the rules of application, composition, type raising, substitution and coordination. Hoffman [1995] uses multi-sets CCGs to define a system that is able to handle scrambling elegantly, capturing languages with freer word order than English, like Turkish. CCGs are also used by Doran and Srinivas [1998] and Hockenmaier et al. [2000] as wide-coverage grammars.

4.2 UB-GCGs

Unification-Based Generalised Categorial Grammars (UB-GCGs), the particular version of CG used in this work, extend basic CGs, by defining the basic categories in terms of bundles of attribute-value pairs, as in CUG, and by using an extended set of operators and rules, also in terms of attribute-value pairs: directional slashes and the product operator, the rules of application, composition and generalised weak permutation. There are five basic categories: **S** (sentence), **N** (noun), **NP** (noun phrase), **PRT** (particle) and **PP** (prepositional phrase); other categories are defined in terms of these. This particular configuration of CG was chosen because it could successfully capture the constructions found in the corpus, as discussed in section 4.6, without adding unnecessary descriptive power.[2]

The UB-GCG implemented follows the sign-based approach to linguistic analysis [Pollard and Sag 1987], where words and categories are represented in terms of TDFSs, with different kinds of linguistic information being simultaneously represented as a conjunction of attribute-value pairs that form a **sign**. A sign is composed of three attributes (figure 4.16):

- ORTH encodes the orthographic description of words,
- CAT encodes syntactic aspects related to the categories, and
- SEM encodes the semantics associated with a particular word.

[1] For an extensive description of the Lambek Calculus see [Moortgat 1988].

[2] The descriptive power of a very similar system is investigated by Hoffman [1995], and she demonstrates that such system is able to successfully handle long-distance scrambling, generating some mildly context-sensitive languages.

$$\begin{bmatrix} \textbf{sign} \\ \text{ORTH} : \textbf{diff-list} \\ \text{CAT} : \textbf{cat} \\ \text{SEM} : \textbf{sem} \end{bmatrix}$$

Figure 4.16: Sign

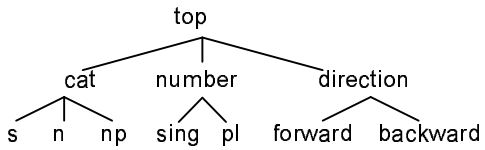

Figure 4.17: A Type Hierarchy

The values that these attributes take correspond to types in a hierarchy. Figure 4.17 shows a small fragment of the complete hierarchy. An attribute specified as taking a type as value can also take any of its subtypes as valid values. This is shown in figures 4.16, for the general description of a sign, and 4.18, for the more specific description of a noun phrase sign. In these FSs **np** is a subtype of **cat** in the hierarchy defined in figure 4.17.

In the next sections, the theories used to encode the syntactic, semantic and linking information in the grammar are presented.

4.2.1 Syntax

The syntactic properties of signs are formalised in the attribute CAT. Categories are divided in terms of valence, depending on whether they are saturated or unsaturated. Atomic categories are saturated which means that they do not subcategorise for any other categories [Wood 1993]. The categories in the set of basic categories are saturated: S, NP, N, PP and PRT. They are distinguished in terms of appropriate feature instantiation, as shown for the NP sign in figure 4.18.

Complex categories are unsaturated, subcategorising for other categories. For instance, an intransitive verb subcategorises for an NP subject to the left and results in an S category, with its category being S\NP, while a preposition sub-

$$\begin{bmatrix} \textbf{np-sign} \\ \text{ORTH} : \textbf{diff-list} \\ \text{CAT} : \textbf{np} \\ \text{SEM} : \textbf{sem} \end{bmatrix}$$

Figure 4.18: NP sign

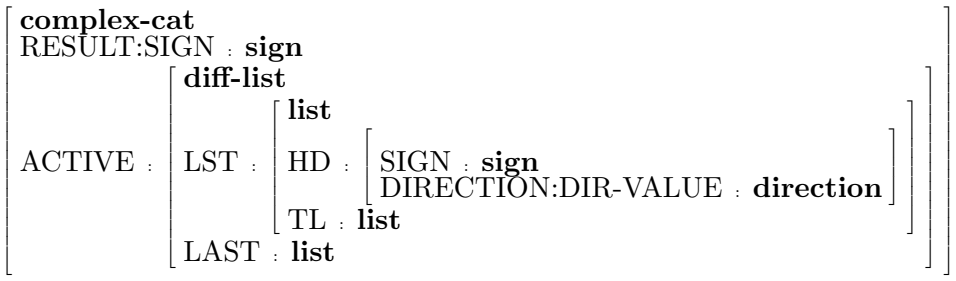

Figure 4.19: Complex Category Type

categorises for an NP to the right, to form a PP, with category PP/NP. In terms of feature structures, complex categories are recursively defined in terms of two attributes following Steedman [1988]:

- RESULT describes the functor category, and
- ACTIVE describes the list of subcategorised argument categories,

both defined in CAT. RESULT has one attribute, SIGN (with ORTH, CAT and SEM), which is instantiated according to the particular result category. ACTIVE encodes the subcategorisation list of the category, where each of the subcategorised categories has two attributes: SIGN and DIRECTION, figure 4.19. The attribute DIRECTION encodes the direction in which the category is to be combined, where DIR-VALUE can be **forward** or **backward**, corresponding to the forward and backward slash operators, respectively. The categories defined in RESULT and in ACTIVE can be either atomic or complex themselves. As an example, an intransitive verb (S\NP) is encoded as shown in figure 4.20. The ACTIVE attribute is implemented as a difference list, with attributes LST (list) and LAST (last). Both can have as value a list, which can be either a non-empty list with head (HD) and tail (TL), or an empty-list with value **e-list**. In a non-empty list the head takes any valid value, and the tail takes a list which can be either empty or non-empty. A difference list is a special structure that maintains a pointer to the end of the list, with the last TL element in LST being coindexed with LAST (figure 4.20). In unification-based encoding, in general, difference lists are used as a way of appending lists using only unification [Copestake 2002]. Figure 4.20 shows the encoding of an intransitive verb whose only argument is defined in LST:HD. It has the LAST attribute specified by default as **e-list**, marking the end of the list, and coindexed with LST:TL. This notation for difference lists can be represented in an abbreviated form where '<!' marks the beginning of the list (LST:HD) and '!>' marks the end of the list (LAST), and the elements in the subcategorisation list are separated by commas (figures 4.21 and 4.22).

This notation for CGs is based on the one used by Steedman [1991a], which was modified to allow a concise implementation of a default inheritance hierarchy

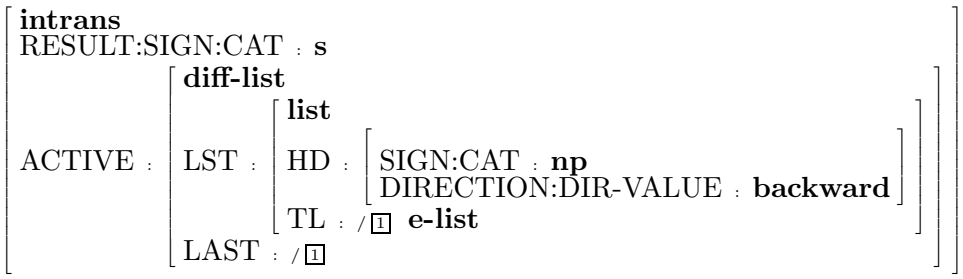

Figure 4.20: Intransitive Verb Type

Figure 4.21: Intransitive Verb Type Abbreviated

of types, where information is defined only once and propagated through the hierarchy. In Steedman's notation, a complex category is defined as a nested structure, as can be seen in figures 4.23 and 4.24 for intransitive verbs and transitive verbs, respectively. On the other hand, in this work a complex category is defined as a list of sequentially defined subcategorised elements, with each element encoding not only the SIGN, but also the DIRECTION attribute. In terms of the inheritance hierarchy, this means that a given subtype inherits the ACTIVE list from its supertype. If the subtype subcategorises for more categories, these are added in the end of the list, overriding the inherited default end of the list ($/ ! >$). This can be seen in figure 4.22 that shows **trans**, which is a subtype of **intrans**, encoding transitive verbs. In this figure, the first subcategorised element in the active list is the most general type $\top$ which can represent any category. **Trans** is expanded as shown in figure 4.25, having all the information contained in **intrans**, where $\top$ is unified with the first subcategorised element in **intrans**, the NP subject. **Trans** also adds to the end of the ACTIVE list another NP category, with direction **forward**, and a default end of the list ($/! >$).

These defaults are only used for encoding generalisations and are transformed into indefeasible information using the *DefFill* operation. *DefFill* can be used, for example, at the interface with the lexicon, so that, unless otherwise required, lexical entries have only non-default information. For instance, a transitive verb has the default specifications in **trans** changed to non-default (figure 4.26), where the end of the list is made indefeasible.

$$\left[\begin{array}{l}\textbf{trans} \\ \text{ACTIVE} : <^{!}\left[\top\right], \left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np} \\ \text{DIRECTION:DIR-VALUE} : \textbf{forward}\end{array}\right]/^{!}>\end{array}\right]$$

Figure 4.22: Transitive Verb Type

$$\left[\begin{array}{l}\textbf{intrans} \\ \text{RESULT:SIGN:CAT} : \textbf{s} \\ \text{DIRECTION} : \textbf{backward} \\ \text{ACTIVE:SIGN:CAT} : \textbf{np}\end{array}\right]$$

Figure 4.23: Traditional Encoding for Intransitive Verbs

$$\left[\begin{array}{ll}\textbf{trans} & \\ \text{RESULT} : & \left[\begin{array}{l}\text{RESULT:SIGN:CAT} : \textbf{s} \\ \text{DIRECTION} : \textbf{backward} \\ \text{ACTIVE:SIGN:CAT} : \textbf{np}\end{array}\right] \\ \text{DIRECTION} : & \textbf{forward} \\ \text{ACTIVE:SIGN:CAT} : & \textbf{np}\end{array}\right]$$

Figure 4.24: Traditional Encoding for Transitive Verbs

4.2.2 Semantics

The semantic properties of words are represented using a neo-Davidsonian style event-based approach to semantics [Parsons 1980] [Sanfilippo 1990], [Copestake et al. 1998], [Sag and Wasow 1999]. It is based on the approach developed by Davidson [1967], where the basic idea is that propositions involve explicit reference to events. Davidson proposed that events should be treated as primitive ontological elements and should be explicitly defined in the logical form of verbs expressing actions, with such verbs including an argument place for event terms. For example, in the logical form of the sentence *Fido chases the cat in the park* the event term 'e' is an argument of both the verbal and the prepositional predicates:

λe.chase(e,Fido,the(cat)) $\wedge$ in(e,the(park))

Such an approach provides a natural account of the entailment relation between sentences. For example, the sentences 4.1 and 4.2 differ only in terms of the modifier *in the park*.

- (4.1) *Fido chases the cat in the park*
- (4.2) *Fido chases the cat*

The first sentence entails the second: if the first sentence is true (that Fido

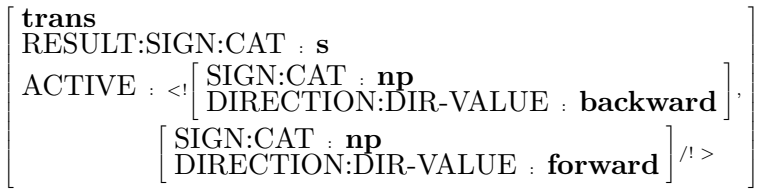

Figure 4.25: Transitive Verb Type Expanded

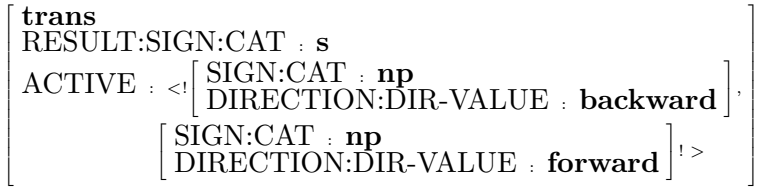

Figure 4.26: Transitive Verb Type Expanded and *DefFilled*

is chasing the cat in the park), then the second one (that Fido is chasing) is also true, but the converse does not hold. This is reflected in their logical forms, respectively:

- λe.chase(e,Fido,the(cat)) $\wedge$ in(e,the(park))
- λe.chase(e,Fido,the(cat))

In this work a neo-Davidsonian style semantics is implemented in terms of a variant approach of Minimal Recursion Semantics (MRS) [Copestake et al. 1998]. MRS is characterised as being a flat semantic framework that does not use embedded structures, with the elements within a flat representation being relations. To simulate recursive embedding, MRS uses identifiers, which are unified with the role arguments of other relations. However, unlike other approaches to flat semantics, MRS provides a treatment of standard quantification. Moreover, MRS can be easily integrated in a feature-based grammar.

The MRS variant implemented in this work follows Sag and Wasow's version [Sag and Wasow 1999] which is a simplification of the MRS defined by Copestake et al. [1998]. Although the grammar implemented uses this simplified version, it could be easily extended to a complete MRS account. In Sag and Wasow's version, semantic objects are implemented as TDFSs describing events or situations (SIT), e.g., a running event in *Bill is running*. Each of these events is classified in relation to its semantic mode (MODE) as:

- propositions (**prop**),
- questions (**quest**), or

- directives (**dir**).

Besides the semantic mode, two other attributes are also used to describe semantic objects: an index (INDEX), which corresponds to the event or individual referred to, and a restriction (RESTR), which specifies the list of conditions that has to be fulfilled for the semantic expression to be applicable. INDEX can take one of three values, according to the object it is referring to: **index**, corresponding to situations, **null-index**, corresponding to a null object and **imp-index** corresponding to the implicit subject of an imperative sentence. The conditions associated with individuals and situations that have to be satisfied for the expression to hold are described by RESTR. Figure 4.27 shows the specification of the general semantic type. These restrictions are described in terms of predications, which provide a general way of specifying relations among semantic objects. A predication specifies what kind of relation is involved and which objects are taking part in it, as shown in figure 4.28 for a *love* relation. Verbs have attributes representing the actor or agent role (e.g. LOVER) and the undergoer or patient role (e.g. LOVED). A more general nomenclature for semantic roles is being adopted in this thesis, but it is compatible with and can be easily mapped to that defined by Davis [1996] or Sag and Wasow [1999]. Thus, for instance, LOVER is mapped to ARG1 and LOVED to ARG2. As a consequence, the predication in figure 4.28 is generalisable to the one in figure 4.29.

$$\begin{bmatrix} \textbf{sem} \\ \text{MODE} : \textbf{mode} \\ \text{INDEX} : \textbf{index} \\ \text{RESTR} : \textbf{diff-list} \end{bmatrix}$$

Figure 4.27: Semantic Type

$$\begin{bmatrix} \textbf{restr} \\ \text{RELN} : \textbf{love} \\ \text{SIT} : \textbf{index} \\ \text{LOVER} : \textbf{index} \\ \text{LOVED} : \textbf{index} \end{bmatrix}$$

Figure 4.28: Sample Predication in RESTR

The role arguments within predications are represented in the attributes ARG1 to ARGN. An element with predication semantics, such as a verb, specifies links between the indices of the role arguments in its predication and the other semantic predicates on the RESTR list. This can be seen in figure 4.30, which shows the semantics for the verb *love* in the sentence *Bill loves Mary*. In this figure, the *love* predicate specifies that ARG1 (the LOVER role) is coindexed with the predicate representing *Bill*, with tag $\boxed{1}$, while ARG2 (the LOVED role) is coindexed with the predicate representing *Mary*, with tag $\boxed{2}$.

Figure 4.29: Equivalent Sample Predication RESTR

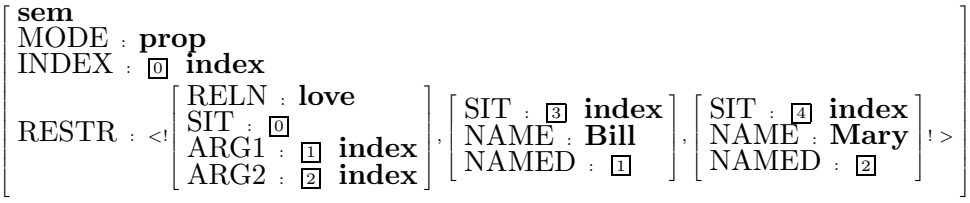

Figure 4.30: Logical Form of the sentence *Bill loves Mary*

The construction of the RESTR list for a sentence adopts the *Compositionality Principle*, which states that the meaning of a constituent is composed of the meaning of its parts. Thus, in a derivation, the resulting RESTR value (figure 4.30) is the sum of the RESTR values of the combined categories, with each word in the sentence being assigned an appropriate semantic value.

4.2.3 Linking

Linking theories try to capture regularities in the mapping between the semantic argument structure and the syntactic subcategorisation frame of a predicator. For instance, the agent role in an action denoted by a verb is usually realised as the subject of the verb, as in *The dog chases the cat* where the verb's subject, *the dog*, is the agent role of the *chasing* action, while *the cat* is the affected patient role of the action and is realised as the verb's direct object. The linking generalisations adopted in the grammar are represented as constraints in an inheritance hierarchy, following the approaches defined by Wechsler [1995] and Davis [1996]. The specified linking types encode constraints between semantic roles and syntactic arguments, with each linking type partly defining the mapping between them. The constraints associated with linking types specify the semantics of the predicators they apply to, and the various constraints that apply to a given type combine to determine, at least partly, its subcategorisation frame. On the other hand, subcategorisation is not considered to be completely semantically driven, being partly arbitrary, as in the case of semantic roles that are optionally realised syntactically. Thus it is also necessary to specify some subcategorisation information in the linking types, taking into account the notion that subcategorisation can be partly independent of semantics. In this way, it is the combination of the independent specifications of a predicate semantic arguments and syntactic

subcategorisation frame that determine the predicate's linking pattern.

Wechsler's HPSG formulation of linking constraints uses four different structures to encode and link the relevant mapping between semantic and syntactic arguments. On the syntactic side, subcategorisation frames are defined in two attributes: SUBJ, containing the subject, and COMPS, containing the other complements. These two attributes are the locus of phrase structure concatenation, as determined by the Valence Principle [Pollard and Sag 1994]. Another attribute, ROLES, contains the union of the SUBJ and COMPS lists. This list is the locus of lexicosemantic generalisations about argument structure. CONTENT contains the appropriate argument structure. For example, the verb *love* is defined as shown in figure 4.31.

$$\begin{bmatrix} \text{SUBJ} : \textbf{<NP[nom]>} \\ \text{COMPS} : \textbf{<NP>} \\ \text{ROLES} : \textbf{<NP[nom],NP>} \\ \text{CONTENT} : \begin{bmatrix} \text{REL} : \textbf{love} \\ \text{LOVER} : \textbf{index} \\ \text{LOVED} : \textbf{index} \end{bmatrix} \end{bmatrix}$$

Figure 4.31: Lexical Description of *love*

The Linking Principle as defined by Wechsler specifies that:

- the SUBJ list item is reentrant with the leftmost syntactically possible ROLES list item (for English, NPs or complementiser-introduced clauses);
- the COMPS list contains any remaining ROLES list items.

The ordered elements in ROLES are mapped to the semantic arguments in the CONTENT attributes. These constraints are specified as follows, with the semantic arguments in CONTENT generically specified as ARG1 to ARGN:

$$\begin{bmatrix} \text{SUBJ} : \langle \boxed{1} \rangle \\ \text{COMPS} : \langle \boxed{2} \ldots \boxed{n} \rangle \\ \text{ROLES} : \langle \boxed{1}_{\boxed{3}}, \boxed{2}_{\boxed{4}}, \ldots \boxed{n}_{\boxed{m}} \rangle \\ \text{CONTENT} : \begin{bmatrix} \text{REL} : \textbf{rel} \\ \text{ARG1} : \boxed{3} \\ \text{ARG2} : \boxed{4} \\ \vdots \ \ldots \\ \text{ARGN} : \boxed{m} \end{bmatrix} \end{bmatrix}$$

Figure 4.32: The Linking Principle

When this principle is applied to a lexical sign like *love*, the sign shown in figure 4.33 is produced.

This HPSG formulation has to be translated to the UB-GCG equivalent, where the SUBJ and COMPS lists are combined together in the CAT:ACTIVE list,

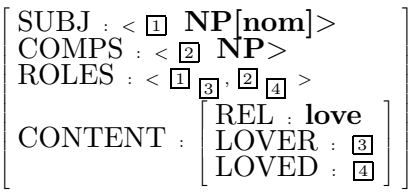

Figure 4.33: Linking Principle Applied to *love* - HPSG Formulation

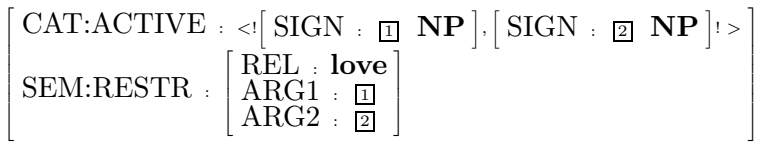

Figure 4.34: Linking Principle Applied to *love* - UB-GCG Formulation

ROLES is non-existent, and CONTENT corresponds to SEM:RESTR, as shown in figure 4.34. The basic difference between these two formulations is that the HPSG version encodes separately the subject, in SUBJ, and the other verbal complements, in COMPS, and they are ordered for linking according to the roles they perform in ROLES. The UB-GCG formulation specifies all these aspects directly in CAT:ACTIVE, where each subcategorised element is ordered and is directly linked to its role in SEM:RESTR.

The formalisation of the linking theory adopted follows Davis' [1996] use of inheritance hierarchies to define linking relations. In this way, linking generalisations can be encoded in types higher up the hierarchy to be propagated throughout the network. As a consequence, the linking constraints defined in a given type are inherited by its subtypes which can also add extra constraints, making effective use of the inheritance mechanism.

4.2.4 The Grammar

The UB-GCG is implemented as a default inheritance network of types in order to structure the linguistic information defined and reduce the problem of redundancy in lexical specification [Villavicencio 1999], [Villavicencio 2000b]. The network of types is implemented using YADU, the symmetric default unification operation described in section 3.6, with different hierarchies defined for different levels of description. The main hierarchies are for syntax, semantics, linking, and for their combination in the resulting signs. The constraints on the types are defined in terms of TDFSs, which contain only the appropriate attributes associated with a particular type. The mechanisms of grammar implementation discussed in chapter 3 are all employed in the construction of this grammar and their application is discussed in the following sections. Even though these mechanisms

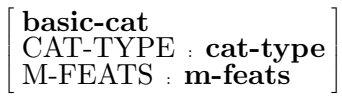

Figure 4.35: Basic Category

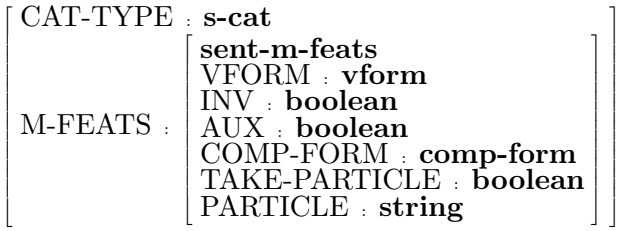

Figure 4.36: S Category

have been successfully used in the description of several linguistic phenomena, the discussion in this section concentrates on their application to structure the lexicon. In the next sections, the main hierarchies are described, focusing mostly on the verbal hierarchy which is rich enough to illustrate the advantages of the approach chosen.

Syntactic Hierarchy

The syntactic dimension of the grammar is described using the attribute CAT in the types associated with the different parts-of-speech. As **S** (sentence), **NP** (noun phrase), **N** (noun), **PRT** (particle) and **PP** (prepositional phrase) are the basic categories, as defined in section 4.2.1, the other categories are defined in terms of these. The basic categories are distinguished in terms of the values of their attributes CAT-TYPE that encodes the category type, and M-FEATS that encodes morphosyntactic information (figure 4.35). The attribute M-FEATS can be further specified to reflect the particular characteristics of each of the basic categories: SENT-M-FEATS for the verbal categories (**S**), NOMINAL-M-FEATS for the other categories (**N**, **NP**, **PRT** and **PP**).

S categories need to encode aspects such as the particular form (VFORM) of a verb (e.g. base, past, and so on), if the verb is an auxiliary (AUX), if it is in the inverted or canonical form (INV) and so forth (figure 4.36).

The other categories encode information about agreement (AGR) and case (CASE), among other aspects. By default a nominal category is defined as having third person singular agreement (AGR:/**3sg**), being countable (COUNT:/**true**), not being a pronoun (PRON:/**false**), not being a locative word (LOC:/**false**) or a *wh*-word (WH:/**false**) (figure 4.37). These defaults are overridden by more specific information or by lexical rules. CAT-TYPE and M-FEATS also indicate whether the sign corresponds to:

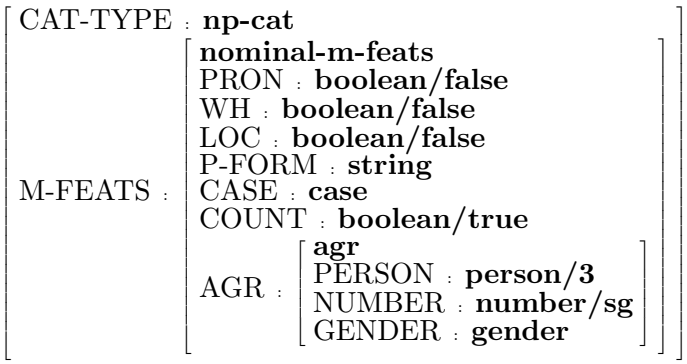

Figure 4.37: NP Category

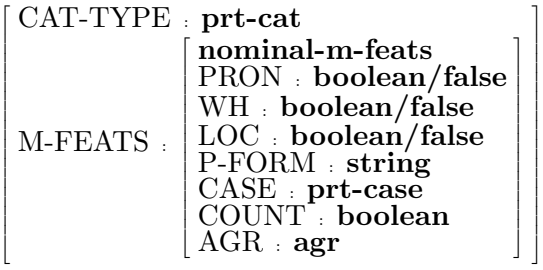

Figure 4.38: PRT Category

- a regular noun (e.g. *apple*, in *He ate the apple*, where CAT-TYPE:**n-cat** and CASE:**reg-case**),
- a noun phrase (e.g. *Bill*, in *Bill runs*, where CAT-TYPE:**np-cat** and CASE:**reg-case**),
- an oblique prepositional phrase argument of a verb (e.g. *to Mary*, in *Bill talks to Mary*, where CAT-TYPE:**p-cat** and CASE:**p-case**), or
- a particle (figure 4.38) attached to a verb (e.g. *up* in *Jane warmed up the milk*, where CAT-TYPE:**prt-cat** and CASE:**prt-case**).

This approach follows Sanfilippo [1993] and Carpenter [1998]. Thus, Ns, NPs, PPs and PRTs are distinguished in terms of the value of the attributes CAT-TYPE and CASE. For regular nouns and noun phrases, the case can be further specified as being accusative or nominative, as appropriate. For both prepositional phrases (PPs) and particles (PRTs), it is important to define the specific preposition or particle form being used and this is done in the P-FORM attribute (e.g. P-FORM:**to** in *Bill talks to Mary*).

For reasons of clarity, throughout this document only the relevant attributes are shown, with the morphological attributes of the basic categories being omit-

ted, unless necessary. Regular nouns are represented as Ns, noun-phrases as NPs, oblique argument prepositional phrases as PPs and particles as PRTs.

Complex categories subcategorise for other categories, which are specified in the ACTIVE list. In the syntactic hierarchy, types defining complex categories are ordered according to the result constituent and the category and number of the subcategorisation arguments they specify. A fragment of the syntactic dimension of the verbal hierarchy, which is based on the sketch by Pollard and Sag [1987], is shown in figure 4.39.

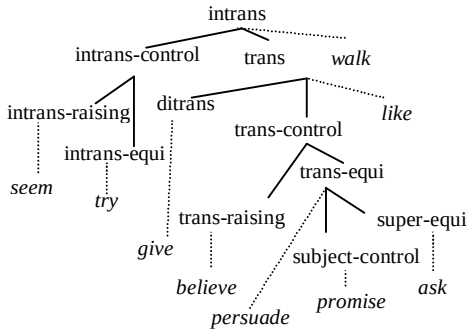

Figure 4.39: The Proposed Hierarchy

In the verbal hierarchy the subcategorised elements in the active list are ordered according to obliqueness [Dowty 1982], where subjects are the least oblique, objects are the next most oblique, followed by indirect objects and finally by oblique objects. The basic category, **intrans**, corresponds to the one assigned to intransitive verbs (S\NP), figure 4.21, repeated in figure 4.40. It specifies the need for exactly one subcategorised argument, the NP subject. The last element of the subcategorisation list is, by default, '! >', which marks the end of the ACTIVE list.

In terms of morphosyntactic attributes, verbs are defined as being by default in the base form (VFORM:/**base**), non-inverted (INV:/**false**) main verbs (AUX:/**false**), figure 4.40. These defaults can latter be overridden by more specific information or by lexical rules. For instance, auxiliary verb signs override the default AUX:/**false** with value **true**. All other syntactic verbal types are defined as subtypes of **intrans**, representing the generalisation that all verbs subcategorise for at least an NP category. All the attributes specified in **intrans**, such as that the NP should be in the nominative case, are inherited by instances of this type and by its subtypes[3]: for example, **trans** for transitive verbs, and **intrans-**

[3]In this work the coverage condition, or closed world assumption [Carpenter 1992], which

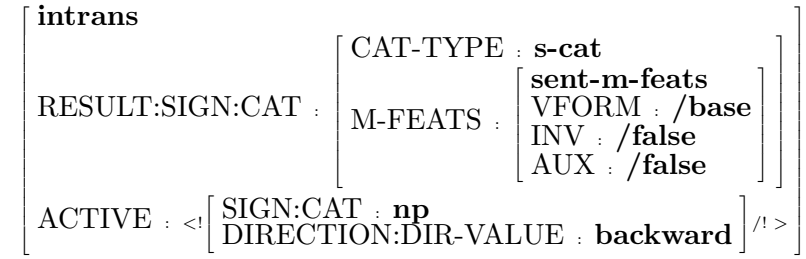

Figure 4.40: Intransitive Verb Type

Figure 4.41: Transitive Verb Type

control for intransitive control verbs, among others. However, since these types subcategorise for 2 arguments, they need to override the inherited default end of the list (/! >) and specify the addition of an extra complement: **trans** specifies an extra NP complement ((S\NP)/NP) (figure 4.22 repeated here in figure 4.41), and **intrans-control** an extra (S\NP) complement ((S\NP)/(S\NP)). Furthermore, each of these types also specifies the default end of the list (/! >), signalling that these categories subcategorise by default for exactly 2 elements.

Similarly, the instances and subtypes of **trans-control**, which is a subtype of **trans** (figure 4.39), inherit the default information that the predicative complement is controlled by the object. This is the case of the subtype **trans-equi** (figure 4.42), for transitive-equi verbs, exemplified in sentence 4.3 with the verb *persuade*. In this sentence, *Mary* is the object of *persuade* and is also the subject of *help* (*Bill persuaded Mary that she should help John*).

• (4.3) *Bill persuaded Mary to help John*

Nevertheless, one of the subtypes of **trans-equi**, the type **subject-control** for subject control verbs (figure 4.43) differs from the general case. For subject control verbs, it is the subject of the transitive verb that controls the predicative complement. Sentence 4.4 shows an example of the subject-control verb *promise*.

• (4.4) *Bill promised Mary to help John*

In this case, *Bill*, the subject of *promise*, is also the subject of *help* (*Bill promised Mary that he will help John*). This behaviour is captured with the use

states that any type in a hierarchy has to be resolvable to a most specific type, is not used.

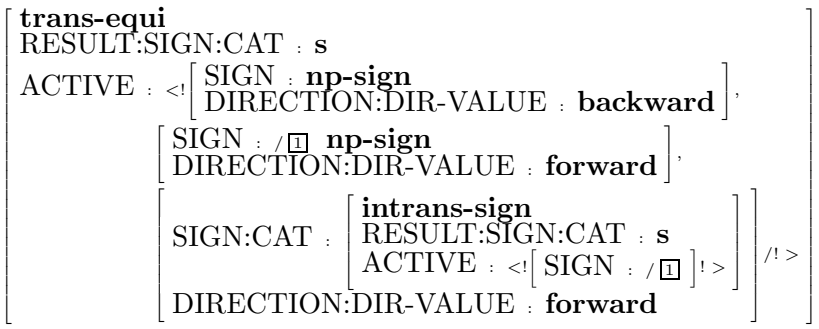

Figure 4.42: Transitive Equi Verb Type Partially Expanded

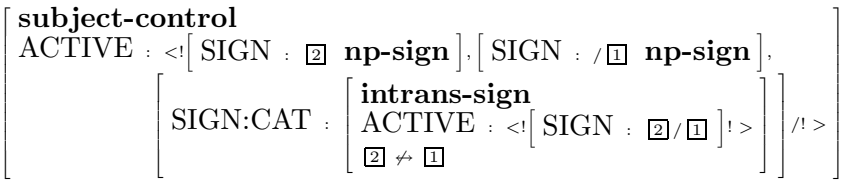

Figure 4.43: Subject-Control Verb Type

of inequalities, which negate the inherited coindexation between the object of the control verb and the subject of the predicative complement allowing the coindexation between the two subjects to hold (figure 4.43). This is a straightforward encoding that allows for the definition of this type as a subtype of **trans-equi** and which captures the common characteristics displayed by these types. The possibility of using defaults and inequalities allows the capture of these generalisations, while avoiding a considerable amount of redundancy that would be present in an alternative encoding. For instance, encoding **subject-control** as a subtype of **trans** would require it to contain most of the information already encoded in **trans-control** and **trans-equi**. It would also lose the generalisation that all these types are closely related, where **subject-control** is a subregularity of the general case.

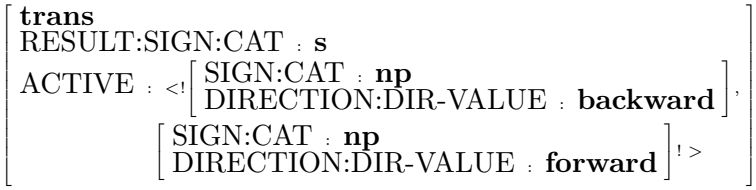

Figure 4.44: Transitive Verb Type Expanded and *DefFilled*

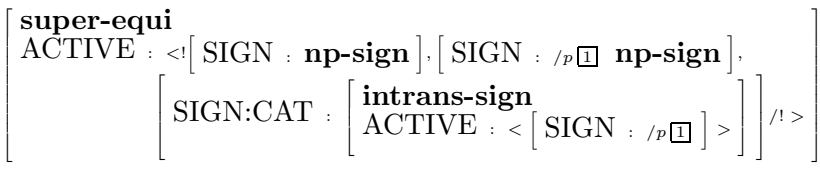

Figure 4.45: Super-Equi Verb Type

The defaults just discussed are used to capture lexical generalisations, but outside the lexicon we want them to act as indefeasible constraints. Therefore, we use *DefFill* to make these default specifications indefeasible, except where marked as persistently default. Thus, after *DefFill*, a type like **trans** has the consistent default specification of the end of the list (/! >) incorporated and specifies indefeasibly the need for exactly two arguments (figure 4.44). However, this is not the case for the type **super-equi**, for super-equi verbs that accept both object control or subject control, as is extensively discussed by Pollard and Sag [1994, p. 308-317]. Sentence 4.5 shows the verb *ask* in a context where the object of *ask*, *Bill*, is also the subject of *go* with object-control. In sentence 4.6, it is the subject of *ask* (*Mary*) that is the subject of *go*, with subject-control. For this type the information inherited from **trans-equi**, specifying default object-control, is allowed to persist until discourse interpretation, using the p operator (figure 4.45). Then, the default will survive if there is no conflicting information, as in sentence 4.5, otherwise it will be overridden, e.g. if it occurs in the context of sentence 4.6.

- (4.5) *Mary asked Bill to go home. His mother was expecting him there.*
- (4.6) *Mary asked Bill to be allowed to go home. She was expected there.*

This results in a direct encoding of such a phenomenon, without the need for extra types being defined to allow for the alternative behaviour.

In comparison with Pollard and Sag's hierarchy, shown in figure 3.1, repeated in figure 4.46, the implemented encoding is much more compact. The syntactic hierarchy defined has no need for types like **strict-trans**, which in Pollard and Sag's hierarchy is a subtype of **trans** and encodes exactly the same information. The only difference between these two types is that **trans** works as an intermediate type, specifying **at least** two arguments ($< np, np... >$) and offering its subtypes the possibility of adding extra arguments, as is the case for **ditrans** and **trans-control**. At the same time, **strict-trans** specifies the subcategorisation list as containing **exactly** two arguments ($< np, np >$), which is appropriate for transitive verbs. Defaults automatically provide this possibility, by defining the end of the list as default (/! >), avoiding the need for these redundant types. In this way, while a hierarchy like Pollard and Sag's is defined using 23 nodes, a similar fragment of the implemented verbal hierarchy is defined using only 19

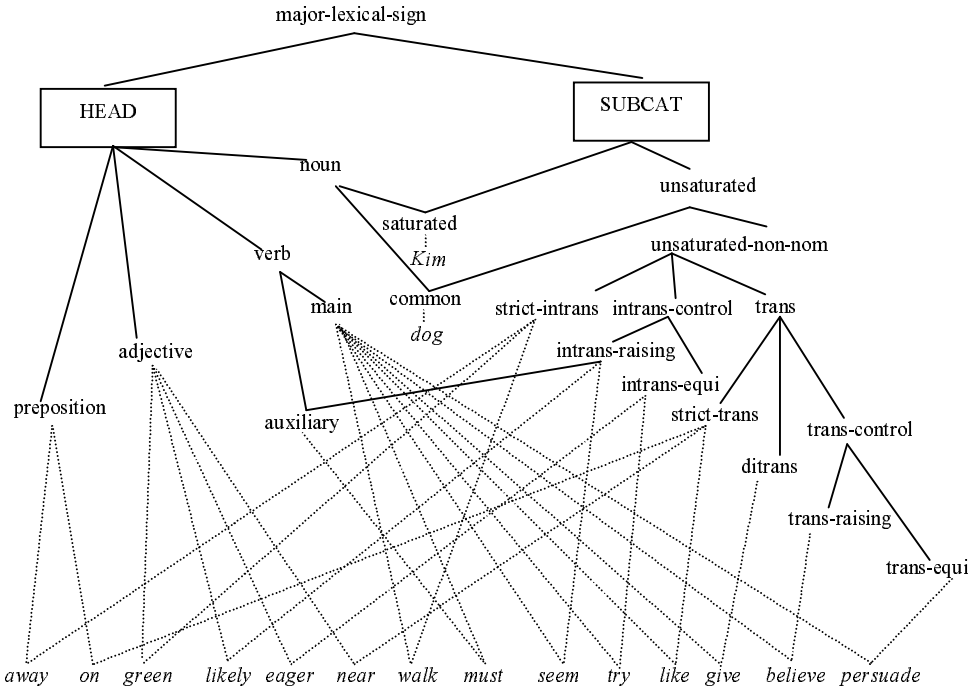

Figure 4.46: Pollard and Sag's Hierarchy

nodes. This is a small fragment of the complete hierarchy, but it already shows the advantages of using defaults. This hierarchy eliminates redundant specifications, like the *strict-trans* type, which are needed in a monotonic encoding. By avoiding this redundancy, there is a real gain in conciseness, with the resulting hierarchy capturing linguistic regularities and sub-regularities, such as the **super-equi** and **subject-control** types, in a compact encoding.

Semantic Hierarchy

The semantic representation used in the grammar is a variant of the MRS, as presented in section 4.2.2, encoded in the attribute SEM (figure 4.47).

$$\begin{bmatrix} \textbf{sem} \\ \text{MODE} : \textbf{mode} \\ \text{INDEX} : \textbf{index} \\ \text{RESTR} : \textbf{diff-list} \end{bmatrix}$$

Figure 4.47: Semantic Type

The predication types in the semantic hierarchy are defined according to their predicate-argument structure and organised according to the number of arguments they take. This information is encoded in the RESTR attribute of the

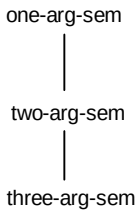

Figure 4.48: Fragment of the Verbal Semantic Hierarchy

semantics for a given verbal type. Figure 4.48 shows a fragment of the verbal semantic hierarchy. The basic type in the verbal hierarchy is the **one-arg-sem** (figure 4.49), which encodes generalisations about verbal categories:

- verb predicates have at least one argument (ARG1), and
- by default the semantic mode corresponds to a proposition (MODE:**/prop**).

This is the semantic type associated with, for instance, intransitive verbs and intransitive-raising verbs.[4]

$$\begin{bmatrix} \textbf{one-arg-sem} \\ \text{MODE} : \textbf{/prop} \\ \text{INDEX} : \textbf{index} \\ \text{RESTR} : \langle ! \begin{bmatrix} \textbf{verb-pred} \\ \text{RELN} : \textbf{reln} \\ \text{SIT} : \textbf{index} \\ \text{ARG1} : \textbf{index} \end{bmatrix} ! \rangle \end{bmatrix}$$

Figure 4.49: One-arg-verb Semantics

$$\begin{bmatrix} \textbf{two-arg-sem} \\ \text{RESTR} : \langle ! \begin{bmatrix} \textbf{verb-pred} \\ \text{ARG2} : \textbf{index} \end{bmatrix} ! \rangle \end{bmatrix}$$

Figure 4.50: Two-arg-verb Semantics

The subtypes of **one-arg-sem** inherit this information, as can be seen in the semantic verbal hierarchy (figure 4.48), while also defining extra arguments. Thus, **two-arg-sem** shown in figure 4.50 requires an extra argument (ARG2)

[4]The type **one-arg-sem** contains a very general description and can be generalised to other categories, such as adjectives, which also have only one semantic argument. This and some other of the ideas presented in this chapter can be successfully achieved in principle, even though the implemented grammar does not fully represent them as described.

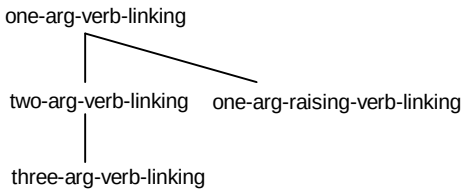

Figure 4.51: Fragment of the Verbal Linking Hierarchy

and it is the appropriate type for transitive verbs and intransitive equi verbs, among others.

By defining the different semantic types using hierarchies, semantic information is propagated effectively from type to type, in an economic encoding of semantic generalisations.

Linking Hierarchy

The mapping between syntactic subcategorised elements and semantic arguments is defined in the linking hierarchy. It is based on Wechsler's [Wechsler 1995] and Davis' [Davis 1996] proposals, as explained in section 4.2.3. The linking theory is implemented using coindexations between the syntactic attributes in CAT and the semantic attributes in SEM.

Given that the subcategorised elements in the ACTIVE list are ordered according to their obliqueness, the basic type in the linking hierarchy (figure 4.51) is the **one-arg-verb-linking** type. It encodes the generalisation specified by the Linking Principle that, for verbal categories in general, the first semantic argument (ARG1) is linked with the first syntactic complement, the subject, figure 4.52. This type is appropriate for intransitive verbs, where the subject NP, which is the first syntactic complement, is linked with the first semantic argument. All its subtypes inherit this information. However, the **one-arg-raising-verb-linking** type does not follow this pattern, since it defines that its argument is linked to the second syntactic argument, which is the predicative complement, figure 4.53. This is done using inequalities which negate the inherited coindexation between the subject and the first semantic argument. This type is appropriate for intransitive raising verbs, which do not assign a semantic role to the NP subject, their first syntactic complement, but instead link their first semantic argument to the second syntactic complement. An example is found in the sentence *Bill seems to run*, where *seems* has only one semantic argument and it is linked to its verbal complement *run*.

The type **two-arg-verb-linking** inherits the linking constraint from the **one-arg-verb-linking** type and adds a further constraint that the second semantic role, ARG2, is linked to the second syntactic complement, figure 4.54. In the case

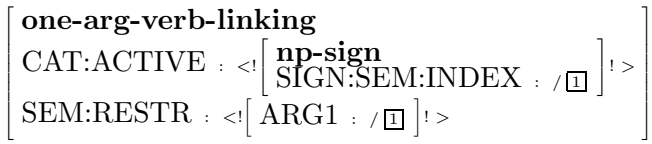

Figure 4.52: One-arg-verb-linking Type

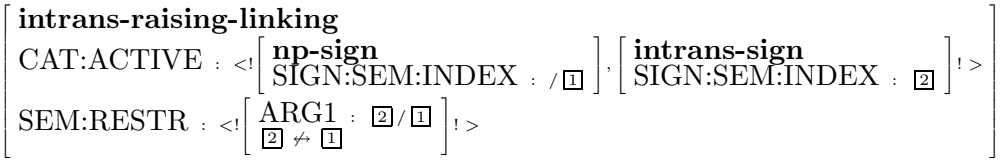

Figure 4.53: Intrans-raising-linking Type

of transitive verbs, the second syntactic complement corresponds to the direct object and in the case of intransitive equi verbs to the predicative complement.

More linking constraints are introduced by the type **three-arg-verb-linking**, which determines that the third semantic role (ARG3) is linked to the third syntactic complement (figure 4.55). This is appropriate for both oblique transitive verbs, where the third subcategorised element is an oblique object, and ditransitive verbs, where the third subcategorised element is the indirect object.

As in the case of the verbal semantic hierarchy, the use of a default inheritance hierarchy for encoding a linking theory resulted in generalisations being concisely defined only once, in more general types and being propagated to more specific types. Moreover, the use of default specifications provided a means of implementing subregularities, such as the **intrans-raising-linking** type.

The Signs

The three different hierarchies defined for syntactic, semantic and linking information are combined together in the hierarchy of signs. Thus, an intransitive verb of type **intrans-sign** is cross classified as:

- **intrans**, which is a type in the syntactic hierarchy that defines the need for one subcategorised complement,

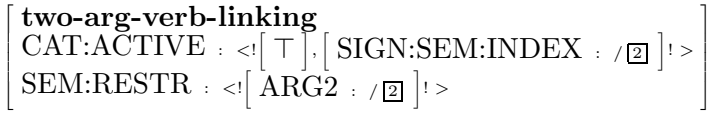

Figure 4.54: Two-arg-verb-linking Type

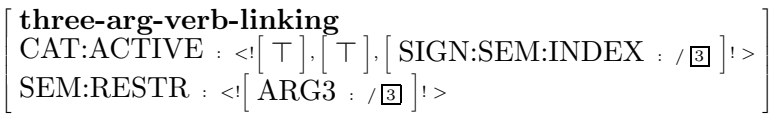

Figure 4.55: Three-arg-verb-linking Type

- **one-arg-sem**, which specifies the requirement for one semantic argument in the semantic hierarchy, and
- **one-arg-verb-linking** type, which in the linking hierarchy maps the first semantic role to the first subcategorised element.

The resulting **intrans-sign** type is shown in figure 4.56, with only the relevant features being shown, and a fragment of the verbal sign hierarchy is shown in figure 4.57.

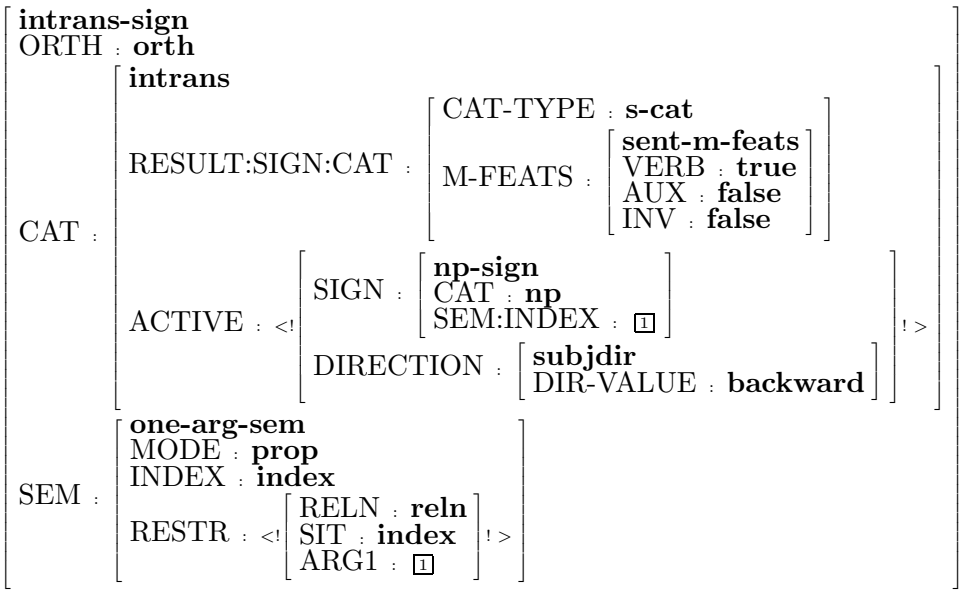

Figure 4.56: Intrans-sign Type Expanded and *DefFill*ed

Default specifications play a major role in the hierarchies, allowing subregularities and exceptions to be straightforwardly defined in a concise encoding. As a consequence of using multiple inheritance hierarchies, these three different kinds of linguistic information are contained in three different hierarchies, which interact with one another combining their information at the level of signs. This modularity means that any subsequent change or addition to any of these hierarchies will be localised, not affecting the others.

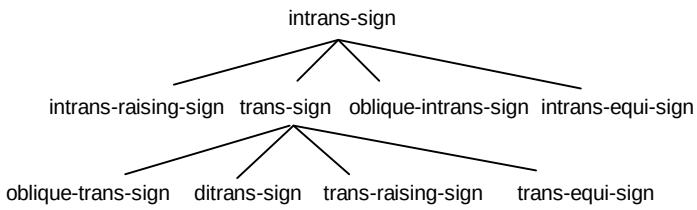

Figure 4.57: Fragment of the Verbal Sign Hierarchy

4.3 Rules

Rules are defined as typed feature structures that represent relationships between two or more signs. There are three different types of rule implemented: morphological, lexical and grammatical. Morphological and lexical rules are unary rules used for generating derived or inflected items from the lexical entries encoded in the lexicon; they are discussed in section 4.3.1. Grammatical rules are used to combine constituents together; they are described in section 4.3.2.

4.3.1 Morphological and Lexical Rules

Morphological and lexical rules are implemented using asymmetric unification, following Briscoe and Copestake [1999]. Using this approach a lexical entry that is successfully unified with the input description of a lexical rule, is then default unified with the output of the rule, where the input is treated as defeasible and the output as indefeasible. The resulting sign has all the information in the output of the lexical rule, plus all the information in the input that is consistent with the output. However, all the information in the input that is inconsistent with the output is removed, as explained in section 3.5. Morphological and lexical rules, being unary rules, are represented throughout the document with an input TDFS, specified in the attribute '1:NODE', an output TDFS, in the attribute 'NODE', and a function for spelling change in the attribute 'FUNCTION' (figure 4.58), using a notation similar to that employed in [Pollard and Sag 1987]. The use of asymmetric unification ensures that the input and output categories are related, with just some modification that follows from unifying the defeasible input with the indefeasible output, rather than just copying substructures from the input to the output structures, which are not necessarily related.

Morphological rules are treated as a special case of lexical rules, potentially involving affixation and, consequently, being applied before parsing. Rules of this kind that capture both nominal and verbal inflections are defined. One such rule is the **plural-noun-rule**. Since noun lexical entries are defined as having a third person singular agreement by default, the **plural-noun-rule** generates plural forms from these third person singular nouns, adding the appropriate

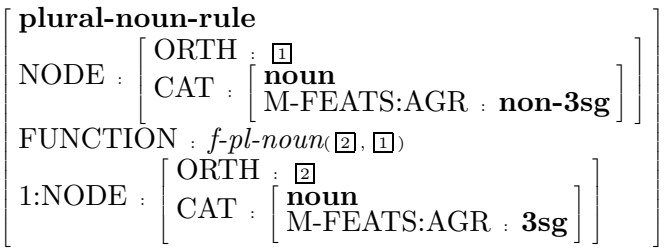

Figure 4.58: Plural Noun Morphological Rule

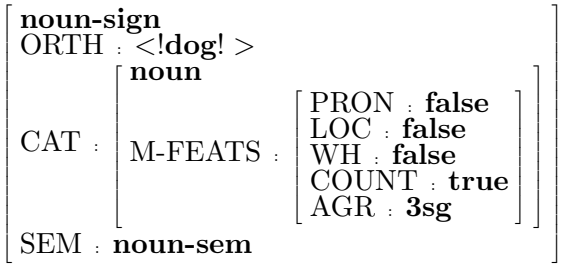

Figure 4.59: *dog* Sign

suffixes and specifying the required agreement values (figure 4.58). In **plural-noun-rule** the input is a noun with agreement defined as third person singular (CAT:M-FEATS:AGR:**3sg**). When the defeasible input is unified with the indefeasible output specification, the input agreement information is incompatible with **non-3sg** and consequently absent from the resulting TDFS, which contains the indefeasibly defined **non-3sg**. However, all the remaining information from the input is present in the result TDFS. For instance, when the lexical entry for *dog* (figure 4.59), is given as input to this rule, the resulting output sign contains the plural form *dogs* (figure 4.60).

Verbal lexical entries are defined by default as being in the base form and several rules are defined for generating the other inflected forms. One such case is the rule **3sg-v-rule** that generates the third person singular inflection from the base form, including affixation when appropriate (figure 4.61). Similarly the rule **non3sg-v-rule** generates lexical entries that define the other person and number agreement cases (figure 4.62). Both these rules specify the appropriate agreement for the subcategorised NP subject. Other finite forms are also generated from the base form. For example, **past-prt-v-rule** generates past participle forms (figure 4.63), and **past-v-rule** generates past forms, with these rules including appropriate affixation when necessary.

The resulting set of rules, defined with asymmetric unification, is more concise and clear than a traditional equivalent, where much more information would need to be defined in each of the rules. Moreover, with the use of default specifications

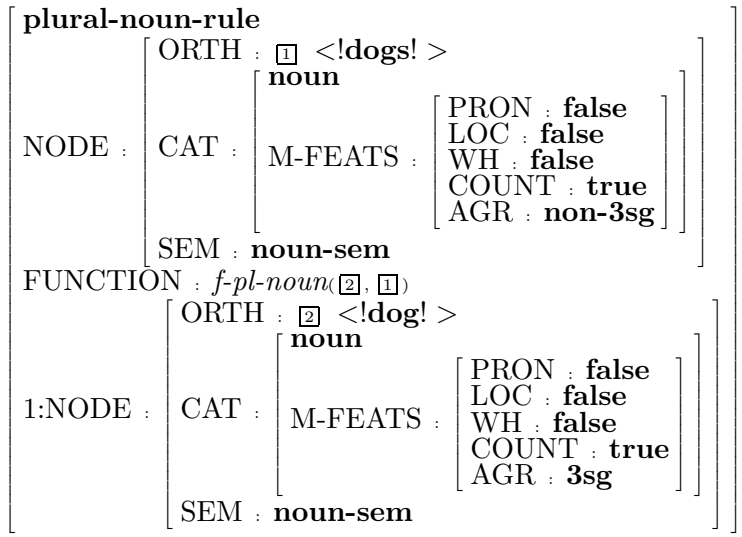

Figure 4.60: Plural Noun Morphological Rule Applied to *dog*

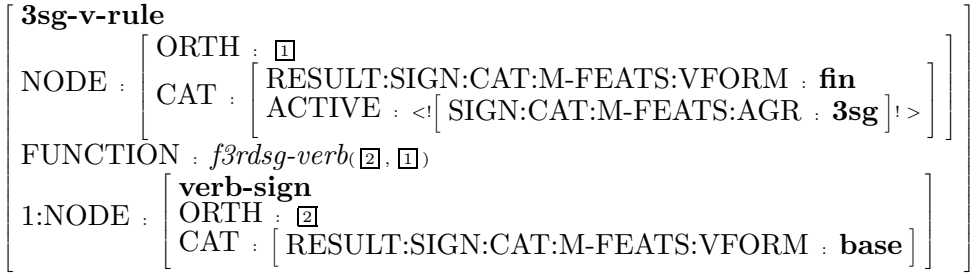

Figure 4.61: Third Singular Verb Morphological Rule

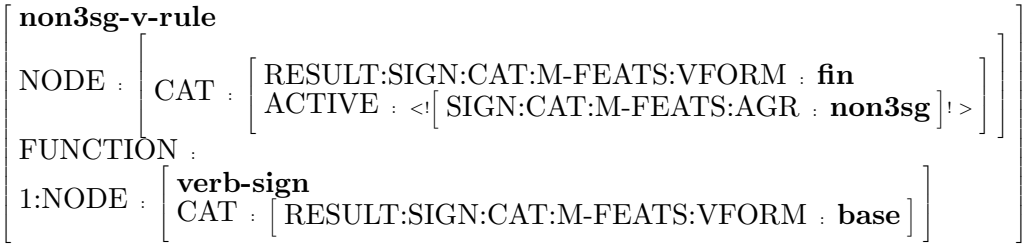

Figure 4.62: Non Third Singular Verb Morphological Rule

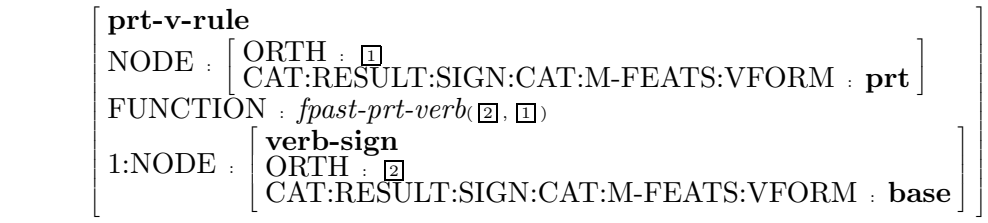

Figure 4.63: Past Participle Verb Morphological Rule

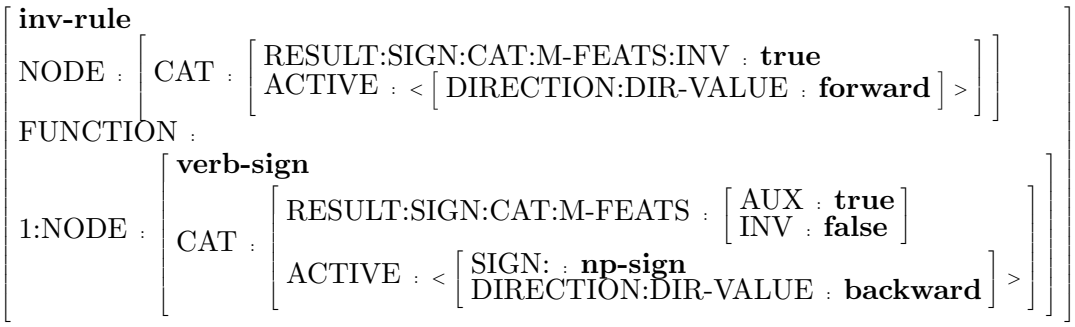

Figure 4.64: Inverted Modal Rule

in the basic lexical entries, fewer rules have to be defined. In this case, rules for generating base form verbal signs and for generating third person singular forms for nouns do not need to be defined, since these forms are defaults in the type definitions, with verbs as VFORM:/**base** and nouns as CAT:M-FEATS:AGR:/**3sg**.

Lexical Rules are applied lexically, either before and/or after affixation or during parsing, as appropriate. Among the rules defined, verbal lexical rules create lexical entries from a basic canonical form. For instance, the **inv-rule** generates the inverted form of a verb which is appropriate for capturing yes-no questions, where the auxiliary verb is fronted preceding the subject: *Will you buy this car?*, figure 4.64. The category of the auxiliary verb *will* is changed from (S\NP)/(S\NP) to (S/NP)/(S\NP).

The **imperative-rule** generates the imperative form for verbal categories (figure 4.65), for capturing sentences like *Put that down*, where the subject is not explicitly realised. This rule removes the subject category from the subcategorisation list and marks the agent role in the semantic specification of the sentence as an imperative one (**imp-index**). Because this rule is implemented with asymmetric unification, inequalities need to be used to negate the existing default coindexations between the corresponding elements in the subcategorisation list in the input and output descriptions.

Following Briscoe and Copestake [1999], lexical rules are also defined for generating the dative alternation (figure 4.66). The **dative-rule** generates the dative construction from the oblique frame of a verb. The result of applying this rule

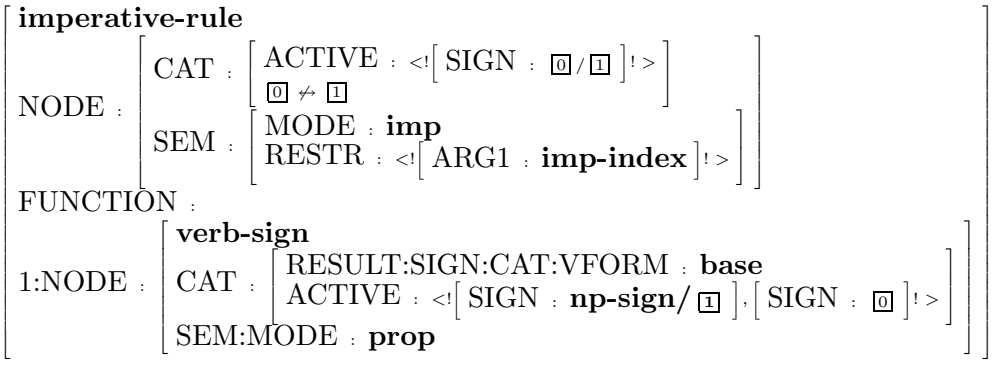

Figure 4.65: Imperative Rule

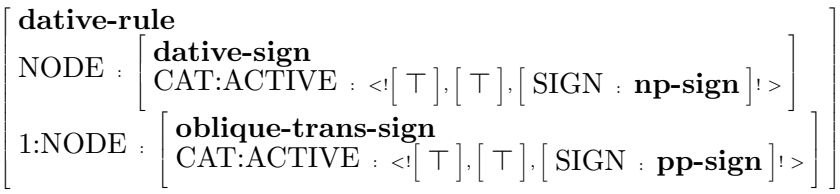

Figure 4.66: Dative Rule

to an oblique verb (((S\NP)/NP)/PP) is that the oblique PP object is replaced by an NP object in the dative type (((S\NP)/NP)/NP).

Two rules are defined to account for the case of passivisation. The first one, the **passive-rule**, takes verbs (transitives or verbs with higher transitivity) and generates the passive equivalent with the oblique argument (*The vase was broken by him*), figure 4.67. The second rule, **passive-no-by**, generates the passive form without the oblique argument which corresponds to the subject of the active form (*The vase was broken*), figure 4.68. It also marks ARG1 as having a **null-index**, since the agent role is not realised syntactically. These rules also require the use of inequalities to negate the default coindexation that exists between the NP subjects in the input and output subcategorisation lists, since they do not have conflicting values.

These are some of the morphological and lexical rules defined and a fragment of the set of rules is included in Appendix B.

4.3.2 Grammatical Rules

In CG some of the categories can be viewed as arguments and some as functions, and the set of rules is responsible for combining functions with arguments or with other functions. In this work five rules are defined: forward application, backward application, forward composition, backward composition and generalised

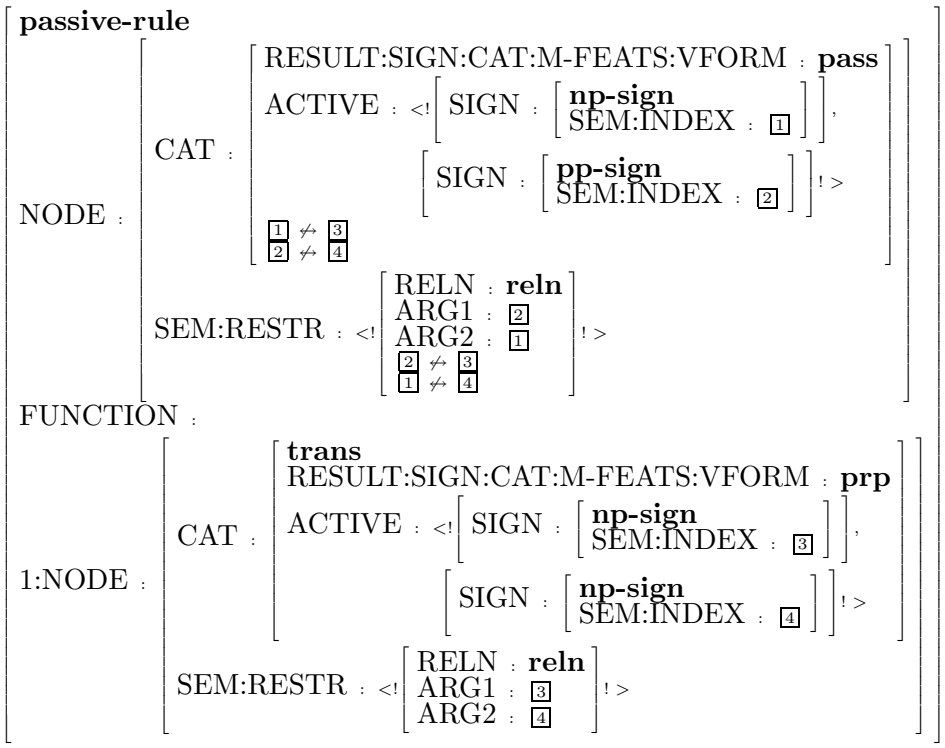

Figure 4.67: Passive Rule

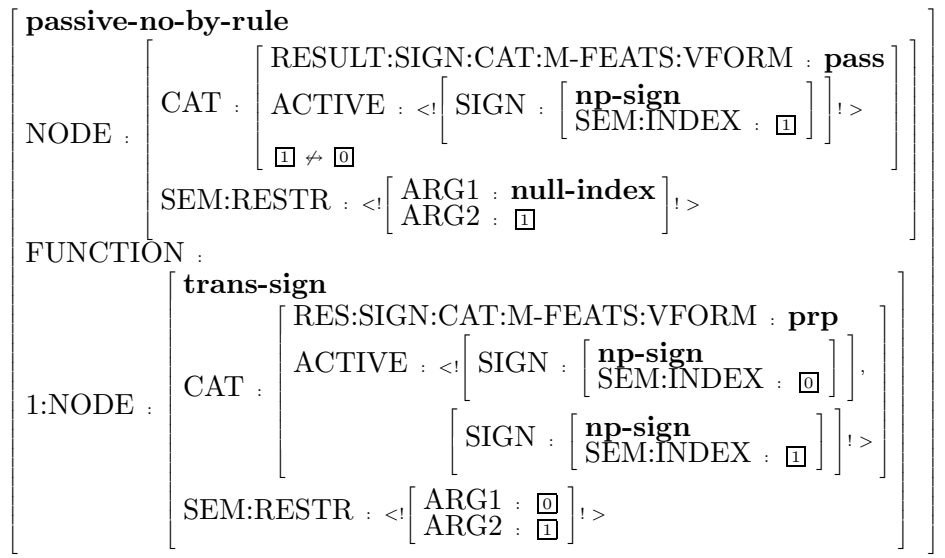

Figure 4.68: Passive without Oblique Argument Rule

weak permutation. These rules are defined as types in the grammar, in terms of TDFSs. The binary rules of application and composition have two input TDFSs defined, respectively, in '1:NODE' and '2:NODE', according to the linear order of the constituents to be combined. The output TDFS, which is the result of the combination of the input TDFSs, is defined in 'NODE'. The unary rule of GWP has one input TDFS defined in the attribute '1:NODE', and an output TDFS defined in 'NODE'. Moreover, in these rules '...' represents one or more arguments in the subcategorisation list.

In the complex categories defined, the subcategorised arguments are ordered in the ACTIVE list, with the first element in the list being the one considered for combination with adjacent categories.

The **Functional Application** rule simply combines a function with its argument, as explained in section 4.1:

- **Forward Application**: X/Y Y → X is encoded as shown in figure 4.69, where the category X/Y, to the left, is represented in the attribute 1:NODE; the category Y, to the right, is represented in 2:NODE; and the resulting category X is represented in NODE.
- **Backward Application** rule: Y X\ Y → X is encoded as in figure 4.70.

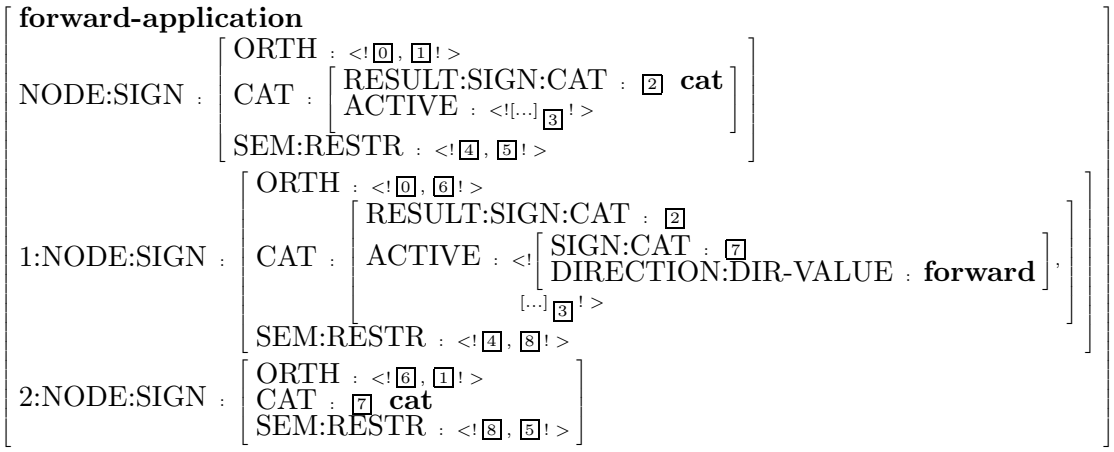

Figure 4.69: Forward Application

Functional application captures a wide range of constructions, being used to combine, for example, verbs with their complements, determiners with nouns, and adjective with nouns, among others.

The rule of **Functional Composition** allows a functor category missing an argument to compose with an adjacent function that outputs that argument as its result:

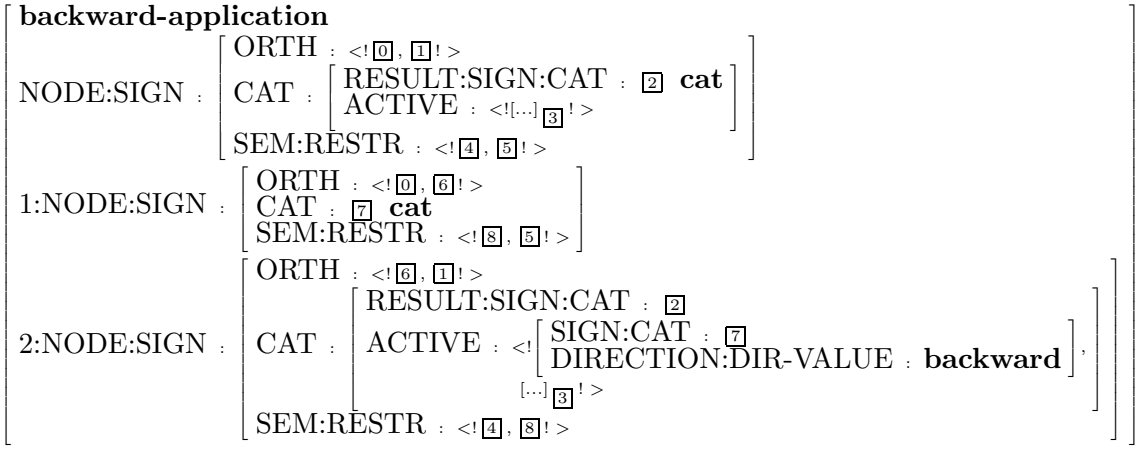

Figure 4.70: Backward Application

- **Forward Composition**: X/Y Y/Z → X/Z (figure 4.71).
- **Backward Composition**: Y\Z X\Y → X\Z.

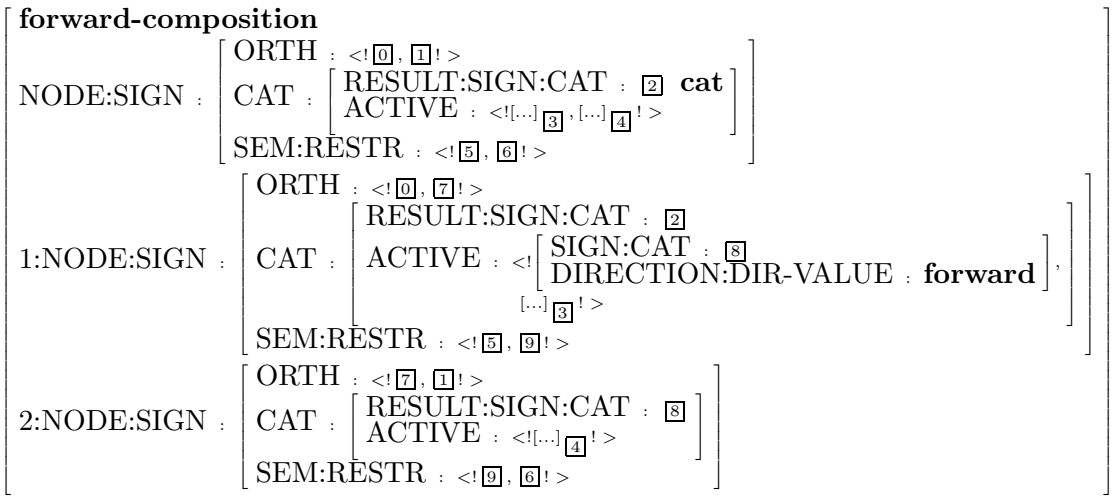

Figure 4.71: Forward Composition

Functional composition is used mainly to capture unbounded dependencies and coordinations, since it allows non-conventional constituency.

The unary rule of **Generalized Weak Permutation** allows a complex category to combine with its arguments in any order, as is discussed in section 4.4. GWP rotates the arguments of a complex category, with the number of possible permutation operations being finite and bounded by the number of arguments

the category has. While rotating the arguments, this rule maintains the original direction associated with each of them.

- **Generalised Weak Permutation** (P): $((X|Y_1)\ldots|Y_n) \rightarrow (((X|Y_n)|Y_1)\ldots)$, shown in figure 4.72

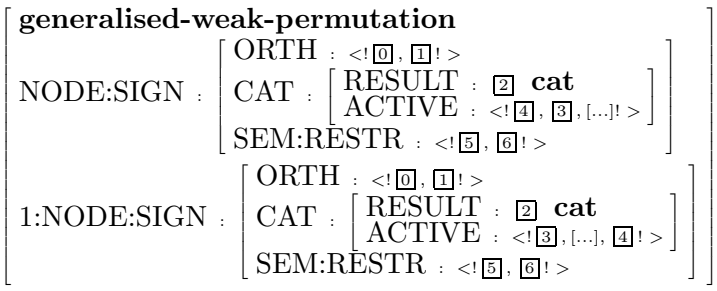

Figure 4.72: Generalised Weak Permutation

The arguments in the subcategorisation list are ordered and the first argument is the one considered for combination with adjacent categories. Thus, if the order of elements in the subcategorisation list does not allow a category to be combined, the subcategorised arguments are rotated until the argument required for combination is in the first position. For instance, for verbal categories, the subject is always the first element in the subcategorisation list (followed by the direct object, indirect object and so on, as required for a particular verbal category). This is defined in the intransitive type (**intrans**) and inherited by all verbal types in the syntactic hierarchy. In this initial ordering, the subject is always the first to be combined in any verbal category. In an atomic representation of the categories, for instance, a transitive verb following this ordering is represented as $S/NP_O\backslash NP_S$. [5] For some sentences, this initial ordering is not appropriate and does not produce a derivation. In these cases, the GWP rule is necessary to rotate the elements in the subcategorisation list until an appropriate ordering is achieved. One example is the case of the coordination in the sentence *He bought fish and cooked dinner*, whose derivation is shown in figure 4.73. Both verbs have category $S/NP_O\backslash NP_S$ and need to be combined with their direct objects first. This is achieved by rotating the arguments so that the object (NP_O) is the first in the subcategorisation list: $S\backslash NP_S/NP_O$.

This rule allows the grammar to capture more flexible word orders, without the need for defining extra categories. As a result the grammar can capture, for example, verb-particle constructions where the particle can be either before or after the object, using the same category. For instance, the verb *back*, which has category $((S/PRT)/NP)\backslash NP$ and takes the particle *up*, not only accepts a

[5] In the remainder of this document this ordering is adopted.

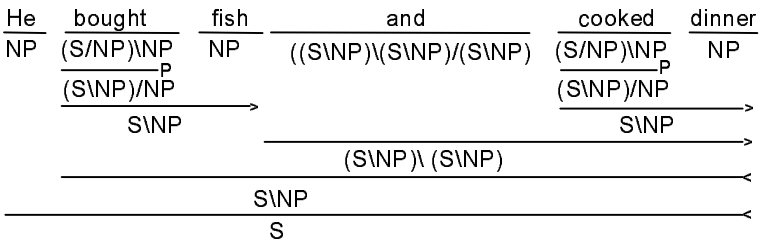

Figure 4.73: Derivation of Sentence *He bought fish and cooked dinner*

sentence such as *Bob backed the team up* (figure 4.74) which follows the initial word order specified, but also accepts *Bob backed up the team*, when its subcategorised arguments are rotated (figure 4.75). On the other hand, by accepting freer word orders this rule can also overgenerate. This is the case, for example, of verb-particle constructions that have a fixed order and only accept the particle in a certain position in relation to the verb. This rule overgenerates, accepting both the grammatical and the ungrammatical orderings, such as in *She came up with the idea* and **She came with the idea up*, respectively.[6] Moreover, for categories which subcategorise for similar categories with the same directions, this rule allows derivations where the categories are interchanged. This is the case of ditransitive verbs, for example, with category $((S/NP_{O2})/NP_{O1})\backslash NP_S$, where the two NP objects are to the right: NP_{O1} corresponds to the direct object and NP_{O2} to the indirect object. GWP also allows NP_{O1} to be linked with the indirect object and NP_{O2} to the direct object. Thus, a sentence such as *He gave Cathy a flower* has two different logical forms generated. The first one is where *Cathy* is the indirect object being linked with the recipient role, and *a flower* is the direct object being linked with the patient role, where a flower is given to Cathy. In the other, *a flower* is the indirect object being linked with the recipient role and *Cathy* is the direct object linked with the patient role, where Cathy is given to a flower. In spite of this overgeneration caused by the GWP rule, the benefits that it brings in terms of the coverage obtained without the need for extra categories, greatly outweighs that. Furthermore, there are possible ways of constraining the application of this rule to control overgeneration. For instance, one way of doing this while at the same time ensuring termination is to store, for each category, the number of arguments it has, and maintain a count of the number of times the arguments are rotated against the maximum number of possible operations for the category, which is obtained by subtracting one from the number of arguments. In this way, the application of GWP is bound by the number of arguments of a category, and for a category such as $((S/NP_{O2})/NP_{O1})\backslash NP_S$, for ditransitive verbs, containing 3 arguments, only 2 applications of GWP are allowed, generat-

[6]Ungrammatical sentences are marked with *.

ing the following categories besides the original one:

$((S\backslash NP_S)/NP_{O2})/NP_{O1}$

$((S/NP_{O1})\backslash NP_S)/NP_{O2}$

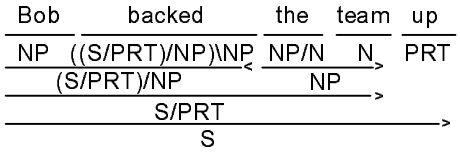

Figure 4.74: Derivation of Sentence *Bob backed the team up*

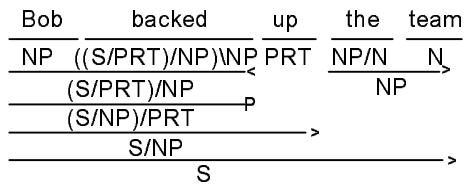

Figure 4.75: Derivation of Sentence *Bob backed up the team*

These grammatical rules are used to combine or allow the combination of constituents and the result of a derivation using the backward application rule is shown in figure 4.76, for the sentence *Bill runs*. In this figure, both the orthography (ORTH) and the semantics (SEM:RESTR) of the resulting sign contain the conjunction of the orthographic and semantic specification of the signs that were combined. This conjunction is obtained through the use of difference lists, which append the signs being combined according to the rule definition: NODE:ORTH:< !$\boxed{0}$, $\boxed{1}$! > = 1:NODE:ORTH:<!$\boxed{0}$, $\boxed{6}$! > + 2:NODE:ORTH:<!$\boxed{6}$, $\boxed{1}$! >.[7]

4.4 Coverage

The implemented UB-GCG has 89 lexical categories defined, covering several linguistic constructions. For instance, it can handle several verbal constructions, it captures declarative, imperative and interrogative sentences, and successfully treats constructions such as unbounded dependencies (wh-questions and relative

[7]Throughout this document, for reasons of clarity, the atomic formulation of the rules will be used, unless otherwise required.

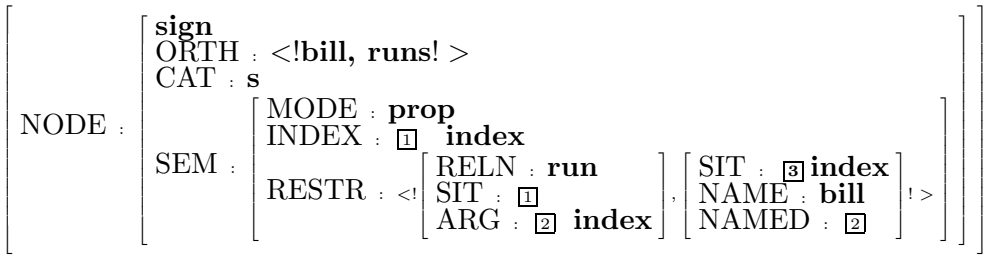

Figure 4.76: Sentence *Bill runs*

clauses) and coordinations. The lexical categories encode the relevant linguistic information and they are defined in terms of five basic categories (S, NP, N, PRT and PP) and three operators (forward and backward slashes and product operator). In this section, the treatment given in the implemented UB-GCG to some of these constructions is discussed.

4.4.1 Verbal Constructions

Most verbal constructions are implemented in the grammar directly as lexical category types and assigned to the appropriate lexical entry for a verb. Thus the verb *loves* in the sentence *John loves Mary* is a transitive verb, with category S/NP\NP, and one derivation for this sentence is shown in figure 4.77.

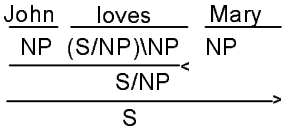

Figure 4.77: Derivation for a Transitive Construction

Similarly oblique transfer verbs and dative verbs are defined as types in the verbal hierarchy. However, following Briscoe and Copestake [1999] these verbs are treated as part of a family of dative constructions, which have the same syntactic and related semantic properties [Goldberg 1995]. Most lexical entries will be of type **oblique-trans-sign**, for oblique transitive verbs with category ((S/PP)/NP)\NP. For example, the verb *give*, which occurs in oblique transitive constructions, such as the sentence *Bill gave the dog to Mary.* The dative alternation is captured by the dative lexical rule, which maps an **oblique-trans-sign** into a **dative-sign** with category ((S/NP)/NP)\NP. Then one possible derivation for the sentence *Bill gave Mary the dog* is shown in figure 4.78, where the dative lexical rule is represented in the derivation as **Dat Rule**. The thematic roles are correctly linked to the syntactic arguments, with *Bill* being linked to the

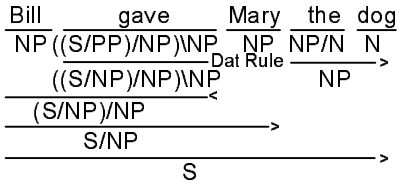

Figure 4.78: Derivation of Sentence *Bill gave Mary the dog*

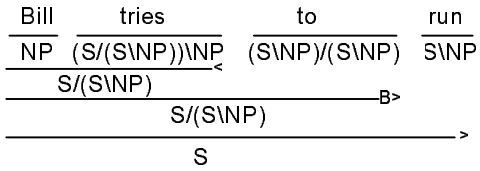

Figure 4.79: Derivation of Sentence *Bill tries to run*

agent role (ARG1), *the dog* to the patient (ARG2) role, and *Mary* to the oblique role (ARG3).

Raising and equi constructions are represented by appropriate types defined in the verbal hierarchies. In the case of intransitive (or subject) equi verbs, such as *try*, with category (S/(S\NP))\NP, its subject is also the subject of the embedded verb. One possible derivation for the sentence *Bill tries to run* is shown in figure 4.79. As can be seen in the logical form for this sentence (figure 4.80), both the control and the embedded verb assign a semantic role to the shared subject.

Intransitive raising verbs are similar to equi verbs, also with category (S/(S\NP))\NP, except that the subject of the embedded verb is not assigned a semantic role. This can be seen in one derivation for the sentence *Bill tends to run*, shown in figure 4.81, and in the corresponding logical form, in figure 4.82. The infinitival *to* is also treated as a raising verb with category (S/(S\NP))\NP.

The same patterns are observed with transitive (or object) control verbs,

$$\left[\begin{array}{l}\textbf{sem}\\ \text{MODE} : \textbf{prop}\\ \text{INDEX} : \boxed{0}\ \textbf{index}\\ \text{RESTR} : \left\langle !\left[\begin{array}{l}\text{RELN} : \textbf{try}\\ \text{SIT} : \boxed{0}\\ \text{ARG1} : \boxed{1}\ \textbf{index}\\ \text{ARG2} : \boxed{2}\ \textbf{index}\end{array}\right], \left[\begin{array}{l}\text{RELN} : \textbf{to}\\ \text{SIT} : \boxed{2}\\ \text{ARG1} : \boxed{3}\ \textbf{index}\end{array}\right], \right.\\ \qquad\qquad \left.\left[\begin{array}{l}\text{RELN} : \textbf{run}\\ \text{SIT} : \boxed{3}\\ \text{ARG1} : \boxed{1}\end{array}\right], \left[\begin{array}{l}\text{SIT} : \boxed{4}\ \textbf{index}\\ \text{NAME} : \textbf{Bill}\\ \text{NAMED} : \boxed{1}\end{array}\right] !> \right\rangle\end{array}\right]$$

Figure 4.80: Logical Form of Sentence *Bill tries to run*

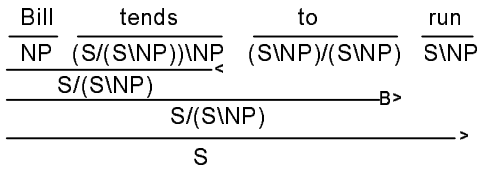

Figure 4.81: Derivation of Sentence *Bill tends to run*

$$\left[\begin{array}{l}\textbf{sem}\\ \text{MODE} : \textbf{prop}\\ \text{INDEX} : \boxed{0}\ \textbf{index}\\ \text{RESTR} : \left\langle !\ \left[\begin{array}{l}\text{RELN} : \textbf{tend}\\ \text{SIT} : \boxed{0}\\ \text{ARG1} : \boxed{1}\ \textbf{index}\end{array}\right], \left[\begin{array}{l}\text{RELN} : \textbf{to}\\ \text{SIT} : \boxed{1}\\ \text{ARG1} : \boxed{2}\ \textbf{index}\end{array}\right], \right.\\ \qquad\qquad \left.\left[\begin{array}{l}\text{RELN} : \textbf{run}\\ \text{SIT} : \boxed{2}\\ \text{ARG1} : \boxed{3}\ \textbf{index}\end{array}\right], \left[\begin{array}{l}\text{SIT} : \boxed{4}\ \textbf{index}\\ \text{NAME} : \textbf{Bill}\\ \text{NAMED} : \boxed{3}\end{array}\right] !\ \right\rangle\end{array}\right]$$

Figure 4.82: Logical Form of Sentence *Bill tends to run*

except that it is the object of a transitive control verb that controls the subject of the embedded verb.

Verbs that expect a particle have an extra argument, with category PRT, in the subcategorisation list. For instance, the transitive particle taking verb *warm* has category ((S/PRT)/NP)\NP. When combined with the particle *up*, the verb can be either immediately followed by the particle, as in *warm up the milk*, or be followed by the direct object and then by the particle, as in *warm the milk up*. These two cases are captured using the same category, but in the first case using the permutation rule, as shown in figures 4.83 and 4.84. Figure 4.85 shows the specification of the verb *warm* with the particle *up*. These cases differ from those where verbs take a complete PP as complement. For example, the verb *put* has category ((S/PP)/NP)\NP and a derivation for the sentence *He put the book on the shelf* can be seen in figure 4.86. In this case, the preposition, with category PP/NP, needs to be combined first with its NP complement to be then combined with the verb.

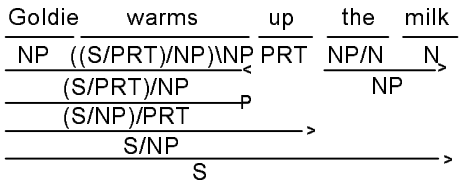

Figure 4.83: Derivation of Sentence *Goldie warms up the milk*

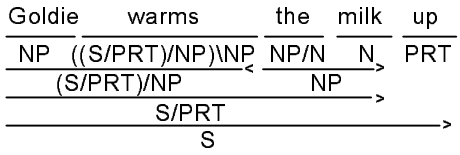

Figure 4.84: Derivation of Sentence *Goldie warms the milk up*

$$\left[\begin{array}{l}\textbf{trans-prt-sign}\\ \text{CAT} : \left[\begin{array}{l}\text{RESULT:SIGN:CAT:M-FEATS} : \left[\begin{array}{l}\textbf{sent-m-feats}\\ \text{TAKE-PARTICLE} : \textbf{true}\\ \text{PARTICLE} : \boxed{1}\ \textbf{up}\end{array}\right]\\ \text{ACTIVE} : \left\langle !\left[\top\right], \left[\top\right], \left[\text{SIGN} : \left[\begin{array}{l}\textbf{particle-sign}\\ \text{CAT:M-FEATS:P-FORM} : \boxed{1}\end{array}\right]\right]_{!>}\right\rangle\end{array}\right]\end{array}\right]$$

Figure 4.85: Abbreviated Lexical Entry for *warm*, with PFORM:**up**

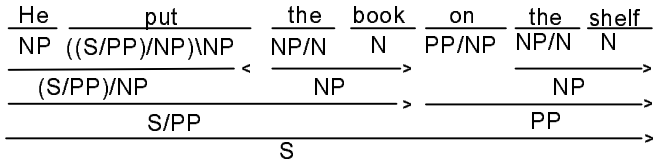

Figure 4.86: Derivation of Sentence *He put the book on the shelf*

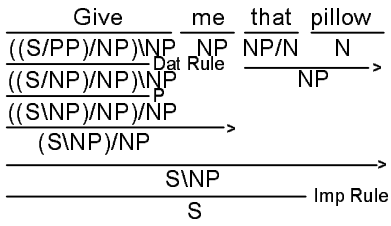

Figure 4.87: Derivation of Sentence *Give me that pillow*

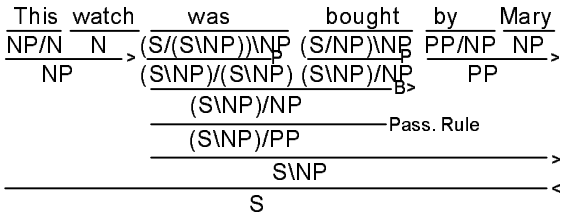

Figure 4.88: Derivation of Sentence *This watch was bought by Mary*

Imperative sentences are captured by having a lexical rule that applies to the main verb, removing the subject NP from the list of subcategorised elements and marking the agent role as implicit. Figure 4.87 shows one derivation for the sentence *Give me that pillow*, where the initial category for *give* is $((S/NP_{O2})/NP_{O1})\backslash NP_S$ and where NP_S is removed after the application of the imperative lexical rule (**Imp Rule** in the derivation).

Passive sentences are also captured by means of lexical rules that relate them to their active counterparts, so that the semantic roles of the arguments of the active and passive pairs are the same. However, the syntactic realisation of these roles is different, with the subject of the passive corresponding to the object of the active and the subject of the active, if present at all in the passive, corresponding to the argument of the prepositional phrase containing the preposition *by*. For instance, in the sentences *This watch was bought by Mary* (passive) and *Mary bought this watch* (active), the subject of the first sentence is *this watch*, while in the second it is *Mary*. However, in both cases the agent role of *bought* (ARG1) should be linked to the predicate associated with *Mary*. Figures 4.88 and 4.89 show derivations for the first and second sentences, respectively, where **Pass Rule** represents the passive lexical rule.

Auxiliary verbs are encoded as types in the verbal hierarchies which capture the generalisation that auxiliaries take verbal complements. Moreover, different auxiliaries require different forms of verbal complements. For instance, *have* requires a past participle verbal complement (*Jane has studied English literature*), while *be* requires a progressive form (*Jane is studying English literature*). This

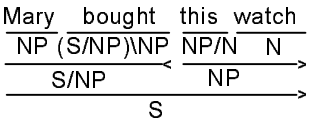

Figure 4.89: Derivation of Sentence *Mary bought this watch*

$$\left[\begin{array}{l}\textbf{aux-prp-sign}\\ \text{CAT:ACTIVE} : {<!}\left[\ \top\ \right],\\ \qquad\qquad \left[\ \text{SIGN:CAT:RESULT:SIGN:CAT:M-FEATS:VFORM} : \textbf{prp}\ \right]{!>}\end{array}\right]$$

Figure 4.90: Sign for the Auxiliary *have*

constraint is encoded in the VFORM attribute of the subcategorised verbal complement, as shown in figure 4.90 for *have*. In terms of semantics, auxiliaries are treated as raising verbs, not assigning a semantic role to their subjects.

Interrogative sentences are handled by means of lexical rules. In this kind of sentence, a finite auxiliary verb precedes the subject (*Will Mary buy the car?*). This constraint can be easily and straightforwardly achieved by changing the direction of the subject NP from '**backward** to '**forward**. This is done by a lexical rule (**Inv Rule**) that has as input an auxiliary with category (S/(S\NP))\NP and generates as output the category (S/(S\NP))/NP. One derivation for the sentence *Will Mary buy the car?* is shown in figure 4.91.

4.4.2 Unbounded Dependencies

The term **Unbounded Dependencies** refers to the several families of natural language constructions where there is a relationship among distant subexpressions within an expression. In these constructions a fronted constituent can "belong to" another constituent embedded arbitrarily deep. For instance, in the sentences *Who do you like?*, *Who do you think you like?*, *Who does Jane believe you like?* and *Who does Kate say Jane believes you like?* the distance between *who* and the position it binds, as the object of *like*, is embedded deeper and deeper in each

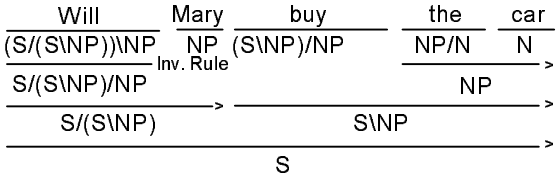

Figure 4.91: Derivation of Sentence *Will Mary buy the car?*

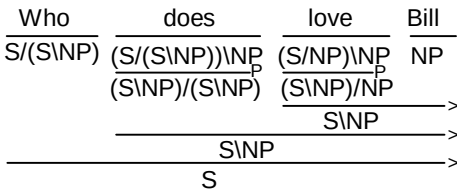

Figure 4.92: Derivation of Sentence *Who does love Bill?*

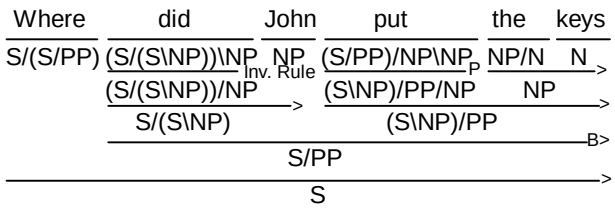

Figure 4.93: Derivation of Sentence *Where did John put the keys*

of them. CGs are well known for providing a simple and elegant way of capturing unbounded dependencies[8]. In this section, the treatment given for wh-questions and relative clauses is discussed.

In **wh-questions**, wh-words have complex categories that take as argument a sentence missing a constituent and whose result is an S category. There are several different constructions, corresponding to different extraction patterns for the fronted object. For each of these constructions, an appropriate category is defined. Thus for **Subject Extraction**, exemplified in sentence 4.8, the fronted wh-pronoun (*who*) corresponds to the subject of the verb (*love*), and the pronoun is assigned category S/(S\NP), as shown in figure 4.92.

• (4.8) *Who does love Bill?*

Another case of verbal argument extraction is that of **PP Extraction**, where the fronted wh-pronoun corresponds to a PP argument of the verb. The pronoun is assigned category S/(S/PP). One derivation of the sentence 4.9 is shown in figure 4.93.

• (4.9) *Where did John put the keys?*

There is also the case of **Adjunct Extraction**, where the fronted wh-word corresponds to an adjunct and is assigned category S/S. An example of this kind

[8]For an extensive discussion of unbounded dependencies and other phenomena in CGs, see Steedman [1985 to 2000].

of extraction can be seen in sentence 4.10 whose derivation is shown in figure 4.94.

• (4.10) *Where does John live?*

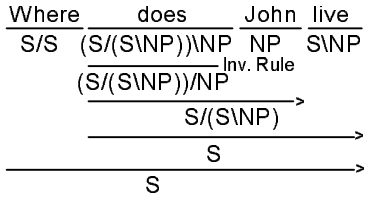

Figure 4.94: Derivation of Sentence *Where does John live?*

Relative clauses are noun modifiers, usually involving a relative pronoun such as *who*, *which* or *that*. The different kinds of relative clauses are modelled by different lexical category types. One case of relative clauses contains **Subject Extraction**, where the relative pronoun, with category (N/(S\NP))\N, corresponds to the subject of the embedded verb. This is exemplified in sentence 4.11, whose derivation is shown in figure 4.95, where the relative clause *who loves Bill* modifies the noun *person*.

• (4.11) *The person who loves Bill hates Bob*

Another case is that of **Adjunct Extraction**, exemplified in sentence 4.12 whose derivation is shown in figure 4.96. In this case the relative pronoun corresponds to an adjunct with category (N/(S/PP))\N.

• (4.12) *The house where John lives has a garden*

These are some of the unbounded dependencies captured in the grammar and

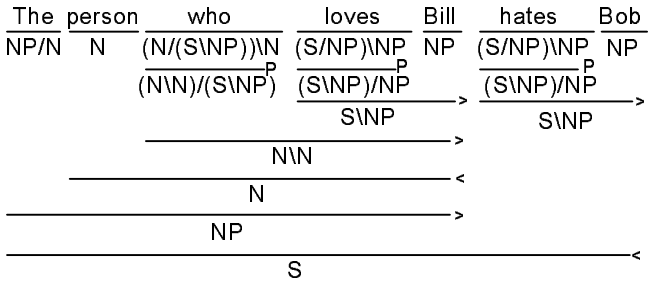

Figure 4.95: Derivation of Sentence *The person who loves Bill hates Bob*

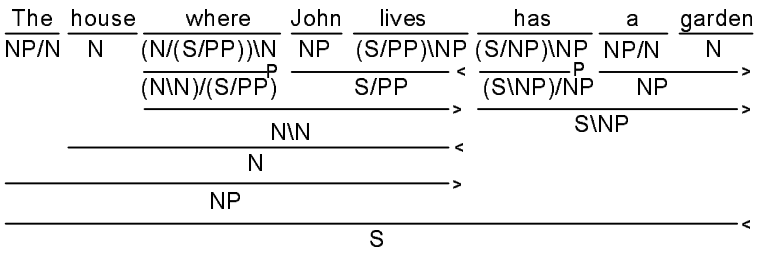

Figure 4.96: Derivation of Sentence *The house where John lives has a garden*

each of these cases is encoded in an appropriate lexical category type. These types are organised in a hierarchy according to their valence and to the type of the subcategorised elements. In the lexicon, these pronouns are assigned as many categories as necessary to describe each of the different constructions in which they take part.

All these cases of unbounded dependencies are captured with the definition of appropriate lexical category types, based on the treatment proposed by Steedman [1985 to 2000], which is implemented in the wide-coverage grammars developed by Doran and Srinivas [in press] and Hockenmaier et al. [2000]. The treatment adopted in this work, however, differs from that proposed by Steedman because he uses the type raising rule for capturing cases of unbounded dependencies. This rule allows, for instance, the verbal category to be combined first with the NP subject, by type raising the NP, and then with the other subcategorised categories. However, since we use the permutation rule that allows the verb to combine with any of its arguments in any order, the effect required for such combination is already obtained. By using GWP we avoid the problem with raising rules which are recursive and, as a consequence, categories can be raised *ad infinitum* unless these rules are constrained. The GWP rule, on the other hand, in spite of being recursive, is bounded by the maximum number of arguments of a given complex category and, consequently, will only be performed a certain number of times. Moreover, the flexibility provided by the use of type raising can allow many unwanted derivations, as found by Murgatroyd [2000], who investigated the use of prosodic phrasing to help resolve parsing ambiguity.

4.4.3 Coordination

Coordination is one of the best known linguistic applications of CGs. In this section the treatment of coordination adopted is discussed, focusing on the so-called coordination of 'like categories', where the conjuncts being coordinated are of the same category.

Coordination involves the combination of constituents using conjunctions like *and* and *or*. The coordination of like categories is implemented using an approach

equivalent to that used in [Gazdar 1981], which regards a coordinate structure as a string of any syntactic category composed of two or more substrings of that category linked by a conjunction. To encode this generalisation, the polymorphic functor category (X\X)/X is used, where X stands for any category. In the grammar different types are used to capture different instantiations of this category, such as NP\NP/NP, ((S\NP)\(S\NP))/(S\NP) and S\S/S, among others.[9] A derivation for the sentence *My brother and sister gave me a book*, that contains a coordination, is shown in figure 4.97. This approach is similar to the ones employed in [Carpenter 1998] and [Doran and Srinivas in press].

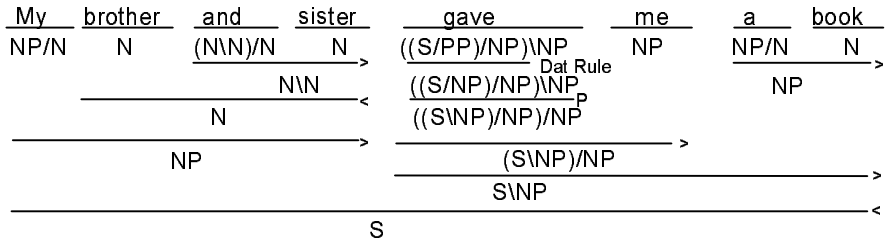

Figure 4.97: Derivation of Sentence *My brother and sister gave me a book*

Due to the weaker notion of constituency in CGs, this approach accounts not only for the cases of constituent coordination, but also for those of non-constituent coordination, which refers to the cases where the conjuncts do not correspond to constituents in the more traditional sense. For instance, in the sentence *Bill likes and Jane hates popcorn*, the conjuncts are composed of subjects and verbs in *Bill likes* and *Mary hates*. Although they are considered valid constituents in CGs, this is not the case for other formalisms that employ a more traditional notion of constituency. In what follows, cases of constituent and non-constituent coordination are discussed.

In the **coordination of NPs**, the conjunction is assigned category NP\NP/NP, as shown in figure 4.98 containing a derivation of the sentence 4.13.

• (4.13) *I like* ***John*** *and* ***Mary****.*

For **coordinating Ns**, the category required for the conjunction is N\N/N, as can be seen in figure 4.97 with a derivation of sentence 4.14.

• (4.14) *My* ***brother*** *and* ***sister*** *gave me a book.*

[9]The use of the category (X\X)/X to treat coordinations causes overgeneration, as in the case of the sentence ****A man who walks*** *and* ***he*** *talks* which would be incorrectly treated as a case of coordination of NPs.

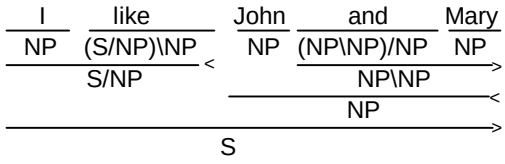

Figure 4.98: Derivation of Sentence *I like John and Mary*

When **coordinating Ss** (sentence 4.15), the conjunction has category S\S/S (figure 4.99).

• (4.15) ***Kim brought a pizza*** *and* ***John ate a slice****.*

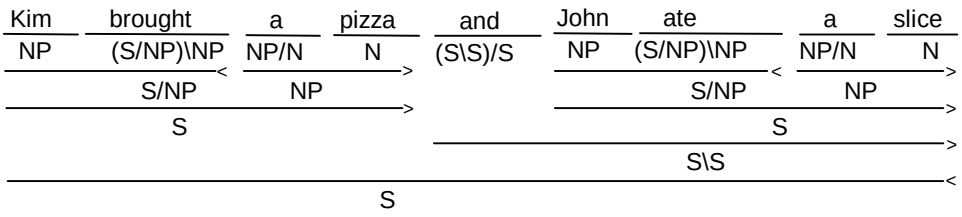

Figure 4.99: Derivation of Sentence *Kim brought a pizza and John ate a slice*

One case of **verbal coordination** is that of Right Node Raising (RNR) (sentence 4.16), where the conjunction has category ((S/NP)\(S/NP))/(S/NP), and the conjuncts are of category S/NP (*You read* and *I taped*) (figure 4.100).

• (4.16) ***You read*** *and* ***I taped*** *the story.*

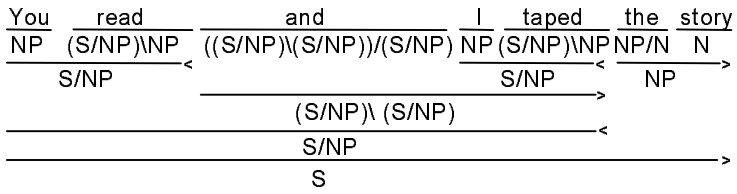

Figure 4.100: Derivation of Sentence *You read and I taped the story*

RNR is another point where the approach to treat coordination we adopted differs considerably from the one used in [Doran and Srinivas in press] and [Steedman 2000]. These works use subject type raising in order to enable the

subject to be combined with the verb, forming a subject-verb conjunct, which is then able to be coordinated. However, as the permutation rule allows the arguments of a given complex category to be combined in any order, the effect obtained is the same: the subject can be combined with the verb, resulting in the subject-verb conjunct that is needed.

Another case of verbal coordination is shown in sentence 4.17, with coordination of verbal conjuncts, where both conjuncts are composed of transitive verbs (*cooked* and *ate*), and the conjunction has category (((S\NP)/NP)\((S\NP)/NP))/((S\NP)/NP) (figure 4.101).

• (4.17) *The man* ***cooked*** *and* ***ate*** *potatoes.*

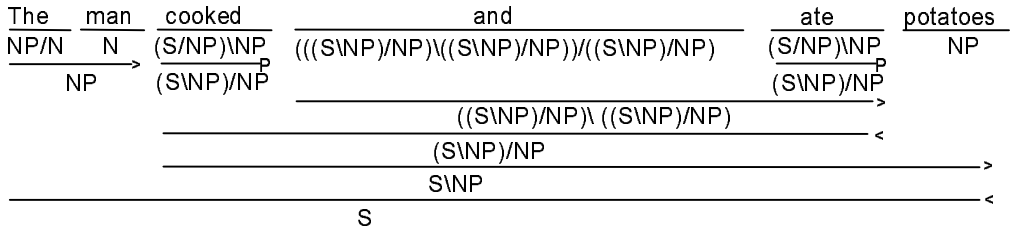

Figure 4.101: Derivation of Sentence *The man cooked and ate potatoes*

A case where the verbal conjuncts have different subcategorisation requirements, is shown in sentence 4.18. The first conjunct is composed of an intransitive verb and the second is composed of a transitive verb and its object. In this case, the object is not shared between the conjuncts; it is actually part of the second conjunct, since only the transitive verb requires a direct object (figure 4.102).

• (4.18) *She* ***smiled*** *and* ***gave him a gift.***

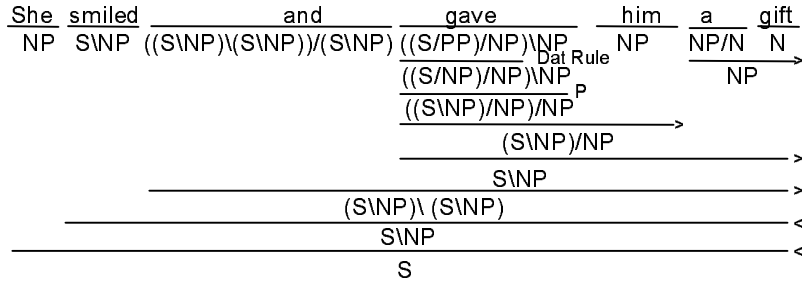

Figure 4.102: Derivation of Sentence *She smiled and gave him a gift*

To capture the case of a ditransitive verb with conjoined pairs of objects, the approach proposed by Wood [1989] is adopted. In this approach the product operator, $*$, is used to form a product from a pair of objects. The products resulting from each pair of objects are coordinated and can serve as argument to a verb looking for a product. Thus, in the lexicon, ditransitive verbs also have the category (S/NP*NP)\NP. This is the case of sentence 4.19, whose derivation is shown in figure 4.103.

• (4.19) *Jane gave* ***Bob a dog*** *and* ***Sue a cat****.*

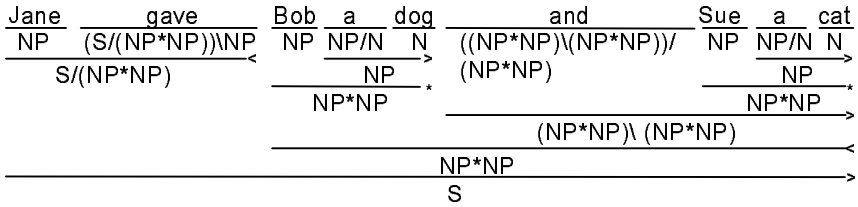

Figure 4.103: Derivation of Sentence *Jane gave Bob a dog and Sue a cat*

These are only some of the constructions that are captured in the grammar. With 89 lexical categories, 19 lexical and morphological rules and 5 grammatical rules, this grammar is able to successfully treat a significant range of linguistic phenomena. Moreover, due to the use of default inheritance hierarchies, the categories are defined in a concise implementation that captures generalisations shared by groups of categories. Subregularities and exceptions are also straightforwardly defined. A subset of the lexical category types defined in the grammar is shown in Appendix A.

4.5 A Possible Universal Grammar

In this section, a possible formalisation of the theory of Universal Grammar (UG) is described. The UG consists of **principles** and **parameters**, and the latter are set according to the linguistic environment [Chomsky 1965], as discussed in chapter 2. This proposal suggests that human languages follow a common set of principles and differ among one another only in finitely many respects, represented by a finite number of parameters that can vary according to a finite number of values.

In this work, UG is represented as a partially underspecified Unification-Based Generalised Categorial Grammar, embedded in a default inheritance network of types. The categories and rules are represented as types in the hierarchies and encoded in terms of TDFSs, as described in the previous sections. The parameters

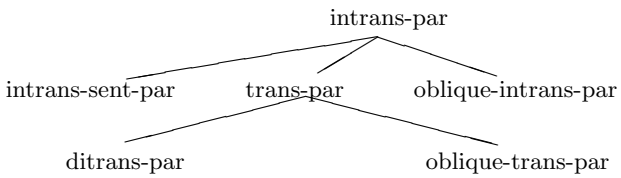

Figure 4.104: Fragment of the Categorial Parameters Hierarchy

of the UG are formalised as types in the grammar also in terms of TDFSs, and this allows the finite values they take to be specified as unset, as default, or as non-default (absolute) values, as required.

The description of how the parameters are implemented in terms of a UB-GCG starts with the encoding of the categorial parameters, which determine the categories that are allowed for a particular language at a given time in learning. It is followed by a description of word order parameters, which reflect the underlying order in which constituents occur in a given language.

Categorial parameters define the categories allowed by the grammar at a particular time during the learning process. Each of the categories in the grammar has one categorial parameter associated with it and its value determines whether the category is present in the grammar or not. If the parameter corresponding to a particular category is active, then the category is allowed as part of the grammar. Otherwise, if it is inactive, the category is not allowed. Then, if the learner, which is discussed in chapter 6, detects that a given categorial parameter is inactive, it does not include the corresponding category as part of its current grammar. As a consequence, it is possible to determine different configurations for the grammar, by activating different parameters, and different languages may use different subsets of categories, as appropriate.

The 89 categorial parameters defined are grouped according to the syntactic type of the categories they are associated with and ordered according to the valence of these categories. The parameters of a given group are organised in increasing order of complexity, reflecting the way the corresponding categories are placed in the syntactic hierarchy, discussed in section 4.2.4. The higher the valence of a category compared to the others with which it is grouped, the more complex it is and the lower in the hierarchy it is placed. Figure 4.104 shows the ordering of a fragment of the verbal categorial parameters, and table 4.1 shows the correspondence between these parameters and their categories. In this hierarchy, **intrans-par** is the basic verbal parameter being associated with intransitive verbs. If it is active, intransitive verbs are part of the grammar. As transitive verbs are defined in terms of intransitive verbs, being more complex in terms of valence than the latter, **trans-par**, the corresponding parameter, is placed lower in the hierarchy than **intrans-par**.

In relation to the grammar, these parameters have the attribute 'VALUE',

Table 4.1: Categorial Parameters

PARAMETERS	CATEGORIES
intrans-par	intrans-sign (S\NP)
trans-par	trans-sign ((S/NP)\NP)
intrans-sent-par	intrans-sent-sign ((S/S)\NP)
oblique-intrans-par	oblique-intrans-sign ((S/PP)\NP)
ditrans-par	ditrans-sign (((S/NP)/NP)\NP)
oblique-trans-par	oblique-trans-sign (((S/PP)/NP)\NP)

taking a boolean as value, which can be specified as either '**true**' or '**false**' (figure 4.105). In each of the categories in the grammar, the relevant categorial parameter is defined with the morphosyntactic attributes (CAT:M-FEATS), in the attribute 'ALLOWED' (figure 4.106 for the intransitive sign type). If ALLOWED is set as **true**, then the category is allowed in the grammar, but if it is set as **false**, then the category is inactive and is not part of the grammar.

$$\begin{bmatrix} \textbf{intrans-par} \\ \text{VALUE} : \textbf{/true} \end{bmatrix}$$

Figure 4.105: Specification of intrans-par Type

During the learning process, these parameters are activated, starting from less complex to more complex, as the corresponding categories are required for analysing sentences. Initially, as only some of the supertype categorial parameters are active, with VALUE:**/true**, the subtypes that are inactive override this value with **false**, breaking the inheritance chain. As learning progresses and more categories become active, the corresponding subtype categorial parameters are

$$\begin{bmatrix} \textbf{intrans-sign} \\ \text{RESULT:SIGN:CAT:M-FEATS} : \begin{bmatrix} \text{ALLOWED} : \begin{bmatrix} \textbf{intrans-par} \\ \text{VALUE} : \textbf{/true} \end{bmatrix} \\ \text{V2} : \textbf{false} \\ \text{VFORM} : \textbf{/base} \\ \text{AUX} : \textbf{/false} \\ \text{INV} : \textbf{/false} \end{bmatrix} \\ \text{ACTIVE} : \langle ! \begin{bmatrix} \text{SIGN} : \textbf{np-sign} \\ \text{DIRECTION} : \begin{bmatrix} \textbf{subjdir} \\ \text{DIR-VALUE} : \textbf{backward} \end{bmatrix} \end{bmatrix} ! \rangle \end{bmatrix}$$

Figure 4.106: Specification of intrans-sign Type

also activated and consequently can inherit the value **true** from their supertypes. Thus, as more categories become active, more categorial parameters inherit their values by default from their supertypes, with the grammar becoming increasingly more concise. For example, if **trans-par** is active and a ditransitive category is necessary for analysing a sentence, its corresponding parameter, **ditrans-par**, can only be activated if the supertype **trans-par** is active, since this is the categorial parameter corresponding to transitive verbs and ditransitive verbs are defined in terms of transitive verbs. However, a parameter can only be activated if its supertype is active, but, if the latter is inactive, then the subtype has to remain inactive too. Thus, if a ditransitive category is needed and neither **ditrans-par** nor **trans-par** are active, then the former cannot be activated and the category for ditransitive verbs is not allowed in the grammar. This means that ditransitive verbs, being defined in terms of transitive verbs, can only become available in the grammar after transitive verbs also are. As a consequence, categories become available in the grammar in increasing order of complexity. Moreover, at any given time, it is possible to know which categories are available by examining the current parameter values.

In terms of learning, such an encoding of categorial parameters in the grammar allows for an account of the incremental nature of language acquisition, by gradually activating these parameters from less complex to more complex. Thus, given a specification of the categories that are initially allowed as part of the grammar, with the corresponding parameters being set as active (VALUE:/**true**), all the other categorial parameters are initialised as inactive (VALUE:/**false**). Then, as learning progresses, the parameters are gradually activated and increasingly more complex categories are made available as part of the grammar.[10]

Word order parameters determine the underlying order in which constituents appear in a given language. For example, for English, an SVO language, these parameters specify that, in the canonical order, subjects appear to the left of the verb, and objects to the right. On the other hand, they specify that, for German, an SOV language, both the subject and the object appear to the right of the verb [Croft 1992]. These parameters are based on the ideas proposed by Briscoe [1997 to 1999], taking into account typological considerations [Comrie 1981], [Croft 1992].

There are 18 word order parameters defined and they are implemented as types in a hierarchy. The supertype is **gendir** that specifies, by default, the general direction for a language. Among its subtypes, we have **subjdir** that specifies the direction of the subject, **vargdir** the direction of the other verbal arguments, and **ndir2** the direction of nominal categories. A fragment of the parameters hierarchy can be seen in figure 4.107.

Word order parameters have one attribute, DIR-VALUE, that takes a value

[10]Chapters 6 and 7 provide a more detailed explanation of the learning of these parameters, looking at the construction of the categorial parameter hierarchy.

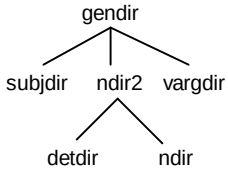

Figure 4.107: Fragment of the Parameters Hierarchy

Table 4.2: Types and Parameters

SIGN	PARAMETERS
intransitive signs	(subjdir = S\\NP)
transitive signs	(subjdir = (S/NP)\\NP) (vargdir = (S//NP)\NP)
oblique intransitive signs	(subjdir = (S/PP)\\NP) (vargdir = (S//PP)\NP)
intransitive with sentential complement signs	(subjdir = (S/S)\\NP) (vargdir = (S//S)\NP)
ditransitive signs	(subjdir = ((S/NP)/NP)\\NP) (vargdir = ((S/NP)//NP)\NP) (vargdir = ((S//NP)/NP)\NP)
oblique transitive signs	(subjdir = ((S/PP)/NP)\\NP) (vargdir = ((S/PP)//NP)\NP) (vargdir = ((S//PP)/NP)\NP)

of type **direction**, which can be either **backward** or **forward** (figure 4.108). Regarding the categories of the UB-GCG, word order parameters specify the direction of each element in the subcategorisation list of a complex category. If the parameter has value **forward** the subcategorised category is to be found to the right and, if the value is **backward**, the category is to the left. This can be seen in figure 4.106, for the intransitive sign, whose subject NP has direction set as **backward** by the **subjdir** parameter. Table 4.2 shows the correspondence between some of these parameters and some of the complex categories, where the double slash indicates the position of the parameter in the category.

Moreover, since the parameters are organised in a default hierarchy, in the absence of conflicting more specific values, they inherit their values by default from their supertypes. For instance, in the case of English, the value of **gendir** is defined, by default, as **forward**, capturing the fact that it is a predominantly right-branching language (figure 4.108) and all its subtypes, like **subjdir** (figure 4.109) and **vargdir** (figure 4.110) inherit this default information. As the categories are also defined in terms of an inheritance hierarchy, the parameters and their values in these categories are propagated throughout the hierarchy. Thus subtypes inherit this information by default in the absence of the definition of a more specific value. For example, an intransitive verb, which has the direction of the subject specified by **subjdir** (DIRECTION:**subjdir**), will be defined as S/NP if **subjdir** has default value **forward** (figure 4.111). However, if as in English the subject occurs to the left of the verb, utterances with the subject to the left will trigger a change in **subjdir** to **backward**, which overrides the default value, breaking the inheritance chain (figure 4.112). As a result, intransitive verbs are defined as S\NP, figure 4.106, for the grammar to account for these sentences. In the syntactic dimension of this network, intransitive verbs are considered the basic case of verbs and the information defined in this node is propagated through the hierarchy to its subtypes, such as the transitive verbs. This can be seen in figure 4.113, which shows the specification of the transitive verb type, where **vargdir** defines the direction of the direct object. Figure 4.114 shows the transitive verb type expanded with the information that it inherits from the intransitive verb type, including the direction of the subject, defined by **subjdir**.

$$\begin{bmatrix} \textbf{gendir} \\ \text{DIR-VALUE} : \textbf{/forward} \end{bmatrix}$$

Figure 4.108: Gendir Parameter

For the learner, the information about subjects (DIR-VALUE:/**backward**), figure 4.112, has already been acquired while learning intransitive verbs, figure 4.106. Thus the learner does not need to learn it again for transitive verbs (figure 4.114) which not only inherit this information but also have the direction for the

$$\left[\,\textbf{subjdir}\,\right]$$

Figure 4.109: Subjdir Parameter

$$\left[\,\textbf{vargdir}\,\right]$$

Figure 4.110: Vargdir Parameter

$$\left[\begin{array}{l}\textbf{subjdir}\\ \text{DIR-VALUE} : \textbf{/forward}\end{array}\right]$$

Figure 4.111: Subjdir Parameter Expanded

$$\left[\begin{array}{l}\textbf{subjdir}\\ \text{DIR-VALUE} : \textbf{/backward}\end{array}\right]$$

Figure 4.112: Subjdir Parameter after Trigger

$$\left[\begin{array}{l}\textbf{trans}\\ \text{RESULT:SIGN:CAT} : \textbf{s}\\ \text{ACTIVE} : \,<!\left[\,\top\,\right], \left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \textbf{vargdir}\end{array}\right]!>\end{array}\right]$$

Figure 4.113: Interaction of the Transitive Verb Type with the Vargdir Parameter

$$\left[\begin{array}{l}\textbf{trans}\\ \text{RESULT:SIGN:CAT} : \textbf{s}\\ \text{ACTIVE} : \,<!\left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \left[\begin{array}{l}\textbf{subjdir}\\ \text{DIR-VALUE} : \textbf{backward}\end{array}\right]\end{array}\right],\\ \qquad\qquad \left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \left[\begin{array}{l}\textbf{vargdir}\\ \text{DIR-VALUE} : \textbf{forward}\end{array}\right]\end{array}\right]!>\end{array}\right]$$

Figure 4.114: Transitive Verb Type Expanded

object defined by **vargdir** (DIR-VALUE:/**forward**), figure 4.110.

Another parameter that is related to the canonical order of constituents in different languages is the V2 parameter. This parameter regulates verb-second phenomenon in main clauses. If it is activated, it allows the tensed verb to be placed after the first constituent in main clauses. As a result, for a language like German, which is SOV, V2 allows the verb to occur after the first constituent, resulting in orders like SVO, Adverb-VSO, and OVS ([Hyams 1986], [Roeper 1992]

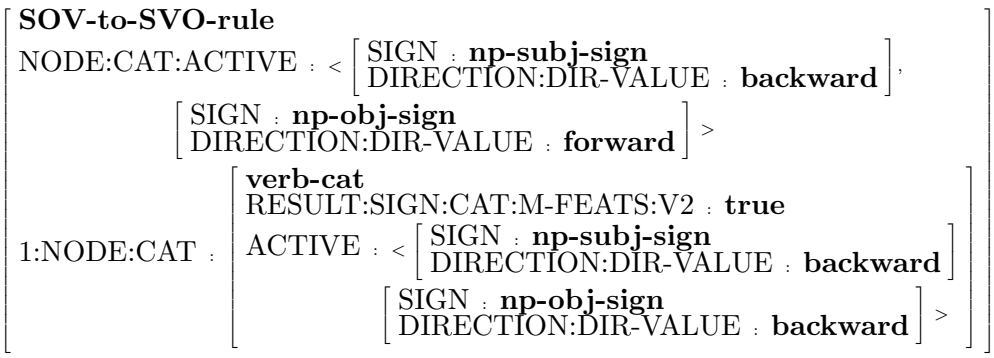

Figure 4.115: SOV to SVO Rule

and [Kapur and Clark 1996]). If the parameter is inactive, this phenomenon is not possible, as is the case for English. In terms of the categories in the grammar, this parameter is encoded in the v2 attribute as a verbal morphological attribute with a boolean value (figure 4.106). If the parameter is activated, the attribute has value **true** and the language allows the verb-second phenomenon, which is generated using lexical rules. Otherwise, if the parameter is inactive, the attribute has value **false** and this movement is not allowed in the language. For example, let us assume that a lexical rule such as the putative rule shown in figure 4.115 is used to change the canonical order of the constituents from SOV to SVO. This rule is restricted to apply only to lexical signs whose V2 attribute is set as **true** and that have SOV order, with the directions of both the subject and the object set as **backward**. This is the case for German and the rule can be successfully applied. However, such a rule could not be applied to a language such as English, in which V2 is set to **false** and where the direction of the object is set as **forward**. There may be many such rules to capture the movement of different V2 constructions, in different V2 languages, and they are applied as appropriate for a given language. For example, the rule shown in figure 4.115 could only be applied to an SOV V2 language such as German because the direction of both the subject and the object are required by this rule to be set as **backward**. In this way, a language learner would have access to all such rules and these would be selected as appropriate for each language.[11]

The UG has to be general enough to capture the grammar for any language. Then, the parameters have to be set to account for a particular language, based

[11]The question of the learning of lexical rules, investigating how the learner would learn which rules are appropriate for a given language, is outside the scope of this work. This constitutes a second stage in the learning process, when the learner has acquired enough information for the patterns occurring in the lexicon to become apparent, building on the basis created by this investigation for such a learning task. The learning of patterns in the lexicon is investigated by Schütze [1995], however, he uses a different approach, employing data-intensive methods that make it incompatible with this work.

on exposure to that language. Among these sentences, some will be triggers for certain parameters, in the sense that some of the parameters will have to be set to a certain values in order to capture these sentences, generating the appropriate logical form. Thus, given a triggering sentence, some of the categorial parameters may have to be set as **true** for the necessary categories to be available, and for those available categories, some of the word order parameters may need to have a specific value if the sentence is to be successfully analysed. In terms of the learning process, having the parameters embedded in a default inheritance hierarchy means that, even before a parameter is set by triggering data, it has a value inherited by default from a supertype. This value will hold until the occurrence of a trigger reinforces or disconfirms it. Furthermore, such an encoding of the UG allows the definition of unset, default and non-default absolute values, being compatible with many proposals in the language acquisition literature, such as [Chomsky 1981], [Hyams 1986], and [Lightfoot 1991] among others, that advocate that some parameters need to be initialised with default values, in the beginning of the learning process.

The use of a default inheritance schema reduces the amount of information to be acquired by the learner, since the information is structured and what is learned is not a single isolated category, but a structure that represents a candidate category set. This is a clear and concise way of defining the UG with the parameters being straightforwardly defined in the categories. It uses effectively the default inheritance mechanism to propagate information about parameters throughout the lexical inheritance network. This approach is well suited for a Principles and Parameters Theory of UG, with very general grammar rules and categories defined as types arranged in a default inheritance hierarchy which is a kind of structure that is likely to have an important role in the way people organise many kinds of information, as pointed out by Sag and Wasow [1999]. For a language learner, it is convenient to organise linguistic information in multiple default inheritance hierarchies, since when learning new information, the learner can use the existing hierarchies to classify the new information and needs only to encode the minimum amount of arbitrary information.

4.6 The Annotated Sachs Corpus

The Principles and Parameters Theory suggests that children set the parameters of the UG to a particular language, based on exposure to sentences provided by their linguistic environment, spoken in a certain context. One of the ways in which to approximate that is to use a corpus of utterances annotated with logical forms, based on which parameters would be set to account for that specific language. This section describes the particular corpus used in this work, as well as the pre-processing and annotation required for the corpus to be used as the basis for learning.

The corpus employed in this work is a subset of the Sachs corpus [Sachs 1983] [MacWhinney 1995]. The Sachs corpus consists of Jacqueline Sachs' naturalistic longitudinal study of Naomi, her daughter, which contains spontaneous interactions between the child and her parents, phonemically transcribed, from the age of 1 year and 1 month to 5 years and 1 month. This corpus contains a total of 29,814 sentences in English. From these sentences we extracted material for 2 different corpora: one with the parents' sentences, containing 12,105 sentences and the other with the child's sentences, containing 17,709 sentences.

The use of the parents' corpus as the basis for learning allows us to approximate the linguistic input that a child receives when learning her native language. Moreover, by annotating this corpus with logical forms, it is possible to simulate a part of the context a child has when listening to a sentence.

In order to annotate the parents' corpus, it first needed to be pre-processed to remove noise, phonological annotations, and some grammatical constructions that are not relevant to this project. As this work concentrates on the investigation of verbal predicate argument structures, sentences such as 4.20 to 4.25 are not suitable for this study, being removed from the corpus to be annotated. The original parents' corpus contains 12,105 sentences, and the first stage of pre-processing removed around 15% of the sentences containing noise. From the remaining 10,292, a subset containing 4,650 sentences was selected from a period ranging from when Naomi was 14 months to when she was 24 months of age. These sentences went through a second stage of pre-processing and sentences that contained only nominal constructions, interjections, elliptical constructions, or that were incomplete were removed. From the 4,650 sentences, around 65% were removed after the second stage of pre-processing, and the resulting corpus to be annotated contained the remaining 1,517 sentences.

- (4.20) *woofo woofo*
- (4.21) *baby birdie*
- (4.22) *and a cow says ...*
- (4.23) *yuck!*
- (4.24) *give me a kiss and I'll get the xxx for you.*
- (4.25) *mmhm snow at the window huh?*

After the pre-processing, the selected subset of the parents' corpus was annotated with logical forms. The annotated Sachs corpus contains 1,517 sentences with a wide variety of constructions, with for example, declarative, imperative and interrogative sentences with rich verbal constructions, and unbounded dependencies, as can be seen in sentences 4.26 to 4.28, extracted from the corpus.

- (4.26) *Are you done with looking at the pictures?*
- (4.27) *He is going to turn off the lights*

• (4.28) *Go in the kitchen and see Daddy*

In order to annotate the parents' corpus with the associated logical forms, a UB-GCG for English that covers all the constructions in the corpus was built.[12]

The parents' corpus contains sentences annotated with logical forms, and an example of the annotation can be seen in figure 4.116, for the sentence '*I will take him*', showing only some of the features for reasons of clarity. The logical form associated with a sentence is constructed using the Sag and Wasow's version of MRS, as explained in section 4.2.2. Thus, each predicate in the semantics list is associated with a word in the sentence, and, among other things, it contains information about the identifier of the predicate (SIT), the required arguments (e.g. ARG1 for the agent role and ARG2 for the patient role, for the verb *take*), as well as about the interaction among the predicates, specified by the boxed indices in the argument roles.

$$\left[\begin{array}{l}\textbf{sign}\\ \text{ORTH} : \langle !\textbf{i, will, take, him} ! \rangle\\ \text{CAT} : \textbf{s}\\ \text{SEM} : \left[\begin{array}{l}\text{MODE} : \textbf{prop}\\ \text{INDEX} : \boxed{1}\ \textbf{index}\\ \text{RESTR} : \langle ! \left[\begin{array}{l}\text{RELN} : \textbf{r-will}\\ \text{SIT} : \boxed{1}\\ \text{ARG1} : \boxed{2}\ \textbf{index}\end{array}\right], \left[\begin{array}{l}\text{SIT} : \boxed{3}\ \textbf{index}\\ \text{PRON} : \textbf{i}\\ \text{NAMED} : \boxed{4}\textbf{index}\end{array}\right], \\ \qquad\qquad \left[\begin{array}{l}\text{RELN} : \textbf{r-take}\\ \text{SIT} : \boxed{2}\\ \text{ARG1} : \boxed{4}\\ \text{ARG2} : \boxed{5}\ \textbf{index}\end{array}\right], \left[\begin{array}{l}\text{SIT} : \boxed{6}\ \textbf{index}\\ \text{PRON} : \textbf{him}\\ \text{NAMED} : \boxed{5}\end{array}\right] ! \rangle\end{array}\right]\end{array}\right]$$

Figure 4.116: Sentence *I will take him*

The annotated sentences have a mean length of around 4.5 words, and from the 6,893 words in the annotated corpus there are 706 different word forms being used in these sentences. Apart from determiners, prepositions, and similar closed class words, the most frequent categories occurring in these sentences are all verbs, as can be seen in table 4.3. In relation to verbal constructions, verbs occur 2,247 times in the corpus. Most of the main verbs are transitive verbs, occurring approximately 24% of the time, while intransitive verbs occur only 7.7% of the time. Sentences containing copula constructions are especially frequent, accounting for 33% of the sentences. Unbounded dependencies occurred in 9% of the sentences, with the vast majority of these cases being wh-questions with object extractions, occurring in 7.5% of the sentences.

[12]The characteristics of such a grammar, as well as a discussion of the constructions it covers, are presented in the previous sections (4.1 to 4.4).

Table 4.3: Ten Most Frequent Open-Class Words

Word	Frequency
is	553
are	159
do	157
want	108
can	64
doing	59
put	64
don't	47
say	44
see	42

4.7 Summary

In this chapter, the implemented Unification-Based Generalised Categorial Grammar was described. It consists of categories and rules, implemented as types in a default multiple inheritance hierarchy which contains information about orthographic, syntactic, semantic and linking constraints. Generalisations are encoded in types higher up in a hierarchy and are propagated to their subtypes, while subregularities and exceptions are encoded by means of default specifications. As a consequence, the resulting grammar captures concisely linguistic phenomena, with different types of linguistic information being localised in different hierarchies, in an encoding that is easier to maintain and modify. With 89 lexical categories, 19 lexical and morphological rules and 5 grammatical rules, this grammar is able to capture elegantly a wide variety of linguistic constructions, including unbounded dependencies and coordinations.

This chapter also discussed how such an encoding is used to formalise a theory of the Universal Grammar and associated parameters. They correspond to types in the hierarchies, in an approach that provides a straightforward integration of the parameters with the categories in the grammar. Moreover, the use of default inheritance hierarchies allows information about the parameters to be propagated from supertype to subtype, which is an important characteristic from a learning perspective. In order to provide a linguistic environment for parameter setting, a subset of the Sachs corpus, containing interactions between a child and her parents, was annotated with logical forms. With this annotated corpus, it is possible to simulate the environment in which a child acquires her native language.

Chapter 5

Learning a Lexicon

In this chapter, two systems are described. The first one, the semantics learner, attempts to learn the meanings of words from data. The second system, the syntax learner, attempts to learn possible syntactic categories for the words in a sentence so that a complete derivation is obtained. The semantics learner, was implemented by Waldron [1999], based on an algorithm developed by Siskind [1996], while the syntax learner was both developed and implemented by Waldron [Waldron 1999 and 2000].

These two systems, as implemented by Waldron, are used to pre-process the annotated Sachs corpus, assigning to each word in a sentence putative semantic predicates and syntactic categories, which can be used as the basis for setting the parameters of the UG. The descriptions provided in this chapter are based on Waldron's implementations, and concentrate especially on the semantics learner, since it can be used on its own as the starting point for the investigation proposed in this thesis. The syntax learner is only used for reasons of expediency, proposing putative syntactic categories for words in a sentence that result in possible derivations.

5.1 Learning the Meanings of Words

Siskind [Siskind 1996] developed a computational model of lexical acquisition. He investigated the learning of the mapping between words and logical forms primarily from a computational perspective, but also related this task to that faced by children when learning the meanings of words in their native language.

The algorithm he developed attempts to learn a lexicon containing for each word an appropriate logical form, by analysing input sentences paired with possible meanings. The algorithm does not rely on the use of single word utterances for the meaning of words to be unambiguously determined, or on prior access to language specific information.

The use of sentence-meaning pairs relates to the problem of referential uncer-

tainty [Siskind 1993] that is faced by children who, when hearing an utterance, may not be able to uniquely determine the meaning of the utterance from context. Thus, each sentence is paired with a set of possible meanings, among which there is usually the correct one. However, the algorithm can also handle cases where a sentence is paired with only incorrect meanings, which correspond to noisy input, only requiring that utterances be non-noisy for a sufficient fraction of the time. The algorithm also deals with polysemy, which is a lexical semantic ambiguity where words may have more than one single meaning.

The learning process is regarded as a mapping problem, where a mapping from words to meanings has to be learned, such that the meanings of words, when combined, form the meanings of utterances containing those words. Semantic interpretation is assumed to be compositional, with the meaning of an utterance being composed of the meaning of each word in the utterance, and no information can be added or deleted from the meanings of these words. As the lexicon learned has each word mapped to a set of meanings denoting different senses for that word, the meaning of an utterance is determined by choosing exactly one sense for each word in that utterance.

For example, if the first sentence the learner receives is:

• (5.1) *Mary lifted the block*,

paired with the meaning:

$$\left[\begin{array}{l}\textbf{sem}\\ \text{RESTR} : \left\langle ! \begin{bmatrix}\text{RELN} : \textbf{lift}\\ \text{SIT} : \boxed{0}\ \textbf{index}\\ \text{ARG1} : \boxed{1}\ \textbf{index}\\ \text{ARG2} : \boxed{2}\ \textbf{index}\end{bmatrix}, \begin{bmatrix}\text{SIT} : \boxed{3}\ \textbf{index}\\ \text{NAME} : \textbf{Mary}\\ \text{NAMED} : \boxed{1}\end{bmatrix}, \right.\\ \qquad\qquad \left. \begin{bmatrix}\text{RELN} : \textbf{the}\\ \text{SIT} : \boxed{4}\ \textbf{index}\\ \text{BV} : \boxed{2}\end{bmatrix}, \begin{bmatrix}\text{RELN} : \textbf{block}\\ \text{SIT} : \boxed{5}\ \textbf{index}\\ \text{INST} : \boxed{2}\end{bmatrix} ! \right\rangle\end{array}\right]$$

which can be represented in a linear form as:

{lift($\boxed{0}$, $\boxed{1}$, $\boxed{2}$), mary($\boxed{3}$, $\boxed{1}$), the($\boxed{4}$, $\boxed{2}$), block($\boxed{5}$, $\boxed{2}$)}

the learner is left with the task of finding out which word in the sentence maps to which portion of the meaning of the sentence. The meaning of a sentence is presented inside curly brackets, with predicates having their arguments represented inside brackets and separated by commas. For instance, in the logical form associated with this sentence, lift($\boxed{0}$, $\boxed{1}$, $\boxed{2}$) has an event variable, $\boxed{0}$, and two arguments, ' $\boxed{1}$ ' and ' $\boxed{2}$ ', where ' $\boxed{1}$ ' is coindexing the agent of 'lift' (in ARG1) to 'mary', and $\boxed{2}$ the undergoer of 'lift' (in ARG2) to 'the block'. The learner needs

to determine which of the predicates (lift([0] , [1] , [2]), mary([3] , [1]), the([4] , [2]) and block([5] , [2])) to associate with each of the words in the sentence. For reasons of clarity, the linear notation is used throughout this document, with event variables not being explicitly represented unless otherwise required. Thus the logical form associated with the sentence is represented as:

{lift([1] , [2]), mary([1]), the([2]), block([2])}

The meaning paired with the sentence is decomposed into its component meanings (predicates), where the bound variables, represented by indices such as [1] and [2] , are replaced by free variables, represented by *a*, *b*, ..., given the following Decomposition Rule:

- In a predicate, every variable argument that is bound has its value replaced by a free variable.

Applying the Decomposition Rule to the sentence *Mary lifted the block*, with meaning {lift([1] , [2]), mary([1]), the([2]), block([2])}, results in {lift(a,b), mary(c), the(d), block(e)} where the bound variable arguments in the logical form ([1] , [2] , ...) are replaced by free variables (*a, b, ...*). Each of these elements of the meaning set can be associated with any of the words in the utterance, and thus initially, the hypotheses the learner has are:

Mary$_1$: {lift(a,b), mary(c), the(d), block(e)}
lifted$_1$: {lift(a,b), mary(c), the(d), block(e)}
the$_1$: {lift(a,b), mary(c), the(d), block(e)}
block$_1$: {lift(a,b), mary(c), the(d), block(e)}

where each word sense in the lexicon is represented with a subscript number, and has a set of possible hypothesised meanings, inside curly brackets, separated by commas. Then, when hearing the utterance:

• (5.2) *John lifted the ball*,

paired with the meaning:

{lift([1] , [2]), john([1]), the([2]), ball([2])}

which is decomposed into {lift(a,b), john(c), the(d), ball(e)}, the learner would hypothesise that:

John$_1$: {lift(a,b) john(c), the(d) ball(e)}
lifted$_1$: {lift(a,b) john(c), the(d) ball(e)}

the_1: {lift(a,b) john(c), the(d) ball(e)}
ball_1: {lift(a,b) john(c), the(d) ball(e)}

This sentence contains two words that were seen before, and for which the learner has already formulated hypotheses: *the* and *lifted*. By comparing the newly hypothesised meanings for *lifted* and *the* with the hypotheses created after the previous utterance, the learner determines that two of the meaning elements for these words are common between the two hypotheses:

lifted_1: {lift(a,b) john(c), the(d) ball(e)} $\cap$ {lift(a,b), mary(c), the(d), block(e)}
the_1: {lift(a,b) john(c), the(d) ball(e)} $\cap$ {lift(a,b), mary(c), the(d), block(e)}

The learner can then reduce the set of possible meanings for both *lifted* and *the* to:

lifted_1: {lift(a,b), the(c)}
the_1: {lift(a,b), the(c)}

In this case, the learner used common meaning elements across each occurrence of a word, doing **cross-situational inference** to try to determine the meaning of the word.

Informally, the learning system does not know the meaning of any word initially, so it implicitly hypothesises that each word can mean anything. Partial knowledge of word meanings can allow the learner to filter out impossible hypothesised meanings of utterances that contain those words. Then, as sentences are processed, the set of possible meanings for each word sense - which starts as containing all possible meanings - is reduced until there is only one possible meaning for each sense, which is when the word sense is determined. The intuition behind the algorithm is that the use of intersection in all the contexts in which a particular word appears helps to determine the meaning of that word. Thus a word cannot mean something that is not contained in the meaning of an utterance containing that word. Such an approach uses two kinds of implicit information: the different utterances in which a word appears, and the different words appearing in an utterance. The learner assumes that words in an utterance must contribute non-overlapping portions of the utterance meaning, adopting the **Principle of Exclusivity**, which uses the knowledge about the meaning of some words in an utterance to constrain the possible meanings of other words in the same utterance.

When different senses of a word are used, problems will occur if the word becomes inconsistent. A word is considered to be inconsistent when, for a given sense, its meaning set becomes empty. This happens when an utterance is processed and a new sense for a word is used, because the intersection between the possible meanings hypothesised for this new sense and another previously used sense for the same word may not contain common elements. A sentence is consid-

ered to be inconsistent if any of its words is inconsistent. For example, assuming that later on the learner is given the sentence:

- (5.3) *These cars block this road*

paired with the meaning:

{block(1,2), these(1), cars(1), this(2), road(2)}

the learner would hypothesise that:

these$_1$: {block(a,b), these(c), cars(d), this(e), road(f)}
cars$_1$: {block(a,b), these(c), cars(d), this(e), road(f)}
block$_1$: {block(a,b), these(c), cars(d), this(e), road(f)}
this$_1$: {block(a,b), these(c), cars(d), this(e), road(f)}
road$_1$: {block(a,b), these(c), cars(d), this(e), road(f)}

By intersecting the newly hypothesised meanings for *block*, with the hypotheses created after the first utterance:

$$\begin{aligned} \text{block}_1 &= \{\text{block(a,b), these(c), cars(d), this(e), road(f)}\} \\ &\cap \{\text{lift(a,b), mary(c), the(d), block(e)}\} \\ &= \{\} \end{aligned}$$

the result is the empty set.

A word that becomes inconsistent will affect the learning of other words, with the learner not being able to reliably determine the meanings of these words, as discussed in Siskind [1996]. In order to deal with this polysemy and to avoid learning noisy meanings, the algorithm first performs a consistency test that checks if, for a given sentence-meaning pair, inconsistencies will occur during learning. If that is the case, the algorithm determines the minimum number of word senses that need to be hypothesised for the utterance to be no longer inconsistent and then processes the utterance as normal. Moreover, as each word sense converges on meaning, it is assigned a confidence factor, which is initially zero, but which is incremented every time it is used. The confidence factor collects the evidence for each of the word senses, reducing the effects of noise which could mislead the learner into converging to an incorrect meaning for a particular word sense.

Let us assume that several words in the lexicon converged on meaning and were assigned a confidence factor. Then, the following is a fragment of the cur-

rent lexicon, after the new sense of "*block*", block_2, is added:

block_1: {block(a)}, C=2
block_2: {block(a,b)}, C=0
he_1: {he(a)}, C=7
bought_1: {buy(a,b)}, C=4
the_1: {the(a)}, C=10

When, subsequently, a sentence like:

• (5.4) *He bought the block*

paired with the meaning:

{(buy(1,2), he(1), the(2), block(2)}

is given to the learner, as there are two distinct senses of *block*, the learner does not know which one to use. As a consequence, it needs to consider the two possibilities:

- He_1 bought_1 the_1 block_1,

whose meaning has as components:

{he(a), buy(b,c), the(d), block(e)}, and,

- He_1 bought_1 the_1 block_2,

whose meaning includes:

{he(a), buy(b,c), the(d), block(e,f)}

However, only the first one

He_1 bought_1 the_1 block_1 = {he(a), buy(b,c), the(d), block(e)},

contains all the necessary meaning components, and can be recomposed into:

{(buy(a,b) he(a) the(b) block(b)),
(buy(a,b) the(a) block(a) he(b),
(buy(a,b) the(a) he(a) block(b)),
(buy(a,b) block(a) the(b) he(b)),
(buy(a,b) he(a) the(c) block(c)),

(buy(a,b) he(b) the(c) block(c)),
(buy(a,b) the(a) block(a) he(c)),
(buy(a,b) the(b) block(b) he(c)),
(buy(a,b) the(a) block(b) he (c),
(buy(a,b) block(a) the(b) he(c)),
(buy(a,b) block(a) he(b) the(c)),
(buy(a,b) he(c) the(d) block(e)), ... }

performing an operation that corresponds to a reverse application of the Decomposition Rule, instantiating the variables. In the resulting set of meanings, the first element (buy(a,b) he(a) the(b) block(b)) is the desired sense assignment, which corresponds to the meaning paired with the utterance. Having found the sense assignment that is consistent with the meaning paired with the utterance, the learner continues processing as usual.

The learner needs to consider all possible senses for the words in the utterance being processed, aiming to find the combinations of sense assignments that are compatible with the meaning paired with the utterance. In order to do so, the learner verifies the consistency of each sense assignment in the cross product of the senses of the words in the utterance (e.g. $\{\text{He}_1\ \text{bought}_1\ \text{the}_1\ \text{block}_1\}$ and $\{\text{He}_1\ \text{bought}_1\ \text{the}_1\ \text{block}_2\}$). If no consistent sense assignment is found, then the smallest number of new meanings is incrementally added to the set of possible senses for words in the utterance until the utterance becomes consistent. Then, if there is exactly one consistent sense assignment as in this example, the word senses contained in the sense assignment are used for processing the sentence, and processing continues as normal. If there are more consistent sense assignments, the best sense assignment is selected and processing continues as normal. Given that each word sense is assigned a confidence factor corresponding to the frequency with which this word sense is used after convergence, the best sense assignment is defined as the one that has the highest overall confidence factor, which is computed as the sum of the confidence factors for each word sense used.

5.1.1 The Algorithm

In this section, the algorithm developed by Siskind [Siskind 1996] is described, as implemented by Waldron [1999]. Waldron's implemented algorithm differs slightly from Siskind's. The main difference between them is that Siskind differentiates between words with a null logical form (e.g. determiners and infinitivals) and those with non-null logical form (e.g. nouns and verbs). Waldron considers all the words to have a non-null semantics, providing a more uniform treatment for all words.

The algorithm is on-line in the sense that each utterance is processed and discarded before the next utterance can be processed, in a single pass of the

corpus, retaining only information about the mappings from words to senses. These mappings are stored in three tables:

- $L_{sem}(w)$ - maps each word w to a set of senses.
- $P_{sem}(s)$ - maps each sense s to a set of meanings. Each sense is initially mapped to any allowed meaning, and the algorithm monotonically removes elements of $P_{sem}(s)$ until it is a singleton, which is when sense s converged. When a sense converges, it is placed in $L_{sem}(w)$.
- $T_{sem}(s)$ - maps each sense s to a confidence factor C, indicating the evidence for the sense. The confidence factor of a sense is initially mapped to zero, and gradually increases as the evidence for the sense increases. When the temperature T_{sem} for a given sense s reaches a constant, μ, the sense *freezes* and is considered to be learned.

Each sense passes through three stages, starting unconverged, converging on meaning and being frozen. Different senses can be in different stages at a given time. Senses that are not frozen, that is, whose confidence factor did not reach the threshold μ, are subject to a garbage collection mechanism and removed from consideration.

The algorithm used for learning word senses is **ProcessUtterance**:

Procedure ProcessUtterance(Utterance, Meaning)

Step 1 - *If there is at least one consistent sense assignment* $\{s_1,..., s_n\}$ *for the utterance, choose the one with the highest confidence factor (computed as the sum of the confidence factors of each word sense* $\{T_{sem}(s_1) + ... + T_{sem}(s_n)\}$ *), and then execute* ***Process****(*$\{s_1, ..., s_n\}$*, Meaning).*

This step verifies if there is at least one sense assignment that is consistent with the meaning of the utterance, and after determining the one that has the highest confidence factor, the algorithm processes the utterance.

Step 2 - *Otherwise, find the smallest subset u' ⊆ Utterance such that if a unique sense is added to* $P_{sem}(w)$ *for each* $w \in u'$*, the utterance becomes consistent.*

The second step finds the smallest number of new word senses that need to be added to $P_{sem}(w)$ to make the utterance consistent.

Step 2.1 - *Add a new sense to* $P_{sem}(w)$ *for each* $w \in u'$*.*

This step adds the new senses to the words that were determined to be necessary for the utterance to be consistent.

Step 2.2 - *Go to Step 1.*

After adding the necessary new senses, the algorithm returns to **Step 1** to process the utterance.

To determine the appropriate meaning for each word in the utterance, the algorithm **Process** is used.

Procedure Process(Sense,Meaning)

Step 1 - *For each word w in the utterance, remove from* $P_{sem}(w)$ *any meanings that do not appear in the utterance meaning.*

This step removes any meaning hypothesised as possible for a word that is not part of the meaning of the utterance.

Step 2 - *For each word w in the utterance, remove from* $P_{sem}(w)$ *any meaning that appears only once in the utterance meaning and is contributed by some other word that has already converged on that meaning, also in the utterance.*

For each converged word in the utterance this step checks if its meaning is contained in the set $P_{sem}(w)$ of possible meanings of other words in the utterance, and if so, removes it from the latter.

Step 3 - *For each word w in the utterance, increment the confidence factor* $T_{sem}(w)$ *if the word sense has converged.*

This step handles noise, minimising the effect of incorrectly converged senses.

The idea is that the learning system assumes that, by default, each word has a single sense, and tries to build a lexicon that covers all the utterances in the corpus. If the corpus has polysemy, some words have more than one sense. When a new word sense occurs, the algorithm tries to map it to an existing sense, but this leads to the existing sense becoming inconsistent if the two senses do not have semantic elements in common. When this happens, the new sense is added to the lexicon $P_{sem}(w)$. Noisy utterances, not being paired with the correct meaning, are treated as introducing new word senses, resulting in spurious senses being created and added to $P_{sem}(w)$. As these senses are unlikely to be encountered again, they will not be sufficiently reinforced to become frozen, and will eventually be discarded by the garbage collection mechanism.

As an example, supposing the algorithm is part-way through the learning process, and that the following is a fragment of the P_{sem} lexicon:

John$_1$: {john(a), red(b), ball(c)}
took$_1$: {take(a,b), look(c,d)}
the$_1$: {the(a)}, C= 9
red$_1$: {red(a), ball(b)}

ball$_1$: {kim(a), ball(b)}

Then, the algorithm receives the utterance:

• (5.5) *John took the red ball*

paired with the following meaning:

{take(1, 2), john(1), the(2), red(2), ball(2)}

By executing **Step 1** of **ProcessUtterance**, the learning system finds that there is a sense assignment of the words in the utterance that is consistent with the meaning:

• John$_1$ took$_1$ the$_1$ red$_1$ ball$_1$

The algorithm then executes **Process(Sense, Meaning)**. In **Step 1**, the intersection between the set of meanings previously hypothesised and the ones newly hypothesised results in:

John$_1$ = {john(a), red(b), ball(c)} ∩ {john(a), red(b), ball(c), take(d,e), the(f)}
= {john(a), red(b), ball(c)}
since john(a) = john(a), red(b) = red(b), and ball(c) = ball(c)

took$_1$ = {take(a,b), look(c,d)} ∩ {john(a), red(b), ball(c), take(d,e), the(f)}
= {take(a,b)}
since take(a,b) = take(d,e)

the$_1$ = {the(a)} ∩ {john(a), red(b), ball(c), take(d,e), the(f)}
= {the(a)}
since the(a) = the(f)

red$_1$ = {red(a), ball(b)} ∩ {john(a), red(b), ball(c), take(d,e), the(f)}
= {red(a), ball(b)}
since red(a) = red(b) and ball(b) = ball(c)

ball$_1$ = {kim(a), ball(b)} ∩ {john(a), red(b), ball(c), take(d,e), the(f)}
= {ball(b)}
since ball(b) = ball(c)

with the resulting lexicon being:

John$_1$: {john(a), red(b), ball(c)}
took$_1$: {take(a,b)}
the$_1$: {the(a)}, C=9
red$_1$: {red(a), ball(b)}
ball$_1$: {ball(a)}

where the meanings of *took* and *ball* have converged. In **Step 2**, the algorithm removes from the set of meanings of each of the words those meanings to which other words have converged:

John$_1$ = {john(a), red(b), ball(c)} - ({take(a,b)}, {the(a)} , {ball(a)})
= {john(a), red(b)}
took$_1$ = {take(a,b)}
the$_1$ = {the(a)}
red$_1$ = {red(a), ball(b)} - ({take(a,b)}, {the(a)} , {ball(a)})
= {red(a)}
ball$_1$ = {ball(a)}

and the resulting lexicon is:

John$_1$: {john(a), red(b)}
took$_1$: {take(a,b)}
the$_1$: {the(a)}, C=9
red$_1$: {red(a)}
ball$_1$: {ball(a)}

with the meaning of *red* having converged also. After that, **Step 3** is applied and the confidence factors of the senses used for *took*, *the*, *red* and *ball* are incremented.

John$_1$: {john(a), red(b)}
took$_1$: {take(a,b)}, C=1
the$_1$: {the(a)}, C=10
red$_1$: {red(a)}, C=1
ball$_1$: {ball(a)}, C=1

In this example there was one sense assignment that was consistent with the meaning associated with the utterance. Thus, no new senses needed to be added, and the algorithm processes the utterance as normal.

In a second example the algorithm has to deal with polysemy, but all the necessary word senses are already in the lexicon. Let us assume that the current

lexicon is:

John$_1$: {john(a)}, C= 9
saw$_1$: {hammer(a), wood-cutting-tool(b)}
saw$_2$: {see(a,b), go(c,d)}
had$_1$: {possess(a,b), want(c,d)}
had$_2$: {conduct(a,b), cause(c,d)}
Mary$_1$: {mary(a)}, C= 9
arrive$_1$: {arrive(a,b)}, C= 4
at$_1$: {at(a,b)}, C= 10
the$_1$: {the(a)}, C= 10
a$_1$: {a(a)}, C= 10
ball$_1$: {dance-party(a)}, C=4
ball$_2$: {toy(a)}, C=10
party$_1$: {political-organisation(a)}, C=4

This lexicon is polysemous, containing two senses for *saw*, *had* and *ball*. Then, the algorithm receives the utterance:

• (5.6) *John saw the ball*

paired with the following meaning:

{see(1,2), john(1), the(2), dance-party(2)}

In **Step1** of **ProcessUtterance**, the algorithm needs to analyse four sense assignments, as this utterance contains two polysemous words, each with two senses in the lexicon:

- John$_1$ saw$_1$ the$_1$ ball$_1$,
- John$_1$ saw$_1$ the$_1$ ball$_2$,
- John$_1$ saw$_2$ the$_1$ ball$_1$,
- John$_1$ saw$_2$ the$_1$ ball$_2$.

However, only one is consistent, containing all the semantic predicates needed to obtain the meaning paired with the utterance:

- John$_1$ saw$_2$ the$_1$ ball$_1$

As this is the only consistent sense assignment, the algorithm proceeds executing **Process**((*John*$_1$ *saw*$_2$ *the*$_1$ *ball*$_1$) {see(a,b) john(a) the(b) dance-party(b)}),

which results in the following lexicon, where saw_2 converged:

$John_1$: {john(a)}, C= 10
saw_1: {hammer(a), wood-cutting-tool(b)}
saw_2: {see(a,b)} C= 1
had_1: {possess(a,b), want(c,d)}
had_2: {conduct(a,b), cause(c,d)}
$Mary_1$: {Mary(a)}, C= 9
$arrive_1$: {arrive(a,b)}, C= 5
at_1: {at(a,b)}, C= 10
the_1: {the(a)}, C= 10
a_1: {a(a)}, C= 10
$ball_1$: {dance-party(a)}, C=5
$ball_2$: {toy(a)}, C=10
$party_1$: {political-organisation(a)}, C=4

In this next example, the algorithm has to deal with polysemy, but for a particular word in the utterance where none of the senses in the lexicon is the desired one, and the algorithm has to add the new sense. If the algorithm receives the utterance:

• (5.7) *Mary had a party*

paired with the following meaning:

{conduct(1 , 2), mary(1), a(2), celebration(2)}

where there is a polysemous word, *had*, with two senses. **Step1** of **ProcessUtterance** finds that none of the sense assignments is consistent with the paired meaning:

- $Mary_1$ had_1 a_1 $party_1$,
- $Mary_1$ had_2 a_1 $party_1$.

In this case, **Step 2** tries to find the smallest set of words in the utterance for which a new sense has to be added to resolve the inconsistency. Adding a new sense to *party* resolves that, and by executing **Process**(($Mary_1$ had_2 a_1 $party_2$), {conduct(a,b) mary(a) a(b) celebration(b)}), $party_2$ converges to celebration(a), had_2 converges to conduct(a,b), and the lexicon is updated to:

$John_1$: {john(a)}, C= 10
saw_1: {hammer(a), wood-cutting-tool(b)}

saw$_2$: {see(a,b)} C= 1
had$_1$: {possess(a,b), want(c,d)}
had$_2$: {conduct(a,b)}, C= 1
Mary$_1$: {mary(a)}, C= 10
arrive$_1$: {arrive(a,b)}, C= 5
at$_1$: {at(a,b)}, C= 10
the$_1$: {the(a)}, C= 10
a$_1$: {a(a)}, C= 10
ball$_1$: {dance-party(a)}, C=5
ball$_2$: {toy(a)}, C=10
party$_1$: {political-organisation(a)}, C=4
party$_2$: {celebration(a)}, C=1

In this case, the algorithm added a new sense to a word already in the lexicon.

5.1.2 Evaluation of the Semantics Learner

The system implemented by Waldron [Waldron 2000] learns a lexicon that maps words to semantics, while dealing with noise and polysemy. Waldron's algorithm is used to pre-process the annotated corpus, assigning a logical form to each word in a sentence, as discussed in chapter 6. In this section, an evaluation of its performance is discussed.

Waldron's semantics learner was tested on the annotated Sachs corpus, containing 1,517 parents' sentences paired with logical forms, as described in section 4.6. This corpus was given as input to the semantics learner, which analysed each sentence and logical form pair in turn, trying to determine an appropriate mapping between words and logical forms. Thus for each word in a sentence, the learner tried to determine the appropriate logical form according to the algorithm described in the previous sections. If the learner could determine a putative semantic predicate for every word in a sentence, then these predicates were returned as result. Otherwise, if a predicate could not be determined for every word in a sentence, no result was returned. From the 1,517 sentences, the learner returned a putative word-logical form mapping for 63.6% of the corpus, with 95.23% of the results returned being correct, and the remaining 4.55% containing at least one incorrect predicate. Most of the incorrect mappings were due to the very frequent co-occurrence of some words in only a particular context, which meant that the learner could not use cross-situational inference to distinguish them. This is the case of the control verb *want* and the infinitival *to*, in sentences like *Do you want to eat?*. As they occur very often together, the learner could not reliably distinguish them and assigned *want* the logical form for *to* and vice-versa. Otherwise, the semantics learner has a good performance, assigning correct logical forms for 95.23% of the sentences that it successfully processes.

5.2 Learning the Syntactic Category of Words

By using the closely linked syntax and semantics in CGs, Waldron developed and implemented an algorithm to learn the mapping between a semantic predicate and a syntactic category. From the mapping between a word and its semantics, the algorithm tries to infer the syntactic category associated with the word. The algorithm is partly algebraic, and tries to construct derivations, given the current knowledge that it has about the syntactic categories and semantic predicates of the words in the utterance. The algorithm regards categories in their atomic formulation, ignoring feature structure information, and it assumes that the syntactic category of nouns, names, pronouns and particles are known. Another assumption made is that the algorithm knows whether an utterance is imperative, interrogative or declarative, since this information is sometimes necessary in order to construct the derivation of an utterance. To construct derivations, the algorithm uses generalised weak permutation, forward application, backward application, forward composition and backward composition, and the set of lexical rules described in chapter 4.

Initially, the algorithm starts with knowledge only about nouns, names, pronouns and particles, and the predicates associated with the words of a sentence. Then, by applying the grammatical rules and some heuristics, the algorithm tries to obtain possible derivations. Each application of a rule or heuristic is assigned a weight (or confidence factor) and each derivation is assigned a weight computed by multiplying the weights used to generate it. Sometimes alternative derivations can be generated, resulting in different applications of rules or heuristics, each derivation having an associated weight. For example, if the algorithm receives as input the sentence:

- (5.8) *I see Bill*,

paired with the semantics:

{ i($\boxed{1}$), see($\boxed{1}$, $\boxed{2}$), bill($\boxed{2}$)}

and the current lexicon is:

i_1: {i(a)}, NP
$bill_1$: {bill(a)}, NP

the algorithm tries to determine the appropriate categories for each word in the utterance that allows the learner to produce a successful derivation. It generates an initial hypothesis, using the current state of knowledge:

I	see	Bill
i(a)	see(b,c)	bill(d)
NP	?	NP

where the words in the sentence are in the first line, the corresponding semantic predicates in the second, and the syntactic categories in the third. After determining all words in the utterance for which the syntactic categories are known, the algorithm has to infer the categories for the remaining words, in this case *see*, so that the derivation ends in S:

I	see	Bill
i(a)	see(b,c)	bill(d)
NP	?	NP
	S	

Therefore, with NP ? NP the algorithm has to generate a derivation that results in S:

NP ? NP = S
NP+?+NP = S

where '+' is a variable over the slash operators. Then, using the grammatical rules defined, the algorithm generates further hypotheses. Among the possible categories that can be inferred from NP+?+NP=S, which can also be written as ?= S+NP+NP, there are:

?= S/NP/NP,
?= S\NP\NP,
?= S/NP\NP, and
?= S\NP/NP

with the latter two being able to generate a derivation resulting in S, when combined with the other categories. By using generalised weak permutation, these two are considered equivalent by the syntax learner, which does not make any use of the thematic roles in the undecomposed logical form ({i(1) see(1,2) bill(2)}), choosing only one of them:

NP S\NP/NP NP = S

As the category for *see* is inferred, the resulting lexicon is updated to:

i_1: {i(a)}, NP

see_1: {see(a,b)}, S\NP/NP
$bill_1$: {bill(a)}, NP

This is a simple case where the category for only one word in the utterance has to be inferred. However, more complex cases can arise, with categories having to be inferred for several words, and with different possibilities for each word. In these cases, the algorithm may not be able to establish uniquely the appropriate category for each word sense, keeping all the possibilities inferred, each with an appropriate confidence factor. Eventually, if the word sense appears in enough different syntactic contexts, one of the possibilities gains enough evidence to have a significantly higher confidence factor than the others, which is when the word sense converges to that category.

For example, in the case of the sentence:

- (5.9) *The big dog slept*

with logical form:

{the(1), big(1), dog(1), sleep(1)}

assuming that the categories for the words *the, big, dog* are known, only the category for *slept*, with meaning sleep(d), has to be inferred:

The	big	dog	slept
the(a)	big(b)	dog(c)	sleep(d)
NP/N	N/N	N	?
		S	

Among the possibilities for the category for sleep(d), the following ones are all consistent with NP/N N/N N ? = S:

- ?= S\(NP/N)\(N/N)\N,
- ?= S\(NP/N)\N, and
- ?= S\NP

As there is no way for the algorithm to determine which one should be used based on the information available, all of these putative categories are considered valid hypotheses. Then, they are stored with their confidence factors, and the algorithm waits for more utterances that can disambiguate them.

For each utterance, the algorithm keeps alternating between applying grammatical rules, trying to infer categories and substituting the newly inferred categories in the derivation, until nothing more can be done. At this stage, if the

algorithm succeeds in inferring categories for all the words in the utterance, all possible derivations for the utterance are generated. However, there are occasions where the algorithm is unable to generate a complete derivation. When this happens, it may be because the algorithm needs to infer the categories of several words in the utterance, and the current state of knowledge is insufficient to account for the word senses used. Even if the algorithm cannot generate a successful derivation, it may be able to combine some of the constituents together into fragments of a derivation. In this case, the algorithm keeps track of the fragments that were generated, with corresponding weights, since these can be potentially useful in the processing of subsequent utterances.

5.2.1 The Algorithm

In this section, a formalisation of the algorithm developed by Waldron [Waldron 1999] is described. This algorithm is also on-line, and after analysing a sentence it retains only information about the mappings from senses to syntactic categories. These mappings are also stored in three tables:

- $L_{syn}(ws)$ - the syntactic lexicon, which maps each word-sense pair, ws, to an appropriate syntactic category c.
- $P_{syn}(ws)$ - for each possible word-sense pair, ws, this table stores a set of possible categories c. Each word-sense pair is initially mapped to any allowed category, and the algorithm monotonically removes elements of $P_{syn}(ws)$ until it is a singleton, which is when the category c converged, and can be placed in $L_{syn}(ws)$.
- $T_{syn}(c)$ - maps each putative category c, associated to a word-sense pair, to a confidence factor, C_{syn}, that indicates the evidence for the category. A putative category is initially mapped to zero, but as evidence for this hypothesis is provided its confidence factor increases. A constant μ_{syn} indicates the freezing point, and a putative category c becomes *frozen* when $T_{syn}(c) \geq \mu_{syn}$.

Each possible category starts unconverged, and as the evidence for it increases, so does its confidence factor. When the confidence factor of a hypothesis c reaches the freezing point, μ_{syn}, the hypothesis is selected as the appropriate category for the word-sense pair ws. Putative categories that are not frozen and that have a low confidence factor are subject to a garbage collection mechanism.

The algorithm used to learn syntactic categories is **ProcessSenseAssignment**. It receives as input an utterance and associated meaning, and the word-sense mapping for each word in the utterance. During the processing of an utterance, the algorithm updates each of the three tables as necessary.

Procedure ProcessSenseAssignment(Utterance, Meaning, Mapping)

Step 1 - *Find if there is at least one possible syntactic category assignment* $\{c_1, ..., c_n\}$ *for the utterance, which produces a successful derivation.*

This step verifies if there is at least one putative category assignment that results in a successful derivation.

Step 1.1 - *For each putative category c which contributes to a successful derivation of the utterance, increment the confidence factor* $T_{syn}(c)$.

Step 1.1 reinforces those putative categories that can be used in a successful derivation.

Step 2 - *Otherwise, create new putative categories for a subset ws'* $\subseteq$ *Word-sense pairs, such that when the new categories are added to* $P_{syn}(c)$ *for each* $c \in$ *ws', one or more successful derivations for the sentence are produced.*

This step creates new putative categories to try to find successful derivations for the sentence.

Step 2.1 - *Add the newly created putative categories to* $P_{syn}(c)$ *for each* $c \in$ *ws'.*

This step adds to the lexicon P_{syn} the new putative categories to the word-sense pairs that were determined to be necessary for producing successful derivations.

Step 2.2 - *Go to Step 1.*

Step 3 - *If no complete derivations were obtained, ignore this utterance.*

If no derivations could be produced, the algorithm stops processing the current utterance.

In the following example, the algorithm is part-way through the learning process, and has the following hypotheses:

bill$_1$: NP, $C_{syn} = 100$
mary$_1$: NP, $C_{syn} = 100$
john$_1$: NP, $C_{syn} = 100$
block$_1$: NP, $C_{syn} = 100$
block$_2$: S\NP/NP, $C_{syn} = 3$
likes$_1$: S\NP/NP, $C_{syn} = 10$
sees$_1$: S\NP/NP, $C_{syn} = 7$

when it receives the utterance:

• (5.10) *Bill likes Mary*,

paired with the semantics:

{like($\boxed{1}$, $\boxed{2}$), bill($\boxed{1}$), mary($\boxed{2}$)},

where *Bill* is paired with sense bill_1, which corresponds to the predicate bill(a); *Mary* is paired with sense mary_1, which corresponds to the predicate mary(a), and *likes* is paired with sense likes_1, which corresponds to the predicate like(a,b). In **Step 1**, the algorithm tries to determine a possible category assignment that allows for a successful derivation of the utterance. It generates an initial hypothesis, using the current state of knowledge:

Bill	likes	Mary
bill(a)	likes(b,c)	mary(d)
bill_1	likes_1	mary_1
NP	S\NP/NP	NP
NP	S\NP	
	S	

As this category assignment produces a derivation, the confidence factors of each of the categories used is updated, following **Step 1.1**:

bill_1: NP, $C_{syn} = 100$
mary_1: NP, $C_{syn} = 100$
john_1: NP, $C_{syn} = 100$
block_1: NP, $C_{syn} = 100$
block_2: S\NP/NP, $C_{syn} = 3$
likes_1: S\NP/NP, $C_{syn} = 11$
sees_1: S\NP/NP, $C_{syn} = 7$

In this case only the confidence factor of *likes* was updated, because the categories of both *Bill* and *Mary*, being names, are known by the learner, and assigned a high confidence factor, which makes them frozen.

The algorithm in this first example has all the necessary categories to provide a successful derivation that is consistent with the word-senses used. Let us consider now that the next sentence the algorithm receives is:

• (5.11) *Mary runs*

paired with meaning:

{run($\boxed{1}$), mary($\boxed{1}$)},

where *Mary* is paired with sense mary_1, which corresponds to the semantic predicate mary(a), and *runs* is paired with sense run_1, which corresponds to the predicate run(a). In **Step 1** the algorithm determines that there are no possible category assignments that generate a derivation given the current state of knowledge, and it proceeds to **Step 2**. The algorithm then determines that it can generate the following derivation fragment:

Mary	runs
mary(a)	run(b)
NP	?

where the category for run_1 has to be found. The algorithm then determines that a successful derivation can be obtained if S\NP is added as a possible category for run_1:

Mary	runs
mary(a)	run(b)
NP	?
NP	S\NP
	S

Step 2.1 adds this hypothesis to the lexicon:

bill_1: NP, $C_{syn} = 100$
mary_1: NP, $C_{syn} = 100$
john_1: NP, $C_{syn} = 100$
block_1: NP, $C_{syn} = 100$
block_2: S\NP/NP, $C_{syn} = 3$
likes_1: S\NP/NP, $C_{syn} = 11$
sees_1: S\NP/NP, $C_{syn} = 7$
run_1: S\NP, $C_{syn} = 0$

Step 2.2 returns the execution to **Step 1**, which now can find a successful derivation, and as a result in **Step 1.1** the confidence factors are updated:

bill_1: NP, $C_{syn} = 100$
mary_1: NP, $C_{syn} = 100$
john_1: NP, $C_{syn} = 100$
block_1: NP, $C_{syn} = 100$
block_2: S\NP/NP, $C_{syn} = 3$
likes_1: S\NP/NP, $C_{syn} = 11$

sees_1: S\NP/NP, $C_{syn} = 7$
run_1: S\NP, $C_{syn} = 1$

In this example the algorithm cannot find a successful derivation for the sentence given the current state of knowledge, and has to add a new entry to the lexicon containing a putative category for the sense run_1.

5.2.2 Evaluation of the Syntax Learner

The syntax learner was evaluated using the annotated Sachs corpus. The corpus contains 1,517 sentences which were annotated using a grammar where 89 distinct lexical categories capture the constructions found in the corpus. The words in the lexicon exemplify these lexical categories defined. Each utterance is first analysed by the semantics learner, and if the semantics of each word in the utterance can be determined, then the syntax learner is activated. Thus every sentence in the corpus, for which an appropriate mapping between words and logical forms could be determined, was given as input to the syntax learner, as well as a lexicon containing for each word sense the associated logical form, and when available, its putative syntactic categories. Initially, the lexicon contains only the syntactic category for nouns, names, pronouns and particles, but, as the syntactic categories of words corresponding to other semantic predicates are determined, they are also added to the lexical entries. As the semantics learner returned a result for 965 sentences out of the 1,517, the syntax learner is evaluated in terms of these. For each sentence given as input, the syntax learner attempted to return a mapping between the word-logical form pairs in a sentence and putative syntactic categories in a way that produced a complete derivation. If the syntax learner could find a putative category for each word-logical form pair in a sentence that produced a derivation, it returned the derivation as result. For many sentences there was more than one possible category assignment for each word and logical form pair that produced a derivation, and when this was the case, the learner returned all possible category assignments. On the other hand, if there was at least one word-logical form pair in the sentence for which the learner could not find a category, then no result was returned. The space of possible categories contains all the atomic categories **S**, **NP**, **N**, **PP** and **PRT**, and all the complex categories that can be generated from them.

When the syntax learner was tested in this unrestricted search space, in a typical run, from the 965 input sentences, putative category assignments were returned for 52.6% of the sentences. However, from these 508 sentences that received putative category assignments, only 4.7% had the correct category assignment for every word in the sentence. For the remaining 95.3%, the putative category assignments returned contained at least one incorrect mapping between word-logical form and syntactic category. The incorrectly hypothesised categories usually contained a large number of arguments, and were too complex, with sev-

eral levels of embeddings. For example one of the categories hypothesised for *red* in *red ball* is (((((S\NP)\(NP/N))\((S/NP)/NP))/N)/NP), instead of N/N. In this unrestricted search space, the syntax learner does not perform well. The putative syntactic categories are too complex and long, and do not correspond to linguistic intuitions. For example, one of the possible categories hypothesised for an auxiliary verb like *are* in *are they eating*, is the category NP/NP, instead of the more conventional (S\NP)/(S\NP). This overgeneration is a significant problem if the output of the syntax learner is to be used as a basis for setting the parameters of the UG. However, as is discussed in chapter 6, the principles and parameters learner analyses this output and only derivations containing categories permitted by the UG are considered for setting the parameters, with all the others being discarded.

5.3 Summary

In this chapter, two systems were discussed and evaluated. The first, the semantics learner, attempts to learn the mapping between words and meanings and to build a lexicon containing the meaning associated with each word sense. The second system, the syntax learner, tries to learn possible syntactic categories for a word sense using an algebraic approach. In this work they are used to pre-process the annotated Sachs corpus, but there is a significant problem of overgeneration, and they are only able to assign correct semantic and syntactic categories for a small portion of the corpus. Therefore, some mechanisms discussed in chapters 6 and 7 have to be used to deal with this problem. Moreover, the output produced by these systems does not contain any information about the linking between the semantic arguments and syntactic complements of the categories, and this is something that that needs to be learned, as discussed in section 7.1.

Chapter 6

The Learning System

The aim of this work is to investigate the process of grammatical acquisition from data. For this purpose, a computational learning system was implemented, which is composed of a Universal Grammar with its principles and parameters, and a learning module. This system receives input from a corpus annotated with logical forms, and this data is used to fix the values of the parameters of the UG to the particular language expressed in the corpus, following the Principles and Parameters Theory. The UG, as described in section 4.5, is implemented as a UB-GCG embedded in a default inheritance network of lexical types. The learner is exposed to an environment provided by the annotated Sachs corpus (section 4.6) that simulates the environment to which a child is exposed. In this environment the learner may have to deal with noise and ambiguity. The learner needs to be able to overcome these problems if it is to converge to the language of the environment. To guide the learner in this convergence, a Bayesian interpretation of the learning process is used. At the end of the acquisition process, the performance of the learner is evaluated by comparing the parameter settings of the UG with those of the target grammar, for the learning tasks performed. The grammar defined in chapter 4 is considered to be the target to which the learner has to converge, with respect to the learning aspects being investigated.[1] This framework can serve as a basis for the investigation of several aspects of language acquisition.

In this chapter the learning system is described, starting with an explanation of the architecture of the learning system (section 6.1), followed by a description of the learning approach employed (section 6.2), and its implementation (section 6.2.1).

[1] In this work we concentrate on the learning of subcategorisation frames and word order information. This builds the basis for a second stage of learning, such as the acquisition of lexical rules, which is when the learner has enough information for the generalisations occurring in the lexicon to become apparent, and this second learning phase is not addressed in this work. The learning of lexical rules is investigated by Schütze [1995], but he uses data intensive methods that are not compatible with this work.

6.1 Architecture of the Learning System

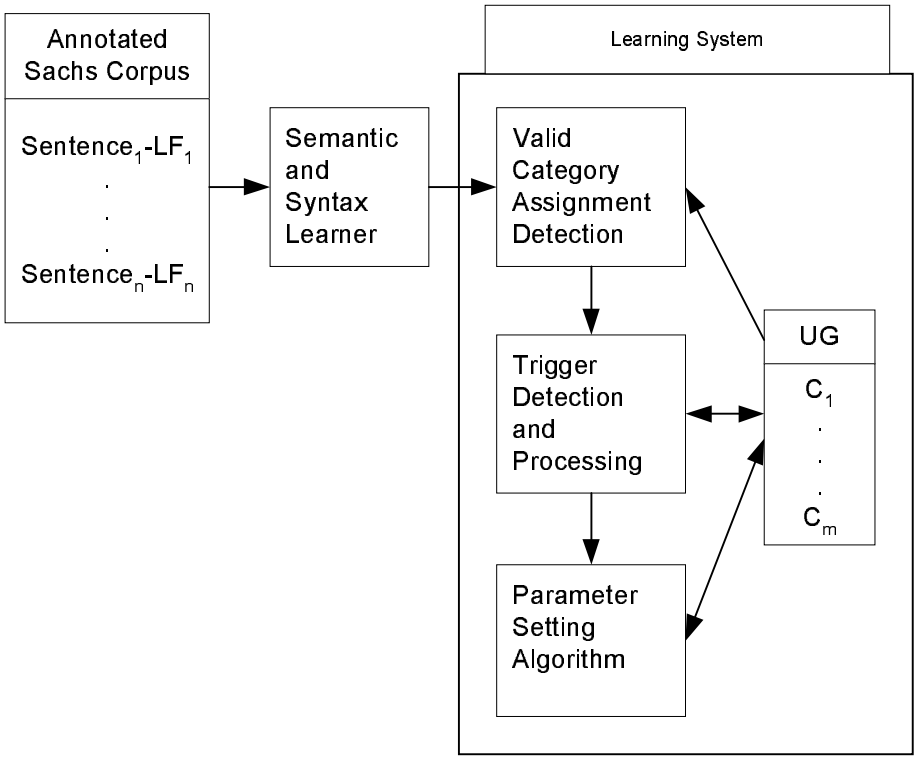

Figure 6.1: Architecture of the Learning System

The learning system is equipped with a UG with its principles and parameters, and a learning module that, given the input data, sets the parameters of the UG to the language exemplified in this input. This process is shown in figure 6.1.

The input data consists of sentences annotated with logical forms that are pre-processed by the semantics and the syntax learners, described in chapter 5. They are combined together to try to determine semantic and syntactic category assignments for each of the words in a sentence. For example, given the sentence:

- (6.1) *Do you see a cup*

paired with the logical form:

$$\left[\begin{array}{l}\textbf{sem}\\ \text{RESTR} : \left\langle !\left[\begin{array}{l}\text{RELN} : \textbf{do}\\ \text{SIT} : \boxed{1}\ \textbf{index}\\ \text{ARG1} : \boxed{2}\ \textbf{index}\end{array}\right], \left[\begin{array}{l}\text{RELN} : \textbf{see}\\ \text{SIT} : \boxed{2}\\ \text{ARG1} : \boxed{3}\ \textbf{index}\\ \text{ARG2} : \boxed{4}\ \textbf{index}\end{array}\right],\right.\\ \qquad\left.\left[\begin{array}{l}\text{SIT} : \boxed{5}\ \textbf{index}\\ \text{NAME} : \textbf{you}\\ \text{NAMED} : \boxed{3}\end{array}\right]\left[\begin{array}{l}\text{SIT} : \boxed{6}\ \textbf{index}\\ \text{RELN} : \textbf{a}\\ \text{BV} : \boxed{4}\end{array}\right]\left[\begin{array}{l}\text{SIT} : \boxed{7}\ \textbf{index}\\ \text{NAME} : \textbf{cup}\\ \text{NAMED} : \boxed{4}\end{array}\right]!\right\rangle\end{array}\right]$$

whose corresponding linear notation is:

{do(1 , 2), see(2 , 3 , 4), you(5 , 3), a(6 , 4) , cup(7 , 4)}

the semantics learner determines semantic assignments for each word in the sentence:

{do_1=do(a,b), you_1=you(c,d), see_1=see(e,f,g), a_1=a(h,i), cup_1=cup(j,k)}

The syntax learner determines, among others, the following syntactic assignments for each of the words:

{(do_1=(NP/NP), you_1=NP, see_1=S\NP/NP, a_1=NP/N, cup_1=N),

(do_1=(S\NP)/(S\NP), you_1=NP, see_1=S\NP/NP, a_1=NP/N, cup_1=N)}

If the semantics and syntax learners are successful in determining possible semantic and syntactic category assignments for the sentence, these assignments are evaluated by the **valid category assignment detection module**. In order to check if there is at least one valid category assignment (VCA) for the sentence this module uses two principles [from Steedman 2000]:

- the Principle of Lexical Head Government assumes that the responsibility for specifying both bounded and unbounded dependencies lies in the lexical specification of the syntactic type of the head of these dependencies, and
- the Principle of Categorial Type Transparency states that, for any language, the semantic interpretation together with a number of language-specific directional parameter settings uniquely determines the syntactic category of a word.

The idea is that, in CGs, lexical entries for words are responsible for specifying (bounded and unbounded) dependencies, containing the mapping between words and their semantic interpretations. From the semantic interpretation of

a word and some directional information for a language, it is possible to determine the syntactic form of the corresponding category. For instance, in English an intransitive verb, such as *run*, is a verbal one-place predicate from an entity (the subject) to a proposition, where the subject occurs to the left of the verb, and the verb is syntactically realised as S\NP. Thus, in this work, we assume that these principles, that we refer to as Categorial Principles, determine how the semantic predicate of a word is syntactically realised. These principles are closely related to the Projection Principle [Chomsky 1981] that states that the selectional requirements of a word are projected onto every level of syntactic representation, and to the Extended Projection Principle [Chomsky 1982] that also requires that all sentences have a subject. The task executed by the VCA detection module is necessary because the syntax learner returns every possible assignment for the words in a sentence that results in complete derivations, given the space of possible categories. Thus, this module searches for VCAs, where a category assignment is considered to be valid if it not only provides a complete derivation for the sentence, but also has each word in the sentence paired with a syntactic category that is appropriate for the associated semantic predicate, according to the Categorial Principles. All other assignments are discarded. For example, in the syntactic category assignments for sentence 6.1, because *do* is a verb, with verbal predication, it is not compatible with category NP/NP which has nominal predication. Thus, only the second assignment is a VCA and can subsequently be used to set the parameters of the UG.

If there is a consistent category assignment for each word in the sentence, the **trigger detection and processing module** checks which triggers the sentence provides. As discussed in chapter 2, the notion of triggers varies in the literature. In the scope of this work, a triggering sentence is a sentence paired with a logical form, with corresponding valid semantic and syntactic assignments, which provide evidence for one or more parameters. Moreover, a particular trigger does not necessarily provide evidence for a change in the current settings of a parameter, but can also give favourable evidence for keeping this setting. A sentence can provide triggers for several parameters, depending on the constructions it contains. For instance, sentence 6.1 contains a transitive verb (*see*) that provides, among other triggers, those for the direction of the subject and of the object in relation to the verb. This module also extracts frequency information about the words in the sentence and the associated semantic and syntactic predicates. The learner may need to know the frequency with which a word is associated with a particular semantic predicate and a given syntactic category for some of the learning tasks.

Triggers that are detected are used by the **parameter setting module** to set the parameters of the UG.

This same process is repeated for each sentence in the corpus, with the learning system using information about the words, the semantic predicates, the syntactic categories and other triggers expressed in the sentence, for the learning tasks. All

these activities are performed by the algorithm **Process-Input(Corpus)**.

Procedure Process-Input(Corpus)

Step 1 - *Read Sentence S from corpus with logical form LF*
Given an input corpus, this step reads a sentence-logical form pair.

Step 2 - *Find a semantic assignment Sem for each word in sentence S, based on the logical form LF, by executing* ***ProcessUtterance****(S,LF).*
In this step the semantics learner receives a sentence and logical form pair as input and tries to determine a semantic assignment for each word in the sentence.
Step 2.1 - *If Sem = {} Go to Step 1.*
If the semantics learner could not find an assignment, the sentence is ignored, and the learner returns to **Step 1** to process the next sentence.

Step 3 - *Find syntactic assignments Syn for each word in sentence S, given its semantic assignment in Sem, by executing* ***ProcessSenseAssignment****(S,LF,Sem).*
If the semantics learner returned assignments for the words in the sentence, these are given as input to the syntax learner, together with the sentence and logical form. The syntax learner tries to determine all the syntactic assignments for the words in the sentence that result in derivations that span through the whole sentence.
Step 3.1 - *If Syn = {} Go to Step 1.*
If the syntax learner could not find possible assignments, the sentence is ignored, and the learner returns to **Step 1** to process the next sentence.

Step 4 - *For each semantic predicate in Sem, from Syn determine* Syn_{valid}*, the syntactic assignment that is valid according to the Categorial Principles.*
The VCA detection module, in this step, establishes if there is a valid syntactic category assignment for every word in the sentence, given all the assignments returned, by evaluating each syntactic assignment according to the Categorial Principles. Those assignments that are not considered valid are discarded.
Step 4.1 - *If* $Syn_{valid} = \{\}$ *Go to Step 1.*
If there is no valid category assignment the sentence is ignored, and the learner returns to **Step 1** to process the next sentence.

Step 5 - *Determine set of triggers T expressed in sentence S, logical form LF, semantic assignment Sem and syntactic assignment* Syn_{valid}*.*
This step determines the triggers expressed in the sentence. Following Dresher and Kaye [1990], we assume that the learner knows how to detect triggers in a triggering sentence, and that it also knows which parameter is being expressed

in a given trigger. In this way, the trigger detection module analyses a sentence-logical form pair, searching for triggers that can set the values of the parameters.

Step 5.1 - *Store frequency information about each word w, its semantic assignment Sem_w, and its syntactic assignment Syn_w.*

The learner stores frequency information about the words in the sentence and their occurrence with particular semantic and syntactic assignments.

Step 6 - *Use the triggers in T to set the parameters of the UG, by executing* ***Process-Triggers****(S,T).*

The triggers found are processed and used to set the parameters of the UG.

Step 7 - *Go to* ***Step 1****.*

The learner goes back to **Step 1** to process the next sentence.

As an example, consider that the learner is given the sentence:

• (6.2) *Bill likes Beth*

paired with the logical form

{like(1 , 2 , 3), bill(4 , 2), beth(5 3)}.

After **Step 1** reads this sentence and logical form pair, these are given as input to the semantics learner in **Step 2**, where the following semantic assignments for each word in the sentence are returned:

Sem = {like$_1$=like(a,b,c), bill$_1$=bill(d,e), beth$_1$=beth(f,g)}

In **Step 3**, the syntax learner receives these as input and determines that the following is one assignment that provides a possible derivation for the sentence:

Syn = {(like$_1$=(S/NP)\NP, bill$_1$=NP, beth$_1$=NP)}

Step 4 determines that this syntactic assignment in *Syn* is valid for the semantic predicates in *Sem*, according to the Categorial Principles.[2] The algorithm

[2]In this simplified example, **Step 4** is straightforwardly applied. If, however, the sentence was *Mary likes the red ball*, with logical form and putative assignments shown below, the learner would have to analyse each possible assignment, determine that only the first assignment is valid, and discard all the others:
Sem = {like(1 , 2), mary(1), the(2), red(2), ball(2)}
Syn = {(like$_1$=(S/NP)\NP, mary$_1$=NP, the$_1$=NP/N, red$_1$=N/N, ball$_1$=N),
...
(like$_1$=NP\NP, mary$_1$=NP, the$_1$=((S\NP)/((((S\NP)\(NP/N))\((S/NP)/NP))/N)),

then determines, in **Step 5**, the triggers expressed in the sentence: T = {NP, NP, S/NP\NP}. **Step 5.1** stores frequency information about each word and corresponding category assignments, in this case: $like_1$ = {like(a,b), S/NP\NP }, $bill_1$ = {bill(c), NP} and $beth_1$ = {beth(d), NP}. **Step 6** gives the triggers as input to **Process-Triggers** that sets the parameters of the UG, and **Step 7** goes back to **Step 1** to process the next sentence.

This is the algorithm that controls the learning process. It calls the procedures **ProcessUtterance** and **ProcessSenseAssignment**, already described in sections 5.1.1 and 5.2.1 respectively, and **Process-Triggers**, that is explained in section 6.2.1.

At the end of the learning process when the learner has received all the sentences in the corpus, the performance of the learner is evaluated. This evaluation is related to the state of the parameter settings of the UG in relation to the target grammar, as discussed in chapter 7 .

6.2 Bayesian Incremental Parameter Setting

The acquisition process uses a Bayesian interpretation of the learning problem, which guides the learner in the parameter setting task. The Bayesian Incremental Parameter Setting (BIPS) algorithm, defined by Briscoe [1999], determines how the learner is to proceed when faced with triggers for the parameters, given the current parameter settings. Some of the triggers will reinforce the current settings, while others will negate them, and the learner may have to change some of the current parameter values. As the parameters are embedded in a default inheritance network, the learning algorithm changes the parameter settings in constrained ways according to the positions of the parameters in the hierarchy, from subtypes to supertypes. As many of the changes to the parameters will result in more than one possible grammar reflecting the current parameter settings, the learner has to choose one among these grammars. For example, the fragment of hierarchies shown in figures 6.2 to 6.4 are all compatible with **subjdir** set to **backward**, and **vargdir**, **detdir** and **ndir** set to **forward**. In these figures '/' corresponds to **forward** and '\' to **backward**. Subtypes that do not adopt the value inherited by default from the supertype, break the inheritance chain, and specify their own values, as is the case of **vargdir** in figure 6.2. The problem for the learner is to decide which possible option is more appropriate, given the goal of converging to the target. One criterion (bias) to help the learner in this decision is to prefer compact grammars whenever possible, using a Minimum Description Length (MDL) style bias [Rissanen 1989].

The MDL Principle states that the best hypothesis to infer from the data is the one which does not make too many errors in relation to the data seen, and at the same time can be described concisely. In terms of grammars, this can be thought

red_1=(((((S\NP)\(NP/N))\((S/NP)/NP))/N)/NP), $ball_1$=NP)}.

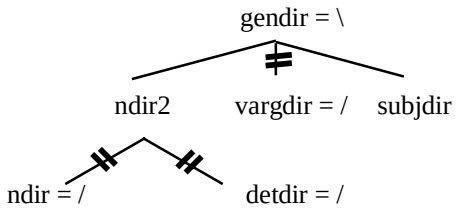

Figure 6.2: Word Order Parameters - Hierarchy 1

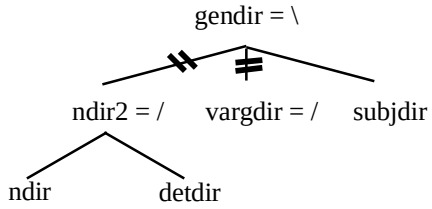

Figure 6.3: Word Order Parameters - Hierarchy 2

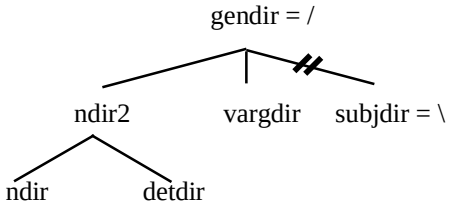

Figure 6.4: Word Order Parameters - Hierarchy 3

of as selecting the grammar that not only covers the input triggers, but is also concisely described. In this way, the most probable grammar is that consistent with the parameter settings and where the default inheritance hierarchy has the maximum number of parameters that inherit their value by default, and the minimum number of parameters that override this default and need to explicitly specify a value.

Thus, assuming that what the learner is looking for is a grammar that captures well the input sentences and that is compact, the learner uses a Bayesian (MDL-style) interpretation of the learning process to evaluate the changes being made to the grammar during the setting of the parameters. In the formalisation of the BIPS algorithm [from Briscoe 1999] the most probable grammar **g** in the grammar space **G**, given the set of triggers **T** seen so far is:

$$p(g \in G \mid T) = \frac{p(g)p(T \mid g)}{p(T)} \tag{6.1}$$

where:

- the *prior probability of a grammar*, $p(g)$, in this interpretation represents the compactness of a particular grammar, in terms of the number of attribute-value specifications that need to be explicitly defined,
- the *likelihood probability*, $p(T \mid g)$, expresses how well the grammar represents the triggering input,
- the *probability of the data*, $p(T)$, is a constant factor among all the hypotheses, and can be ignored, and
- the *posterior probability*, $p(g \mid T)$, represents the combination of the prior and likelihood probabilities.

In effect, the most probable grammar is computed as:

$$g = argmax_{g \in G} p(g)p(T \mid g) \tag{6.2}$$

The prior probability of a grammar is based on its compactness and this measure is used to order the space of possible grammars: the more compact the grammar, the higher its prior is. The compactness of a grammar is measured in terms of the amount of information that needs to be explicitly defined. Each of the grammars is defined in terms of a hierarchy (denoted by $CON(Type, \subseteq)$) where its categories are types. Each type is encoded as a set of attribute-value specifications (AVSs), and each of the AVSs has a probability associated:

- a non-default (absolute) specification, corresponding to a principle, has probability 1 ($p(AVS_i = abs) = 1$),

- a categorial parameter has probability 1,
- a word order parameter can be set to either **forward** (AVS_i=forward) or **backward** (AVS_i=backward), or it can be left unset (AVS_i=?), and each of these settings has a probability associated, such that the sum of the probabilities of these values must be 1:

$$p(AVS_i = backward) + p(AVS_i = forward) + p(AVS_i =?) = 1,$$

A parameter that has not yet been set by triggers (an unset parameter) has a probability of 0.5 ($p(AVS_i =?) = 0.5$) for the unset value and a probability of 0.25 for each of the two other values.

After a parameter is set, the unset value has probability 0 and then the probability is distributed as appropriate between the other two values:

$$p(AVS_i = backward) + p(AVS_i = forward) = 1,$$

The prior probability of a category c is defined as the product of the probabilities of the AVSs that are explicitly defined in the category, normalised with respect to the entire category set in UG:

$$p(c) = \frac{\prod_{AVS \in CON(c,\subseteq)} p(AVS)}{\sum_{c \in CON(Type,\subseteq)} \prod_{AVS \in CON(c,\subseteq)} p(AVS)} \tag{6.3}$$

where the probability of each AVS is assumed to be independent. In relation to a particular grammar, the prior probability of a category is defined by restricting the normalisation to the grammar:

$$p(c \mid g) = \frac{\prod_{AVS \in CON(c,\subseteq)} p(AVS)}{\sum_{c \in g} \prod_{AVS \in CON(c,\subseteq)} p(AVS)} \tag{6.4}$$

The prior probability of the grammar is the product of the probabilities of all AVSs explicitly defined in the grammar:

$$p(g \in G) = \prod_{AVS \in CON(Type,\subseteq)} p(AVS) \tag{6.5}$$

where $CON(Type, \subseteq)$ defines the inheritance network, denoting a minimal set of feature structures representing the category set for a given grammar. The grammar space contains the grammars allowed by the UG that have the principles in common and vary in relation to the setting of their parameters. These grammars

are in effect differentiated in terms of the product of the probabilities of the default and absolute parameter specifications defined. Following these constraints the prior probabilities of the grammars are assigned to ensure that:

$$p(G) = \sum_{g \in G} p(g) = 1 \tag{6.6}$$

This gives a higher prior to those grammars where the number of explicitly defined AVSs is lower, making a more efficient use of the inheritance hierarchy to define and propagate information through the grammar.

The likelihood probability is obtained by computing the amount of triggering data, **T**, generated by the target grammar that a particular grammar is able to model. It is defined as the product of the probabilities of each input trigger **t**:

$$p(T \mid g) = \prod_{t \in T} p(t \mid g) \tag{6.7}$$

The probability of a triggering sentence is defined as the product of the probabilities of the categories assigned to it, its valid category assignment (VCA):

$$p(t \mid g) = \prod_{c \in VCA(t)} p(c \mid g) \tag{6.8}$$

This gives a preference for grammars that can account with high probability for the triggering data. In these grammars the parameter settings are at least partially compatible with the target, given the triggers seen so far.

Using this interpretation of the learning problem, the most probable grammar is the one consistent with the parameter settings and where the default inheritance hierarchy is the most concise. This grammar has the maximum number of parameters that inherit their value by default, and the minimum number of parameters that override this default.

6.2.1 The Implemented Learning Algorithm

In this section we describe an implementation of the BIPS algorithm as an on-line incremental algorithm which uses the triggers detected in the input data to set and update the parameter values [Villavicencio 2000a, 2000c and 2000d].

The learning system receives as input a sentence annotated with the appropriate logical form, a semantic assignment, and a set of putative syntactic assignments. The learning system analyses the category assignments and gets those that are valid according to the UG. These are processed by the trigger detection module which determines if there are triggers for any of the parameters. The parameter setting module uses these triggers to reinforce the appropriate parameter values, as formalised by the procedure **Process-Triggers**.

There are two classes of parameters: categorial parameters and word order parameters, and they are treated slightly differently by the learning algorithm. Categorial parameters, as explained in section 4.5, determine which categories are allowed as part of the grammar. These parameters are encoded in each of the categories and are binary-valued. If they are set as **true**, the corresponding categories are part of the UG, but if they are set as **false** the categories are not part of the grammar. The grammar space contains all the possible grammars that contain the five basic categories S, NP, N, PP and PRT, and we assume that the grammars have complex categories containing at most 5 arguments, since this seems to be the maximum arity observed in a language such as English. We further assume that the definition of the rules and categories are predefined in the UG, but that their corresponding categorial parameters must be set as **true** for them to become available. Initially the UG is only partially specified, with only the parameters of some of the categories set as **true**, which means that the learner has to use a restricted set of categories to process the sentences. However, as the learner is exposed to more complex categories, the parameters for these categories are also set as **true**. If a categorial parameter associated with a given category type is **true**, when a trigger containing that category is processed the triggers provided by the sentence can be used to set the word order parameters. On the other hand, if the parameter is **false** when a sentence containing that category type is processed, and the supertype of the category is **true**, then the appropriate categorial parameter is set to **true**, and the triggers provided by the category are used as evidence for other parameters. A third case is when the subtype is **false** and its supertype is also **false**, which results in the triggers provided for the sentence being ignored, since the categories required to parse the sentence are not yet part of the learner's grammar. The algorithm used to set these parameters is **Process-Triggers**. It receives as input a sentence and the triggers that are expressed in this sentence, that were detected by the trigger detection module, as described in **Step 5** of the **Process-Input** procedure.

Procedure Process-Triggers(S,T)

Step 1 - *For all triggers t_i in the set of triggers T, expressed in sentence S, determine the subset of parameters Pr_{act} from the set of parameters Pr that is activated by the triggers*

This step finds all the parameters that are activated by the triggers that are present in the current sentence.

Step 2 - *Determine the subset of parameters Pr_{cat} from Pr_{act} that are Categorial Parameters.*

This step determines all the categorial parameters for which there are triggers in the sentence, where the categorial parameters determine which categories are currently part of the grammar, as discussed in section 4.5.

Step 2.1 - *If all parameters in Pr_{cat} have* VALUE=***true****, then determine the subset of parameters Pr_{wo} from the set of parameters Pr_{act} that are word order parameters and the subset T_{wo}, with the triggers from T that are triggers for word order parameters, and execute* ***Reinforce-Parameters****(T_{wo},Pr_{wo}).*

If all the categorial parameters in Pr_{cat} have their corresponding categories as part of the grammar, i.e., are set as **true**, then the learner has all the categories that are required to process the sentence as part of its current grammar. In this way, it can continue processing the remaining triggers.

Step 3 - *Otherwise, determine the subset of parameters Pr_{false} from Pr_{cat} that have* VALUE=***false****.*

If there are categorial parameters for which there are triggers in the sentence, but whose categories are not yet part of the grammar, determine these parameters.

Step 3.1 - *Determine the subset of parameters Pr_s from the set of parameters Pr, which are optimal supertypes for the parameters in Pr_{false}, such that when the corresponding categories are added to the grammar, the resulting grammar is the most concise.*

For these categorial parameters that are not part of the grammar, the algorithm determines suitable supertypes such that the addition of the corresponding categories to the grammar requires the minimum number of AVSs to be explicitly encoded in the grammar.

Step 3.2 - *If all the parameters in Pr_s have* VALUE=***true****, then set the categorial parameters in Pr_{false} to* VALUE=***true****.*

This step determines if all the supertypes of the categorial parameters have the corresponding category as part of the grammar, and then activates the categorial parameters in Pr_{false}, making their corresponding categories also part of the grammar.

Step 3.3 - *Determine the subset of parameters Pr_{wo} from the set of parameters Pr_{act} that are not categorial parameters, and the subset T_{wo}, with the triggers from T that are not triggers for categorial parameters, and execute* ***Reinforce-Parameters****(T_{wo},Pr_{wo}).*

Then the remaining triggers are processed for those parameters for which there are triggers in the sentence, but that are not categorial parameters.

Step 4 - *Otherwise ignore triggers in T for parameters in Pr_{act}.*

If there is any supertype parameter from Pr_s that is set as **false**, then the learning algorithm ignores the sentence.

As an example, continuing the execution in the previous example, in **Step 1**, the learner identifies that NP is a trigger for **np-par=true**, S/NP\NP for **trans-par=true**, **subjdir=backward** and **vargdir=forward**, and NP for **np-par=true**, resulting in the set {**np-par=true**, **trans-par=true**, **sub-**

jdir=backward, **vargdir=forward**}. In **Step 2** the learner determines that the set of categorial parameters Pr_{cat} is {**np-par**, **trans-par**}, and in **Step 2.1** it verifies if these categorial parameters are set as **true**. This is the case for **np-par**, the categorial parameter associated with the basic category NP. However, **trans-par**, the parameter associated with the transitive verb category, is set as **false** and is not yet part of the grammar, so the learner has to execute **Step 3**, where it determines that the set of parameters Pr_{false} that are set as **false** is {**trans-par**}. Thus, transitive verbs have to be added to the grammar. In the case of the transitive verb category, there are several different possible insertion places, as shown in figures 6.5, 6.6 and 6.7, in some fragments of possible hierarchies. In these hierarchies the categories are shown in an atomic formulation for reasons of clarity, but they are defined in terms of feature structures. In these figures the specifications that need to be explicitly encoded are represented in bold font, and those specifications that are inherited are in normal font. As can be seen, some of these hierarchies are more compact then others, making more effective use of the inheritance mechanism. For instance, the one shown in figure 6.5 is the most compact while that in figure 6.7 is flatter, and much less concise. One assumption made in order to find the most concise encoding for the grammar is that the learner uses Light's [1993] algorithm. This algorithm determines the optimal insertion place in the hierarchy for a new type and its required attributes. The optimal place is computed as the one where the minimum number of parent types can provide the maximum number of required attributes by inheritance, while blocking the minimum number of attributes from being inherited, because of conflicting values. In **Step 3.1** the learner determines that an optimal supertype for **trans-par** is **intrans-par**, which is the parameter associated with the intransitive verb category, with Pr_s={**intrans-par**}. In this way, the most concise grammar results from the addition of **trans-par** as a subtype of **intrans-par** and of the corresponding category, the transitive verb category, as a subtype of the intransitive verb category, which corresponds to the hierarchy shown in figure 6.5. Due to the use of default inheritance hierarchy, all the information about the transitive verb category (S|NP|NP), shown in figure 6.8, that is already contained in the intransitive verb category (S|NP), in figure 6.9, is inherited by the former. This encoding requires the minimum number of AVSs to be explicitly added to the grammar, as shown in figure 6.10. The learner then determines in **Step 3.2** if **intrans-par** is set as **true**. As this is the case, the learner defines **trans-par** as a subtype of **intrans-par** and sets the former to **true**, which makes the corresponding category part of the grammar. In **Step 3.3** the learner proceeds by executing **Reinforce-Parameters**({S/NP\NP}, {**subjdir=backward**, **vargdir=forward**}).

At any given moment it is possible to determine the grammar the learner has by checking which categorial parameters are set as **true**.

Before defining the second algorithm, **Reinforce-Parameters**, we now consider the second class of parameter: the word order parameters. These parameters

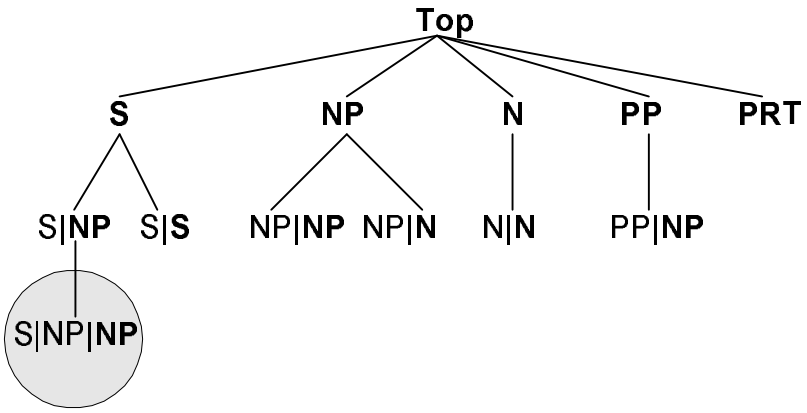

Figure 6.5: Possible Hierarchy 1

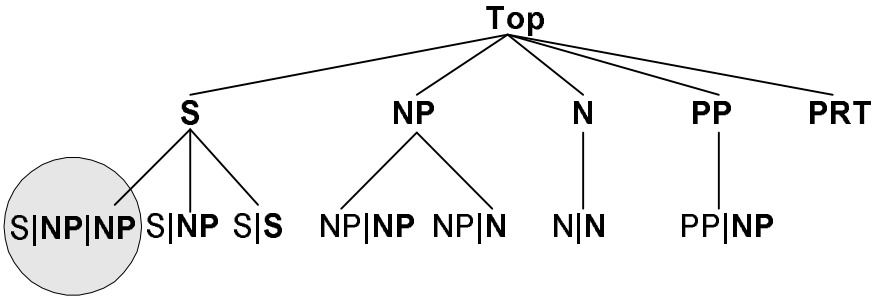

Figure 6.6: Possible Hierarchy 2

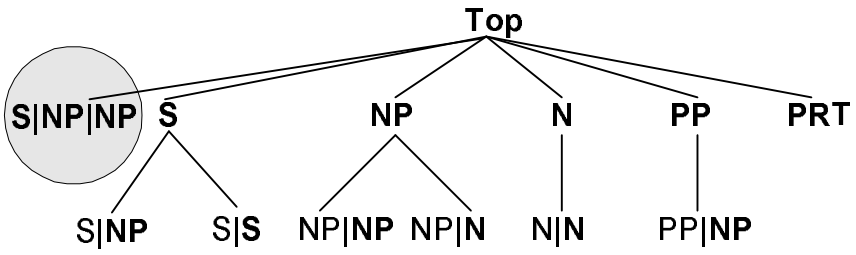

Figure 6.7: Possible Hierarchy 3

$$\left[\begin{array}{l}\textbf{trans}\\ \text{RESULT:SIGN:CAT} : \left[\begin{array}{l}\text{CAT-TYPE} : \textbf{s-cat}\\ \text{M-FEATS:ALLOWED} : \textbf{trans-par}\end{array}\right]\\ \text{ACTIVE} : \text{<!}\left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \textbf{subjdir},\end{array}\right]\\ \qquad\qquad \left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \textbf{vargdir}\end{array}\right]\text{!>}\end{array}\right]$$

Figure 6.8: Transitive Verb Type Partially Expanded

$$\left[\begin{array}{l}\textbf{intrans}\\ \text{RESULT:SIGN:CAT} : \left[\begin{array}{l}\text{CAT-TYPE} : \textbf{s-cat}\\ \text{M-FEATS:ALLOWED} : \textbf{intrans-par}\end{array}\right]\\ \text{ACTIVE} : \text{<!}\left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \textbf{subjdir}\end{array}\right]\text{/!>}\end{array}\right]$$

Figure 6.9: Specification of Intransitive Verb Type

$$\left[\begin{array}{l}\textbf{trans}\\ \text{RESULT:SIGN:CAT:M-FEATS:ALLOWED} : \textbf{trans-par}\\ \text{ACTIVE} : \text{<!}\left[\ \top\ \right], \left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \textbf{vargdir}\end{array}\right]\text{/!>}\end{array}\right]$$

Figure 6.10: Specification of Transitive Verb Type

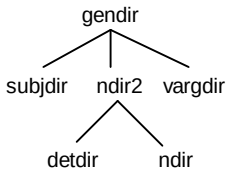

Figure 6.11: Fragment of the Parameters Hierarchy

specify the direction of each element in the subcategorisation list of a complex category, as described in section 4.5. These parameters are binary-valued and can be set as **backward** or **forward**, defining if a subcategorised element is to be found to the left or to the right, respectively. They are defined in a hierarchical relationship, with the supertype being **gendir**, which specifies, by default, the general direction for a language and from which all the other parameters inherit. Among the subtypes we have **subjdir**, which specifies the direction of the subject, **vargdir**, the direction of the other verbal arguments and **ndir2**, the direction of nominal categories. A fragment of the parameters hierarchy can be seen in figure 4.107, repeated here as figure 6.11. As the categories are defined in terms of an inheritance hierarchy, the parameters (and their values) in these categories are propagated throughout the hierarchy, from supertypes to subtypes, which inherit this information by default.

For the word order parameters, the learning algorithm stores frequency information: each of the word order parameters has a prior and a posterior probability associated with each of its values, and a current setting which corresponds to the value with the highest posterior probability. Following Briscoe [1999], the probabilities associated with the word order parameter values correspond to weights which are represented in terms of fractions, with the denominator storing the total evidence for a parameter and the numerator storing the evidence for a particular value of that parameter. For instance, if the value **backward** of the **subjdir** parameter has a weight of 9/10, it means that 9 out of 10 triggers for **subjdir** provided evidence for the value **backward**, and only 1 out of 10 for the other value, **forward**. Table 6.1 shows a possible initialisation for the **subjdir** parameter, where the prior has a weight of 1/10 for **forward**, corresponding to a probability of 0.1, and a weight of 9/10 for **backward**, corresponding to a probability of 0.9.

An approximation is used for computing the posterior probability associated with each parameter, where the incremental computation of the likelihood probability is smoothed with the prior probability, as we now explain. Initially, in the learning process the posterior probabilities associated with each parameter are initialised to the corresponding prior probabilities. These probabilities are going to define the parameter settings initially used, with the learner setting a parameter to the value with the highest posterior probability. For instance, in table 6.1,

as **backward** has a higher posterior probability, it is used as the initial current value for the parameter. Then, as triggering sentences are successfully parsed, the posterior probabilities of the parameter settings that allowed the sentence to be parsed are reinforced. The evidence provided by the trigger of a certain parameter is represented as an addition to the denominator and/or numerator of each of the posterior weights of the parameter values. In this case, when a sentence provides evidence for one of the settings of a parameter the denominator of the posterior probabilities of the two values are incremented by 1, and the numerator of the value being reinforced is also incremented by 1. For example, if the triggers provide evidence for **backward**, and its current posterior is 9/10, it is updated to 10/11, while the posterior probability of **forward** is updated to 1/11. Table 6.2 shows the status of the parameter after 5 triggers that provided evidence for the value **backward**. The learner uses the evidence provided by the triggers to choose certain parameter values, in order to be able to parse these triggers successfully while generating the appropriate logical form. Further triggers are used to reinforce these values or to negate them, and the learner sets the parameters to the values with the highest posterior probabilities. This approach ensures that a trigger does not cause an immediate change to a different grammar. Instead, the learner has to wait for enough evidence in the data before it can change the value of any parameter. As a consequence, the learner behaves in a more conservative way, being also robust to noise present in the input data, as discussed in chapter 7.

Table 6.1: Initialisation of a Parameter

Value	Prior		Posterior	
	Prob.	Weight	Prob.	Weight
Forward	0.1	$\frac{1}{10}$	0.1	$\frac{1}{10}$
Backward	0.9	$\frac{9}{10}$	0.9	$\frac{9}{10}$

The values of a subtype (leaf) word order parameter are determined by the direct evidence provided by the triggers, as is the case of **subjdir** and **vargdir**. On the other hand, a supertype (non-leaf) parameter such as **gendir**, receives only indirect evidence from its subtypes and its posterior probabilities are set according to the posterior probabilities of the subtypes that inherit its current value by default. As the parameters are defined in a default inheritance hierarchy, each time the posterior probabilities of a given parameter are updated it is necessary to update the posterior probabilities of its supertypes and examine the hierarchy to determine what the most probable grammar for these settings is, given the goal of converging to the target. This has to be done because many changes in

the parameters can result in more than one possible grammar, as shown in figures 6.2 to 6.4.

As the learner has a bias towards a concise grammar, by requiring that the supertype parameter is set to the current value used by the majority of its subtypes, it is possible to ensure that the grammar will have the minimum number of explicitly defined AVSs. In this way, as the values of the subtypes are being set, they influence the value of the supertypes. If the value of a given subtype differs from the value of the supertype, the subtype overrides the inherited default value and breaks the inheritance chain. As a result, this subtype does not contribute to the posterior probability of the supertype. For instance, in figure 6.4, **subjdir** overrides the default value specified by **gendir**, breaking the inheritance chain. Unset subtype parameters inherit, by default, the current value of their supertypes, and while they are unset they also do not influence the posterior probabilities of their supertypes.

The algorithm used to set these parameters is **Reinforce-Parameters**. It receives a set of triggers and the corresponding set of parameters as input.

Procedure Reinforce-Parameters(T_{wo},Pr_{wo})

Step 1 - *For each parameter $pr_i \in Pr_{wo}$, get values v_1 and v_2, associated with pr_i, with probabilities p_{v1} and p_{v2}.*

This step gets the corresponding values and probabilities for the parameters which had triggers in the sentence, and that are not categorial parameters.

Step 2 - *For corresponding trigger $t_i \in T_{wo}$ with value v_i, if v_i is equal to v_1, set the probability of v_1, p_{v1}, to*:

$$p_{v1} = \frac{numerator(v_1) + 1}{denominator(v_1) + 1} \tag{6.9}$$

and set the probability of v_2, p_{v2}, to:

Table 6.2: Status of a Parameter

Value	Prior		Posterior	
	Prob.	Weight	Prob.	Weight
Forward	0.1	$\frac{1}{10}$	0.07	$\frac{1}{15}$
Backward	0.9	$\frac{9}{10}$	0.93	$\frac{14}{15}$

$$p_{v2} = \frac{numerator(v_2)}{denominator(v_2) + 1} \tag{6.10}$$

If the value being reinforced by the trigger t_i, is equal to v_1, then the posterior probability of v_1 is reinforced by adding 1 to both the numerator and denominator of the corresponding fraction. *Numerator(x)* is a function that returns the numerator of a fraction, and *denominator(x)* returns the denominator of a fraction. Moreover, the second value v_2 has its denominator incremented by 1 also.

Step 3 - *Otherwise if v_i is equal to v_2, set the probability p_{v1} to*:

$$p_{v1} = \frac{numerator(v_1)}{denominator(v_1) + 1} \tag{6.11}$$

and set the probability p_{v2} to:

$$p_{v2} = \frac{numerator(v_2) + 1}{denominator(v_2) + 1}. \tag{6.12}$$

Otherwise, the learner reinforces the posterior probability of v_2, by adding 1 to both the denominator and numerator of the corresponding fraction, and by adding 1 to the denominator of v_1.

Step 4 - *If $p_{v1} > p_{v2}$, set the current value v_c in parameter pr_i to v_1 and the probability p_{vc} to p_{v1}.*

If v_1 is the value with the highest posterior probability in parameter pr_i, set the current value v_c of this parameter to v_1, and the probability p_{vc} to p_{v1}.

Step 5 - *Otherwise, if $p_{v2} > p_{v1}$, set the current value v_c of parameter pr_i to v_2 and the probability p_{vc} to p_{v2}.*

Otherwise set the current value v_c of this parameter to v_2, and the probability p_{vc} to p_{v2}.

Step 6 - *Execute* ***Compute-Probabilities-of-Supertypes*** *for the word order parameters, which determines the most concise encoding that reflects the parameter settings.*

After the parameter settings were reinforced, it is necessary to examine the current hierarchy and determine appropriate settings for the supertype parameters so that the most concise encoding for the grammar is obtained.

If the learner continues executing the previous example, **Reinforce-Parameters**({S/NP\NP}, {**subjdir**=**backward**, **vargdir**=**forward**}), in **Step 1** the learner determines that **subjdir** has values v_1=**forward**, with probability $p_{forward}$=1/48 and value v_2=**backward**, with probability $p_{backward}$=47/48.

These values are reinforced in **Step 3**, with v_i=**backward**, as expressed in the trigger, and providing evidence for v_2, whose probability is updated to $p_{backward}$=48/49 and v_1 (**forward**) has its probability updated to $p_{forward}$=1/49. **Step 5** is executed, with posterior probability of **backward** being greater than that of **forward**, and the current value of **subjdir** being set to **backward**.

This same process is executed for reinforcing the value **forward** for the parameter **vargdir** as the trigger provides evidence for v_1, where $p_{forward}$=33/35 and $p_{backward}$=2/35. **Step 2** updates its probability to $p_{forward}$=34/36 and the probability of v_2 to $p_{backward}$=2/36. After all the parameters in Pr_{wo} were processed, the algorithm then proceeds by executing **Compute-Probabilities-of-Supertypes** for the current settings of the UG.

Procedure Compute-Probabilities-of-Supertypes

Step 1 - *For each supertype word order parameter prs_i in the set of parameters Pr find the set of subtype parameters $prwo_{v1}$ in Pr that have v_1 as current value, and the set of subtype parameters $prwo_{v2}$ in Pr that have v_2 as current value.*

The goal of this step is to find all the subtypes of a given supertype, and divide them in two groups according to the value they adopt as current.

Step 2 - *If $|prwo_{v1}| > |prwo_{v2}|$ then set the current value of the supertype prs_i to v_1.*

This step is executed if the majority of the subtypes adopted v_1 as current value, and sets the current value of the supertype also to v_1.

Step 2.1 - *Then set the posterior probabilities p_{prsi} of the supertype parameter prs_i according to the posterior probabilities of all the subtypes that adopted v_1 as current value:*

$$p_{prsi}(v_1) = \frac{\sum_{pr_j \in prwo_{v1}} p_{prj}(v_1)}{\mid prwo_{v1} \mid} \tag{6.13}$$

$$p_{prsi}(v_2) = \frac{\sum_{pr_j \in prwo_{v1}} p_{prj}(v_2)}{\mid prwo_{v1} \mid} \tag{6.14}$$

This step sets the value of the posterior probabilities of the supertype to the normalised sum of the posterior probabilities of the subtypes that adopted v_1 as current value.

Step 3 - *Otherwise if $|prwo_{v2}| > |prwo_{v1}|$ then set the current value of the supertype prs_i to v_2.*

If the majority of the subtypes adopted value v_2 as current value, then the current value of the supertype is also set to v_2.

Step 3.1 - *Then set the posterior probabilities p_{prsi} of the parameter prs_i from the posterior probabilities of all the subtypes that adopted v_2 as current value:*

$$p_{prsi}(v_1) = \frac{\sum_{pr_j \in prwo_{v2}} p_{prj}(v_1)}{\mid prwo_{v2} \mid} \tag{6.15}$$

$$p_{prsi}(v_2) = \frac{\sum_{pr_j \in prwo_{v2}} p_{prj}(v_2)}{\mid prwo_{v2} \mid} \tag{6.16}$$

The posterior probabilities of the supertype are set based on the posterior probabilities of all the subtypes that adopted v_2 as current value.

Step 4 - *Else if* ($|prwo_{v1}| = |prwo_{v2}|$) *and* ($\frac{\sum_{pr_j \in prwo_{v1}} p_{prj}(v_1)}{|prwo_{v1}|} > \frac{\sum_{pr_j \in prwo_{v2}} p_{prj}(v_2)}{|prwo_{v2}|}$)
then set the current value of the supertype prs_i *to* v_1.

If there is an equal number of subtype parameters that adopt each of the values as current values, the learner compares the normalised sum of the posterior probabilities of the current value in each of the cases. If the sum of the posterior probabilities of the set of subtypes whose current value is v_1 is the highest, then v_1 is set as the current value adopted by the supertype.

Step 4.1 - *And set the posterior probabilities* p_{prsi} *of the parameter* prs_i *from the posterior probabilities of all the subtypes that adopted* v_1 *as current value:*

$$p_{prsi}(v_1) = \frac{\sum_{pr_j \in prwo_{v1}} p_{prj}(v_1)}{\mid prwo_{v1} \mid} \tag{6.17}$$

$$p_{prsi}(v_2) = \frac{\sum_{pr_j \in prwo_{v1}} p_{prj}(v_2)}{\mid prwo_{v1} \mid} \tag{6.18}$$

The posterior probabilities of the supertype are set to the normalised sum of the posterior probabilities of the subtypes that adopted v_1 as current value.

Step 5 - *Otherwise if* ($|prwo_{v1}| = |prwo_{v2}|$) *and* ($\frac{\sum_{pr_j \in prwo_{v2}} p_{prj}(v_2)}{|prwo_{v2}|} > \frac{\sum_{pr_j \in prwo_{v1}} p_{prj}(v_1)}{|prwo_{v1}|}$)
then set the current value of the supertype prs_i *to* v_2.

This step is executed if there is an equal number of subtype parameters that adopt each of the values as current value and the sum of the posterior probabilities of the subtypes whose current value is v_2 is the highest. Then v_2 is set as the current value adopted by the supertype.

Step 5.1 - *Set the posterior probabilities* p_{prsi} *of the parameter* prs_i *based on the sum of the posterior probabilities of all the subtypes that adopted* v_2 *as current value:*

$$p_{prsi}(v_1) = \frac{\sum_{pr_j \in prwo_{v2}} p_{prj}(v_1)}{\mid prwo_{v2} \mid} \tag{6.19}$$

$$p_{prsi}(v_2) = \frac{\sum_{pr_j \in prwo_{v2}} p_{prj}(v_2)}{\mid prwo_{v2} \mid} \tag{6.20}$$

The posterior probabilities of the supertype are set to the normalised sum of the posterior probabilities of the subtypes that adopted v_2 as current value.

Step 6 - *Else if* ($|prwo_{v2}| = |prwo_{v1}|$) *and* ($\frac{\sum_{pr_j \in prwo_{v2}} p_{prj}(v_2)}{|prwo_{v2}|} = \frac{\sum_{pr_j \in prwo_{v1}} p_{prj}(v_1)}{|prwo_{v1}|}$)
then set the posterior probabilities p_{prsi} *of the parameter* prs_i *to 0.5.*

If the number of subtypes that adopt each of the values as current value is the same, and the normalised sum of the posterior probabilities of the subtypes in each of these cases is the same, then set the posterior probability of the supertype to 0.5.

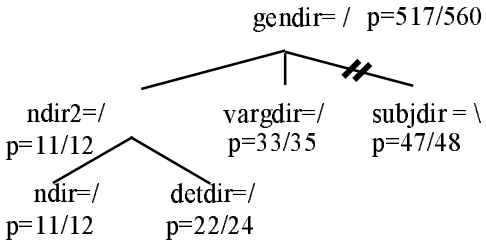

Figure 6.12: Word Order Parameter - Hierarchy 4

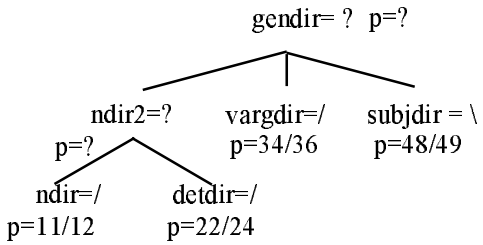

Figure 6.13: Word Order Parameter - Hierarchy 5

As an example, assume that the current status of the word order parameters hierarchy is that shown in figure 6.12, and the status changes after reinforcing the parameters to the one shown in figure 6.13, where **subjdir** is set to **backward** ('\') and **vargdir**, **detdir** and **ndir** to **forward** ('/'). There are several possible grammars that are compatible with these settings, with three possibilities shown in figures 6.2 to 6.4. However, there is a variation in terms of the compactness of

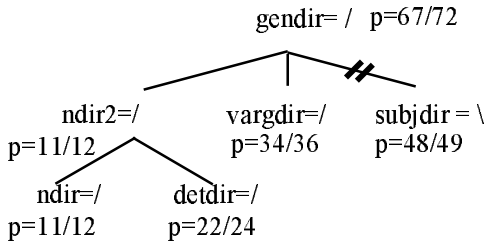

Figure 6.14: Word Order Parameter - Hierarchy 6

each hierarchy. In the first one (in figure 6.2), the supertype parameter **gendir** has value **backward**, which is inherited by **subjdir** and **ndir2**, and overridden by the 3 other parameters, where the inheritance chain is broken. In the second hierarchy (in figure 6.3), the supertype parameter **gendir** has value **backward**, which is inherited by **subjdir**, and overridden by all the other parameters, with the inheritance chain being broken in 2 cases. In the last hierarchy (in figure 6.4), **gendir** is specified as **forward** and is inherited by all its subtypes, with the exception of **subjdir**, which overrides the default value with **backward**, and the inheritance chain is broken in only one place. The learner needs to find the hierarchy that not only is compatible with the parameter settings, but that also has the most concise encoding for the UG, and to achieve this it executes the algorithm **Compute-Probabilities-of-Supertypes**. **Step 1** is executed for each of the word order supertype parameters and it divides the subtypes of a given supertype in two groups according to the value they adopt as current value. For instance, if we assume that v_1 is **forward** and v_2 **backward**, for the supertype **ndir2** both its subtypes **ndir** and **detdir** are part of the subset $prwo_{v1}$, because both are set to **forward**, and $prwo_{v2}$ is empty. The learner then executes **Step 2**, because $|prwo_{v1}| > |prwo_{v2}|$, and sets the current value of the supertype **ndir2** to **forward**. Using the probabilities from figure 6.13, **Step 2.1** sets the posterior probabilities of **ndir2** to:

$$p_{ndir2}(forward) = \frac{\frac{11}{12} + \frac{22}{24}}{2} = \frac{11}{12} \tag{6.21}$$

$$p_{ndir2}(backward) = \frac{\frac{1}{12} + \frac{2}{24}}{2} = \frac{1}{12} \tag{6.22}$$

If the learner now executes **Step 1** for the supertype **gendir**, which has 5 subtypes, these are divided into two groups: $prwo_{v1} = \{$**vargdir**, **ndir2**, **ndir**, **detdir**$\}$, and $prwo_{v1} = \{$**subjdir**$\}$. In this case, the learner also proceeds to **Step 2**, since $|prwo_{v1}| > |prwo_{v2}|$, and the current value of **gendir** is set to **forward**, with its posterior probabilities computed in **Step 2.1**, based on the posterior probabilities of **ndir2**, **ndir**, **detdir** and **vargdir**:

$$p_{gendir}(forward) = \frac{\frac{11}{12} + \frac{11}{12} + \frac{22}{24} + \frac{35}{36}}{4} = \frac{67}{72} \tag{6.23}$$

$$p_{gendir}(backward) = \frac{\frac{1}{12} + \frac{1}{12} + \frac{2}{24} + \frac{1}{36}}{4} = \frac{5}{72} \tag{6.24}$$

The resulting parameter settings are those shown in the hierarchy in figure 6.14 which are exactly the same as those shown in figure 6.4. This is the most concise encoding, specifying the need for exactly 2 AVSs, instead of for example, 3 in figure 6.3, or 4 in figure 6.2.

The BIPS learner has a preference for grammars (and thus hierarchies) that not only model the triggering data well, but are also compact. Therefore, the most probable grammar, among the ones consistent with the parameter settings, is the one where the default inheritance hierarchy is the most concise. This is the grammar that has the minimum number of explicitly defined AVSs, and, consequently, makes more efficient use of the inheritance mechanism.

6.3 Summary

In this chapter we described the architecture of the learner. The learner has a UG with its principles and parameters, and a learning module. Each input sentence-logical form pair is analysed by the learner, which detects the triggers provided and sets the parameters accordingly. The learning modules use the BIPS algorithm during the acquisition process to set the parameters of the UG. There are two different kinds of parameter: the categorial parameters and the word order parameters, and they are treated in slightly different ways. Categorial parameters control the categories that are allowed as part of the grammar and are incrementally set. Word order parameters determine the direction of the subcategorised elements of a complex category, and they are set based on the frequency with which their values are reinforced by the triggers. During the setting of the parameters, the learner is searching for the grammar that is consistent with the triggers seen so far and that is also the most concise.

Chapter 7

The Learning Tasks

During the learning process, there are several different aspects of language that the learning system has to acquire. The learner starts with a UG that is very general and only partially specified, and it has to use the input data to complete this initial knowledge. When the learner receives a triggering input sentence, which is a sentence annotated with a corresponding logical form and the semantic and syntactic category assignments for each word in the sentence, it provides several different levels of information that the learner has to process. The first one relates to the putative categories assigned to the words by the syntax learner. The BIPS learner needs to establish if, among the putative category assignments, there is a valid syntactic category assignment for the sentence. Then the learner has to establish if the categories in the VCA are part of the current grammar by checking the corresponding categorial parameters. If not, these categories need to be learned and the UG extended, as is described in section 7.1.

During this process, there are cases where the learner faces ambiguity between semantic and syntactic structures, with the Categorial Principles not being able to uniquely determine a syntactic form based only on the semantics of a word. One such case is that of locative Prepositional Phrases that are ambiguous between true arguments of the verb or as adjuncts, and this case is investigated in section 7.2. Once the learner establishes which is the appropriate status of the PP in relation to the verb, the corresponding categories and the triggers expressed in them are used for further processing.

When processing the triggers, the learner uses the information available to determine the appropriate word order that captures this language. This learning task is described in section 7.3.

Throughout the acquisition process the learner faces several difficulties, among them ambiguity and noise in the input data. The problem of ambiguous triggers is investigated in relation to PPs as arguments or adjuncts in section 7.2, and the problem of noise is discussed in relation to the acquisition of word order in section 7.3.

In these sections, the performance of the learner in several learning tasks is

discussed. We concentrate our investigations on the conditions for the learner to converge to the target, in relation to the learning tasks defined, and how noise and ambiguity can affect the convergence. The chapter ends with a discussion and analysis of the performance of the learning system in the tasks investigated.

7.1 Learning Categories

The UG has 89 categorial parameters, one for each of the categories, and they define which categories are part of the grammar. As an example, **intrans-par** is a categorial parameter for the intransitive verb type shown in figure 7.1. In the beginning of the learning process, the learner has a limited knowledge of languages, provided by the UG. The learner has a partial specification of the UG that contains information about basic categories and about complex categories with one complement. For these categories the corresponding categorial parameters are set as **true**. Initially only these categories are allowed as part of the grammar. Then as learning progresses, the learner is faced with the task of processing more complex sentences, with the need for more complex categories. As this happens the learner has to look for grammars that extend the UG with the new categories. It is this task that we address in this section, where the learner gradually acquires more complex categories.

$$\left[\begin{array}{l}\textbf{intrans-sign}\\ \text{CAT} : \left[\begin{array}{l}\text{RESULT:SIGN:CAT} : \left[\begin{array}{l}\text{CAT-TYPE} : \textbf{s-cat}\\ \text{M-FEATS:ALLOWED} : \left[\begin{array}{l}\textbf{intrans-par}\\ \text{VALUE} : \textbf{/true}\end{array}\right]\end{array}\right]\\ \text{ACTIVE} : \text{<!}\left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \textbf{subjdir}\end{array}\right]\text{!>}\end{array}\right]\end{array}\right]$$

Figure 7.1: Intransitive Verb Type Partially Expanded

The learner is looking for the grammar that adds the new categories and that is the most compact, as discussed in section 6.2. The most compact grammar is the one where the minimum number of AVSs needs to be explicitly specified. To find this grammar, one assumption that is made is that the learner uses the algorithm defined by Light [1993], which finds an optimal insertion place for a new type in the hierarchy. This is the insertion place where the minimum number of parent types provides the maximum number of attributes required by the new type. In terms of verbs, the learner starts with knowledge about the basic verbal category for intransitive verbs and it has to dynamically expand the hierarchy in order to add the appropriate types corresponding to more complex subcategorisation frames. The learner is guided in this search by the Categorial Principles that find an appropriate syntactic structure for a given semantic predicate. As

an example, suppose the learner receives a sentence like:

• (7.1) *You have one*

paired with the meaning:

{have(1 , 2 , 3), you(4 , 2), one(5 , 3)},

where the predicate of each word is determined to be:

{have$_1$=have(a,b,c), you$_1$ = you(d,e), one$_1$ = one(f,g)}

and they are paired with the following set of putative category assignments:

{(you = NP, have = (S\NP/NP), one = NP)}

According to the Categorial Principles, the categories assigned for these semantic predicates are correct. S\NP/NP is the category for transitive verbs, shown in figure 7.2, and this category provides triggers for the parameter **trans-par** that allows transitive categories as part of the grammar, and for the parameters **subjdir** and **vargdir**, encoding, respectively, the direction for the subject and for the object of the verb.

$$\left[\begin{array}{l}\textbf{trans-sign}\\ \text{CAT} : \left[\begin{array}{l}\text{RESULT:SIGN:CAT} : \left[\begin{array}{l}\text{CAT-TYPE} : \textbf{s-cat}\\ \text{M-FEATS:ALLOWED} : \textbf{trans-par}\end{array}\right]\\ \text{ACTIVE} : \; <!\left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \textbf{subjdir},\end{array}\right]\\ \qquad\qquad \left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np}\\ \text{DIRECTION} : \textbf{vargdir}\end{array}\right]^{!>}\end{array}\right]\end{array}\right]$$

Figure 7.2: Transitive Verb Type Partially Expanded

If transitive verbs are already part of the grammar, the parameter **trans-par** associated with transitive verbs is set as **true**, and the triggers are used to set the word order parameters, **subjdir** and **vargdir**, as explained in section 6.2.1. On the other hand, if **trans-par** is set as **false**, then transitive verbs are not part of the hierarchy yet. For instance, assuming that the current grammar is the one shown in figure 7.3, possible hierarchies showing different insertion places for the transitive verb type are shown in figures 6.5 to 6.7, repeated here as 7.4 to 7.6. In these figures the information that needs to be explicitly specified is displayed in bold font. Since the hierarchy needs to be gradually expanded to account for more complex cases, the learner uses inheritance to take advantage of

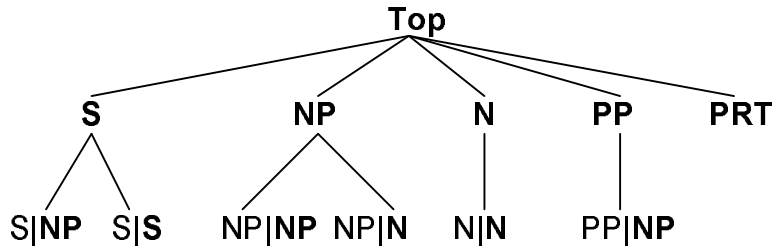

Figure 7.3: Current Hierarchy

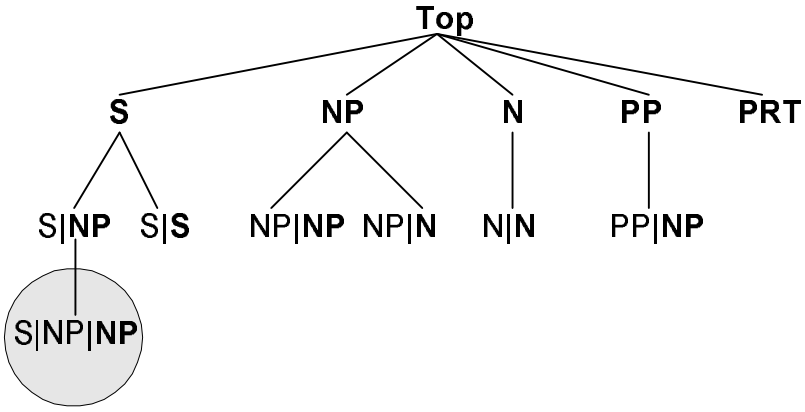

Figure 7.4: Insertion Place for the Transitive Verb Type - 1

the information already acquired, in order to obtain the most concise grammar.[1]

Because the learner is looking for a compact hierarchy, the current hierarchy needs to be expanded so that the minimum number of specifications needs to be added to it, making maximum use of the information already contained in it. The learner finds that the intransitive verb type with category S|NP, shown in figure 7.1, is an optimal supertype for a transitive verb type, as shown in the hierarchy in figure 7.4, because only the information shown in figure 7.7 would need to be added, and it verifies that **intrans-par**, its corresponding categorial parameter, is set as **true**. As a consequence, the learner adds **trans-par** as a subtype of **intrans-par**, setting it to **true**, and adds the transitive verb type as a subtype of the intransitive verb type.

The other options shown in figures 7.5 and 7.6 correspond to the TDFSs

[1]In this work, the assumption is made that the definitions of the categories and categorial parameters are all available as part of the UG. Then, as the learning process takes place, these definitions are added as part of the hierarchy in optimal places so that the resulting hierarchy is the most concise. However, this assumption does not alter the results obtained by the learning system.

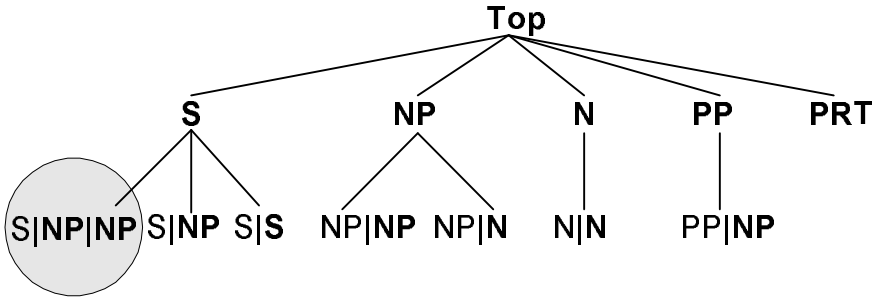

Figure 7.5: Insertion Place for the Transitive Verb Type - 2

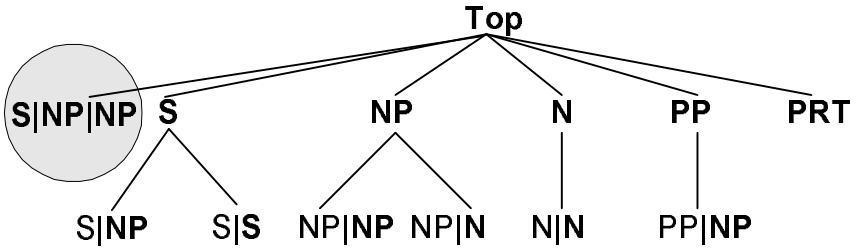

Figure 7.6: Insertion Place for the Transitive Verb Type - 3

$$\left[\begin{array}{l}\textbf{trans-sign} \\ \text{CAT} : \left[\begin{array}{l}\text{RESULT:SIGN:CAT:M-FEATS:ALLOWED} : \textbf{trans-par} \\ \text{ACTIVE} : \left\langle !\left[\top\right], \left[\begin{array}{l}\text{SIGN:CAT} : \textbf{np} \\ \text{DIRECTION} : \textbf{vargdir}\end{array}\right]/!\right\rangle\end{array}\right]\end{array}\right]$$

Figure 7.7: Specification of Transitive Verb Type

shown in figures 7.8 and 7.9, respectively. These options require more information to be encoded in the new type.

With the addition of this category to the hierarchy, the learner becomes able to successfully process sentences containing transitive verbs. After that, when a ditransitive verb is presented to the learner, it needs to extend the hierarchy by adding the appropriate type for the category S|NP|NP|NP. The most concise grammar requires the ditransitive verb type to be inserted as a subtype of the transitive verb type that encodes the category S|NP|NP. Also, **ditrans-par**, the categorial parameter associated with ditransitive verbs, has to be added as a subtype of **trans-par**. This means that a ditransitive verb can only be learned after the category for transitive verbs is learned and added to the hierarchy. Thus, if **trans-par** is set as **true**, then **ditrans-par** can also be set as **true**, as is the case in this example. Otherwise, if **trans-par** is **false**, the ditransitive verb type

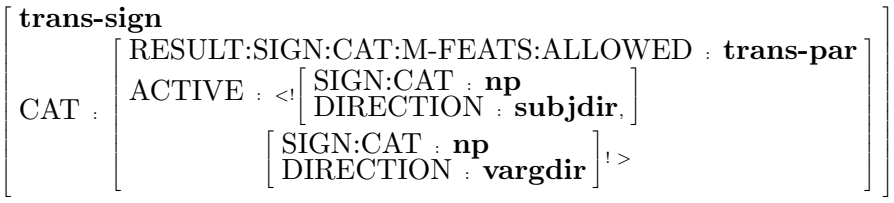

Figure 7.8: Transitive Verb Type - Alternative 1

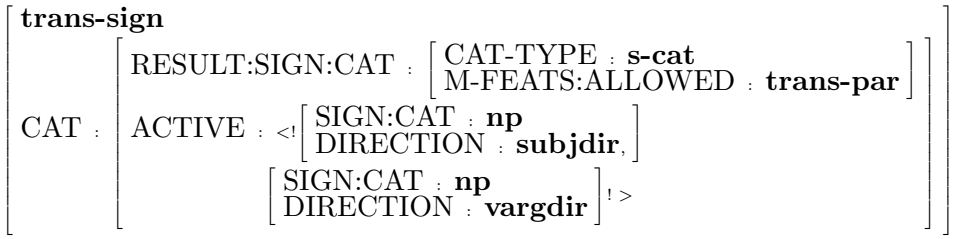

Figure 7.9: Transitive Verb Type - Alternative 2

cannot yet be added as part of the grammar, and the learner ignores the sentence.

As a consequence of the requirement for the most concise encoding, an implicit ordering on the learning of categories is obtained, with categories being learned in incremental order of complexity using the maximum amount of information already acquired by the learner. For example, an intransitive verb subcategorising for one complement is less complex than a transitive verb subcategorising for two complements which is less complex than a ditransitive verb subcategorising for three complements.

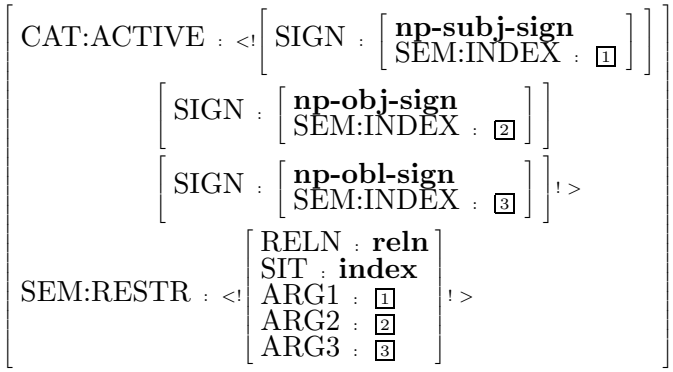

Figure 7.10: Possible Linking Pattern - 1

As the learner extends its grammar with more complex categories, these categories also introduce more complex linking patterns between the semantic argu-

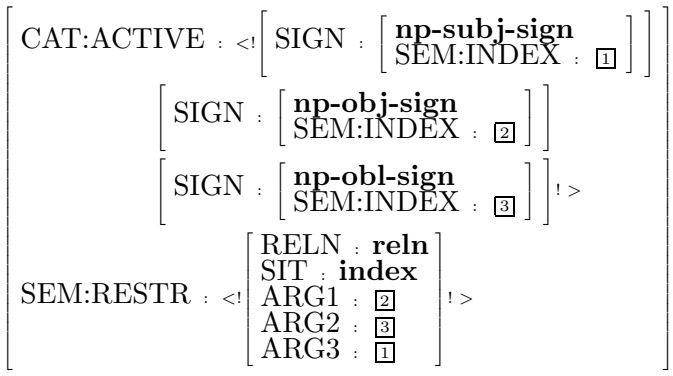

Figure 7.11: Possible Linking Pattern - 2

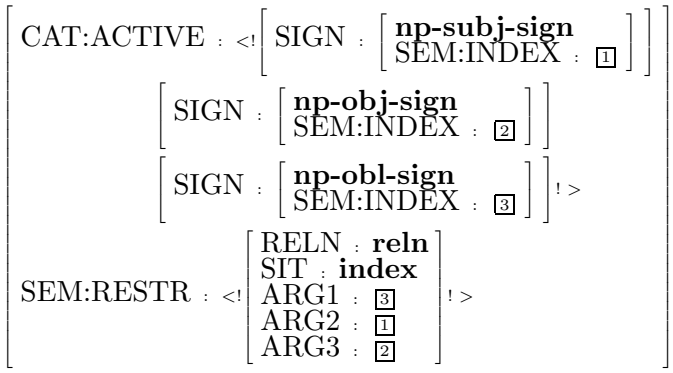

Figure 7.12: Possible Linking Pattern - 3

ments and the syntactic complements.[2] Therefore, when the grammar is incrementally constructed, the linking constraints are also incrementally added to it. When learning new and more complex linking patterns, the learner may be uncertain as to which of the possible linking options, given the semantic arguments and syntactic complements, it should choose. For example, when an oblique transitive verb is learned, there are six different linking possibilities among the syntactic complements and semantic arguments, as shown in figures 7.10 to 7.15. The learner needs to decide which of these six options it is going to adopt, but without any criterion one option is as good as the other. One source of bias for the learner in this decision is the requirement for the most concise grammar, following the MDL-style bias. Thus, the learner examines the linking patterns it already has, and uses this information to decide among these options. Let us assume that the learner has already decided that a transitive verb type is an optimal supertype and has also determined its corresponding linking pattern as that shown in figure 7.16. Then, the learner determines that the most concise

[2]Possible linking patterns in English are described in section 4.2.4.

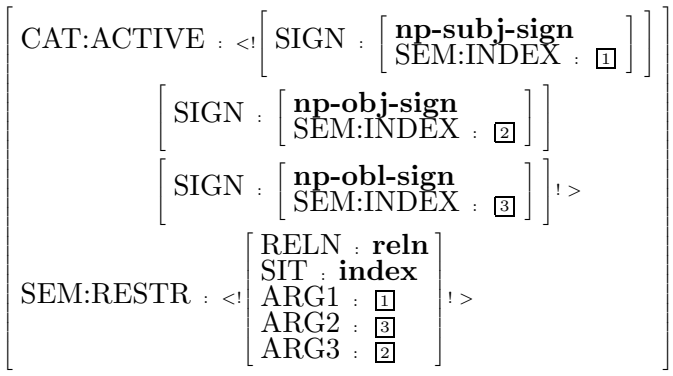

Figure 7.13: Possible Linking Pattern - 4

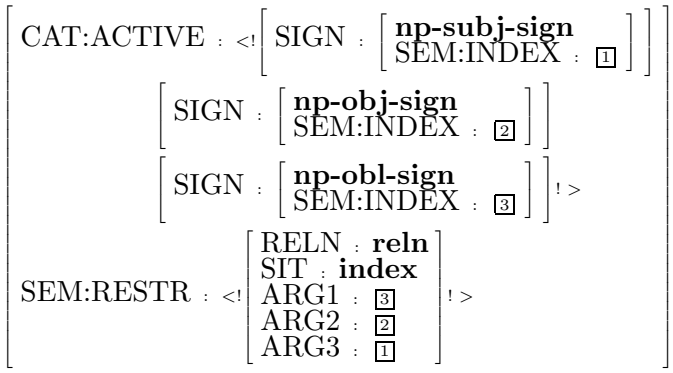

Figure 7.14: Possible Linking Pattern - 5

encoding is obtained by choosing the option in figure 7.10 which only requires a minimum of additional information to be encoded in the grammar[3], as shown in figure 7.17. All the other possibilities would require the addition of more information than that, and they are less interesting for the learner. The use of a default inheritance hierarchy allows the linking information to be passed from supertype to subtype. This means that a subtype will get most of its linking constraints for free from its supertype. It will only need to specify the linking for the remaining arguments, and there are only a finite number of possibilities for this linking. Moreover, from these possibilities, all the patterns that are incompatible with the inherited information are ruled out. In this way, the default inheritance hierarchy not only reduces the amount of linking information that needs to be learned, but also rules out possible patterns that are incompatible with the inherited information. Consequently, the learner makes very efficient use

[3]Also, as discussed in section 4.2.4, the same linking pattern is adopted by ditransitive verbs with the rule of permutation rotating the elements in the subcategorisation list, until an appropriate ordering is obtained.

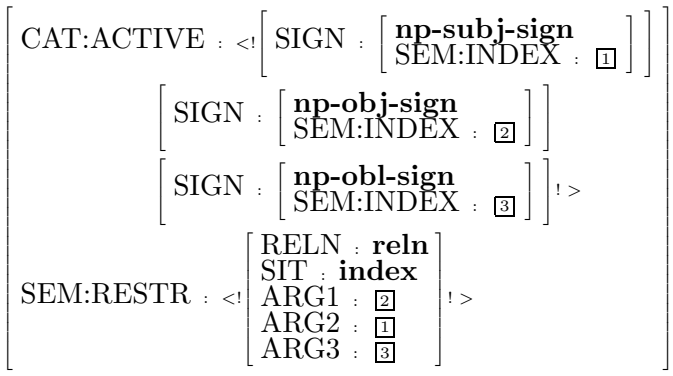

Figure 7.15: Possible Linking Pattern - 6

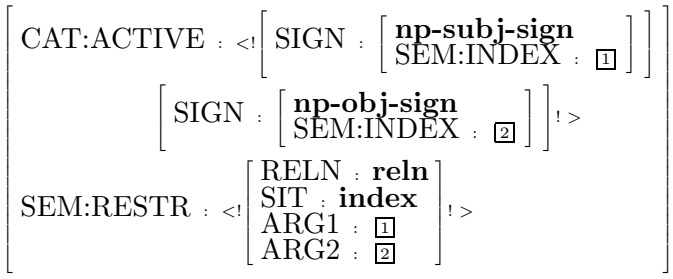

Figure 7.16: Specification of Transitive Linking Pattern Expanded

of the default inheritance hierarchy to learn and encode new linking information.

There are cases where there is a mismatch between the semantic structure, the subcategorisation frame, and the linking constraint. One interesting case is that of raising verbs, which we consider to be in the canonical form in a sentence like *It seems that John runs*, where *seems* is an intransitive raising verb. For such a verb, there is a mismatch between the semantic argument structure and the syntactic subcategorisation frame. It has only one semantic argument, a propositional argument, and it is syntactically realised as an S. However, this cannot be the only subcategorised category of the verb, as the ungrammaticality of the sentence **Seems that John runs* indicates. As required by the Categorial Principles every sentence must have a subject, which means that every verb must subcategorise for a subject. Consequently the subcategorisation frame for such a verb is S/S\NP. Thus, an intransitive raising verb is a one-place predicate, but subcategorises for two complements: an NP and a predicative complement. As such a construction did not occur frequently enough in the processed sentences to be learned, we did not implement a solution for dealing with them, but in what follows we present the outline of a possible treatment.

When the learner is faced with such a raising verb containing only one semantic argument and two syntactic complements, there is an indeterminacy between

$$\left[\begin{array}{l}\text{ORTH} : \textbf{orth} \\ \text{CAT:ACTIVE} : \langle ! \left[\top\right], \left[\top\right], \left[\text{SIGN} : \left[\begin{array}{l}\textbf{np-obl-sign} \\ \text{SEM:INDEX} : \boxed{3}\end{array}\right]\right] ! \rangle \\ \text{SEM:RESTR} : \langle ! \left[\text{ARG3} : \boxed{3}\right] ! \rangle\end{array}\right]$$

Figure 7.17: Specification of Oblique Transitive Linking Pattern

two possible linking constraints:

- the semantic argument is linked to NP, which is the pattern adopted by most verbs, or
- the semantic argument is linked to S.

As the canonical form has the NP subject occurring as a pleonastic or expletive subject, this form indicates the absence of a corresponding semantic role [Hyams 1986], and rejects the first hypothesis. Thus the canonical form of the raising construction with a pleonastic or expletive subject gives evidence for the learner to select the second linking pattern, where the semantic argument is linked to S. Related raising constructions such as that in the sentence *John seems to run* can be obtained by applying a lexical rule to the intransitive raising verb, which generates the derived category S/(S\NP)\NP from S/S\NP. In this way, as the categories are incrementally learned, so are the linking patterns.

7.2 Learning from Ambiguous Triggers

The learner uses the Categorial Principles to determine appropriate syntactic categories for the semantic predicates used as input. In most cases, the Categorial Principles guides the learner to find the required syntactic categories. However, there are cases where it is not possible to uniquely determine an appropriate subcategorisation frame for a given semantic predicate. In this section, we discuss one case where the learner faces ambiguity when choosing an appropriate subcategorisation frame for the semantic structure associated with a particular word: locative Prepositional Phrases (PPs) occurring with verbal constructions as subcategorised complements or non-subcategorised adjuncts. The ambiguity between these cases arises because using this particular logical form representation, in the case of locative PPs, the logical form describing the verb with an argument PP is similar to that describing the verb with an adjunct PP.[4] For example, the sentence:

[4]It may be the case that using another representation for the logical form this ambiguity does not arise.

• (7.2) *Bill swims across the river*

with logical form:

{swim(1,2), bill(3,2), across(1,4), the(5,4), river(6,4)}

shows a case where the PP is an argument of the verb *swim*, and where the appropriate subcategorisation frame for the verb should include the PP: (S/PP)\NP, as a locative intransitive sign. On the other hand, the sentence:

• (7.3) *The dog chases the cat in the garden.*

with logical form:

{chase(1,2,3), the(4,2), dog(5,2), the(6,3), cat(7,3), in(1,8), the(9,8), garden(10,8)}

exemplifies a case where the PP is an adjunct. Thus it should not be included in the subcategorisation frame of the transitive verb *chase*, which is (S/NP)\NP. For both sentences, the logical form has a similar structure, with both a verbal and a locative predicate, and so the logical form cannot be used to help the learner resolve the ambiguity: given the logical forms {swim(1,2) across(1,5)} and {chase(1,2,3) in(1,8)}, which syntactic category should the learner choose for each of these verbs? This ambiguity constitutes a significant problem for the learner, since it has to decide whether or not a given PP is functioning as a complement of a verb and should be included as part of the subcategorisation frame of the verb, or if it is working as an adjunct and should not be included as a subcategorised complement of the verb. Three different cases to which the learner is exposed are identified, based on Pustejovsky [1991 and 1995], Wechsler [1995] and Verspoor [1997], with the PP occurring as an obligatory argument, as an optional argument, or as an adjunct[5]:

1. **The PP is an obligatory argument of the verb.** For certain verbs the PP is an obligatory argument of the verb and should be included in its subcategorisation frame. An instance of this case is the verb *put*, in sentence 7.4:

   • (7.4) *Mary put the book on the shelf*,

   with logical form:

   {put(1, 2, 3), mary(4, 2), the(5,3), book(6, 3), on(1, 7), the(8,7), shelf(9,7)}.

[5]In this work we classify PPs in terms of these three cases, even though more fine-grained classifications can be used as in [Pustejovsky 1995].

In this sentence *put* occurs with a locative PP. Also, as the ungrammaticality of sentence 7.5 suggests, this verb requires a locative PP:

- (7.5) * *Mary put the book*

The appropriate syntactic category for the verb[6] is ((S/PP)/NP)\NP.

2. **The PP is an optional semantic argument of the verb.** For example, a verb such as *float* can occur with a locative PP, as in sentence 7.6, from Pustejovsky [1995, p. 126]. In 7.6 the motion verb *float* is modified by a directional PP which is an optional argument of the verb:

   - (7.6) *The bottle floated into the cave.*

   with logical form:

   {float(1 , 2), the(3 , 2), bottle(4 , 2), into(1 , 5), the(6 , 5), cave(7 , 5)}.

   This verb may also occur without the PP, as in sentence 7.7:

   - (7.7) *The bottle floated.*

   with logical form:

   {float(1 , 2), the(3 , 2), bottle(4 , 2)}

   This is a case of a verb that can occur in both constructions with the PP being a semantic argument, which, when occurring, must be included in the subcategorisation frame of the verb. Consequently, the appropriate category for the verb *float* in the first sentence is (S/PP)\NP, and in the second is S\NP.

3. **The PP is an adjunct.** Adjuncts modify the logical form of the sentence, but are not part of the subcategorisation frame of the verb. The PP *in the park* in sentence 7.8 is an example of an adjunct that is neither part of the semantic argument structure of the verb *kiss* nor part of its subcategorisation frame:

   - (7.8) *Bill kisses Mary in the park*,

   whose logical form is:

   {kiss(1 , 2 , 3), bill(4 , 2), mary(5 , 3), in(1 , 6), the(7 , 6), park(8 , 6)}.

   This verb can also occur without the PP, as in sentence 7.9:

   - (7.9) *Bill kisses Mary*,

   whose logical form is:

   {kiss(1 , 2 , 3), bill(4 , 2), mary(5 , 3)}.

   The appropriate syntactic category for the verb in both sentences is S/NP\NP.

[6] This work does not include, in its investigation, elliptical or noisy constructions. Therefore, the sentences analysed and the frequencies reported exclude these cases.

When faced with a locative PP, the learner has to identify which of these cases is appropriate. The required subcategorisation frame is determined independently for each verb sense. It depends on the semantic type of the verb, and on its frequency of occurrence with a particular subcategorisation frame and predicate argument-structure combination.

In order to determine if a locative PP is an obligatory argument of the verb, the learner uses frequency information about the occurrence of each verb with locative PPs. If the frequency with which they occur together is above a certain threshold, the PP is considered to be an obligatory argument of the verb. In an analysis of all the mother's sentences in the entire Sachs corpus, only two occurrences of *put* without the locative PP were found: one seems to be an instance of an elliptical construction, and the other a derived sense. The frequency with which *put* occurs with a locative PP indicates that the PP is an argument of the verb, and it needs to be included in the subcategorisation frame of the verb. On the other hand, for verbs like *float* and *kiss* in sentences like 7.6 to 7.9, the locative PP is an occasional constituent, with the semantics of the sentence including the location predicate only in these cases. The occasional occurrence of PPs indicates that they are not obligatory PP arguments.

Thus, if the learner detects the occurrence of the verbal predicate above a certain threshold with the locative predicate, then it considers that the latter is required as an argument in the subcategorisation frame of the verb. In this case, the threshold is set to 80% of the total occurrences of a verb. This is high enough for discarding adjuncts and optional arguments that occur occasionally, and at the same time is not high enough to be affected by the occurrence of noise. We also experimented with other values, but this had optimal effect.

If the frequency is not above the threshold, then the PP can be either an optional argument or an adjunct. To determine if a PP is an optional argument, the learner checks if the preposition can be semantically selected by the verb. This approach to identify non-obligatory argument PPs follows Wechsler's proposal of semantically motivated preposition selection [Wechsler 1994 and 1995], where a PP is an argument of a verb if it can be selected by the verb on pragmatic grounds. Pragmatic knowledge is represented in terms of a hierarchy of types and it is used to classify verbs and prepositions. A fragment of such a hierarchy is shown in figure 7.18. Following Wechsler's approach a verb such as *talk*, which is of type **communicative-act**, can select as its argument a preposition such as *to*, which is of type **comm-to**, because these two types unify on the world knowledge hierarchy, as in *Bill talks to Mary*. On the other hand, this verb does not select a preposition such as *across* of type **directed-motion** as its argument, because their types do not unify. However, this preposition can be selected as the argument of the verb *swim* in sentence 7.2, which is a **motion-act** type. In this way, the pragmatic knowledge confirms certain PPs (e.g. *across*) as arguments of some verbs (e.g. *swim*), while rejecting others.

If a locative PP is rejected as argument of a verb on pragmatic grounds, then

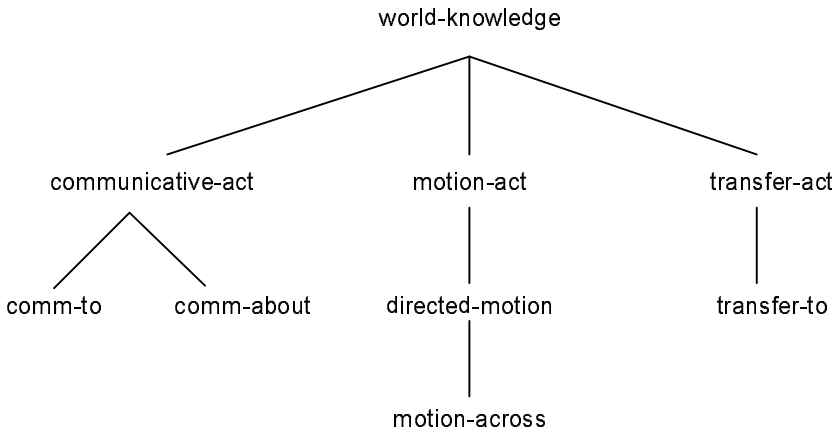

Figure 7.18: Fragment of Hierarchy of World Knowledge

the PP is treated as an adjunct and is not included in the subcategorisation frame of the verb. Once the learner decides which is the case for a particular verb PP combination, it uses the triggering information, including the appropriate subcategorisation frame of the verb, for further processing.

7.2.1 Argument or Adjunct?

This section discusses how the proposed approach helps the learner facing ambiguous triggers in relation to locative PPs as arguments or adjuncts. For this task, the learner is evaluated in terms of three different verbs: *put* where the PP is an obligatory argument, *come*, where the locative PP is an optional argument, and *draw* (in the sense of drawing a picture) where the PP is an adjunct. These verbs are representative of each case and the sentences in which they occur are taken from the mother's section of the complete Sachs corpus, which is the largest of the parents' sections. The status of the locative PPs occurring with these verbs is determined following syntactic tests for argument structure[7]. The specific test used for determining the status of the PP is the "do so" test, which is a standard test for argument structure. In this test, the claim is that a complete complement can be replaced by "*do so*". In the case of obligatory arguments, only the full constituent **verb PP** or **verb NP PP** can be replaced by *do so*, while in the case of adjuncts, the **verb** or **verb NP** constituent can also be replaced by *do so*. The sentences 7.9 to 7.17 indicate that the PPs are arguments of the verbs *put* and *come*, and adjuncts of the verb *draw*.

- (7.9) *You put Goldie through the chimney*

[7]For a discussion about several tests for determining argument structure see Verspoor [1997].

- (7.10) *You put Goldie through the chimney and Bob also did so*
- (7.11) * *You put Goldie through the chimney and Bob did so through the window*
- (7.12) *You came from the garden*
- (7.13) *You came from the garden and John also did so*
- (7.14) * *You came from the garden and John did so from the kitchen*
- (7.15) *You drew in the park*
- (7.16) *You drew in the park and John also did so*
- (7.17) *You drew in the park and John did so in the garden*

In the mother's sentences from the entire Sachs corpus, the occurrence of these verbs with locative PPs is as shown in table 7.1. This table shows in the first column the verb, in the second the frequency of occurrence of the particular verb sense with a locative PP, in the third the frequency of occurrence of the verb sense without a locative PP, and in the fourth the total occurrences. Only sentences which contain the verbs in the relevant sense are considered, and sentences where the verb occurs with the locative PP and with particles are not considered.

Table 7.1: Frequency Information about Locative PPs

Verb	Frequency with PP	Frequency without PP	Total
put	137	2	139
come	24	8	32
draw	9	12	21

The first verb, *put*, is a verb that occurs mostly with locative PPs, as reflected by the frequency in table 7.1, and where the locative PPs are considered to be true arguments of the verb, following the "do-so" test. The sentences that occur without the locative PP (table 7.2) correspond to 1.4% of the total.

Table 7.2: Sentences with *put* without a Locative PP

Sentence
What are you putting in braids
We put chocolate

The first sentence has a derived sense of *put*, where the PP is not locative, in spite of the subcategorisation frame being similar (((S/PP)/NP)\NP). The second of these sentences seems to be elliptical and needs a context in order to

be completely acceptable (from the Sachs Corpus).

- (7.18) *No, we don't put glasses in the milk.*
- (7.19) *We put chocolate.*

The verb *come* occurs in the corpus with both constructions, as exemplified in sentences 7.12 to 7.14 and 7.20, with the locative PP occurring in 75% of the sentences. Consequently, frequency information disconfirms *come* as having obligatory PP arguments. In the case of *come* the PP is an optional argument, and sentences without it are also acceptable, as sentence 7.20 shows. Nonetheless, the locative PP is an argument of the verb *come* [Levin 1993], as can be confirmed by the "do so" test (sentences 7.12 to 7.14).

- (7.20) *I'll come*

As the frequencies in table 7.1 indicate, **draw** is a verb that can frequently occur in each of these constructions, as exemplified in sentence 7.15 to 7.17 and 7.21. From the 22 sentences in the corpus containing *draw*, 57% occur without the locative PP, and when the locative PP occurs, it is considered to be an adjunct according to the syntactic test for argument structure.

- (7.21) *I am not drawing*

Upon receiving an input sentence, the learner attempts to disambiguate the PP as argument or adjunct of the verb by first checking if the frequency of occurrence of the verb with locative PPs is above the threshold of 80%, in which case the PP is considered to be an obligatory argument of the verb. Otherwise, the learner checks if the verb can select the PP on pragmatic grounds. If so, the PP is an optional complement of the verb. On the other hand, if this is not the case, then the PP is considered to be an adjunct. Once the learner determines which is the appropriate case and selects a suitable subcategorisation frame, then this information is used to set the parameters of the UG.

The learner is given as input sentences annotated with logical forms, and semantic and syntactic categories for the words in each sentence. In the input sentences, each of these verbs occurs with the patterns and frequencies presented in table 7.1. The results obtained for each of these three verbs are shown in table 7.3. In this table the first column shows the verb, the second the number of sentences used as input for each of the subcategorisation frames, and in the third, the number of times the learner correctly disambiguates the locative PP for each of the subcategorisation frames.

The learner correctly selects the appropriate subcategorisation frame in all of the cases, which confirms the effectiveness of the proposed approach to disambiguate locative PPs.

This experiment used three verbs representing each of the cases that the learner has to face when a locative PP is encountered in the input data:

- obligatory arguments, disambiguated with frequency information,
- optional arguments, selected on pragmatic grounds, and
- adjuncts.

In terms of frequency of occurrence of the verbs with the locative PPs, other verbs in the mother's sentences from the entire Sachs corpus also have a similar pattern, with the locative PP being frequent for arguments of the verb, and less frequent than the non-locative construction for the case of adjuncts:

- *stay*, which according to the "do-so" test has an obligatory locative PP argument, occurs in 12 sentences in the corpus (such as sentence 7.22), and in all of them the locative PP is present,
- *eat*, which according to the test does not have a locative PP argument, occurs in 81 sentences as a transitive verb (such as sentence 7.23), and in only 1 of these it has a locative PP, and
- *play*, which also does not have a PP argument, is in 10 sentences in the corpus, as an intransitive verb (as in sentence 7.24), and in 4 of those it has a locative PP.

These results indicate that it is indeed possible for the learner to disambiguate between locative PPs as arguments or adjunct based on semantic compatibility and frequency information.

- (7.22) *That's because the piggy had to stay at home.*
- (7.23) *We can't eat it now.*
- (7.24) *I think she just wants to go out to play.*

Table 7.3: Disambiguation of Locative PPs

Verb	Input Sentence	Correctly Disambiguated
put	locative PP = 137	137
	non-locative PP = 2	2
come	locative PP = 24	24
	non-locative PP = 8	8
draw	locative PP = 9	9
	non-locative PP = 12	12

7.3 Learning Word Order Parameters

Word order parameters reflect the underlying order in which constituents occur in different languages. There are 18 word order parameters defined, based on typological considerations ([Croft 1992] and [Comrie 1981]), and they are embedded in an inheritance hierarchy, as explained in section 4.5. As defined, the UG allows for languages with the four basic word orders, defined in terms of the canonical order of the verb (V), subject (S) and object (O): SVO, SOV, VSO and OVS. These major word order groups are obtained by setting the parameter **subjdir**, for the direction of the subject, and **vargdir**, for the direction of the other verbal arguments:

- SVO: **subjdir** = **backward** and **vargdir** = **forward**,
- SOV: **subjdir** = **backward** and **vargdir** = **backward**,
- OVS: **subjdir** = **forward** and **vargdir** = **backward** and
- VSO: **subjdir** = **forward** and **vargdir** = **forward**[8].

Specific languages inside each of these major language groups are allowed by the other word order parameters, such as **ppdir** that defines if the language accepts prepositions or postpositions, and **reldir** that defines if relative clauses are allowed before or after the noun they modify [Comrie 1981]. In the beginning of the learning process, the partial specification of the UG needs to be set for direction according to the input data. As the learner starts receiving input data, it tries to analyse the data, and in order to get successful derivations consistent with the logical form, the learner has to set the direction of the relevant parameters for the target language. If the learner is exposed to English, which is an SVO language, the learner has to set, among other parameters, the **subjdir** parameter to **backward**, as the NP subjects occur to the left of the verb, and the **vargdir** parameter to **forward**, because other verbal complements occur to the right of the verb. The setting of these parameters is also executed in such a way as to get the most concise encoding for the grammar, taking into account the information flow in the inheritance hierarchy. In order to do so, the learner collects frequencies for word order parameters, reflecting the occurrence of each of the possible values of a parameter in the triggering data. It uses these frequencies to decide which of the values should be used to set the parameter, as described in section 6.2.1. These frequencies also help the learner deal with noisy and ambiguous triggers that can be present in the input data.

[8]Other possible word orders such as VOS and OSV can be obtained by adding another parameter to the current model of the UG that inverts the order of the subject and objects of the verb with respect to the semantic arguments, as done by Briscoe [1997-2000]. However, the orders obtained by the current UG are varied enough for the proposed investigation.

In this work we investigate the performance of the learner in setting the word order parameters, in a number of experiments where different environments are defined, with different degrees of noise and with different starting points. In each learning cycle, the learning system receives sentences from the annotated Sachs corpus paired with logical forms as input. This English corpus has an underlying SVO order and a predominantly right branching structure. The sentences in the input corpus are presented to the learner only once, sequentially, in the original order.

After a sentence is assigned semantic and syntactic categories, if a VCA is among the category assignments, then the sentence is processed, with the triggers expressed in the sentence being detected and used to set the parameters of the UG. As the sentences are processed, the learner sets the parameters to reflect the constructions found in these sentences, and the order in which the constituents occurred, using the algorithms described in chapter 6.

This learning task is investigated under several different conditions. These conditions provide different environments for the learner during the acquisition process. They can vary with respect to the starting point for the learning process, and with respect to the amount of noise and ambiguous triggers present in the input data. As the assignments made by the syntax learner for a given sentence may vary slightly at each learning cycle, each condition is run 10 times and the results are reported in relation to the averages obtained. In these experiments, the learner is initialised with all word order parameters unset, unless otherwise stated. In relation to the categories in the grammar, all the categorial parameters are **false**, except those for the basic categories and for the complex categories with one subcategorised complement, unless otherwise stated.

7.3.1 The Unset Learner: a Basic Case

The unset learner is initialised with all word order parameters being unset. It has the convergence displayed in table 7.4 and in figure 7.19, with an average of 129.1 triggers that have VCAs and that can be used to set the parameters, out of 1,517 sentences. With these triggers, 8.1 parameters in average are set out of a total of 18 parameters, resulting in 13.5 parameters being correct according to the target because of the inheritance of default values from the supertype parameters.

Table 7.4: Convergence of the Unset Learner

Learner	Triggers	Parameters Correct	Parameters Set
Unset	129.1	13.5	8.1

As the output of the syntax learner may vary slightly at each run, the learner

may be given a somewhat different group of sentences for setting the parameters at each run. In this way, some sentences that are used to set parameters in one run may not be available in the next one. As a consequence, the parameters that are set in one run may not be set in the next, since the learner may not have triggers for doing that. However, this variation is small, and throughout each of the 10 runs, the same parameters tend to get set.

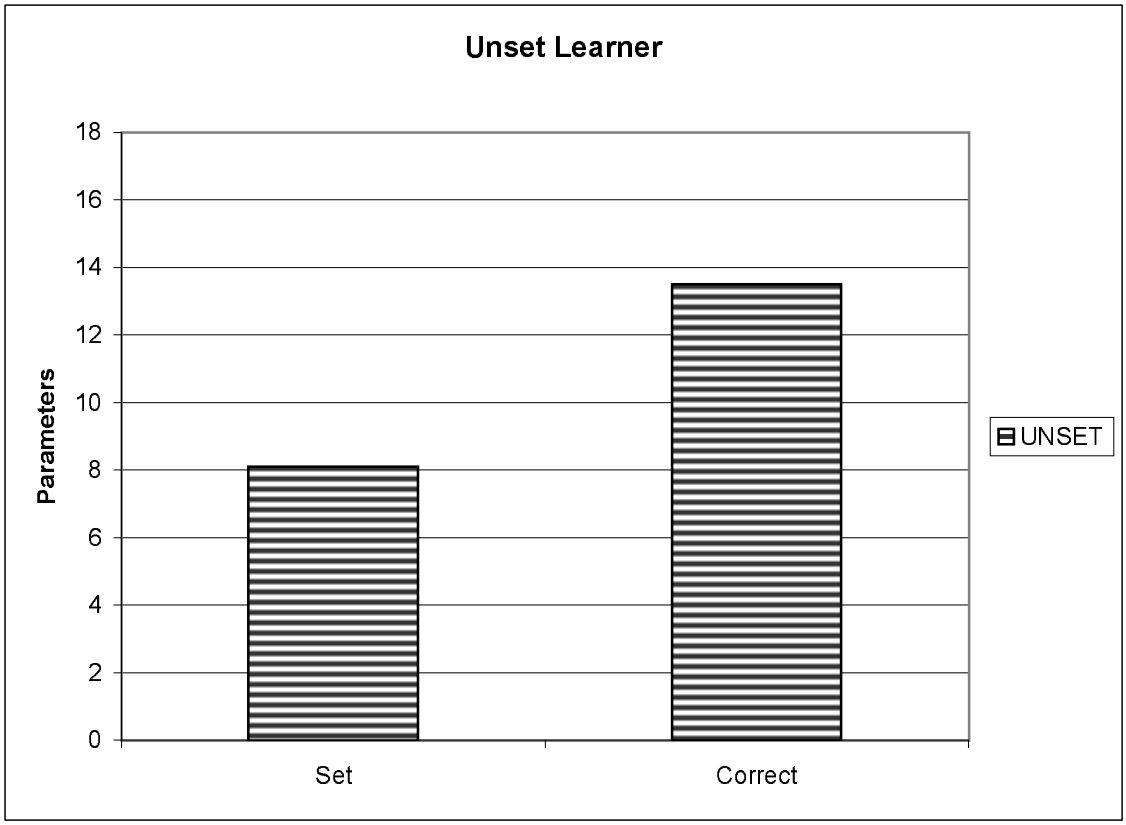

Figure 7.19: Performance of the Unset Learner

Figure 7.20 shows the convergence pattern obtained in a typical learning cycle for the unset learner, for the **subjdir**, **vargdir**, **ppdir** and **gendir** parameters. This figure shows how the **subjdir** parameter is set early on in the learning process, and how the value of **gendir** is strongly influenced by this, so that **gendir** is set as **backward** until there are more parameters set as **forward** than **backward**, around trigger 16.

In the input data there are triggers for an average of 8 parameters, and all of them are correctly set in relation to the target. However, because of the inheritance mechanism, even though only 8 parameters are set this resulted in 13.5 of the 18 parameters having the correct value according to the target, since they inherit by default the value of their supertypes. Thus, the encoding of

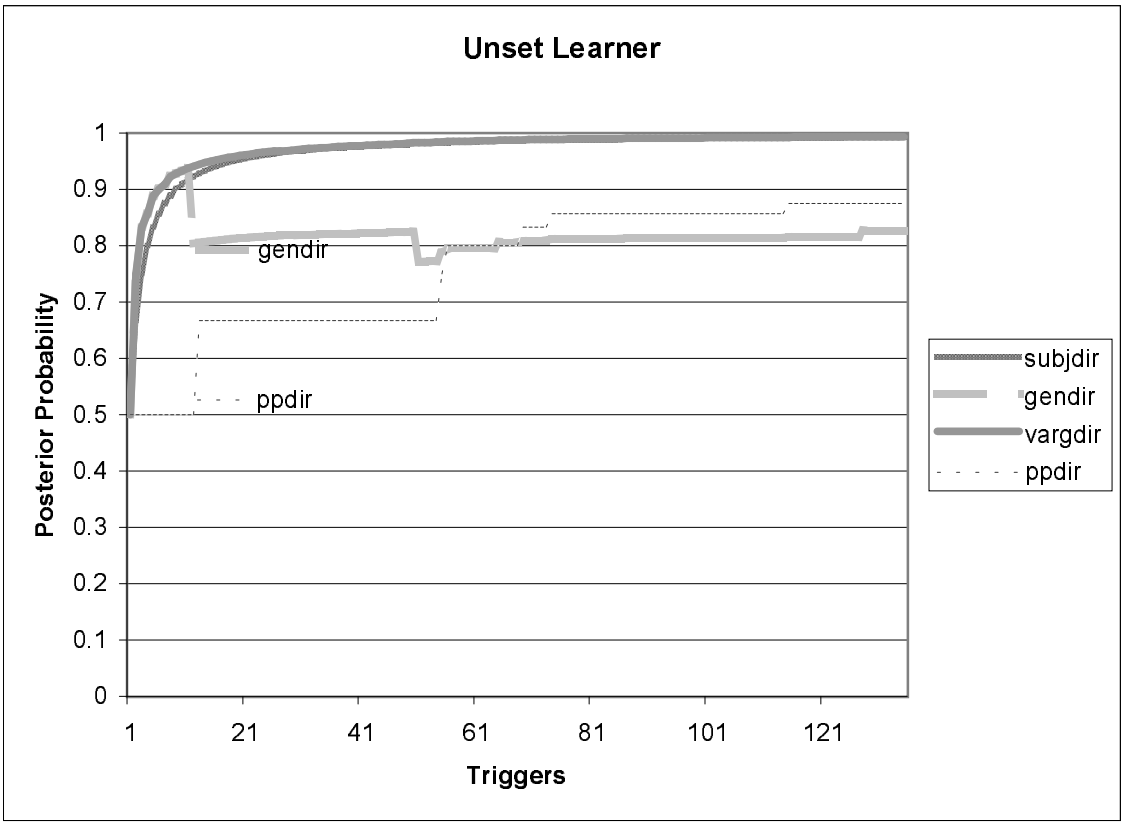

Figure 7.20: Convergence of the Unset Learner

parameters in terms of a default inheritance hierarchy speeds up the convergence to the target. As a consequence, the learner is successfully converging in the direction of the target even with a limited amount of input data.

7.3.2 Different Starting Points

The UG is defined with its parameters that are set according to the data, but how should the parameters be initialised in the beginning of the learning process? In the previous experiment, the learner is initialised with all word order parameters unset, and it succeeds in setting its parameters, given the amount of triggering data available. However, Chomsky [1981] suggests that some of the parameters may have an initial default value, which is retained in the absence of incompatible input during the learning process. Moreover, Hyams [1986] and Lightfoot [1991], among others, also propose the use of default values as a way of preventing the learner from converging to a superset grammar. Should the learner start with unset parameters, or should it have a bias towards a group of languages? How do the different initialisations affect the convergence of the learner? In

this section one set of experiments is described, where five different learners are defined, corresponding to five different initialisations of the parameter settings of the UG. In these experiments we investigate how the initialisations - or starting points - of the learners influence convergence to the target grammar. The first one, the unset learner, is initialised with all parameters unset, as in the previous section. It is the basic case against which the others can be compared. The other learners, the default learners, are each initialised with default parameter values corresponding to one of four basic word orders, defined in terms of the canonical order of the verb, subject and objects: SVO, SOV, VSO and OVS. The parameters **subjdir**, **vargdir** and **gendir** of the default learners are set according to each of the basic orders, with **gendir** having the same direction as **vargdir**, and all the other parameters having unset values. These parameters have the prior and posterior probabilities of 0.1 for one value and 0.9 for the other, as shown in table 7.5. In these tests, two sets of weights corresponding to the prior and posterior probabilities of the parameters are used, as shown in table 7.6. The learners in Learners-10 are initialised with lighter weights (1/10 and 9/10) than those in Learners-50 (5/50 and 45/50), which are initialised with much heavier weights. The latter are used to investigate the effect of such a heavy initialisation in the performances of the learners. In this case, even though the probabilities are the same as in Learners-10, the weights are much stronger and that causes an impact on the number of triggers needed by the learner to converge to the target value. This set of experiments investigates how the different learners perform in a normal environment with a limited English SVO corpus as input. The results obtained in each of these conditions are now described.

Table 7.5: Initialisations of the Different Learners

Learners	Direction	Subjdir	Vargdir	Gendir
SVO	backward	0.9	0.1	0.1
	forward	0.1	0.9	0.9
SOV	backward	0.9	0.9	0.9
	forward	0.1	0.1	0.1
OVS	backward	0.1	0.9	0.9
	forward	0.9	0.1	0.1
VSO	backward	0.1	0.1	0.1
	forward	0.9	0.9	0.9

Condition 1: Learners-10

In the first condition, the parameters **subjdir**, **vargdir** and **gendir** of the default learners are initialised as shown in table 7.7. The results from the first experiment

Table 7.6: Initial Weights of the Different Learners

Learners	Weights
Learners-10	0.1 = 1/10 0.9 = 9/10
Learners-50	0.1 = 5/50 0.9 = 45/50

(table 7.8) show no significant variation in the performances of the different learners. This is the case with the number of parameters that are correct in relation to the target, with an average of 13.5 parameters out of 18, and also with the number of parameters that are set given the triggers available, with an average of 8.1 parameters out of 18, as shown in figure 7.21.

Table 7.7: Initialisations of the Different Learners-10

Learners	Direction	Subjdir	Vargdir	Gendir
SVO	backward	0.9 =9/10	0.1 =1/10	0.1 =1/10
	forward	0.1 =1/10	0.9 =9/10	0.9 =9/10
SOV	backward	0.9 =9/10	0.9 =9/10	0.9 =9/10
	forward	0.1 =1/10	0.1 =1/10	0.1=1/10
OVS	backward	0.1 =1/10	0.9 =9/10	0.9 =9/10
	forward	0.9 =9/10	0.1 =1/10	0.1 =1/10
VSO	backward	0.1 =1/10	0.1 =1/10	0.1 =1/10
	forward	0.9 =9/10	0.9 =9/10	0.9 =9/10

Table 7.8: Convergence of the Different Learners - Condition 1

Learners	Triggers	Parameters Correct	Parameters Set
Unset	129.1	13.5	8.1
SVO	129.1	13.5	8.1
SOV	133	13.6	8.3
OVS	132	13.5	8
VSO	132	13.5	8

The only difference between them is the time needed for each learner to con-

verge: the closer the starting point of the learner is to the target, the faster it converges, as shown in figure 7.22 for the **subjdir** parameter. This figure shows all the learners converging to the target value, with high probability, and with a convergence pattern very similar to the one presented by the unset learner. Even those default learners that are initialised with values incompatible with the target soon overcome this initial bias and converge to the target. A similar pattern occurs for **vargdir** and **gendir**.

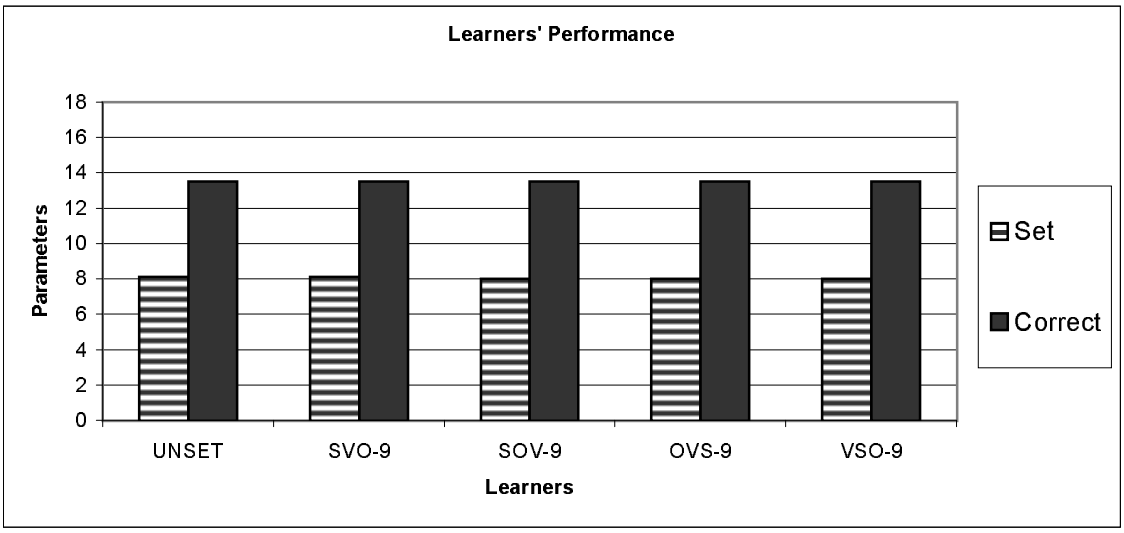

Figure 7.21: Learners-10 in a Normal Environment

Condition 2: Learners-50

The results in the previous condition did not show any significant difference between the performances of the different learners. The next experiment tests how the use of stronger weights to initialise the learners affects their performances, given the limited amount of input data. The parameters **subjdir**, **vargdir** and **gendir** are initialised with stronger weights (table 7.9). These weights provide an extreme bias for each of the learners. In this condition, the learners are tested again in a normal noisy environment.

The results (table 7.10) show that the learners once more have similar performances regardless of the initialisations. An average of 13.5 correct word order parameters is obtained, with 9 out of 18 parameters being set. Figure 7.23 shows the convergence patterns presented by these learners for the **subjdir** parameter. Once again, the learners that are initialised with values compatible with the target have a faster convergence pattern, as can be seen for the SOV and SVO learners. Nevertheless, the overall speed of convergence is much slower than in the previous condition.

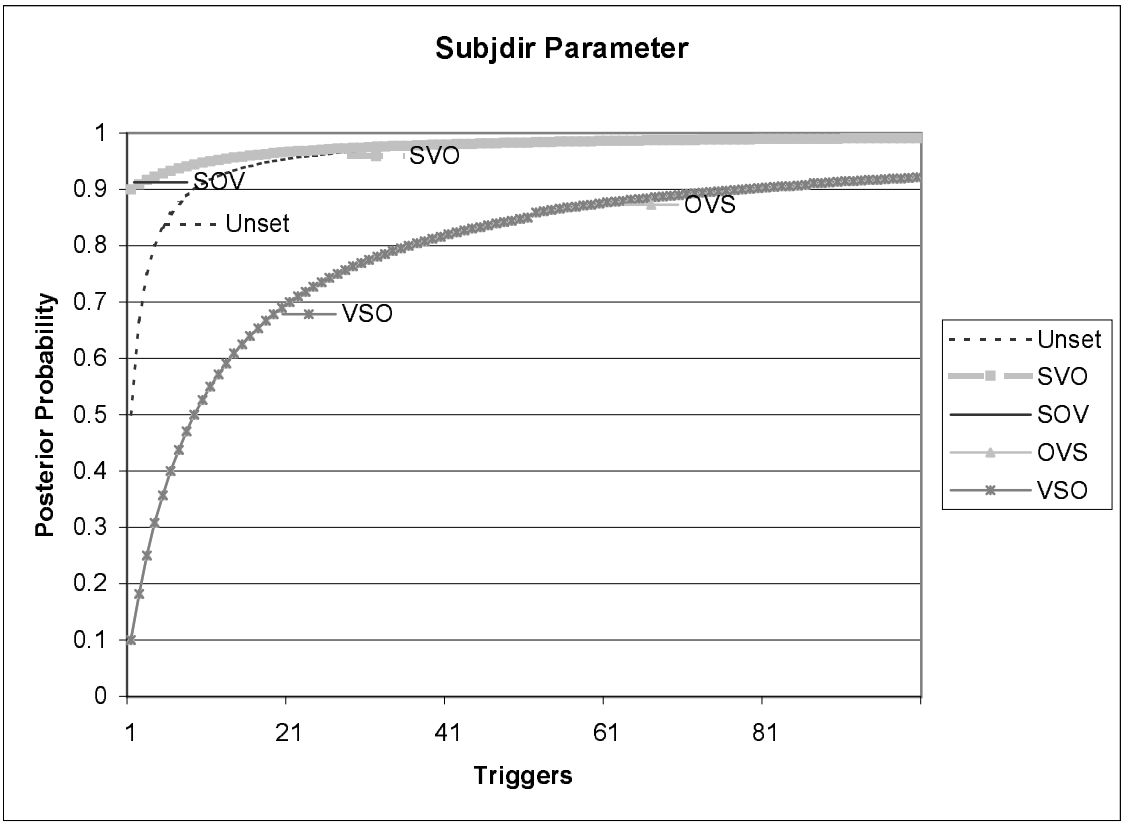

Figure 7.22: Convergence of Subjdir - Learners-10 - Noisy Environment

Table 7.9: Initialisations of the Different Learners-50

Learners	Direction	Subjdir	Vargdir	Gendir
SVO	backward	0.9 =45/50	0.1=5/50	0.1 =5/50
	forward	0.1 =5/50	0.9 =45/50	0.9 =45/50
SOV	backward	0.9 =45/50	0.9 =45/50	0.9 =45/50
	forward	0.1 =5/50	0.1 =5/50	0.1=5/50
OVS	backward	0.1 =5/50	0.9 =45/50	0.9 =45/50
	forward	0.9 =45/50	0.1 =5/50	0.1 =5/50
VSO	backward	0.1 =5/50	0.1 =5/50	0.1 =5/50
	forward	0.9 =45/50	0.9 =45/50	0.9 =45/50

In spite of the strong initialisations, the performances of the learners are only slightly affected by the stronger weights (table 7.10). They have performances similar to those obtained by the learners in condition 1, as shown in figure 7.24, comparing the learners in conditions 1 and 2.

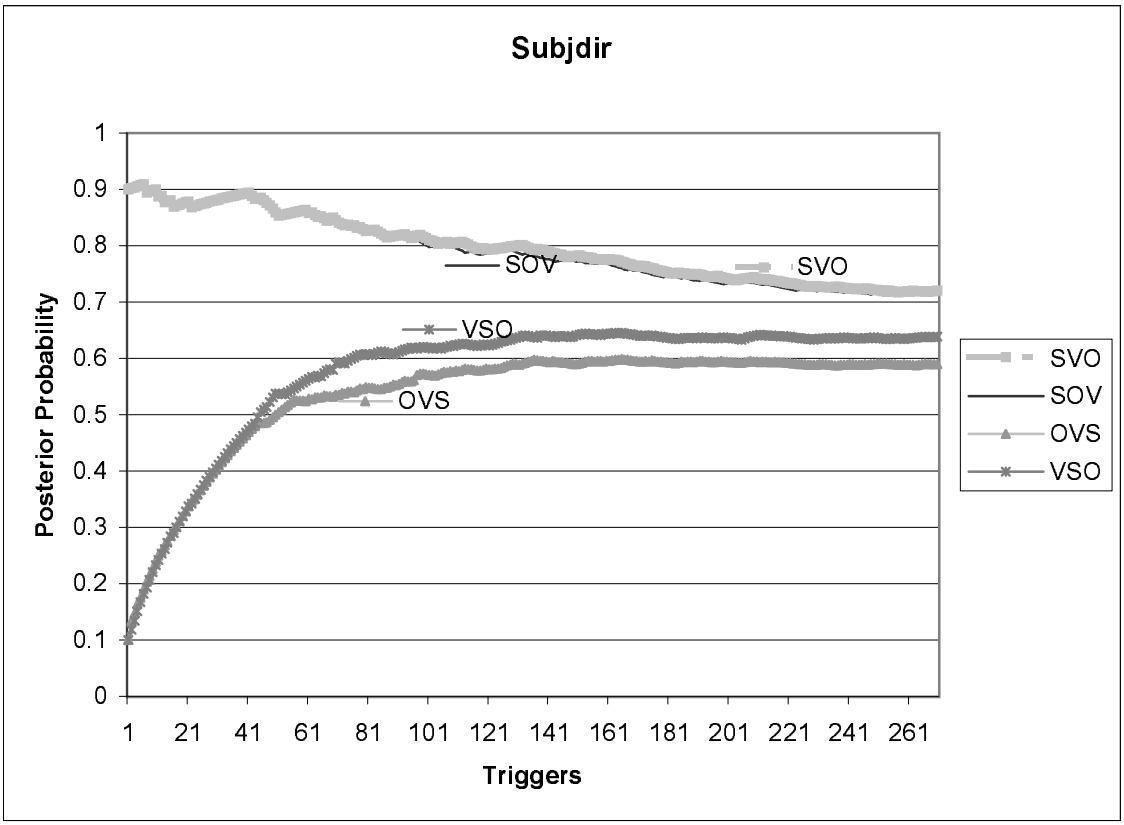

Figure 7.23: Convergence of Subjdir - Learners-50 - Noisy Environment

Table 7.10: Convergence of the Different Learners - Condition 2

Learners	Triggers	Parameters Correct	Parameters Set
SVO-50	264.8	13.5	9
SOV-50	271.3	13.5	9
OVS-50	273	13.5	9
VSO-50	272	13.5	9

Condition 3: Learners-10 in a Noise-free Environment

Figures 7.22 and 7.23 show some peaks in the convergence pattern displayed by the learners. This is particularly acute in figure 7.23, where there is a fall in the posterior probability of both the SVO and SOV learners of almost 20% until the end of the cycle. Conditions 3 and 4 investigate how noise affects the convergence of the learners. In this particular case, we are interested in word

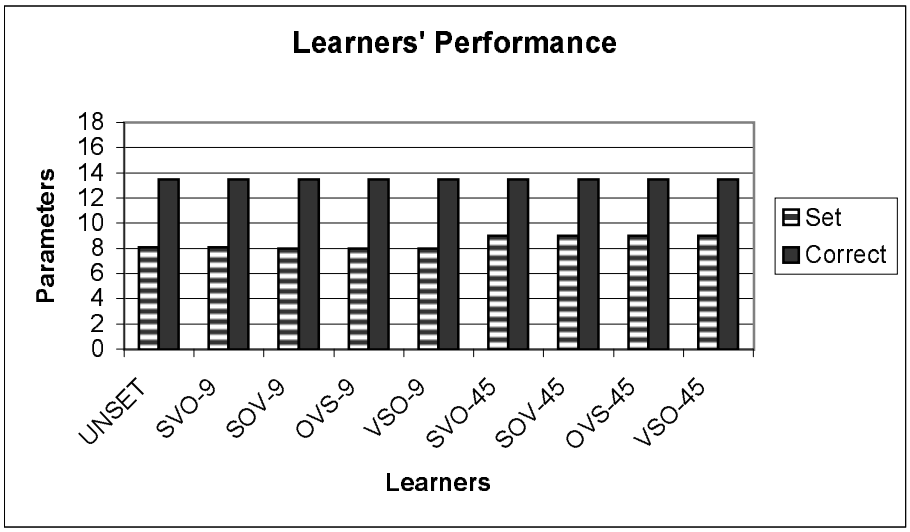

Figure 7.24: Learners' Performances in a Noisy Environment

order noise that occurs because of incorrect directionality specifications for some of the constituents for the target language. In order to test this, the learners are exposed to a noise-free environment, and we compare their performances with those obtained in the first two conditions. To obtain a noise-free environment, as each trigger is processed a module is used for correcting category assignments if incorrect directionality specifications are detected. This module has access to the appropriate directionality specification of each category in English and it uses this information to correct noisy triggers. This condition uses the same initialisations as the ones in condition 1. The results obtained by the learners are shown in table 7.11.

Table 7.11: Convergence of the Different Learners - Condition 3

Learners	Triggers	Parameters Correct	Parameters Set
Unset	132.13	13.5	8
SVO-10	132	13.5	8
SOV-10	128.2	13.5	8.1
OVS-10	131	13.5	8
VSO-10	131.5	13.5	8

These learners have performances similar to those in conditions 1 and 2, with an average of 13.5 of the 18 parameters correct in relation to the target, and

an average of 8 parameters that can be set with the triggers available (figure 7.25). However, the convergence is slightly faster for all learners than in the first condition and considerably faster than in the second condition, as can be seen in figure 7.26. These results show that, indeed, the presence of noise slows down the convergence of the learners, because they need more triggers to compensate for the effect produced by the noisy triggers. The declines in the posterior probabilities of the learners in conditions 1 and 2 are caused by noise in the category assignments of the input triggers, which provides incorrect evidence for the parameter values. Every time a noisy trigger is encountered, it provides evidence for a non-target parameter value, slowing down the learners.

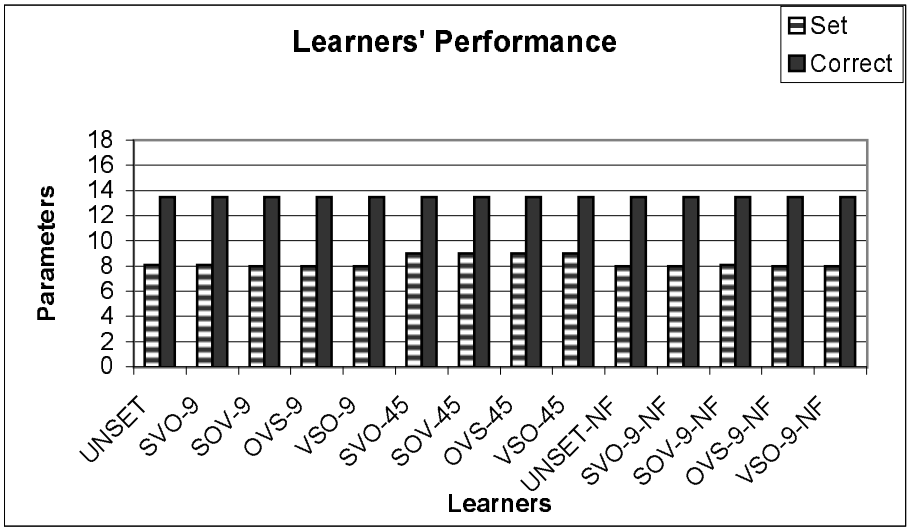

Figure 7.25: Learners' Performances in Different Environments

Condition 4: Learners-50 in a Noise-free Environment

When the noise-free environment is used with these stronger weights, the convergence pattern is much faster for all learners, when compared to condition 2 (which uses a noisy environment), but still slower than conditions 1 and 3 (figure 7.27).

The effect produced by the noise is increased by the stronger weights, such that all learners have a slower convergence to the target in condition 2. All the learners have similar performances to those obtained in all the previous conditions, as can be seen in table 7.12, and in figure 7.28, with learners which are very robust and are converging to the target.

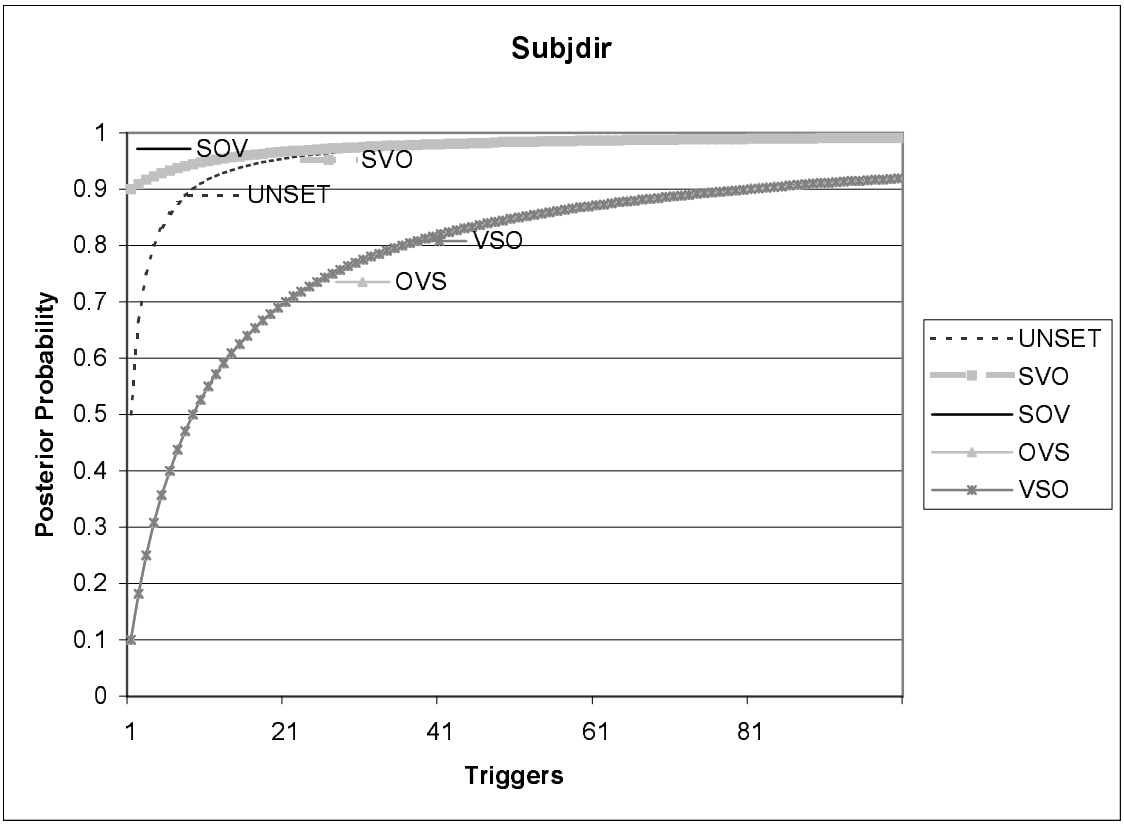

Figure 7.26: Convergence of Subjdir - Learners-10 - Noise-free Environment

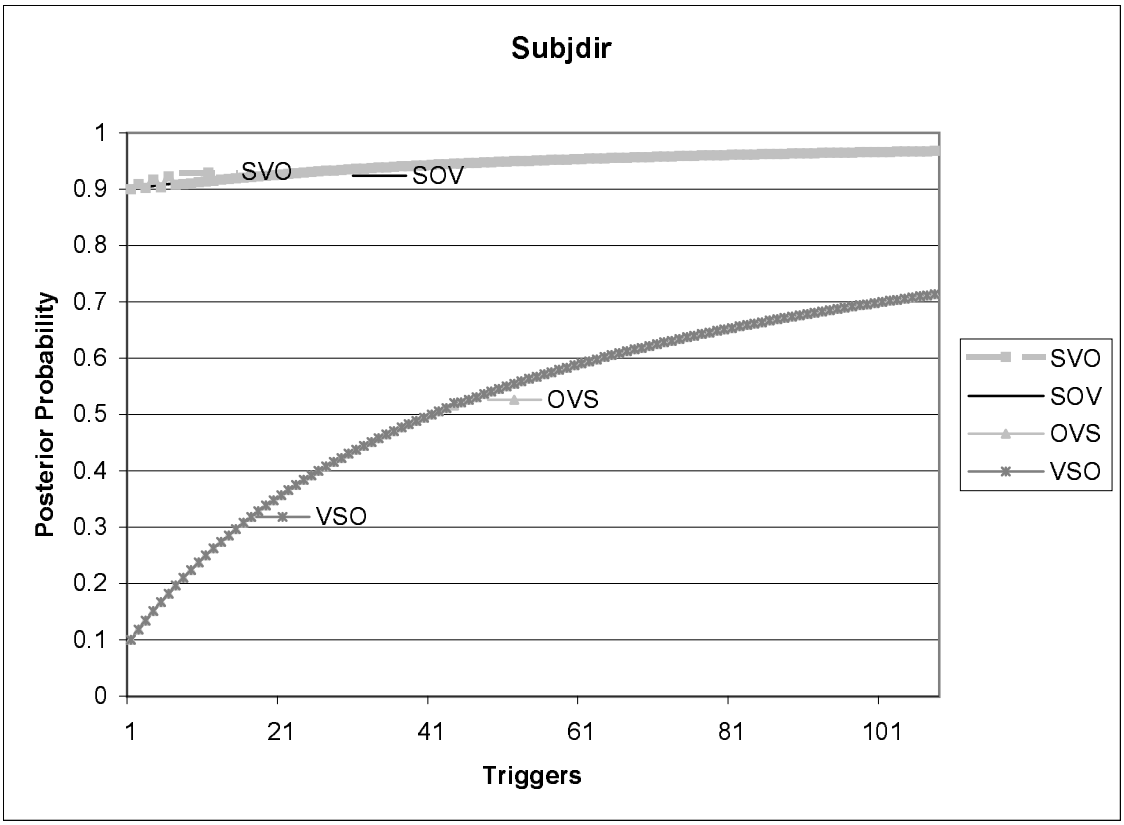

Figure 7.27: Convergence of Subjdir - Learners-50 - Noise-free Environment

Table 7.12: Convergence of the Different Learners - Condition 4

Learners	Triggers	Parameters Correct	Parameters Set
SVO-50	138.5	13.5	8
SOV-50	138.5	13.5	8
OVS-50	138.5	13.5	8
VSO-50	138.5	13.5	8

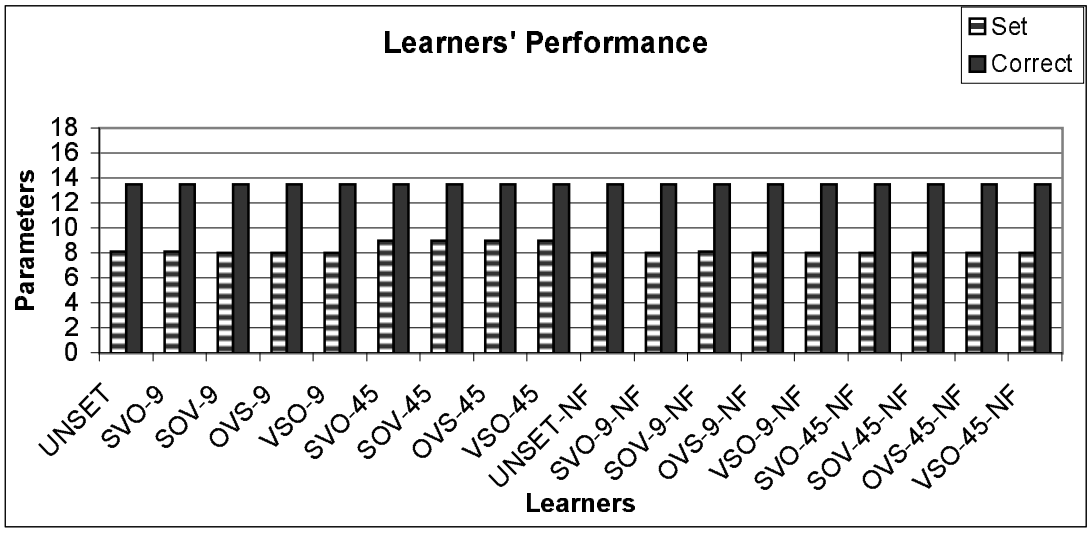

Figure 7.28: Learners' Performances in all the Different Environments

7.3.3 Learning in a Noisy Environment

As suggested by the results obtained in the previous experiments, noise has a significant influence in the learning process. The next set of experiments investigates the influence of noise in the unset learner's convergence pattern. Noise, both in this and in the previous experiments, is related to the directionality specifications, and this corresponds to the cases where the learner is uncertain about the thematic roles expressed in a sentence. For example, in the sentence:

• (7.8) *John kisses Mary*,

the learner may be uncertain as to whether it is *John* that is doing the kissing action, where the logical form is:

{kiss(1 , 2 , 3), john(4 , 2), mary(5 , 3)},

or whether it is *Mary* and the logical form is:

$\{\text{kiss}(\boxed{1}, \boxed{2}, \boxed{3}), \text{mary}(\boxed{4}, \boxed{2}), \text{john}(\boxed{5}, \boxed{3})\}$.

In the first case, the appropriate syntactic category for the verb has the NP subject to the left, in $S/NP_o\backslash NP_s$, expressing an SVO language, while in the second the NP subject is to the right of the verb, whose category is $S\backslash NP_o/NP_s$, expressing an OVS language.

If the learner chooses the second option, and the target language is English, this sentence-logical form pair is going to provide noisy triggers for the learner, giving evidence for another word order that does not correspond to the target SVO. This experiment concentrates on how the convergence pattern of the learner is going to be affected when faced with different degrees of noise introduced by an uncertainty about thematic roles. In this case, these are incorrect triggers for the position of the subject and the object, and they reinforce the non-target resulting order OVS.

In this set of experiments, the unset learner is exposed to increasing levels of noise, ranging from 10% to 50%. Sentences from the corpus are randomly selected according to the appropriate noise level and the noise is generated by changing the directions of the subject and object of the verb, reinforcing non-target values. In this way, when the nosie level is 50%, it means that 50% of the sentences were selected and were made noisy. Table 7.13 shows the performance of the unset learner in each of these noisy environments. As indicated by these results, the learner is robust and has a similar performance in all these environments, even when half of the processed sentences have noise, as shown in figure 7.29. However, the learner's speed of convergence is strongly affected by the noise, as can be seen in figure 7.30 for the **subjdir** parameter in each of these environments. There is a significant difference in the speed of convergence that becomes slower as the amount of noise is increased. There is a difference of up to 45% between the noise-free case and the 50% noise case. The similar performances among the learners are obtained due to the Bayesian framework which ensures that even if a proportion of noisy triggers occurs during the learning process, it will not be enough for the learner to incorrectly set its parameters permanently. Even if the learner sets a given parameter to the non-target value due to the noise, as long as there is a larger proportion of correct triggers, the learner will be able to overcome this problem and re-set the parameter to the target value. In any case, the learner is conservative and waits to set a parameter until a considerable amount of triggers consistently gives evidence for one of the values. This ability to deal with noise provided by the Bayesian framework proves to be very effective in the learning process. This is an important characteristic also in terms of human language acquisition, since it is likely to be the case that children learning a language will be exposed to noise and ambiguity throughout the acquisition

process. As a consequence, they have to be able to deal with these problems and converge to their target languages. Moreover, the environments to which the learner is exposed are extremely noisy, with up to half of the triggers providing evidence for another non-target language. However, even though the noise is a burden to the learner, it does not prevent the learner from eventually setting the parameters correctly, and converging to the target.

Table 7.13: Convergence of the Learner with Different Levels of Noise

Noise Levels	Triggers	Parameters Correct	Parameters Set
10%	280.5	13.5	9
20%	280.5	13.5	9
30%	280.5	13.5	9
40%	245.8	13.5	9
50%	263.25	13.5	9

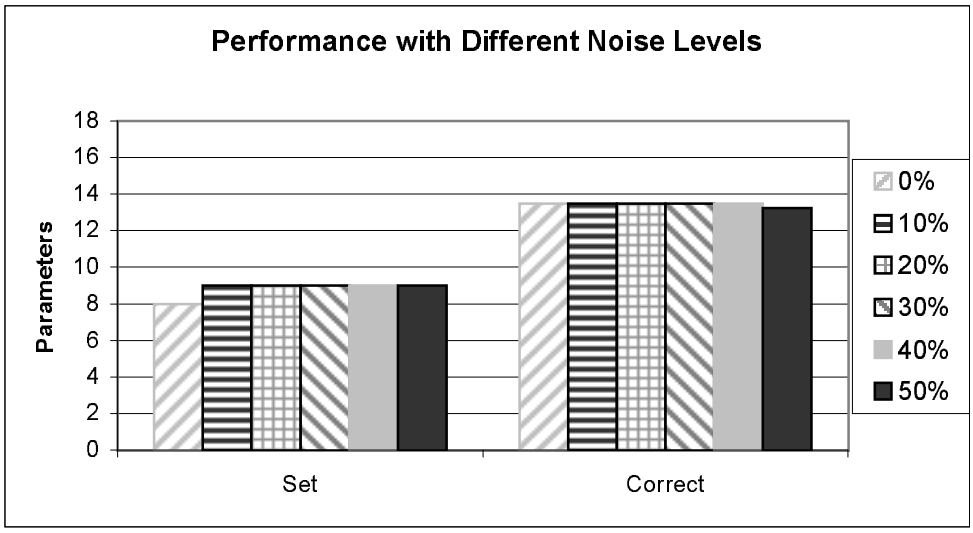

Figure 7.29: Learner's Performance with Different Levels of Noise

Discussion

The results obtained indicate that the different initialisations cause little impact in the learners' performances, in spite of delaying the convergence of those learners that have values incompatible with the target. However, when combining the

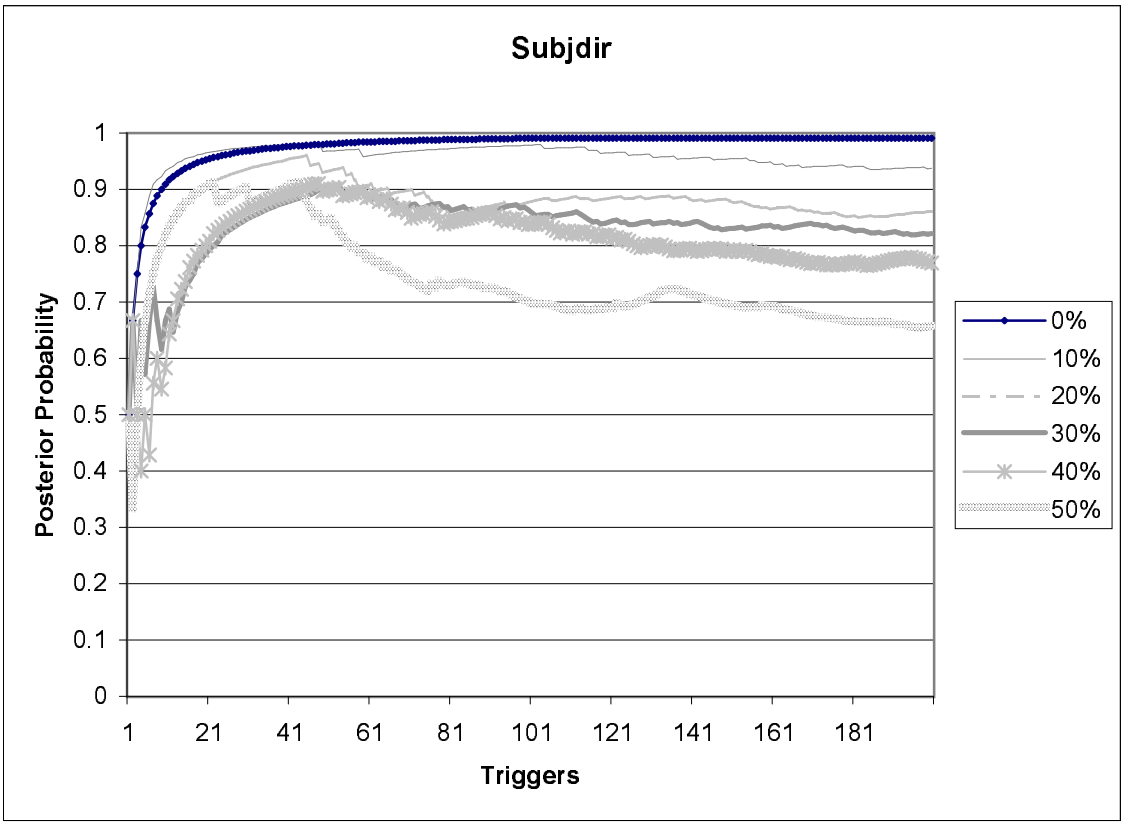

Figure 7.30: Convergence of Subjdir with Different Levels of Noise

presence of noise with the use of stronger weights, there is a significant delay in convergence, where the final posterior probability is up to 10% lower than in the noise-free case (e.g. for the OVS learner), as can be seen in figures 7.23 and 7.27. The noise has a strong influence on the convergence of the learners, slowing down the learning process, since the learners need more triggers to compensate for the effect caused by the noisy ones. There is a difference of up to 45% between the noise-free case and the 50% noise case for the unset learner. This is a considerable difference. Nonetheless, these learners are robust, only selecting or changing a value for a given parameter when there is strong evidence for that. As a consequence, all the learners are converging towards the target, even with the small amount of available triggers, regardless of the initialisations and the presence of noise. This is the case even with an extreme bias in the initial values (section 7.3.2), and in the presence of an extremely noisy environment (section 7.3.3). Moreover, the learners make effective use of the inheritance mechanism to propagate default values in all the conditions.

7.4 Summary

In this chapter some of the tasks that the learning system has to undertake during the learning process were discussed. The learning system starts with an incomplete grammar that has to be extended for the learner to be able to successfully process more complex constructions. During this process, several difficulties are encountered, and the specific case caused by the ambiguity of locative PPs was discussed. An approach to overcome this problem was proposed, which when adopted by the learner proved to be effective and helped the learner decide the appropriate case for the ambiguities found in the data available. Further tests would be necessary to analyse how well the approach would perform given a larger amount of data and more verbs, but from the data gathered so far the proposed approach can successfully deal with this ambiguity.

The learner uses the disambiguated input data to set the parameters of the UG, and several models of learners were evaluated. The evaluation considered the unset learner, and four models of default learners. They were all equally effective in the task of acquiring word order, converging to the target grammar given the data available. The difference between them was only in terms of the speed of convergence, which was faster for those learners compatible or partially compatible with the target. Nevertheless all of them were converging to the target, regardless of the initialisations. This is the case even in the presence of noise, which was investigated by verifying the performance of the unset learner in environments which provided increasing levels of noise. In spite of the slower convergence observed when the noise level was increased, even when half of the triggers processed were noisy, the learner successfully set its parameters to the target values. Such a learner is robust to noise and ambiguity, which are likely to be part of the environment where children acquire their language, and it can be used to investigate further aspects of language acquisition and potentially help to shed light on aspects of human language acquisition.

Chapter 8

Conclusions and Future Work

8.1 Results and Achievements

The purpose of this research was to investigate the process of grammatical acquisition from a computational perspective, employing some ideas from Psychology and Linguistics about human language acquisition. In order to conduct this investigation, a computational learning system, which is equipped with a model of Universal Grammar and a learning algorithm, was implemented. Such a system provides a robust and effective account of parameter setting, using a suitable model of the UG, in a plausible parameter space and learning from a corpus of parents' spontaneous sentences to a child.

In this work we implemented a Unification-Based Generalised Categorial Grammar for English that not only served to annotate the corpus with logical forms, but was also used as the target grammar to which the learner needed to converge. The grammar was implemented as a default inheritance network of lexical types which resulted in a significant reduction in lexical redundancy, in an encoding that is able to describe linguistic regularities, sub-regularities and exceptions, defined by means of typed-default feature structures. The resulting lexicon is succinctly organised and also easier to maintain and modify. This grammar is able to capture successfully a wide range of constructions, including several different coordination structures and unbounded dependencies.

The proposed grammatical encoding is well suited to implementing a theory of Universal Grammar and associated principles and parameters. The UG is represented as a UB-GCG and is embedded in a default inheritance hierarchy. This is a clear and concise way of defining the UG with the parameters being straightforwardly defined in the categories, in a way that makes effective use of the default inheritance mechanism to propagate information about parameters throughout the lexical inheritance network. The use of a default inheritance schema to implement the UG reduced the amount of information to be acquired by the learner, since the information was represented in a structured way. Thus

what is learned is not a single isolated item, but a whole structure that represents a candidate category set and can be propagated through the hierarchies by the inheritance mechanism. Thus the learner can use the existing hierarchies to classify new information learned and needs only to encode the minimum amount of arbitrary information. The proposed model of UG is consistent with typological and linguistic ideas about the space of possible human languages. This approach is well suited for a Principles and Parameters Theory of UG, with very general grammar rules and categories defined as types arranged in a default inheritance hierarchy, a kind of structure that is likely to have an important role in the way people organise many kinds of information.

In order to provide a more realistic linguistic environment for parameter setting, a subset of the Sachs corpus, containing interactions between a child and her parents, was annotated with logical forms. The English UB-GCG was used to parse and annotate a subset of the Sachs corpus consisting of the parents' sentences, with the appropriate logical forms. This corpus of child-directed transcribed speech contains a wide variety of constructions and allows us to simulate the environment in which the child acquired her native language.

The linguistic environments to which children are exposed include a variety of dialects and possibly more than one language, but children are robust to this variety of influence and successfully learn those languages and dialects to which they are consistently exposed. Sentences are produced in linguistic and non-linguistic contexts and these are potential sources of information that children can plausibly employ during the learning process, to minimise the problem of the poverty of stimulus. The linguistic context was partially recreated through the annotation of each sentence with semantic information. These environments also exhibit statistical properties, which are a potentially useful source of information to help learners in the task of acquiring languages on the basis of ambiguous and noisy data. In the implemented learning system, the learner made use of statistical information for dealing with certain ambiguities in the data, such as that caused by locative Prepositional Phrases as arguments or adjuncts. To resolve this ambiguity, the learner used frequency information about verbs and locative PPs and also the idea of semantically motivated preposition selection. This approach uses information that human language learners potentially use, being a plausible mechanism for solving this kind of ambiguity. Another potential use of statistical information lies in helping human language learners deal with noise in their environment. This is incorporated in the setting of the parameters of the UG in the learning system. The learner uses statistical information to set the word order parameters, collecting evidence to set each parameter and only changing the corresponding value when there is enough support in the data for the move.

The use of categorial parameters also provides the learning system with an incremental approach for the learning of subcategorisation frames, where the learner is able to process categories and thus sentences, increasingly more com-

plex, given the bias of obtaining the most concise encoding consistent with the data. Initially the learner has a restricted set of categories that it can process but, as learning progresses, this set is expanded, with categories learned as required by the input data.

Such a learning model presents a computationally tractable possibility for modelling language acquisition, compatible with psychological and linguistic ideas about human language acquisition. The learning system defined is a general computational framework that provided the possibility of exploring putative paths to language acquisition. The learning system was used to investigate certain possibilities in the acquisition of a language, providing the necessary conditions for performing computational simulations of the learning process. Different characterisations of learners were implemented in this framework and some aspects of language acquisition were investigated, concentrating on the starting point for learning and on the presence of noise in the input data. These conditions provided extreme environments for learning, but in all cases, the learners converged towards the target grammar. The different starting points and the presence of noise affected only convergence times, with learners further from the target having a slower convergence pattern. The learners were shown to be effective in the language learning tasks defined, converging in the direction of the target, given the data available. These learners were conservative and only changed a parameter after they collected enough evidence for such change. In this way, the learners were able to deal with noise and still successfully set their parameters according to the target even under unfavourable conditions.

As a result of implementing these ideas, we obtained a basic framework that allows us, among other things, to perform experiments in language acquisition: there is a plausible model of a UG with associated parameters, an English UBGCG implementing several insights into the encoding of linguistic information, a corpus of child-directed sentences annotated with logical forms, based on which the parameters are set and an implementation of a Bayesian learning system, that learns in a unification-based framework. A computational learning system like this can provide the grounds for testing the plausibility and requirements of theories about language acquisition. Thus when a theory under investigation does not provide the necessary conditions for a computational learner to converge to the target, given a reasonable amount of data, then this theory is unlikely to play an important role in human language acquisition. On the other hand, a theory that can be successfully modelled by the learning system and that allows the learner to converge to the target, may not reflect the way human language acquisition occurs; it only highlights possible paths to follow. With the computational study of language acquisition, we expect to gain a better understanding of the human language acquisition process.

The development of learning systems, such as that implemented in the scope of this thesis, that can automatically learn linguistic information from data, opens several possibilities for NLP systems that have traditionally been developed as

static and inflexible systems. The need for more adaptive technology that constantly evolves with its environment is increasing, with the necessity of dealing with larger and larger amounts of data, which may contain noise and novel and flexible uses of languages. Although ours is primarily a cognitive computational model, it is potentially relevant to the development of more adaptive NLP technology.

In summary, there are seven main contributions of this enquiry. First of all, in terms of grammar development, we integrate several ideas on the use of defaults in linguistic description, as well as proposing novel uses of defaults, as discussed in chapter 4. These are implemented in a Unification-Based Generalised Categorial Grammar defined in terms of a default inheritance network of lexical types.

Secondly, we propose a plausible model of the Universal Grammar based on typological and linguistic studies, as presented in section 4.5. We represent it using UB-GCG, which is a formalism that has very general rules and is suited to a description of the UG. In the proposed UG, the parameters are integrated in the grammatical formalism, interacting with the categories and rules in the grammar. The UG and associated parameters are embedded in a default inheritance network of types, in a structure that is likely to have an important role in the way people organise different kinds of information.

Thirdly, the definition of such a model of the UG embedded in a default inheritance network of types is well suited to learning purposes, since the learner can make use of the information already represented in the grammar when acquiring new information. This reduces the amount of information to be learned, because when learning a new structure, the learner does not need to learn again the parts of this new structure that are already encoded in the grammar. Thus, even if the learner sets only a few parameters, the others will automatically have the inherited value, even before being set by triggers. This considerably speeds up the convergence to the target.

Fourthly, learning is defined in terms of a unification-based representation of the UG, which allows featural variation, with categories, rules and parameters all described in terms of feature structures. Such a grammatical representation is suited to capturing natural languages and is consistent with recent developments in computational linguistics [Grover et al. 1993], [Sag and Wasow 1999], [Bouma et al. in press].

Another contribution is to define learning as based on principles, such as the Categorial Principles that were defined in CGs for independent reasons. Moreover, through the use of default inheritance hierarchies and the MDL principle, the learning systems was able to straightforwardly learn several aspects of language, such as the linking between subcategorisation frames and predicate argument structures. Finally, the learning system implemented can successfully learn from a corpus of real child-directed data, dealing with noise and ambiguity, in a computational account of parameter setting that is compatible with several characteristics of human language acquisition.

8.2 Future Work

Future work includes annotating more data to secure a bigger corpus, with triggers for all the parameters. This larger corpus would allow more experiments to be conducted, testing how much data is required for all the triggers to converge with high probability to the target grammar.

In this work, a sentence was only used to set the parameters of the UG if the semantics for all the words in the sentence could be determined. This research can be extended by investigating how the learner would be affected if the semantic information it receives from the semantics learner were incomplete, with sentences containing words for which no semantics could be determined. The learner could benefit from using those words whose semantics could be determined to set the appropriate parameters and this would mean more triggers for the parameters expressed in this data, since such sentences were ignored. However, the question is what the learner should do with the words with no semantics. One possible solution is for the learner to try to approximate their meaning, given the information that it may already have collected for that word. In preliminary experiments we performed, the learner made use of the information (such as the semantic and syntactic category) it had already collected about the words seen so far, so that if a given word had no semantic information assigned, it used the information it had stored about previous occurrences of the word. The results indicated that if the word for which the semantics could not be determined had the same part-of-speech that the one stored by the learner from previous sentences with that word even if in a different context, the learner could minimise the problem caused. For instance, if the semantics for *eat* could not be determined for a sentence such as *He ate apples*, where it is a transitive verb, but the learner has information stored from sentences containing *eat* as an intransitive verb, the learner can use the information stored at least to have a better understanding of the sentence: that *He ate* and that there were *apples* involved in the eating act. A more thorough investigation needs to be conducted.

It is also necessary to have more resources to investigate certain aspects of language learning, such as developmental stages. It would be desirable to have the complete parents' corpus annotated, to investigate how the developmental aspects of learning would be captured by such a model. Certain phenomena in human language acquisition occur at a later stage than others and the constructions that trigger the learning of these phenomena need to be identified in the data. Some work has already been done in the analysis of a small number of triggers for specific languages [Hyams 1986], but larger scale studies are still needed. The question of the frequency of triggers should also be addressed, with investigations of the frequency of possible triggers in child-directed data. If a learning model requires a certain number of triggers for setting its parameters and it is verified that the data to which children are exposed would never provide the required number, then this finding casts doubt on the plausibility of

such a model. Moreover, a detailed analysis of the child data and the learning stages reflected in this data is also needed. Then a proper comparative study could be undertaken of the developmental stages of the child during acquisition and those of the learning system. A further development is the annotation of a corpus of another language, preferably with a word order different from that of English, based on which cross-linguistic comparative studies could be performed, in relation to language acquisition in this framework. The question of learning in a bilingual environment deserves attention and the implemented learning model could be employed for such a task.

In summary, a better understanding of the activities and resources employed in human language acquisition is necessary for the construction of more accurate models of language acquisition.

Bibliography

[Adriaans 1992] Adriaans, P. *Language Learning From a Categorial Perspective.* Academish proefschrift, Universiteit van Amsterdam, 1992.

[Adriaans and de Haas 2000] Adriaans, P. and de Haas, E. Grammar Induction as Substructural Inductive Logic Programming. *Learning Language in Logic.* J. Cussens and S. Dzeroski eds., p. 127-142, Springer, 2000.

[A$\ddot{\text{i}}$t-Kaci 1984] A$\ddot{\text{i}}$t-Kaci, H. *A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures.* Ph.D. Thesis, University of Pennsylvania, 1984.

[Ajdukiewicz 1935] Ajdukiewicz, K. *Die Syntaktische Konnexitat Polish Logic 1920-1939* Storrs McCall ed., p. 207-231, Oxford University Press, 1935. Translated from *Studia Philosophica*, v. 1, p. 1-27.

[Atkinson 2001] Atkinson, M. Learnability and the Acquisition of Syntax. *Language Acquisition and Learnability.* S. Bertolo ed., p. 15-80, Cambridge University Press, 2001.

[Bar Hillel 1953] Bar Hillel, Y. *A Quasi-Arithmetical Notation for Syntactic Description.* Language, v. 29, n. 1, p. 47-58, 1953.

[van Benthem 1988] van Benthem, J. The Lambek Calculus. *Categorial Grammars and Natural Language Structures.* R. Oehrle, E. Bach and D. Wheeler eds., p. 35-68, Dordrecht:Reidel, 1988.

[Bertolo 2001] Bertolo, S. A Brief Overview of Learnability. *Language Acquisition and Learnability.* S. Bertolo ed., p. 1-14, Cambridge University Press, 2001.

[Berwick 1985] Berwick, R. *The Acquisition of Syntactic Knowledge.* MIT Press, 1985.

[Berwick and Niyogi 1996] Berwick, R. and Niyogi, P. *Learning from Triggers.* Linguistic Inquiry, v. 27, n. 4, p. 605-622, 1996.

[Blumer et al. 1987] Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M. K. *Occam's Razor.* Information Processing Letters, v. 24, n.6, p. 377-380, 1987.

[Bouma 1990] Bouma, G. *Defaults in Unification Grammar.* Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL), p. 165-173, Pittsburg, Pennsylvania, 1990.

[Bouma 1992] Bouma, G. Feature Structures and Nonmonotonicity. *Computational Linguistics*, v. 18, n. 2, p. 183-203, 1992.

[Bouma et al. in press] Bouma, G., Flickinger, D. and van Eynde, F. Constraint-Based Lexicons. *Lexicon Development for Speech and Language Processing.* F. van Eynde ed., ELSNET, Leuven.

[Brent 1996] Brent, M.R. *Advance in the Computational Study of Language Acquisition.* Cognition, v. 61, n. 1-2, p. 1-38, 1996.

[Briscoe 1993] Briscoe, T. Introduction. *Inheritance, Defaults and the Lexicon.* T. Briscoe, A. Copestake and V. de Paiva eds., p. 1-12, Cambridge University Press, 1993.

[Briscoe 1997] Briscoe, T. *Co-Evolution of Language and the Language Acquisition Device.* Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL), p. 418-427, Madrid, Spain, 1997.

[Briscoe 1998] Briscoe, T. Language as a Complex Adaptive System: Coevolution of Language and of the Language Acquisition Device. *Proceedings of 8th Computational Linguistics in the Netherlands.* P. Coppen, H. van Halteren, and L. Teunissen eds., p. 3-40, Rodopi, 1998.

[Briscoe 1999] Briscoe, T. The Acquisition of Grammar in an Evolving Population of Language Agents. *Machine Intelligence, 16: Electronic transactions in Artificial Intelligence, Special Issue.* S. Muggleton ed. In http://www.etaij.org/ 1999.

[Briscoe 2000] Briscoe, T. *Grammatical Acquisition: Inductive Bias and Coevolution of Language and the Language Acquisition Device.* Language, v. 76, n. 2, p. 245-296, 2000.

[Briscoe in press] Briscoe, T. Grammatical Acquisition and Linguistic Selection. *Language Acquisition and Linguistic Evolution: Formal and Computational Approaches* T. Briscoe ed., Cambridge University Press.

[Briscoe and Copestake 1999] Briscoe, T. and Copestake, A. *Lexical Rules in Constraint-Based Grammar.* Computational Linguistics, v. 25, n. 4, p. 487-526, 1999.

[Buszkowski 1987] Buszkowski, W. Discovery Procedures for Categorial Grammars. *Categories, Polymorphism and Unification.* E. Klein and J. van Benthem eds., p. 35-64, Centre for Cognitive Science, University of Edinburgh, and Institute for Logic, Language and Information, University of Amsterdam, 1987.

[Buszkowski and Penn 1990] Buszkowski, W. and Penn, G. *Categorial Grammars Determined from Linguistic Data by Unification.* Studia Logica, v. 49, p. 431-454, 1990.

[Carpenter 1990] Carpenter, B. *Typed Feature Structures: Inheritance, (In)equalities and Extensionality.* Proceedings of the First International Workshop on Inheritance in Natural Language Processing, p. 9-18, Tilburg, The Netherlands, 1990.

[Carpenter 1991] Carpenter, B. *The Generative Power of Categorial Grammars and Head-Driven Phrase Structure Grammar with Lexical Rules*, Computational Linguistics, v. 17, n. 3, p. 301-313, 1991.

[Carpenter 1992] Carpenter, B. *The Logic of Typed Feature Structures.* Cambridge University Press, 1992.

[Carpenter 1993] Carpenter, B. Skeptical and Credulous Default Unification with Applications to Templates and Inheritance *Inheritance, Defaults and the Lexicon.* T. Briscoe, A. Copestake and V. de Paiva eds., p. 13-37, Cambridge University Press, 1993.

[Carpenter 1998] Carpenter, B. *Type-Logical Semantics.* MIT Press, 1998.

[Chen 1996] Chen, S. *Building Probabilistic Language Model for Natural Language.* Ph.D. Thesis, Harvard University, 1996.

[Chomsky 1965] Chomsky, N. *Aspects of the Theory of Syntax.* MIT Press, 1965.

[Chomsky 1981] Chomsky, N. *Lectures on Government and Binding.* Foris Publications, 1981.

[Chomsky 1982] Chomsky, N. *Some Concepts and Consequences of the Theory of Government and Binding.* MIT Press, 1982.

[Clark 1992] Clark, R. *The Selection of Syntactic Knowledge.* Language Acquisition, v. 2, n. 2, p. 83-149, 1992.

[Clark 2001] Clark, R. Information Theory, Complexity and Linguistic Descriptions *Language Acquisition and Learnability.* S. Bertolo ed., p. 126-171, Cambridge University Press, 2001.

[Comrie 1981] Comrie, B. *Language Universals and Linguistic Typology.* University of Chicago Press, 1981.

[Copestake et al. 1998] Copestake, A., Flickinger, D. and Sag, I. *Minimal Recursion Semantics: An introduction.* ESSLLI 98, 1998.

[Copestake 2002] Copestake, A. *Implementing Typed Feature Structure Grammars*, 2002. CSLI Publications, 2002.

[Croft 1992] Croft, W. *Typology and Universals.* Cambridge University Press, 1992.

[Culicover 1997] Culicover, P.W. *Principles and Parameters - An Introduction to Syntactic Theory.* Oxford University Press, 1997.

[Daelemans et al. 1992] Daelemans, W., De Smedt, K. and Gazdar, G. *Inheritance in Natural Language Processing.* Computational Linguistics, v. 18, n. 2, p. 205-218, 1992.

[Davidson 1967] Davidson D. The Logical Form of Action Sentences. *The Logic of Decision and Action.* N. Rescher ed., p. 81-210, University of Pittsburgh Press, 1967.

[Davis 1996] Davis, A. *Linking and the Hierarchical Lexicon.* Ph.D. Thesis, Stanford University, 1996.

[Doran and Srinivas in press] Doran, C. and Srinivas, B. *Developing a Wide-Coverage CCG System.* In CSLI Lecture Notes, Cambridge University Press. In http://www.itri.bton.ac.uk/∼Christy.Doran/Pubs/ccg-csli.ps.gz.

[Dowty 1982] Dowty D. Grammatical Relations and Montague Grammar. *The Nature of Syntactic Representation* P. Jacobson and G.K. Pullum eds., p. 79-130, Dordrecht, 1982.

[Dresher and Kaye 1990] Dresher, E. and Kaye, J.D. *A Computational Learning Model for Metrical Phonology.* Cognition, v. 34, n. 2, p. 137-195, 1990.

[Evans et al. 1993] Evans, R., Gazdar, G. and Moser, L. Prioritised Multiple Inheritance in DATR. *Inheritance, Defaults and the Lexicon.* T. Briscoe, A. Copestake and V. de Paiva eds., p. 38-46, Cambridge University Press, 1993.

[Flickinger et al. 1985] Flickinger, D., Pollard, C. and Wasow, T. *Structure Sharing in Lexical Representation.* Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics (ACL), p. 262.268, University of Chicago, 1985.

[Flickinger and Nerbonne 1992] Flickinger, D. and Nerbonne, J. *Inheritance and Complementation: A Case Study of* easy *Adjectives and Related Nouns.* Computational Linguistics, v. 18, n. 3, p. 269-310, 1992.

[Fodor 1998] Fodor, J.D. *Unambiguous Triggers.* Linguistic Inquiry, v. 29, n. 1, p. 1-36, 1998.

[Frank and Kapur 1996] Frank, R. and Kapur, S. *On the Use of Triggers in Parameter Setting.* Linguistic Inquiry, v. 27, n. 4, p. 623-660, 1996.

[Gazdar 1981] Gazdar, G. *Unbounded Dependencies and Coordinate Structure.* Linguistic Inquiry, v. 12, n. 2, p. 155-184, 1981.

[Gazdar 1987] Gazdar, G. Linguistic Applications of Default Inheritance Mechanisms. *Linguistic Theory and Computer Applications.* P. Whitelock, M. M. Wood, H. L. Somers, R. Johnson and P. Bennet eds., p. 37-68, Academic Press, London, 1987.

[Gibson and Wexler 1994] Gibson, E. and Wexler, K. *Triggers.* Linguistic Inquiry, v. 25, n. 3, p. 407-454, 1994.

[Gleitman and Wanner 1982] Gleitman, L. and Wanner, E. The State of the State of the Art. *Language Acquisition: the State of the Art.* E. Wanner and L. Gleitman eds., Cambridge University Press, 1982.

[Gleitman and Landau 1994] Gleitman, L. and Landau, B. Preface to Lingua 92. *The Acquisition of Lexicon, Lingua 92.* L. Gleitman and B. Landau eds., p. 1-4, MIT Press, 1994.

[Gold 1967] Gold, E.M. *Language Identification in the Limit.* Information and Control, v. 10, n. 5, p. 447-474, 1967.

[Goldberg 1995] Goldberg, A. *Constructions.* Chicago University Press, 1995.

[Grover et al. 1993] Grover, C., Carroll, J. and Briscoe, T. *The Alvey Natural Language Tools Grammar.* Technical Report No. 284, University of Cambridge, 1993.

[Hockenmaier et al. 2000] Hockenmaier, J., Bierner, G. and Baldridge *Providing Robustness for a CCG System.* Proceedings of the Workshop on Linguistic Theory and Grammar Implementation, p. 97-112, Birmingham, 2000.

[Hoffman 1995] Hoffman, B. *The Computational Analysis of the Syntax and Interpretation of 'Free' Word Order in Turkish.* Ph.D. Thesis, University of Pennsylvania, 1995.

[Hudson 1990] Hudson, R. *English Word Grammar*, Blackwell, 1990.

[Hyams 1986] Hyams, N. *Language Acquisition and the Theory of Parameters.* Reidel, Dordrecht, 1986.

[Ingram 1992] Ingram, D. *First Language Acquisition: Method, Description and Explanation.* Cambridge University Press, 1992.

[Kanazawa 1998] Kanazawa, M. *Learnable Classes of Categorial Grammars.* CSLI Publications and folli, 1988.

[Kapur and Clark 1996] Kapur, S. and Clark, R. The Automatic Construction of a Symbolic Parser via Statistical Techniques. *The Balancing Act: Combining Symbolic and Statistical Approaches to Language.* J. Klavans and P. Resnik eds., p. 85-118, MIT Press, 1996.

[Kohl 1999] Kohl, K. *An Analysis of Finite Parameter Learning in Linguistic Spaces.* S.M. Thesis, MIT, 1999.

[Krieger and Nerbonne 1993] Krieger, H. and Nerbonne, J. Feature-Based Inheritance Networks for Computational Lexicons. *Inheritance, Defaults and the Lexicon.* T. Briscoe, A. Copestake and V. de Paiva eds., p. 90-136, Cambridge University Press, 1993.

[Lambek 1958] Lambek. J. *The Mathematics of Sentence Structure.* American Mathematical Monthly, v. 65, n. 3, p. 154-170, 1958.

[Landau and Gleitman 1985] Landau, B. and Gleitman, L.R. *Language and Experience.* Harvard University Press, 1985.

[Langley and Carbonell 1987] Langley, P. and Carbonell, J.G. Language Acquisition and Machine Learning. *Mechanisms of Language Acquisition.* B. MacWhinney ed., p. 115-155, Lawrence Erlbaum Associates, 1987.

[Lascarides et al. 1996a] Lascarides, A., Copestake, A, Briscoe, T. Ambiguity and Coherence. *Journal of Semantics*, v. 13, n. 1, p. 41-65, 1996.

[Lascarides et al. 1996b] Lascarides, A., Briscoe, T., Asher, N., and Copestake, A. *Order Independent Persistent Typed Default Unification.* Linguistics and Philosophy, v. 19, n. 1, p. 1-89, 1996.

[Lascarides and Copestake 1999] Lascarides, A. and Copestake, A. Default Representation in Constraint-based Frameworks. *Computational Linguistics*, v. 25, n. 1, p. 55-105, 1999.

[Levin 1993] Levin, B. *English Verb Classes and Alternations.* University of Chicago Press, 1993.

[Li and Vitanyi 1993] Li, M. and Vitanyi, P.M.B. *An Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, 1993.

[Li and Vitanyi 1995] Li, M. and Vitanyi, P.M.B. *Computational Machine Learning in Theory and Praxis*. NeuroCOLT Technical Report Series, NC-TR-95-052, University of London, 1995.

[Light 1993] Light, M. *Classification in Feature-based Default Inheritance Hierarchies*. Technical Report 473, University of Rochester, 1993.

[Lightfoot 1991] Lightfoot, D. *How to Set Parameters: Arguments from Language Change*. Bradford Book, 1991.

[MacDonald 1999] MacDonald, M. Distributional Information in Language Comprehension, Production, and Acquisition: Three Puzzles and a Moral. *The Emergence of Language.* B. MacWhinney ed., p. 177-196, Lawrence Erlbaum Associates, 1999.

[MacWhinney 1995] MacWhinney, B. *The CHILDES Project: Tools for Analyzing Talk*. Second Edition. Lawrence Erlbaum Associates, 1995.

[Malouf 1998] Malouf, R. *Mixed Categories in the Hierarchical Lexicon.* Ph.D. Thesis, Stanford University, 1998.

[Marcus 1993] Marcus, G. *Negative Evidence in Language Acquisition.* Cognition, v. 46, n. 1, p. 53-85, 1993.

[Moens et al 1989] Moens, M., Calder, J., Klein, E., Reape, M. and Zeevat, H. *Expressing Generalizations in Unification-Based Grammar Formalisms*, Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics (EACL), p. 66-71, Manchester, England, 1989.

[Moortgat 1988] Moortgat, M. *Categorial Investigations - Logical and Linguistic Aspects of the Lambek Calculus.* Foris Publications, 1988.

[Morrill 1987] Morrill, G. Meta-Categorial Grammar. *Categorial Grammar, Unification Grammar, and Parsing: Working Papers in Cognitive Science.* N. Haddock, E. Klein and G. Morrill eds., v. 1, p. 1-29, Centre for Cognitive Science, University of Edinburgh, 1987.

[Murgatroyd 2000] Murgatroyd, D. *Exploiting Prosodic Phrasing in Parsing*, MPhil dissertation, University of Cambridge, 2000.

[Osborne and Briscoe 1997] Osborne, M. and Briscoe, T. *Learning Stochastic Categorial Grammars*. Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL), Computational Language Learning (CoNLL97) Workshop, p. 80-87, Madrid, Spain, 1997.

[Parsons 1980] Parsons, T. *Event in the Semantics of English - A Study in Subatomic Semantics.* MIT Press, 1980.

[Pinker 1984] Pinker, S. *Language Learnability and Language Development.* Harvard University Press, 1984.

[Pinker 1994] Pinker, S. *The Language Instinct.* Penguin Books, 1994.

[Pinker 1995] Pinker, S. The Language Acquisition, *An Invitation to Cognitive Science, Language* L.R. Gleitman and M. Liberman eds., v. 1, p. 135-183, MIT Press, 1995.

[Pollard and Sag 1987] Pollard, C. and Sag, I.A. *Information-Based Syntax and Semantics*, CSLI Lecture Notes Series, n. 13, 1987.

[Pollard and Sag 1994] Pollard, C. and Sag, I.A. *Head-driven Phrase Structure Grammar*, Chicago University Press, 1994

[Pustejovsky 1991] Pustejovsky, J. *The generative lexicon.* Computational Linguistics, v. 17 n. 4, p. 409-441, 1991.

[Pustejovsky 1995] Pustejovsky, J. *The Generative Lexicon.* MIT Press, 1995.

[Quinlan and Rivest 1989] Quinlan, J.R. and Rivest, R.L. *Inferring Decision Trees Using the Minimum Description Length Principle.* Information and Computation, v.80, n. 3, p. 227-248, 1989.

[Redington and Chater 1998] Redington, M. and Chater, N. *Connectionist and Statistical Approaches to Language Acquisition: A Distributional Perspective.* Language and Cognitive Processes, v. 13, n. 2/3, p. 129-191, 1998.

[Rissanen 1989] Rissanen, J. *Stochastic Complexity in Statistical Inquiry.* Volume 15 of Series in Computer Science, World Scientific, 1989.

[Roeper 1992] Roeper, T. From the Initial State to V2: Acquisition Principles in Action. *The Acquisition of Verb Placement: Functional Categories and V2 Phenomena in Language Acquisition.* J.M. Meisel ed., p. 333-370, Kluwer Academic Publishers, 1992.

[Sachs 1983] Sachs, J. Talking About the There and Then: the Emergence of Displaced Reference in Parent-Child Discourse. *Children's language, Volume 4.* K. E. Nelson ed., p. 1-28, Lawrence Erlbaum Associates, 1983.

[Sag and Wasow 1999] Sag, I.A. and Wasow, T. *Syntactic Theory - A Formal Introduction.* CSLI Publications, 1999.

[Sanfilippo 1990] Sanfilippo, A. *Grammatical Relations, Thematic Roles and Verb Semantics.* Ph.D. Thesis, University of Edinburgh, 1990.

[Sanfilippo 1993] Sanfilippo, A. LKB Encoding of Lexical Knowledge. *Inheritance, Defaults and the Lexicon.* T. Briscoe, A. Copestake and V. de Paiva eds., p. 190-222, Cambridge University Press, 1993.

[Sakas 2000] Sakas, W. G. *Modeling the Effect of Cross-Language Ambiguity on Human Syntax Acquisition.* Proceedings of Computational Natural Language Learning (CoNLL-2000) and 2nd Learning Language in Logic (LLL-2000) workshop, p. 61-66, Lisbon, Portugal, 2000.

[Sakas and Fodor 2001] Sakas, W. G. and Fodor, J.D. The Structural Triggers Learner. *Language Acquisition and Learnability.* S. Bertolo ed., p. 172-233, Cambridge University Press, 2001.

[de Saussure 1916] de Saussure, F. *Cours de Linguistique Generale.*, 1916. (Translated as Course in General Linguistics. New York: McGraw-Hill, 1959.)

[Schütze 1995] Schütze, H. *Ambiguity in Language Learning Computational and Cognitive Models.* Ph.D. Thesis, Stanford University, 1995.

[Shieber 1986] Shieber, S. *An Introduction to Unification-Based Approaches to Grammar* CSLI Lecture Notes Series, n. 4, 1986.

[Siskind 1993] Siskind, J.M. *Lexical Acquisition as Constraint Satisfaction.* IRCS Report No. 93-41, 1993.

[Siskind 1996] Siskind, J.M. *A Computation Study of Cross-Situational Techniques for Learning Word-to-Meaning Mappings.* Cognition, v. 61, n. 2, p. 39-91, 1996.

[de Smedt 1984] de Smedt, K. Using Object-Oriented Knowledge Representation Techniques in Morphology and Syntax Programming. *Proceedings of the 6th European Conference on Artificial Intelligence (ECAI-84).* T. O'Shea ed., p. 181-184, Elsevier, 1984.

[Steedman 1985] Steedman, M. *Dependency and Coordination in the Grammar of Dutch and English.* Language, v. 61, n. 3, p. 523-566, 1985.

[Steedman 1988] Steedman, M. Combinators and Grammars. *Categorial Grammars and Natural Language Structures* R. Oehrle, E. Bach and D. Wheeler eds., p. 417-442, Dordrecht: Reidel, 1988.

[Steedman 1990] Steedman, M. *Gapping as Constituent Coordination.* Linguistics and Philosophy, v. 13, n. 2, p. 207-263, 1990.

[Steedman 1991a] Steedman, M. *Type-raising and Directionality in Combinatory Grammar.* Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics (ACL), p. 71.78, Berkeley, 1991.

[Steedman 1991] Steedman, M. *Structure and Intonation.* Language, v. 67, n. 2, 1991.

[Steedman 1996] Steedman, M. *Surface Structure and Interpretation.* MIT Press, 1996.

[Steedman 2000] Steedman, M. *The Syntactic Process.* The MIT Press, 2000.

[Stolcke 1994] Stolcke, A. *Bayesian Learning of Probabilistic Language Models* Ph.D. Thesis, University of California, Berkeley, 1994.

[Uszkoreit 1986] Uszkoreit, H. *Categorial Unification Grammars.* Proceedings of Coling86, p. 187-194, Bonn, 1986.

[Valiant 1984] Valiant, L.G. A Theory of Learnable. Communications of the ACM, v. 27, n. 11, p. 1134-1142, 1984.

[Verspoor 1997] Verspoor, C.M. *Contextually-Dependent Lexical Semantics.* Ph.D. Thesis. University of Edinburgh, 1997.

[Villavicencio 1999] Villavicencio, A. *Representing a System of Lexical Types Using Default Unification.* Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL), p. 261-264, Bergen, Norway, 1999.

[Villavicencio 2000a] Villavicencio, A. *The Acquisition of a Unification-Based Generalised Categorial Grammar.* Proceedings of the Third CLUK Colloquium, p. 59-66, Brighton, 2000.

[Villavicencio 2000b] Villavicencio, A. *The Use of Default Unification in a System of Lexical Types.* Proceedings of the Workshop on Linguistic Theory and Grammar Implementation, p. 81-96, Birmingham, 2000.

[Villavicencio 2000c] Villavicencio, A. *The Acquisition of Word Order by a Computational Learning System.* Proceedings of Computational Natural Language Learning (CoNLL-2000) and 2nd Learning Language in Logic (LLL-2000) workshop, p. 209-218, Lisbon, Portugal, 2000.

[Villavicencio 2000d] Villavicencio, A. *Grammatical Learning Using Unification-Based Generalised Categorial Grammars.* Proceedings of AMLaP'2000, p. 38-39, Leiden, 2000.

[Waldron 1999] Waldron, B. *Learning Grammar from Corpora.* MPhil Thesis. University of Cambridge, 1999.

[Waldron 2000] Waldron, B. *Learning Natural Language within the Framework of Categorial Grammar.* Proceedings of the Third CLUK Colloquium, p. 67-74, Brighton, 2000.

[Watkinson and Manadhar 2000] Watkinson, S. and Manandhar, S. Unsupervised Lexical Learning with Categorial Grammars Using the LLL Corpus. *Learning Language in Logic.* J. Cussens and S. Dzeroski eds., p. 127-142, Springer, 2000.

[Wechsler 1994] Wechsler, S. Preposition Selection Outside the Lexicon. *Proceedings of the Thirteenth West Coast Conference on Formal Linguistics.* R. Aranovich, W. Byrne, S. Preuss, and M. Senturia eds., p. 416-431, University of California, Santa Diego, 1994.

[Wechsler 1995] Wechsler, S. *The Semantic Basis of Argument Structures.* CLSI Publications, 1995.

[Wexler and Culicover 1980] Wexler, K. and Culicover, P. *Formal Principles of Language Acquisition.* MIT Press, 1980.

[Wood 1989] Wood, M.M. *A Categorial Syntax for Coordinate Constructions.* Ph.D. thesis, University of London, 1988, University of Manchester Department of Computer Science Technical Report UMCSD-89-2-1, 1989.

[Wood 1993] Wood, M.M. *Categorial Grammars.* Routledge, 1993.

[Yang 1999] Yang, C.D. *A Selectionist Theory of Language Acquisition.* Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL), p. 429-435, Maryland, USA, 1999.

[Zeevat et al 1987] Zeevat, H., Klein, E., Calder, J. An Introduction to Unification Categorial Grammar. *Categorial Grammar, Unification Grammar, and Parsing: Working Papers in Cognitive Science.* N. Haddock, E. Klein and G. Morrill eds., v. 1, p. 195-222, Centre for Cognitive Science, University of Edinburgh, 1987.

[Zeevat 1988] Zeevat, H. Combining Categorial Grammar and Unification. *Natural Language Parsing and Linguistic Theories.* U. Reyle and C. Rohrer eds., p. 202-229, Dordrecht:Reidel, 1988.