The Use of Default Unification in a System of Lexical Types

Aline Villavicencio
Computer Laboratory,
University of Cambridge,
New Museums Site - Pembroke Street,
Cambridge, CB2 3QG, England, UK
Aline.Villavicencio@cl.cam.ac.uk

Abstract

In this paper we describe the encoding of a Unification-Based Generalised Categorial Grammar for English, in terms of a default inheritance network of types, implemented with YADU, which is an order independent default unification operation on typed feature structures. We then propose to use this framework to encode a Universal Grammar (UG) and associated parameters, following the Principles and Parameters Theory and describe how they are implemented. This is a clear and concise way of defining the UG with the parameters being straightforwardly defined in the type network, in a way which takes advantage of the default inheritance mechanism, to propagate information about parameters, throughout the lexical inheritance network. The parameters are set based on exposure to a particular language, and that is obtained by using a corpus of annotated sentences to set the parameters. The corpus was parsed and annotated using the implemented English grammar. The resulting formalisation is being used as the basis to perform studies about language acquisition.

1 Introduction

In a lexicalized linguistic formalism like Categorial Grammar, the lexicon is the focal point, with most of the information being located there, and with the syntactic component being drastically reduced. The issue of how to organise lexical information is essential, since the burden of linguistic description is concentrated in the lexicon and if lexical entries are organised as unrelated lists, there is a significant loss of generalisation and an increase in redundancy. As a lot of the information described in one lexical sign will also be contained in other signs, 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. This idea is employed in Pollard and Sag's [1987] sketch of an HPSG lexicon as an inheritance hierarchy, where information common to a class of items is inherited by its subtypes, figure 1. Proposals like this enrich the structure of the lexicon, allowing generalisations about classes of items to be economically expressed, defined only once and then passed through the inheritance hierarchy. However, these works 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 ([Bouma, 1992], [Krieger and Nerbonne, 1993], [Daelemans et al, 1992], [Briscoe, 1993]). In this paper we use the

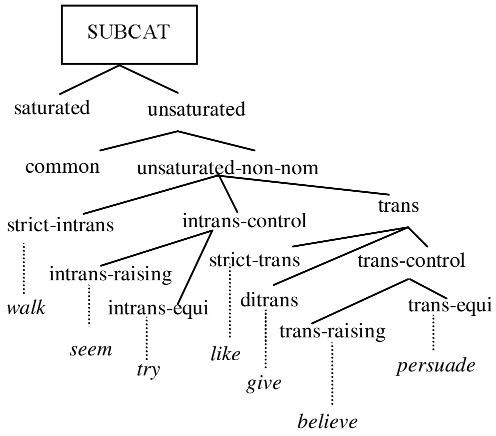

Figure 1: Pollard and Sag's Hierarchy

order-independent and persistent version of default unification described in [Lascarides et al, 1996b] and [Lascarides and Copestake, 1999] to structure linguistic information, concentrating on the encoding of the lexicon, implementing it as a default inheritance hierarchy, that is succinct and able to express elegantly linguistic sub-regularities.

Such an approach can be used not only in the encoding of a grammar for a particular language, in this case English, but also in the encoding of the Universal Grammar (UG). The UG corresponds to our innate knowledge about language, according to Chomsky [1981], who proposed the idea based on observations that children learn languages quickly and easily, and they do it based on exposure to data that is so limited and noisy that it cannot provide an explanation of how it is that children achieve all the aspects of the mature state. Moreover, Gold [1967], when investigating the possibility of learning various classes of languages using only positive examples (which is considered to be the only kind of information available to children), showed that, with this requirement, just a proper subset of regular languages could be learned. This implied that natural languages could not be learned only on the basis of positive evidence. These results were used as a confirmation for the proposal that children must have some innate knowledge about language to help them overcome the problem of the poverty of the stimulus and acquire a grammar on the basis of positive evidence only. 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 exposition to this particular language. In this way, as a child listens to its caretakers' speech, the learning procedure sets the values of the parameters such that the resulting grammar is usually equivalent to that of the caretakers. In this paper, we implement the UG in terms of a Unification-Based Generalised Categorial Grammar, embedded in a default inheritance hierarchy. This formalisation provides a succinct definition of the UG, taking advantage of the default inheritance mechanism to propagate information about the parameters of the UG and their values.

The structure of this paper is as follows. In section 2, we review the concepts involved in inheritance hierarchies and their applications to natural language processing, concentrating on the description of YADU, which is the default unification operation used to implement the lexical inheritance hierarchy proposed. We give a description of the particular version of Unification-Based Generalised Categorial Grammars (UB-GCG) used, in section 3. In section 4, we present the results of representing a UB-GCG for English in terms of default multiple inheritance networks. In section 5, we discuss how to implement the concept of Universal Grammar using this framework. In the next section we describe the application of the implemented English grammar to parse and annotate a corpus of utterances, which serves as the basis to set the parameters of the UG. Finally, we discuss the results achieved and future work.

2 Inheritance Hierarchies

Linguistic objects have a good deal of regularity, with various objects presenting a number of properties in common. 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 and 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, being 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 it has **nonmonotonic inheritance**. This latest type 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. More suited for the description of natural languages are multiple inheritance hierarchies, where different hierarchies are defined for different properties; and nonmonotonic inheritance hierarchies, where the overriding mechanism is used to allow for exceptions. And they can both be used in conjunction in a multiple default inheritance hierarchy, as is the case here.

The use of inheritance in unification-based grammar processing systems can be found in PATR, where, according to Shieber [1986], the use of lexical templates corresponds to a language for defining multiple inheritance hierarchies. In this way, it is possible to define a collection of hierarchically dependent templates that can be incorporated into the definition of lexical entries and grammar rules [Carpenter, 1992]. One of the theories that has made use of inheritance networks is HPSG. In Pollard and Sag [1987], the lexicon is treated as a monotonic multiple inheritance type hierarchy. Monotonic multiple inheritance type networks are also used in UCG [Moens et al, 1989]. Some other frameworks use default multiple inheritance, like, for instance, Word Grammar, by [Hudson, 1990]. DATR also allows multiple default inheritance, and although it requires orthogonality, Evans et al [1993] show how to encode some kinds of prioritised inheritance.

Among the characteristics that make the use of multiple default inheritance networks so appealing, we can cite (from [Daelemans et al. 1992]):

- parsimony,
- ease of maintenance,
- uniformity,

- modularity - different taxonomies can be defined for different levels of description,
- interaction - properties at different levels of description can interact with one another,
- the ability to capture generalisations, linguistic subregularities and exceptions (with the overriding mechanism), and
- the ability to provide non-redundant representations.

Thus, default networks allow properties to be incrementally true, so that things can be considered true by default until further evidence is discovered indicating 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 [1998].

2.1 YADU

In this work, a default multiple orthogonal inheritance hierarchy is used to represent linguistic information. The hierarchy is implemented using YADU ([Lascarides et al, 1996b] and [Lascarides and Copestake, 1999]), which is an order independent default unification operation on typed feature structures (TFS). YADU can be regarded as a default operation that tries to incorporate the maximal amount 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 typed feature structures (TFSs) called typed default feature structures (TDFSs). A TDFS is composed by an indefeasible TFS (*I*), which contains the non-default information, a defeasible TFS (*D*), which contains the default information, and a tail *T*, which is composed by a set of pairs where the first element is a path or path equivalence and the second member 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 feature structures is deterministic, always returning a single value.

The notation *Indefeasible/Defeasible* can be abbreviated to *Indefeasible*, where *Indefeasible* = *Defeasible* and to $/Defeasible$ where $\top/Defeasible$, as can be seen in the example below.

YADU has 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, whose indefeasible part is the result of unifying the indefeasible information defined in the input TDFSs; 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 $\mathbf{t} \sqsubset \mathbf{t}'$ (t is more specific than t′):

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

- $I = \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \sqcap \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} = \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix}$
- $T = \{\langle \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, \mathbf{t'} \rangle, \langle \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix}, \mathbf{t'} \rangle\}$

Then it is possible to calculate the resulting defeasible structure:

2. $D = \mathit{DefFS}(\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \overset{<>}{\sqcap} \begin{bmatrix} \mathbf{t'} \\ \mathrm{F} : \mathbf{/a} \\ \mathrm{G} : \mathbf{/b} \end{bmatrix}) = \sqcup \left(\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \top \\ \mathrm{G} : \top \end{bmatrix} \overset{<}{\sqcap}_{cs} \left\{ \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix} \right\} \right) = \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix}$

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*, which is more specific than the indefeasible TFS:

$$\mathit{DefFill}(\begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{/a} \\ \mathrm{G} : \mathbf{/b} \end{bmatrix} \Big/ \{\langle \begin{bmatrix} \mathrm{F} : \mathbf{a} \end{bmatrix}, \mathbf{t'} \rangle, \langle \begin{bmatrix} \mathrm{G} : \mathbf{b} \end{bmatrix}, \mathbf{t'} \rangle\}\ ) = \begin{bmatrix} \mathbf{t} \\ \mathrm{F} : \mathbf{a} \\ \mathrm{G} : \mathbf{b} \end{bmatrix}$$

This can be done, 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 so 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], which was already shown to be significant, for example, for the interface between the lexicon and pragmatics, where lexically encoded semantic defaults can be overridden by discourse information [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 feature structures (FSs), as described in [Carpenter, 1992]. 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 are allowed as an integrated part of the typed feature structure system implemented. It achieves the perspicuity, declarativity and expressibility that are familiar from unification-based approaches to non-default inheritance. A more detailed description of YADU can be found in [Lascarides and Copestake, 1999].

3 Unification-Based Generalised Categorial Grammars

Categorial Grammar (CG) ([Bar Hillel, 1964] [Wood 1993]) is a lexicalised grammatical formalism that regards language in terms of functions and arguments and allows the representation of semantics directly in syntax with each rule of syntax being a rule of semantics. CGs are composed by a categorial lexicon and a small set of rules. As a lexicalised formalism, the syntactic and semantic behaviour of any item is directly encoded in the lexicon, with each word being assigned one or more categories. Categories can be defined as either atomic or complex, depending on whether they are saturated or unsaturated. Atomic categories are saturated, which means that they do not subcategorise for any other categories. There are three atomic categories, **s** (sentence), **n** (noun), and **np** (noun phrase) and other categories are defined in terms of these. Complex categories, being unsaturated, subcategorise for other categories. They are composed of more than one category separated by forward or backward slashes, which are operators over categories. Complex categories are defined recursively: if X and Y are categories, X/Y and X\Y are also categories. Unification-Based Generalised Categorial Grammars (UB-GCGs) extend the basic CGs by including the use of feature-value pairs associated with each category and by using a larger set of rules and operators.

In this work, we follow the sign-based approach to linguistic analysis [Pollard and Sag, 1987], where words and categories are represented in terms of typed feature structures, with different kinds of linguistic information being simultaneously represented as a conjunction of attribute-value pairs that form a **sign**. Feature structures are a means of representing information about the value of an object's attributes. In the examples that follow, features are displayed in small capital letters and atoms in bold face, e.g. CAT and **s**, respectively, and the numeric tag (e.g. $\boxed{1}$) in a FS is used to indicate structure sharing between substructures, with the coindexation being used for displaying, only once, substructures that are reached by different sequences of features. Throughout this document, only the relevant information is shown in the figures, for reasons of clarity.

A sign is composed of three features: ORTH, which is used to encode the orthographic description of words; CAT which contains features that encode syntactic aspects related to the categories, and SEM, which encodes the semantics associated to a particular word. Complex categories are recursively defined in terms of two attributes, following [Steedman 1988]: result (RESULT), which describes the functor category and active (ACTIVE), which describes the list of subcategorised argument categories, both defined in CAT. RESULT has one feature, SIGN, (with attributes ORTH, CAT and SEM) which is instantiated according to the particular functor category. ACTIVE encodes the subcategorisation list of the category, representing it as a difference list (LST and LAST), with each of the subcategorised categories having two features: SIGN, as before, and DIRECTION, encoding the direction in which the category is combined (**forward** or **backward**). The categories defined in RESULT and in ACTIVE can be either atomic or complex themselves. As an example, an intransitive verb is encoded as shown in figure 2. This notation is based on the one used by Steedman [1988], but it differs from the latter in that it had to be modified to allow an efficient implementation of a default inheritance hierarchy of types, where information is defined only once and propagated through the hierarchy. In this way, a complex category is not defined as an embedded structure, but rather as a list of sequentially defined subcategorised elements, with each element encoding not only the SIGN, but also the DIRECTION feature. In terms of the inheritance hierarchy, it means that a given subtype inherits the ACTIVE list from a supertype, and, if it subcategorises for more categories, these are added in the end of the list, overriding the default **e-list** (end-of-list), as we can see in figure 3, which is a subtype of **intrans** encoding transitive verbs. **trans** is expanded as shown in figure 4, with all the information contained in **intrans**, also adding in the end of the ACTIVE list another np category, with direction **forward**, and a default **e-list**.

$$\begin{bmatrix} \textbf{intrans} & \\ \text{RESULT:SIGN:CAT} : \textbf{s} & \\ \text{ACTIVE} : & \begin{bmatrix} \textbf{diff-list} & \\ \text{LST} : & \begin{bmatrix} \textbf{ne-list} & \\ \text{HD} : & \begin{bmatrix} \textbf{act-args} \\ \text{SIGN:CAT} : \textbf{np} \\ \text{DIRECTION} : \textbf{backward} \end{bmatrix} \\ \text{TL} : & /\boxed{1}\ \textbf{e-list} \end{bmatrix} \\ \text{LAST} : & /\boxed{1} \end{bmatrix} \end{bmatrix}$$

Figure 2: Intransitive Verb type

The set of combinatory rules is responsible for combining functions with arguments or with other

$$\begin{bmatrix} \textbf{trans} & \\ \text{ACTIVE} : & \begin{bmatrix} \textbf{diff-list} & \\ \text{LST:TL} : & \begin{bmatrix} \textbf{ne-list} & \\ \text{HD} : & \begin{bmatrix} \textbf{act-args} \\ \text{SIGN:CAT} : \textbf{np} \\ \text{DIRECTION} : \textbf{forward} \end{bmatrix} \\ \text{TL} : / \boxed{1}\ \textbf{e-list} & \end{bmatrix} \\ \text{LAST} : / \boxed{1} & \end{bmatrix} \end{bmatrix}$$

Figure 3: Transitive Verb Type

$$\begin{bmatrix} \textbf{trans} & \\ \text{RESULT:SIGN:CAT} : \textbf{s} & \\ \text{ACTIVE} : & \begin{bmatrix} \textbf{diff-list} & \\ \text{LST} : & \begin{bmatrix} \textbf{ne-list} & \\ \text{HD} : & \begin{bmatrix} \textbf{act-args} \\ \text{SIGN:CAT} : \textbf{np} \\ \text{DIRECTION} : \textbf{backward} \end{bmatrix} \\ \text{TL} : & \begin{bmatrix} \textbf{ne-list} & \\ \text{HD} : & \begin{bmatrix} \textbf{act-args} \\ \text{SIGN:CAT} : \textbf{np} \\ \text{DIRECTION} : \textbf{forward} \end{bmatrix} \\ \text{TL} : / \boxed{1}\ \textbf{e-list} & \end{bmatrix} \end{bmatrix} \\ \text{LAST} : / \boxed{1} & \end{bmatrix} \end{bmatrix}$$

Figure 4: Transitive Verb type Expanded

functions. In this work, we employ the rules of (forward and backward) application, (forward and backward) composition and generalised weak permutation:

- **forward application** ($>$): X/Y Y $\rightarrow$ X,
- **backward application** ($<$): Y X$\backslash$Y $\rightarrow$ X,
- **forward composition** ($>$**B**): X/Y Y/Z $\rightarrow$ X/Z ,
- **backward composition** ($<$**B**): Y$\backslash$Z X$\backslash$Y $\rightarrow$ X$\backslash$Z.
- **generalised weak permutation** (G): $((\mathrm{X}|\mathrm{Y}_1)\ldots|\mathrm{Y}_n) \Rightarrow ((\mathrm{X}|\mathrm{Y}_n)\ldots|\mathrm{Y}_1)$

where '$|$' is a variable over '/' and '$\backslash$'. They are encoded in terms of TDFSs, as can be seen in figure 5, for the backward application rule, and the result of a derivation using this rule is shown in figure 6, for the sentence *Bill runs*.

The unary rule of generalised weak permutation [Briscoe 1999] allows a functor category with any number of arguments to combine with its arguments in any order, keeping the direction of the argument. Hoffman [1995] uses a similar system, that is able to handle scrambling elegantly and generate some mildly context-sensitive languages.

As the arguments in the subcategorisation list are in a sequential order, the generalised weak permutation rule is particularly useful, since it rotates the arguments until they are in an appropriate order for combining with an adjacent category. Both this rule and the composition rules introduce "spurious ambiguity", which is the property where alternative constituent structures for a sentence have the same semantic interpretation. Wood [1993] notes that in most cases spurious ambiguity will not

$$
\begin{bmatrix}
\textbf{backward-application} & \\
\text{NODE} : & \begin{bmatrix}
\textbf{sign} & \\
\text{ORTH} : & \begin{bmatrix} \textbf{diff-list} \\ \text{LST} : \boxed{0}\ \textbf{list} \\ \text{LAST} : \boxed{1}\ \textbf{list} \end{bmatrix} \\
\text{CAT} : & \boxed{2}\ \textbf{cat} \\
\text{SEM} : & \begin{bmatrix} \textbf{sem-struc} & \\ \text{RESTR} : & \begin{bmatrix} \textbf{list-preds} \\ \text{LST} : \boxed{3}\ \textbf{list} \\ \text{LAST} : \boxed{4}\ \textbf{list} \end{bmatrix} \end{bmatrix}
\end{bmatrix} \\
1 : & \begin{bmatrix}
\textbf{total-sign} & \\
\text{NODE} : \boxed{5} & \begin{bmatrix}
\textbf{sign} & \\
\text{ORTH} : & \begin{bmatrix} \textbf{diff-list} \\ \text{LST} : \boxed{0} \\ \text{LAST} : \boxed{6}\ \textbf{list} \end{bmatrix} \\
\text{CAT} : & \textbf{cat} \\
\text{SEM} : & \begin{bmatrix} \textbf{sem-struc} & \\ \text{RESTR} : & \begin{bmatrix} \textbf{list-preds} \\ \text{LST} : \boxed{7}\ \textbf{list} \\ \text{LAST} : \boxed{4} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
2 : & \begin{bmatrix}
\textbf{total-sign} & \\
\text{NODE} : & \begin{bmatrix}
\textbf{sign} & \\
\text{ORTH} : & \begin{bmatrix} \textbf{diff-list} \\ \text{LST} : \boxed{6} \\ \text{LAST} : \boxed{1} \end{bmatrix} \\
\text{CAT} : & \begin{bmatrix}
\textbf{complex-cat} & \\
\text{RESULT} : & \begin{bmatrix} \textbf{result-args} \\ \text{SIGN:CAT} : \boxed{2} \end{bmatrix} \\
\text{ACTIVE} : & \begin{bmatrix}
\textbf{comps-list} & \\
\text{LST} : & \begin{bmatrix}
\textbf{ne-list} & \\
\text{HD} : & \begin{bmatrix} \textbf{active-args} \\ \text{SIGN} : \boxed{5} \\ \text{DIRECTION} : \textbf{backward} \end{bmatrix} \\
\text{TL} : & \boxed{8}
\end{bmatrix} \\
\text{LAST} : & \boxed{8}
\end{bmatrix}
\end{bmatrix} \\
\text{SEM} : & \begin{bmatrix}
\textbf{sem-struc} & \\
\text{MODE} : & \textbf{mode-cat} \\
\text{INDEX} : & \textbf{index} \\
\text{RESTR} : & \begin{bmatrix} \textbf{list-preds} \\ \text{LST} : \boxed{3} \\ \text{LAST} : \boxed{7} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Figure 5: Backward Application

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.

4 Structuring Linguistic Information with Default Unification

By using such a lexicalised approach to grammar, it is possible to reduce the complexity of the set of rules, which can then be defined as very general schemas of how to combine categories. However, the descriptive burden is then concentrated in the signs, which are composed of rich and elaborate information structures. We organise the signs as a default inheritance network of types to reduce

$$
\begin{bmatrix}
\text{NODE}: \begin{bmatrix}
\textbf{sign} \\
\text{ORTH}: \begin{bmatrix} \textbf{diff-list} \\ \text{LST}: \begin{bmatrix} \textbf{ne-list} \\ \text{HD}: \textbf{bill} \\ \text{TL}: \begin{bmatrix} \textbf{ne-list}\\ \text{HD}: \textbf{runs} \\ \text{TL}: \boxed{0}\ \textbf{list}\end{bmatrix}\end{bmatrix} \\ \text{LAST}: \boxed{0}\end{bmatrix}\\
\text{CAT}: \textbf{sent-cat}\\
\text{SEM}: \begin{bmatrix} \textbf{sem-struc}\\ \text{MODE}: \textbf{prop}\\ \text{INDEX}: \boxed{1}\ \textbf{sit-index}\\ \text{RESTR}: \begin{bmatrix}\textbf{list-preds}\\ \text{LST}: \begin{bmatrix}\textbf{ne-list}\\ \text{HD}: \begin{bmatrix}\textbf{intrans-pred}\\ \text{RELN}: \textbf{r-run}\\ \text{SIT}: \boxed{1}\\ \text{ACT}: \boxed{2}\ \textbf{ref-index}\end{bmatrix}\\ \text{TL}: \begin{bmatrix}\textbf{ne-list}\\ \text{HD}: \begin{bmatrix}\textbf{np-pred}\\ \text{RELN}: \textbf{r-name}\\ \text{SIT}: \boxed{2}\\ \text{NAME}: \textbf{bill}\end{bmatrix}\\ \text{TL}: \boxed{3}\ \textbf{list}\end{bmatrix}\end{bmatrix}\\ \text{LAST}: \boxed{3}\end{bmatrix}\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Figure 6: “Bill runs”

redundancy, implemented it using YADU, with different hierarchies defined for different levels of description. Defaults can be used in the definition of inflectional morphology and in the specification of lexical semantics among others. In this section we concentrate on the use of defaults to structure the lexicon, focusing on the description of verbal subcategorisation information, which serves to illustrate the advantages of the approach chosen.

Types are organised into a hierarchy, with appropriate features associated with particular types and the relevant constraints on each type defined. 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 7. In this hierarchy, types defining complex categories are ordered according to the number and type of the subcategorisation arguments they specify. In the verbal hierarchy, the basic category (**intrans**) corresponds to the one assigned to intransitive verbs (s\np), figure 2, which specifies the need for exactly one subcategorised argument, the NP subject. The last element of the difference-list is, by default, **e-list**, which marks the end of the subcategorisation list (ACTIVE). All the attributes specified for the subcategorised subject NP in **intrans** are inherited by instances of this type and by its subtypes[1]: **trans** (transitive verb), and **intrans-control** (intransitive control verb). However, since these types subcategorise for 2 arguments, they need to override the default of **e-list** and specify the addition of an extra argument type: **trans** specifies an extra NP argument, figure 3, and **intrans-control** specifies an extra predicative complement. Furthermore, each of these types also adds, by default, **e-list** as the last element of the subcategorisation list, signalling that these categories subcategorise, by default, for exactly 2 arguments. Similarly, the instances and subtypes of **trans-control** inherit the default information that the predicative complement is controlled by the object, like **trans-equi** for transitive-equi

[1] In this paper, we are not assuming the coverage condition, that any type in a hierarchy has to be resolvable to a most specific type.

verbs (figure 8), with the exception of the **subject-control** type. Sentences 1 and 2 show examples of a transitive-equi verb (*persuade*) and a subject-control verb (*promise*), respectively. For subject control verbs, it is the subject of the transitive verb that controls the predicative complement. This behaviour is captured with the use of inequalities, which negate the inherited coindexation, (between the object of the control verb and the subject of the predicative complement), figure 9, allowing the coindexation between the two subjects (specified in *subject-control*) to hold [Villavicencio 1999].

1. *Bill persuaded Mary to help John*
2. *Bill promised Mary to help John*

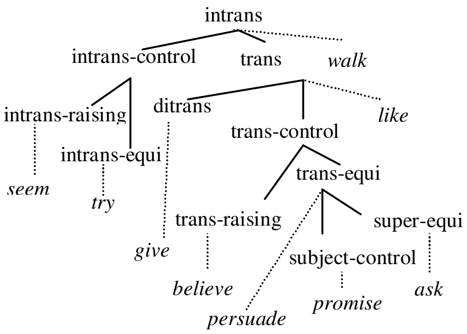

Figure 7: The Proposed Hierarchy

These 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, a type like **trans**, after *DefFill*, has the consistent default **e-list** incorporated and specifies, indefeasibly the need for exactly two arguments, as shown in figure 4. On the other hand, for a type like **super-equi**, which inherits the default object control (sentence 3), but which also allows subject control, sentence 4, we allow the default to persist until discourse interpretation, using the *p* operator, figure 10. Then, the default will survive if there is no conflicting information, otherwise it will be overridden.

3. *Mary asked Bill to go home. His mother was expecting him there*
4. *Mary asked Bill to go home. She was expected there*

In comparison with Pollard and Sag's hierarchy, shown in figure 1, this is much more compact, since it has no need for types like **strict-trans**, which is a subtype of **trans** and encodes exactly the same information as **trans**, except that the latter works as an intermediate type, where the subcategorisation list contains **at least** two arguments ($< np, np... >$), offering its subtypes the possibility of adding extra arguments, while the former specifies the list as containing **exactly** two arguments ($< np, np >$). Defaults automatically provide this possibility, by defeasibly defining the end-of-the-list (**e-list**) argument, avoiding the need for these redundant specifications. This hierarchy achieves total size-elimination of the redundant specifications, like the *strict-trans* type, which are needed in

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{trans-equi-sign}} \\
\text{CAT} : & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{trans-control-cat}} \\
\multicolumn{2}{l}{\text{RESULT:SIGN:CAT} : \textbf{s}} \\
\text{ACTIVE:LST} : & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD} : & \left[\begin{array}{l} \textbf{active-args} \\ \text{SIGN} : \textbf{np-sign} \\ \text{DIRECTION} : \textbf{backward} \end{array}\right] \\
\text{TL} : & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD} : & \left[\begin{array}{l} \textbf{active-args} \\ \text{SIGN} : \boxed{0}\ \textbf{np-sign} \\ \text{DIRECTION} : \textbf{forward} \end{array}\right] \\
\text{TL} : & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD} : & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{active-args}} \\
\text{SIGN:CAT} : & \left[\begin{array}{l} \textbf{intrans-cat} \\ \text{RESULT:SIGN:CAT} : \textbf{s} \\ \text{ACTIVE:LST:HD:SIGN} : \boxed{0} \end{array}\right] \\
\multicolumn{2}{l}{\text{DIRECTION} : \textbf{forward}}
\end{array}\right] \\
\multicolumn{2}{l}{\text{TL} : \textbf{e-list}}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

Figure 8: Transitive Equi Verb Type

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{subject-control}} \\
\text{CAT:ACTIVE:LST} : & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\multicolumn{2}{l}{\text{HD:SIGN} : \boxed{5}} \\
\text{TL} : & \left[\begin{array}{l} \textbf{ne-list} \\ \text{HD:SIGN} : \boxed{0} \\ \text{TL:HD:SIGN:CAT:ACTIVE:LST:HD:SIGN} : \boxed{5} \\ \boxed{5} \nleftrightarrow \boxed{0} \end{array}\right]
\end{array}\right]
\end{array}\right]$$

Figure 9: Subject-control Verb Type

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{super-equi}} \\
\text{CAT:ACTIVE:LST:TL} : & \left[\begin{array}{l} \textbf{ne-list} \\ \text{HD:SIGN} : \boxed{0} \\ \text{TL:HD:SIGN:CAT:ACTIVE:LST:HD:SIGN} : /_{p}\,\boxed{0} \end{array}\right]
\end{array}\right]$$

Figure 10: Super-equi Verb Type

a monotonic encoding. By avoiding this redundancy, there is a real gain in conciseness, with the resulting hierarchy capturing linguistic regularities and sub-regularities in a compact encoding.

The implemented UB-GCG has 84 lexical categories defined, covering several verbal constructions (intransitives, transitives, ditransitives, obliques, control verbs, verbs with sentential complements, etc), declarative, imperative and interrogative sentences, and unbounded dependencies (wh-questions and relative clauses), among others, and is able to successfully parse a corpus of utterances, as discussed in section 6. As a result of using default inheritance to represent lexical information, it is possible to obtain a highly structured and succinct hierarchy.

5 The Universal Grammar, Principles and Parameters and UB-GCGs

To address this issue, Chomsky proposed the idea that people must have some innate knowledge about languages. This core common knowledge, or Universal Grammar (UG), is composed by this common set of principles and by some parameters, which are set based on exposure to a given linguistic environment, according to the Principles and Parameters Theory [Chomsky, 1981]. It implies 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 in relation to a finite number of values.

The approach used here to formalise the UB-GCG for English is well suited for a 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 [1998]. In this section we discuss how the UG and associated parameters can be encoded in terms of a UB-GCG, concentrating on the description of word order parameters, which reflect the order in which constituents occur in different languages (e.g. SVO and SOV languages) [Croft, 1992].

The UG is implemented as a UB-GCG, embedded in a default inheritance network, where the category set and rule schemata are represented in terms of typed feature-structures and the feature-structures associated with any given category are defined by the inheritance chain. It provides a clear and concise way of formalising the UG, taking advantage of the default inheritance mechanism. Regarding the categories of the UB-GCG, word order parameters are those that specify the direction of each element in the subcategorisation list of a complex category. As the categories are defined in the terms of an inheritance hierarchy, the parameters (and its values) in these categories are propagated throughout the hierarchy, from supertypes to subtypes, which inherit this information by default. There are 28 parameters defined, and they are also embedded in the hierarchy, 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**, which specifies the direction of the other verbal arguments and **ndir**, which specifies the direction of nominal categories. This hierarchy makes a distinction between phrasal and clausal parameters. A fragment of the parameters hierarchy can be seen in figure 11. We are formalising the UG with the parameters being straightforwardly defined in the categories, in a way which takes advantage of the default inheritance mechanism, to propagate information about parameters, throughout the lexical inheritance network.

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 on exposure to that language. This can be obtained by means of a corpus of utterances, annotated with logical forms, which is described in section 6. Among these sentences, some will be triggers for certain parameters, in the sense that, to account for that sentence, with that particular logical form, some of the parameters will have to be set to a given value. The parameters are set based on exposure to the triggers, and while they are unset, they inherit their value by default, from their supertypes. Then, when they are set, they can either continue to inherit by default, in case they have the same value as the supertype, or they can override this default and specify their own value, breaking the inheritance chain. 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, and all its subtypes inherit this default information. Then an intransitive verb will be defined as S/NP. However, as in English, the subject NP occurs to the left of the verb, utterances with the subject to the left will trigger a change in **subjdir** to **backward**, resulting in intransitives being defined as S\NP, if the grammar is to account for these sentences.

In the syntactic dimension of this network, intransitive verbs can be considered the general case of verbs, and the information defined in this node is propagated through the hierarchy to its subtypes, such as the transitive verbs. For the learner, the information about subjects (**subjdir = backward**) has already been acquired while learning intransitive verbs, and the learner does not need to learn it again for transitive verbs. The result of using a default inheritance schema is that it reduces the pieces of information to be acquired by the learner, since the information is structured and what it learns is not a single isolated category, but a structure that represents this information in a general manner. 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. And this value is going to hold until the occurrence of a trigger reinforces or disconfirms it. A detailed description of the learning algorithm used to set the parameters of the UG can be found in [Villavicencio 2000]

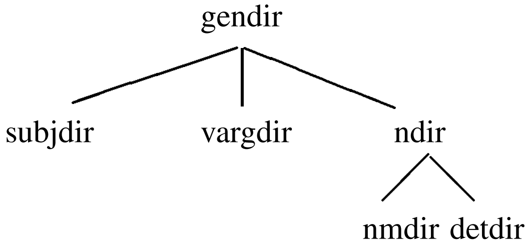

Figure 11: Fragment of Parameters Hierarchy

6 The corpus

The Principles and Parameters Theory suggests that the parameters of the UG are set to a particular language, based on exposure to data. Thus, one of the resources needed to set the parameters, was a corpus of utterances based on which the parameters would be set to account for that particular language. The corpus chosen is a subset of the Sachs corpus [Sachs 1983], which contains interactions between one child and her parents, from the age of 1 year and 1 month to 5 years and 1 month. The Sachs corpus is part of the Child Language Data Exchange System (CHILDES) [MacWhinney 1995] that contains various corpora of child language transcript data from scores of projects in a number of different languages. From this corpus, we extracted material for generating two different corpora: one containing only the child's sentences and the other containing the parents' sentences. It means that we can use the parents' corpus as the basis in which to set the parameters, using the same input that the child receives. The parents' corpus had to be pre-processed to remove phonological annotations and some grammatical constructions that are not relevant to this project, like noise, interjections, elliptical material, incomplete sentences and the like. The original parents' corpus contains 12.163 sentences, and after the first stage of pre-processing 10.292 sentences remained, from which many have been removed for containing elliptical material or for being incomplete sentences. The implementation of the UB-GCG discussed in section 4 was directed by the constructions found in this corpus, and the grammar was used to parse and annotate the sentences in the corpus with logical forms, as can be seen in figure 12, for the sentence *I will take him.* The corpus contains a wide variety of constructions, with declarative, imperative and interrogative sentences with rich verbal constructions and unbounded dependencies, among others, as can be seen in sentences 5 to 9, extracted from the corpus. At the moment, around 1.300 parents' sentences were annotated with logical forms, ranging from when the child is 14 months to when she is 20 months.

5. *Are you done with looking at the pictures*
6. *I think you had better finish your popsicle*
7. *Do you want to give Mommy the popsicle*
8. *He is going to turn off the lights*
9. *Go in the kitchen and see Daddy*

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{sign}} \\
\text{ORTH}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{diff-list}} \\
\text{LST}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD}: & \textbf{i} \\
\text{TL}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD}: & \textbf{will} \\
\text{TL}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD}: & \textbf{take} \\
\text{TL}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD}: & \textbf{him} \\
\text{TL}: & \boxed{0}\ \textbf{list}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{LAST}: & \boxed{0}
\end{array}\right] \\
\text{CAT}: & \textbf{s} \\
\text{SEM}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{sem-struc}} \\
\text{MODE}: & \textbf{prop} \\
\text{INDEX}: & \boxed{1}\ \textbf{sit-index} \\
\text{RESTR}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{list-preds}} \\
\text{LST}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{aux-pred}} \\
\text{RELN}: & \textbf{r-modal} \\
\text{ARG}: & \boxed{2}\ \textbf{sit-index} \\
\text{SIT}: & \boxed{1}
\end{array}\right] \\
\text{TL}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{pron- pred}} \\
\text{RELN}: & \textbf{r-name} \\
\text{SIT}: & \boxed{3}\ \textbf{ref-index} \\
\text{PRON}: & \textbf{i}
\end{array}\right] \\
\text{TL}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{trans-pred}} \\
\text{RELN}: & \textbf{r-take} \\
\text{SIT}: & \boxed{2} \\
\text{UND}: & \boxed{4}\ \textbf{index1} \\
\text{ACT}: & \boxed{3}
\end{array}\right] \\
\text{TL}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{ne-list}} \\
\text{HD}: & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textbf{pron-pred}} \\
\text{RELN}: & \textbf{r-name} \\
\text{SIT}: & \boxed{4} \\
\text{PRON}: & \textbf{him}
\end{array}\right] \\
\text{TL}: & \boxed{5}\ \textbf{list}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{LAST}: & \boxed{5}
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure 12: Sentence: I will take him

7 Conclusion and Future Works

In this paper, we described the implementation of a UB-GCG in terms of a default inheritance network of lexical types. It resulted in a significant reduction in lexical redundancy, as exemplified in the fragment of the verbal hierarchy, which, compared with Pollard and Sag's, needs less nodes to describe the same information, and is also able to describe sub-regularities and exceptions, defined by means of TDFSs. The resulting lexicon is succinctly organised, and is also easier to maintain and modify. Then we described how concepts like the Universal Grammar and associated principles and parameters can be implemented in this framework in a clear and concise way, with the use of a default inheritance hierarchy providing an efficient manner of propagating information about the parameters and structuring the information to be learned. Finally, we discussed how the implemented UB-GCG for English was used to parse and annotate a subset of the Sachs corpus, with the appropriate logical forms. As a result of implementing these ideas, we obtained a basic framework that is allowing us, among other things, to perform experiments about language acquisition: there is the UG and associated parameters, a corpus annotated with logical forms, based on which the parameters are set, and the target English grammar. Some preliminary results are described in [Villavicencio 2000]. The next step now is to annotate more data, in order to have a bigger corpus. Then, one of the possible options to follow is to obtain data for another language, annotate it, and make a comparative study of these two languages in relation to language acquisition in this framework.

Acknowledgments

The research reported on this paper is supported by doctoral studentship from CAPES/Brazil. I would like to thank Ted Briscoe and Fabio Nemetz for their comments and advice on this paper, and to Ann Copestake and John Carroll for some ideas and help with the LKB system. Thanks also to the anonymous reviewers for their comments.

References

[Bar Hillel, 1964] Bar Hillel, Y. *Language and Information.* Wesley, Reading, Mass. 1964.

[Bouma, 1992] Bouma, G. Feature Structures and Nonmonotonicity. *Computational Linguistics*, 18.2 1992.

[Briscoe, 1993] Briscoe, T. Introduction. *Inheritance, Defaults and the Lexicon.* Ted Briscoe, Ann Copestake and Valeria de Paiva, eds.. 1993.

[Briscoe 1999] Briscoe, T. *The Acquisition of Grammar in an Evolving Population of Language Agents.* Linkoping Electronic Articles in Computer and Information Science, http://www.ep.liu.se/ea/cis/1999 , 1999.

[Carpenter, 1992] Carpenter, B. *The Logic of Typed Feature Structures*, 1992.

[Croft, 1992] Croft, W. *Typology and Universals*, 1993.

[Chomsky 1981] Chomsky, N. *Lectures on Government and Binding*. Foris Publications, 1981.

[Daelemans et al, 1992] Daelemans, W.; De Smedt, K. and Gazdar, G. Inheritance in Natural Language Processing. *Computational Linguistics*, 18.2 1992.

[Evans et al., 1993] Evans, R.; Gazdar, G. and Moser, L. Prioritised multiple inheritance in DATR. *Inheritance, Defaults and the Lexicon.* Ted Briscoe, Ann Copestake and Valeria de Paiva, eds.. 1993.

[Gold 1967] Gold, E.M. *Language Identification in the Limit*. Information and Control, v.10, p.447-474, 1967.

[Hoffman, 1995] Hoffman, B. The Computational Analysis of the Syntax and Interpretation of 'Free' Word Order in Turkish. PhD dissertation, University of Pennsylvania. 1995.

[Hudson, 1990] Hudson, R. English Word Grammar.1990.

[Krieger and Nerbonne, 1993] Krieger, H. and Nerbonne, J. Feature-Based Inheritance Networks for Computational Lexicons. *Inheritance, Defaults and the Lexicon.* Ted Briscoe, Ann Copestake and Valeria de Paiva, eds.. 1993.

[Lascarides et al, 1996a] Lascarides, A., Copestake, A, Briscoe, T. Ambiguity and Coherence. *Journal of Semantics*, 13.1,41-65, 1996.

[Lascarides et al, 1996b] Lascarides, A., Briscoe, T., Asher, N., and Copestake, A. Order Independent Persistent Typed Default Unification. *Linguistics and Philosophy* , 19.1,1-89, 1996.

[Lascarides and Copestake, 1999] Lascarides, A. and Copestake, A. Default Representation in Constraint-based Frameworks. *Computational Linguistics*, 25.1, 55-105, 1999.

[MacWhinney 1995] MacWhinney, B. *The CHILDES Project: Tools for Analyzing Talk*. Second Edition. Hillsdale, NJ: Lawrence Erlbaum Associates. 1995.

[Moens et al, 1989] Moens, M., Calder, J., Klein, E., Reape, M. and Zeevat, H. *Expressing Generalizations in unification-based grammar formalisms*, In ACL Proceedings, 4th European Conference, 1989.

[Pollard and Sag, 1987] Pollard, C. and Sag, I.A. *Information-Based Syntax and Semantics*, CSLI Lecture Notes Series, n. 13, 1987.

[Sachs 1983] Sachs, J. *Talking about the there and then: the emergence of displaced reference in parent-child discourse*. In K. E. Nelson editor, Children's language, v.4. 1983.

[Sag and Wasow, 1998] Sag, I.A. and Wasow, T. *Syntactic Theory - A Formal Introduction*, 1998.

[Sanfilippo, 1993] Sanfilippo, A. LKB Encoding of Lexical Knowledge. *Inheritance, Defaults and the Lexicon.* Ted Briscoe, Ann Copestake and Valeria de Paiva, eds.. 1993.

[Shieber 1986] Shieber, S. *An Introduction to Unification-Based Approaches to Grammar* CSLI Lecture Notes Series, n.4. 1986.

[Steedman 1988] Steedman, M. Combinators and Grammars. *Categorial Grammars and Natural Language Structures* Richard Oehrle, Emmon Bach and Deirdre Wheeler, eds.. 1988.

[Villavicencio 1999] Villavicencio, A. *Representing a System of Lexical Types Using Default Unification.* Proceedings of EACL, 1999.

[Villavicencio 2000] Villavicencio, A. *The Acquisition of a Unification-Based Generalised Categorial Grammar.* Proceedings of the Third CLUK Colloquium, 2000.

[Wood 1993] Wood, M.M. *Categorial Grammars*. Routledge, 1993.