Towards a Unification-Based Swedish Grammar in Prolog

Mattias Nilsson
Department of Linguistics and Philology
Uppsala University
tias@stp.ling.uu.se

Abstract
In this paper we review the theoretical aspects and the initial design steps of implementing a unification-based, theory-independent feature structure grammar for Swedish in Prolog.1

1 Introduction
Writing simple phrase structure grammars of natural language in a declarative programming language like Prolog is easily carried out using Definite Clause Grammar (DCG) term notation. We may even add constraints to our DCG by adding arguments representing particular linguistic information. This may be useful, for instance, if we want to account for a particular agreement relation that may hold in a language. However, if our intention is to implement expressive grammars and to make extensive use of features, the term based unification formalism of DCG is hardly of any practical use. A value of a particular feature in a DCG-term is defined by its argument position. Thus, a dcg-term is order dependent since it’s not possible to change the order of the arguments in a term. Additionally, we can neither remove nor add features in a particular term, since only terms with the same number of arguments may be unified. Consequently, DCG-terms with several features become difficult to read and tricky to implement.

2 Unification-Based Grammars
Most formal theories of grammar developed since the 1980’s are unification-based. On the one hand there are unification based theories of syntax developed as complete linguistic theories, such as Lexical Functional Grammar (LFG) and Head-driven Phrase Structure Grammar (HPSG). On the other hand there are unification based formalisms developed as general tools for implementing grammars and theories of grammar, such as PATR-II and Functional Unification Grammar (FUG).

A unification-based grammar structures linguistic information in feature structures, consisting of features (attributes) and their associated values. Further, a unification-based grammar only uses the operation of unification for merging the information content of two feature structures or rejecting incompatible information content.

Any standard algorithm for context free parsing can be used for parsing with unification-based grammars. Thus, parsing unification-based grammars can be considered an instance of context free parsing where the context free rules are associated with order independent unification constraints.

---

1This paper is a report of an assignment carried out at an undergraduate university course in computational grammar.
3 Formal Properties of Feature Structures

A feature structure is a partial function from features to values. Features are defined by their names, not by their position. Thus, feature structures are order independent. Feature structures are sometimes represented as Directed Acyclic Graphs but more commonly they are illustrated as Attribute Value Matrices, as in Figure 1:

\[
\begin{bmatrix}
\text{Feature}_1 : \text{Value}_1 \\
\text{Feature}_2 : \text{Value}_2 \\
\vdots \\
\text{Feature}_n : \text{Value}_n
\end{bmatrix}
\]

Figure 1: AVM feature structure

The value of a feature is either atomic or another feature structure:

\[
\begin{bmatrix}
a : b \\
c : [d : e] \\
f : [g : h]
\end{bmatrix}
\]

Figure 2: Nested feature structure

Two or more features can share a value. This property of feature structures is called reentrancy. In AVMs, reentrancy is illustrated by coindexing the values which are shared:

\[
\begin{bmatrix}
a : \#1 \\
b : \{ [c : d], e : f \}
\end{bmatrix}
\]

Figure 3: Coindexed feature structure

3.1 The Unification Operation

Unification is a partial operation that makes two feature structures identical by combining them into a new feature structure that contains all the information of the original feature structures. The operation is partial since in the case where two feature structures contain incompatible information, the unification fails and no result is returned. The value of a such a unification is said to be undefined. Unification is formally defined as follows:\(^2\)

def: Unification

The most general feature structure \(U\) that is subsumed by two feature structures \(\alpha\) and \(\beta\) constitutes the unification of \(\alpha\) and \(\beta\).

def: Subsumption

A feature structure \(\alpha\) subsumes another feature structure \(\beta\) if \(\alpha\) consists of a subset of the information in \(\beta\).

Examples of unification of feature structures:

1. \[
\begin{bmatrix}a : b \\
c : [d : e] & f : g\end{bmatrix} \cup \begin{bmatrix}h : i \\
c : [d : e]\end{bmatrix} = \begin{bmatrix}a : b \\
c : [d : e] \\
h : i\end{bmatrix}
\]

2. \[
\begin{bmatrix}a : [b : c] \\
d : [a : [b : c]]\end{bmatrix} \cup \begin{bmatrix}d : [a : [e : f]]\end{bmatrix} = \begin{bmatrix}a : [b : c] \\
d : [a : [b : c] \\
e : f]\end{bmatrix}
\]

3. \[
\begin{bmatrix}a : X \\
b : [c : d]\end{bmatrix} \cup \begin{bmatrix}b : [c : e] \\
f : g\end{bmatrix} = \text{Undefined}
\]

4 Destructive Unification

There are several different algorithms for implementing unification.\(^3\) We have chosen to implement a destructive unification algorithm.\(^4\) The basic procedure of the algorithm is to take two feature structures as input, unify them, and return the unified feature structure or else fail if the structures can not be unified. This is achieved by recursing through the subparts of one of the input feature structures and for each feature ensure that a corresponding feature in the second feature structure has the same value. In Prolog, the matching process is performed using ordinary Prolog term unification. If all features match, the unification is successful. If any feature fails to match, then the whole unification fails. The algorithm is destructive in the sense that it destroys the original input structures in the computing process, so at the

\(^2\)Shieber 1986.

\(^3\)See e.g. [Gazdar and Mellish 1989] or [Striegnitz 2003].

\(^4\)see e.g [Jurafsky and Martin 2000].
end of the computation, each of the original feature structures holds the result of the unification. The main procedure of the algorithm is roughly described by the following piece of code:

unify(FS1,FS2):-
    unifies(FS1,FS2),
    write(FS1).

unifies([Feature|Rest],FS2):-
    find_corr_feature(Feature,FS2),
    value_of_feat(Feature,Val1),
    value_of_corr_feat(Feature2,Val2),
    Val1 == Val2,
    unifies(Rest,FS2).

5 Feature Structures in Prolog

A feature structure can be represented as an open-ended list in which each element corresponds to a feature value pair. An open list is a list whose tail is always a variable. Thus, new feature value pairs can always be added to an open list by instantiating the tail to another open list. For example, instantiating the tail of the list [a:b,c:d,e:f|X] with the list [g:h|Y] gives the open list [a:b,c:d,e:f,g:h|Y]. A feature structure for the Swedish noun *bil* and its Prolog representation is given below in 1. In 2, a more elaborate feature structure in HPSG style for the same word is illustrated.

\[
1. \begin{align*}
\text{cat: noun} & , \\
\text{word: bil} & , \\
\text{agr: [species: indef, numerus: singular]} & .
\end{align*}
\]

\[
2. \begin{align*}
\text{head: [cat: noun,} & \\
\text{agr: [per: 3rd,} & \\
\text{num: sg,} & \\
\text{gen: non_neut | X] | _]} & , \\
\text{val: [spr: [head: [cat: det,} & \\
\text{per: 3rd,} & \\
\text{num: sg,} & \\
\text{gen: non_neut,} & \\
\text{species: indef | X] | _] | _]} & .
\end{align*}
\]

6 Noun Phrase Agreement

In this section we discuss agreement properties of the Swedish simple noun phrase and review its Prolog representation in our feature structure grammar. We will associate a noun phrase such as *Ett stort fint hus* with the syntactic structure in 1. The corresponding phrase structure rules are given in 2.

1. \[ NP \rightarrow \text{Det} \text{ N} \text{bar} \]\[ NP \rightarrow \text{AdjP} \text{ N} \text{bar} \]\[ \text{N} \text{bar} \rightarrow \text{AdjP} \text{ N} \text{bar} \]\[ \text{N} \text{bar} \rightarrow \text{N} \]\[ \text{AdjP} \rightarrow \text{Adj} \]

An obvious difference between the Swedish and the English simple noun phrase is the gender distinction on Swedish nouns. Each common noun belongs to one of four categories: neuter (t-gender), non-neuter (n-gender), masculine or feminine. Only neuter and non-neuter are grammatical gender categories. In singular, the premodifiers (determiners and adjectives) agree with the gender of the head noun. Adjectives in the definite form show no gender distinction. Gender distinction is not the only feature of internal noun phrase
agreement. Premodifiers and the head noun also agree in number and definiteness (species). In the grammar we will account for these separate agreement features by associating a general feature $agr$ with features of $gend$, species and num respectively. Each lexical item is introduced via a 2-place predicate \texttt{lex(Word,FeatStruct)}, whose first argument is a word and second argument an associated feature structure.

Lexical entries for the words \textit{ett}, \textit{stort}, \textit{fint}, \textit{hus} are given below:

\begin{verbatim}
%% Lexical Entries
lex(ett,[cat:det,
        agr:[gend:neut,
             species:indef,
             num:sg|_|_]).

lex(stort,[cat:adj,
          agr:[gend:neut,
               species:indef,
               num:sg|_|_]).

lex(fint,[cat:adj,
          agr:[gend:neut,
               species:indef,
               num:sg|_|_]).

lex(hus,[cat:n,
        agr:[gend:neut,
             species:indef,
             num:sg|_|_]).
\end{verbatim}

It is now possible to account for noun phrase internal agreement in the phrase structure rules in terms of projecting the single attribute $agr$. The head noun and its premodifiers are assigned the same values for $agr$ and so is the complete noun phrase.

\begin{verbatim}
%% Phrase Structure Rules
NP ---> [Det,N_bar]:-
    NP = [cat:np,agr:Agr|_],
    Det = [cat:det,agr:Agr|_],
    N_bar = [cat:n_bar,agr:Agr|_].

N_bar ---> [N]:-
    N_bar = [cat:n_bar,agr:Agr|_],
    N = [cat:n,agr:Agr|_].
\end{verbatim}

\section{Summary}

In this paper we reviewed the theoretical aspects and the initial design steps of implementing a unification-based, theory independent feature structure grammar for Swedish in Prolog. Initially, we discussed unification-based grammars and the formal properties of feature structures. Then, an informal description of the destructive unification algorithm for implementing the operation of unification was followed by a brief discussion on the representation of feature structures in Prolog. In the closing section we analyzed noun phrase internal agreement in the Swedish simple noun phrase and showed how it can be resolved in our Prolog feature structure grammar.

A small fragment of Swedish grammar has been tried out on a bottom-up chart recognizer\footnote{Striegnitz, Blackburn 2003}. No statistical evaluation of the grammar has been carried out. However, for the current fragment, the grammar has the status of being observationally adequate\footnote{A grammar that generates all grammatical sentences and rules out the ungrammatical ones is generally said to be observationally adequate}.

\section*{References}


