Expression of syntactic and semantic features in logic-based grammars

Patrick Saint Dizier

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(3) 954 90 20

Rapports de Recherche

N° 449

EXPRESSION OF SYNTACTIC AND SEMANTIC FEATURES IN LOGIC - BASED GRAMMARS

Patrick SAINT-DIZIER

Octobre 1985

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

Publication Interne n° 266

Septembre 1985 - 19 pages

EXPRESSION OF SYNTACTIC AND SEMANTIC FEATURES IN LOGIC-BASED GRAMMARS

P. SAINT-DIZIER

ABSTRACT

In this paper, we introduce and motivate a formalism to represent syntactic and semantic features in logic-based grammars. We also introduce technical devices to express relations between features and inheritance mechanisms. This leads us to propose some extensions to the basic unification mechanism of PROLOG. Finally, we consider the problem of long distance dependency relations between constituents in Gapping Grammars rules from the point of view of morphosyntactic features that may change depending on the position occupied by the "moved" constituents: What we propose is not a new linguistic theory about features, but rather a formalism and a set of tools that we think to be useful to grammar writers to describe features and their relations in grammar rules.

RESUME

Dans ce document, nous introduisons et motivons un formalisme pour représenter dans les grammaires logiques les traits syntaxiques et sémantiques. Nous introduisons en complément des outils pour exprimer les relations entre traits, ainsi que des mécanismes d'héritage. Ceci nous conduit à augmenter le mécanisme d'unification de base de PROLOG. Enfin, nous considérons le problème des dépendances lointaines entre constituants, du point de vue de l'évolution de leurs traits syntaxiques. Ce que nous proposons ici n'est pas une nouvelle théorie linguistique des traits, mais plutôt un formalisme et des outils que nous pensons pouvoir être utiles pour l'écriture de grammaires du langage naturel.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

EXPRESSION OF SYNTACTIC AND SEMANTIC FEATURES IN LOGIC-BASED GRAMMARS

Patrick SAINT-DIZIER
I.R.I.S.A.
Campus de Beaulieu
35042 RENNES -Cedex
FRANCE

1 INTRODUCTION :

Most linguistic theories use features and feature notations for the description of several aspects of phonology, morphology, syntax and semantics. The features used differ slightly from a theory to another. For instance, some theories use semantic filtering via semantic features (human, animate, ...), others use only morphosyntactic features. The structure of features vary also : some theories use very few features and very refine syntactic categories [Gross 84] whereas others have a system of complex features associated with inheritance mechanisms. A theory such as Unification Grammars [Kay 85] is essentially based on features and unification : no hierarchy is made between syntactic categories and other features.

The formalisms and the tools we develop can be used for basic morphosyntactic feature systems as well as for complex systems of features including Unification Grammars. The elaboration of a general theory of features seems to us a promising approach to the problem of natural language understanding because the use of features permits considerable elegance without excessive power. Of course, the means of expression and of implementation are also

crucial. In particular, the means of expression of feature control have to be independant of the parsing strategy.

The usefulness of features has no longer to be demonstrated. At the last TANLU workshop, Dan Flickinger insisted on the fact that *"every element in a sentence must be accounted for, since it may not be redundant"*. As an illustration, he gave the two following sentences that differ only in the form of the auxiliary *"to be"* :
(1) *"The scientists who suggested this theory was proven wrong by all the data returned from the moon. "*
(2) "The scientists who suggested this theory were proven wrong by all the data returned from the moon. "
He shown also the interest of using lexical rules in the lexicon to express generalizations, including inflectional and derivational rules.

In this paper, we first define what we call basic features. Then, we introduce the notion of descriptor and prototype of descriptor for the set of features of a given language. Next, we give some properties of a descriptor and, finally, we introduce functions for instanciations and comparisons of descriptors. These functions are practical tools for grammar writers. Tools and structures are viewed in the spirit of a declarative programming language and a programming methodology is introduced through examples. Finally, we consider the problem of long distance dependency relations between constituents in Gapping Grammars from the point of view of morphosyntactic features that may change depending on the position occupied by the *"moved"* constituents, and that may be used to express constraints on gaps.

2_ FEATURES.

Features are traditionally viewed as attribute-value pairs [Gazdar and al 82], [Karttunen 84], [Pereira and al 84], for example : (gender=masc), (person=2nd). Most current theories make no longer a distinction between features and values : values are themselves features. Thus, from this point of view, it is possible to define a complex feature system represented as an acyclic graph or as an oriented tree :
(agreement=((gender=masc),(number=plural)))
This approach is adopted in Lexical Functional Grammar (LFG) [Kaplan 82], Generalized phrase Structure Grammar (GPSG) [Gazdar and al 82] and Unification Grammars, among others and also in PATRII, a grammar for english [Shieber and al. 83].

We now define the set L of features for a given language. L is a finite set of distinct constants vi whose nature is language dependent. For instance, for french, we have syntactic features such as : singular, feminine, 1st, 3rd, ... We may also have semantic features (human, abstract idea, ...), if we wish to have semantic filtering. This notion of semantic features is subject to

several controversies : their nature is not really well known and their number is a priori unbounded. Their role also is not very clear. However, it seems to me that semantic features can be used in an efficient way for natural language front ends, where the vocabulary and the underlying concepts are very limited [Dahl 77]. It is also possible to define a hierarchy between semantic features with a partial order relation associated with inheritance mechanisms. Works in cognitive science [Le Ny 80] confirm the interest of this approach. As our long term goal is to build interfaces that supports a natural language communication, we will also consider here the use of semantic features. More precisely, in our examples, we assume that to a verb is associated a list of pairs :
(sem-t-subj,list-of-sem-t-of-cpt)
where :
sem-t-subj is an acceptable semantic feature for the subject,
list-of-sem-t-of-cpt is the corresponding list of acceptable semantic features for complements.
we also assume that to an NP is associated a proper semantic feature.

A feature value may be a single value (complex or not), a disjunction of possible values, or the negation of a value : NOT(lst) means that the value associated with that feature is any value except the value *"lst"*. Several justifications about this point and examples are developed in [Karttunen 84]. As we will see later, these possibilities have consequences on the unification mechanism we need.

Another possible feature is the expression of the X-bar notation, with values : 1, 2, 3, lexical. The syntactic function (subject, object, ...) is also a feature in the sense given here but it has also other uses. In fact, the syntactic function is also used to build the resulting structure of the parsing process (ex. : f-structure in LFG), or to control some aspects of the syntactic analysis in Gapping Grammars [Dahl and Abramson 84], [Saint-Dizier 85].

3_ PROTOTYPE OF A DESCRIPTOR OF FEATURES.

We now give a structure to the set of features. We introduce the notion of preterminal feature, descriptor and prototype of descriptor of features for a given natural language.

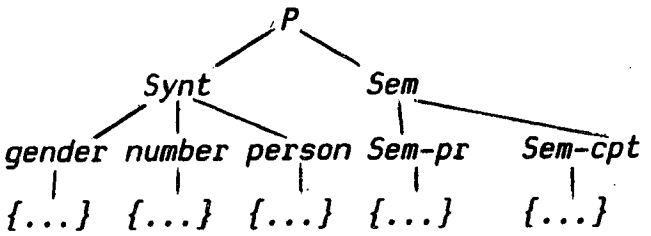
A preterminal feature P is a feature to which a value or a disjunction of values is directly associated. We think that features should be designed so as to be independent, and, in fact, they are naturally so in most cases. However, if, for a specific language, or a particular context, there are relations between features and their respective values, then integrity constraints can be formulated. These constraints are specified apart from the

rules themselves and intervene as a control process when descriptors are built.

The prototype of descriptors of the features of a given language is an oriented tree where :

- the root node is an arbitrary node we note FR,
- nodes are features. The lowest nodes are the preterminal features.
- each feature is used only once.
- to each preterminal symbol is associated the set of possible values it may have.

Example :

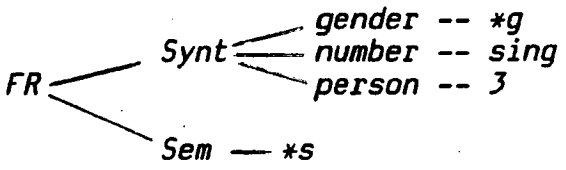


This prototype depends on the language considered and also, for natural language front ends, on the precision and the sharpness of the treatment. Then, once defined, the prototype of descriptors is fixed for the grammar and, in fact, act as a process that controls the coherence of descriptors when they are elaborated during the parse. It may be considered as a guideline for the description of feature structures in grammar rules, or, in programming terms, as a predefined typed structure, with sets of predefined values attached to its leaves.

Finally, to each terminal or non-terminal symbol in the grammar rules is associated a descriptor D which is a particular instance of the prototype of descriptor and where preterminal features are linked to precise values. However, and most importantly, a descriptor may be only partially instanciated : an unknown subtree is then represented by a variable. The variables we use within the logic programming framework are logical variables and their use has several meanings. Logical variables [Palmer and al 83] allow one to create structures with free constituents. These free constituents represent structures that have not yet been discovered, they may also mean that the subtree represented by a logical variable is any structure or of no present interest.

Finally, it is also possible to formulate constraints on such variables, and, in logic-based grammars, these constraints may percolate in other rules. In fact, logical variables and terms, associated with unification, are the major strength of logic-based grammars. From a computational point of view, using appropriately logical variables results in a gain in efficiency, because instanciated structures are used only when they are necessary.

Let's consider, for example, the lexical item *"the"*, its associated descriptor is :



(a * before a symbol denotes a variable indentifier) (FR is the root of the descriptor; FR= Feature Representation.)

4_ *PROPERTIES OF DESCRIPTORS.*

Theorems and properties given below follow straightforwardly from the definitions given above, thus, they will be stated without proofs.

Theorem 1 : The set of possible descriptors, completely instanciated or not, is finite. This follows from the fact that feature names and feature values are finite.

Definition 1 : A descriptor is empty if it is represented by a free variable.

Definition 2 : A descriptor is maximal if is it completely instantiated.

Definition 3 : To create an extension Dj of a descriptor Di is to limit its degree of freedom : a free variable is instantiated to a value or a disjunction of values is decreased of one or several elements.
More formally, for any descriptor Di let :
Pi be the set of preterminal symbols linked to a single value. Pi is a set of pairs (pt_i, v_i) where pt_i is the name of the feature and v_i, its value.
Fi be the set of symbols that dominate only a free variable.
LEi be the set of preterminal symbols linked to a disjunction of values.
Let Ei = Pi U Fi U LEi.
Then, Dj is an extension of Di (noted EXT(Di,Dj)) iff for all e_{ik}, element of Ei, one of these conditions is true :
(i) if $e_{ik} \in$ (Pi U LEi), then $e_{jk} = e_{ik}$
(ii) if $e_{ik} \in$ LEi, $e_{jk} \subset e_{ik}$
(iii) if $e_{ik} \in$ Fi, then ($e_{jk} \in$ Fi) or ($e_{jk} \in$ (Pi U LEi)) or e_{jk} is the root of a non-elementary tree.

Definition 4 : Dm belongs to the set of minimal extensions of Di iff :
$\exists$ Dj such that EXT(Di,Dj) and EXT(Dj,Dm)

Definition 5 : Two descriptors Di and Dj are equal iff EXT(Di,Dj) and EXT(Dj,Di).

Theorem 2 : The empty descriptor is unique whereas the set of maximal descriptors is finite and equal to the set of all the possibilities of distribution of values on adequate preterminal symbols.

Theorem 3 : for all Di : EXT(Di,Di).

Definition 6 : Two descriptors Di and Dj A-unify (noted A-Unify(Di,Dj) for *"augmented unification")* iff they unify except for the subtrees in Di or Dj that are either disjunctions or negations. In this case, the classical notion of unification has to be augmented . Then we say that :

- a subtree A-unifies with a disjunction of subtrees if it A-unifies with one subtree of the disjunction.
- a subtree A-unifies with the negation of a subtree iff it does not A-unify with this subtree.
- two disjunctions A-unify iff A-unification is a bijection between these two sets of elements.
- two negations A-unify iff the subtrees dominated by the negation A-unify.

Notice that A-unification on subtrees dominated by a negation is computed as late as possible in order to have as less free variables as possible in these subtrees.

Theorem 4 : A-unification is reflexive, transitive and not symmetric.

Theorem 5 : If Di and Dj A-unify, then the result of the unification process is a descriptor Dk such as :
(i) EXT(Di,Dk)
(ii) EXT(Dj,Dk)
(iii) Dk is the minimal common extension of Di and Dj.

Theorem 6 : Dk, as defined above, is unique. This theorem is essential for the coherence of the system.

5 INSTANTIATIONS AND COMPARISONS OF DESCRIPTORS :

We now define functions for instantiations and comparisons between descriptors. These functions are presented here, for more convenience, as PROLOG predicates. One main feature of these functions is that they are designed to apply on completely or partially instantiated descriptors as well as on parts of descriptors (proper subtrees we call sub-descriptors).

When they are used in grammar rules, some functions can be viewed as the expression of constraints on feature values. Other functions are used to build the resulting descriptor of the structure described by the rule. They quite directly encode the

main functions stated by Gazdar [Gazdar and al. 82] : the head feature convention, the control agreement principle and the foot feature principle. Up to now, in Definite Clause Grammar (DCG) [Pereira 80], Extraposition Grammar (XG) [Pereira 82] and Gapping Grammar (GG) [Dahl and al 84], features were represented by distinct variables, and most of them in distinct arguments. The relations between features were directly encoded in the arguments : if two distinct symbols have the same value for a given feature, then the same variable identifier appears in the arguments of the respective symbols. These means of expression are insufficient and expensive in processing time.

A very different point of view to describe relations between features is that of [Miyoshi and Furukawa 85]. They propose an object-oriented parsing mechanism for the logic programming language ESP. Each component of the parser is abstracted as a class and access between classes is performed by a message-passing mechanism. Features, viewed as slots, are described as particular instances of category classes. Another point of view is that of [Uehara and al. 85] designed for LFG. They view parsing, and especially control procedures, as passing messages among actors. Each context free rule corresponds to an actor. Features are represented by predefined structures. A distinction is made between two kinds of schemata : defining schemata and constraining ones, which roughly correspond to our inheritance mechanism and expression of constraints. In both approaches, the mechanisms they have defined are transparent to the user, but the power of expression for feature instantiations and control and the elegance of the "surface " grammar remain unchanged. We view these two works as the specification of efficient and reliable interpretors for grammars.

We now define the functions we need to completely define grammar rules :

Union of two descriptors :
UNION(<feature name>,<descriptor D1>,<descriptor D2>).
If the subtrees s1 and s2 of <descriptor D1> and <descriptor D2> with root node <feature name> A-unify, then <descriptor D1> is rewritten into a new descriptor where s1 is replaced by the minimal common extensions to s1 and s2. The remainder of <descriptor D1> remains unchanged. If the A-unification is not possible, then the predicate UNION is evaluated to false and the rule is not applicable. This predicate is used to build the descriptor of a structure from the descriptors of the subtrees it is composed of. As we explain in section 6, this resulting descriptor is not always a strict extension of the descriptors from which it is built. Classical compositionality applied to features has to be extented so as it takes into account *"contextual syntactic information"*, in other terms, informations about the syntactic structure of other constituents in the sentence and on the position of constituents in the sentence itself.

Coordination of two descriptors

COORD(<descriptor D1>,<descriptor D2>,<descriptor D3>).

This function, language dependent, is specifically designed for coordination. COORD has not to be confused with the intersection (or generalization) process defined in [Pereira and al 84] which is, in our terminology, the most instantiated descriptor containing all the elements the two descriptors have in common. This operation is not directly useful in itself for our purpose, it has rather to be considered as a by-product of COORD.

When COORD is applied, <descriptor D3> is the result of the partial functions applied on specific features of <descriptor D1> and <descriptor D2>. If at least one of these functions cannot be applied then COORD is evaluated to false. Some of these partial functions are conditions of A-unification. For example, two structures that have a proper semantic feature have to A-unify for this feature (or, more generally, to be in relation if there exists an hierachy of semantic features). Another function is the construction of the intersection of two disjunctions. Finally, the last class of functions are those of the form :

Fct(f1,f2,f3)

where the fi represent subdescriptors or simply values of preterminal features. f3 is the result of the application of the function Fct. A very simple example is coordination in french, where masculine prevails on feminine. Thus, we have :

Fct(masc, fem, masc).

This predicate is used to make feature checking in coordination as well as to build the resulting descriptor of the structure that dominates the coordination.

For coordination, L. Karttunen [Karttunen 84] proposes a mechanism based on generalization, linguistically sound for english, but not for other languages, such as arabic, that justify the complex apparatus developed here. He uses artifices (negative constraints) for the generalization process to work properly, these artifices make the system less clear and more specific to english. In addition, generalization cannot be applied directly on semantic features because there exists an hierarchical relation between semantic feature values. For instance, from human + animate it does not result any feature (a variable) but the feature animate [Dahl 77].

Transformation of features :

TRANSF(<feature>,<D1>,<new-subtree>,<D2>).

where Di are descriptors. This predicate is always evaluated to true. D2 is equal to D1 except for the subtree with root node <feature> which is equal to <new-subtree>. This function is used for modifications of features when constituents are *"moved"* in a sentence.

Equality :

AUNIF(<feature>,<D1>,<D2>).

This function is evaluated to true if the sub-descriptors of D1 and

D2 dominated by the node <feature> A-unify. This function is often used for feature agreement. D1 has to A-unify with D2 or conversely.

Inclusion :
INCL(<feature f1>(D1),<feature f2>(D2)).
This function is used for features agreement. It is evaluated to true if the subdescritor with root node <feature f1> of D1 A-unifies with the subdescritor with root node <feature f2> of D2.

6_ HOW TO WRITE GRAMMAR RULES WITH DESCRIPTORS :

In this section, we give examples of rules for french. What we want to show is that the functions we have defined above can be used in the spirit of a declarative programming language, which is really convenient for grammar rule writers. The main principle is modularity in rules, thus, we have the following parts in a rule :

- the derivation part itself,
- the agreement control,
- the construction of the resulting descriptor(s).

The same principle remains valid for other aspects, not mentioned here, such as the construction of the output representation. The programming style is that of logic programming. Finally, the use of the type of functions defined above allows the grammar writer to make a distinction between descriptors of symbols that appear in the left and in the right hand part of the rule. Thus, this makes easier to make the system of rules applicable forward as well as backward, for natural language synthesis. Descriptors are represented by logical variables which make the grammar independent of the parsing strategy. The rules are here classical DCG, augmented by an argument that represents the descriptor .

```
NP(*D) ---> Det(*D1)  Adj(*d2)  Noun(*D3)
            AUNIF(FR,*D1,*D3)      (Notice that FR is the root node
            AUNIF(FR,*D2,*D3)       of the whole descriptor)
            UNION(FR,*D,*D3).

VP(*D) ---> V(*D1)  NP(*D2)  PP(*D3)
             INCL(*D1(sem-cpt),*D2(sem-pr))
             INCL(*d1(sem-cpt),*D3(sem-pr))
             UNION(R,*D1,*D).
```

For state verbs followed by a verb in the infinitive form in french, we have the rule :

```
V(FR(*d3,*d4)) ---> V_ETAT(*D1)  V(*D2)
        AUNIF(nature-verb,nature-verb(state),*D1)
        AUNIF(mode,mode(infinitive),*D2)
        AUNIF(sem-pr,*D1,*D2)     (The 2 verbs have the same semantic
        UNION(Synt,*d3,*D1)        feature for an acceptable subject)
        UNION(Sem,*d4,*D2).       (The VP inherits of the syntactic
  features of V_ETAT and the semantic features of V it
  dominates.)
```

7_ FEATURES AND LONG DISTANCE DEPENDENCIES :

We now consider the problem of the transmission and, in some cases, of the variation of features when constituents are *"moved"* in a sentence, depending on the new position occupied by these constituents. Although the terminology used here is often that of the transformational theory, what we present here is applicable to non-transformational theories as well, since the formalism of the logic-based grammar we consider is that of Gapping Grammars. In the remainder of this section, we first present Gapping Grammars, then consider the problem of feature tranmissions and modifications in long distance dependencies, and finally show how features may intervene to limit the freedom of expansion of gaps.

7.1_ Gapping Grammars :

Gapping Grammars (GG) are a generalization of the Metamorphosis Grammars (MG) [Colmerauer 78], [Dahl 77], Definite Clause Grammars (DCG) [F. Pereira 80] and Extraposition Grammars (XG) [F. Pereira 83]. A GG rule allows one to *"indicate where intermediate, unspecified substrings can be skipped, left unanalysed during one part of the parse and possibly reordered by the rulés application for later analysis by other rules" [Dahl and Abramson 84], [Dahl 84].* The left hand part of a GG rule is composed of a non-terminal symbol followed by any string of non terminal and terminal symbols and gaps. PROLOG calls can also be used. The right hand side of a GG rule is a string of non terminal and terminal symbols, of gaps in any position and PROLOG calls. GG are a powerfull formalism that can be used to describe in a elegant and concise way complex natural language sentences as well as formal languages. A Gapping rule is of the form :

$$\alpha_1\ gap(x_1)\ \alpha_2\ gap(x_2) \text{ ---- } \alpha_n \rightarrow \beta$$

where :

$\alpha_1 \in V_N$

α_i , $i \in]1, n]$, $\alpha_i \in V_N \cup V_T$

$x_i \in V_T^*$

$G = \{ gap(x_i) , 1 \leq i < n \}$

$\beta \in (V_N \cup V_T \cup G)^*$

For example, the left extraposition of an adjective in french: "Cette jolie demeure agréable ... "
(this nice, pleasant home ...) which becomes : "Agréable, cette jolie demeure ... "
is represented by the following GG rule :

Det Gap(X) Adj ---> Adj [,] Det Gap(X).

The basic form appears in the left hand part of the rule while the form with the movements appears in the right hand part. In fact, the arrow ---> has to be thought of as : <---.

Gapping Grammars are a notational tool to express movements of constituents or long distance dependencies. They are not transformational grammars. We can say, in fact, that the explicit constituents of a GG rule are in relation and that the type of relation involved is directly expressed by the GG rule. Gaps in Gapping Grammars has not to be confused with gaps or trace in linguistics. The term gap represents here variable that abbreviate symbols in grammar rules.

7.2 Feature transmissions and modifications :

Gapping Grammars adopt a classical approach to account for a *"missing"* constituent X in a position P which is to posit that at some other level, X is in this position P. These two levels are respectively represented by the right and the left hand parts of a GG rule. Thus, such a representation can account for feature transmissions without positing an additional abstract derivational level. This is sufficient (1) since a GG rule contains a symbol X of an appropriate type in the *"extraction site"* position and (2) since features, like case-functions, are not only considered as being properties of specific nodes but as being properties of all the nodes dominated by these specific nodes. For instance, if an adjective belongs to an NP subject, then it inherits of the function subject. Conversely, features of terminal symbols may percolate up in the derivational tree.

In GG, it is possible to represent transmissions of features, modifications and inheritance mechanisms, by the functions defined above which can apply on any two descriptors. In

addition, inherited features can percolate in other GG rules when several *"transformations"* are involved. This gives a little more power to GG than to classical GPSG. In GPSG, the dependence between the *"extracted"* node and the extraction site is captured by derived categories. One problem is that it has never been made explicit how features interact with the rule deriving machinery.

Let's now consider the main phenomena that arise when constituents are moved in a sentence. The first phenomenon is direct transmission of features. This is done in GG rules by using the same variable to denote the descriptor of the *"moved"* constituent in the right and in the left hand part of the GG rule. It is also possible to use, for more clarity, the function UNION. Thus, the left extraposition of an object NP is represented by the following GG rule :

NP(*D1) V(*D2) gap(X) NP(*D3) ---> NP(*D3) NP(*D1) V(*D2) gap(X).

Next, inheritance mechanisms are expressed in GG rules by the same means and for the same reasons than in rules without gaps. Inheritance mechanisms can operate on any two partial or complete descriptors. An interesting example that illustrates the use of inheritance mechanisms is what P. Jacobson [Jacobson 82] calls the strong connectivity. Strong connectivity occurs *"when properties of material dominated by the extracted node are determined by the position of the extraction site. "* She gives the following example, where the case marking on the fronted pronoun is determined by the position of the extraction site :

* *"Who did John see ?"*
"Whom did John see ?"
"Who did John say was coming ?"
* *"Whom did John say was coming ?"*

The use of "who " or "whom" is accounted for by the NP's case-marking. When it has the case "obj", "whom " is generated and when it has the case "subj", "who " is produced. The way how to represent this phenomenon by a GG rule rises an interesting, well known, methodological problem. The question is to what extend should one use features to describe properties of constituents. If we consider, in the example above, the way how to represent the fact that an explicit NP has been transformed into a wh-pronoun, we can express this in two distinct ways in GG:

- by a new symbol, "wh-pronoun", that inherits of the features of the NP it represents:

*NP(*D1) V(*D2) gap(X) NP(*D3) --->*
*wh-pro(*D3) Aux(*D4) V(*D1) gap(X)*
*UNION(Synt,*D4,*D5)*
*UNION(FR,*D5,*D2)*
*TRANSF(mode,*D5,(mode=infinitive)).*

(Notice that, in addition, Aux inherits of the syntactic features of V, and V is itself put in the infinitive form.)

- by a feature "expression" with possible values: explicit, wh-pronoun, ... "Pronominalization" is here considered as a feature. The GG rule is then:

*NP(*D1) V(*D2) gap(X) NP(*D3) --->*
*NP(*D6) Aux(*D4) V(*D5) NP(*D1) gap(X)*
*UNION(Synt,*D4,*D5)*
*UNION(FR,*D5,*D2)*
*UNION(FR,*D6,*D3)*
*TRANSF(mode,*D5,(mode=infinitive))*
*TRANSF(expression,*D6,(expression=wh-pro)).*

This rule is equivalent to the previous one. However, the mean of expression illustrated by this latter rule has several practical and theoretical advantages for GG. First, we think that it is closer to the linguistic reality and easier to understand. Next, and most importantly, it permits to express with much more clarity and efficiency constraints on long distance dependencies [Saint-Dizier 85] in GG with a higher economy in non terminal symbols. Finally, we will see in the next section that features may be used to limit the freedom of expansion of gaps, whereas symbols may not.

Thus, we think that the use of features should be pushed to its limits for all the factors that may changed when constituents are moved. What is then crucial to notice is that nodes whose features differ are not nodes of the same type.

A more complex case is the agreement of a verb conjugated with the auxiliary *"avoir"* (to have) in french. This verb agrees with the object NP gender and number only if the object complement occurs before it, otherwise the verb is not agreed, i.e. it is in the neutral form. Example :

"J'ai vu des personnes. " (I saw people)
which becomes:
"Les personnes que j'ai vues. "
A solution to this problem is to consider that, depending on the position of some constituents, some of the features of other constituents (or of these very constituents) are inhibited. This can be realized by adding an argument to nodes of a descriptor which specifies if the features dominated by this node are inhibited or not. If they are inhibited, then the constituents involved are in the neutral form. The function TRANSF is used here with a slightly different role : it transforms a subtree into the same subtree except that the *"inhibition"* argument is changed. The corresponding GG rule is:

*NP(*D1) V(*D2) gap(X) NP(*D3) --->*
*NP(*D3) Rel_pr NP(*D1) V(*D4) gap(X)*
*UNION(FR,*D4,*D2)*
*TRANSF(mode,*D2,(mode(inhibited)=!!))* *etc...*

(This example is given for anlysis, i.e., the rule should apply in the opposite side of the arrow --->)
(!! specifies that the subtree dominated by mode remains unchanged.)

Finally the case argument can also be used to constrain the application of a rule so as to prevent incorrect derivations. This adjunction is necessary, for instance, to allow the elision of the relative pronoun "who " in a sentence such as :

"The student I saw left. "
but not in :
** "The student saw me left. "*
where "who " is obligatory :
"The student who saw me left. "

This means that only relative pronouns that replace NP complements may be elided. The corresponding GG rules, where we only make appear the case argument, are :

*Rel-marqueur gap(X) NP(*case) --->*
*Rel-pronoun(*case) gap(X).*
Rel-pronoun(subj) ---> [who].
Rel-pronoun(obj) ---> [who] / [].
(Terminal symbols are written between square brackets.)

7.3 Using the case argument as a restriction on the expansion of gaps:

It may be convenient to add a descriptor argument to gap variables themselves. If we consider gap(X) as an element of the non-terminal vocabulary, we can then add arguments to it. We will here focus on the case argument of descriptors. Adding a case argument to gaps implies that gaps represent any string of symbols that have a specific case-function, or that belong to a set of specific case-functions, if the case-argument may be a list.

The semantics of a case argument in a gap is that the nearest symbol, up in the tree, which dominates entirely the gap and to which a case function is assigned, has to have the function stated in the case argument of that gap.

We can express that a given gap has a specific function, i.e. the case argument is instantiated. We can also say that a gap has an unknown function but which is equal (or different) to that of another constituent or another gap. This can be done because the

variables we use are logical variables [Palmer 83]. A gap is then noted :

gap(X(*case))
(For more clarity, we only represent the case feature.)

An illustration of this point is the extraposition of an adjective in a subject NP in french :

"Cette belle maison confortable est l'orgueil de ses proprietaires. "
(This nice, comfortable house is the pride of its owners.)
which becomes :
"Comfortable, cette belle maison est l'orgueil des ses proprietaires. "

To prevent the extraposition of adjectives in object NPs, we can then write the following rule :

Det Gap(X(subj)) Adj(subj) --->
Adj(subj) [,] Det gap(X(subj)).

It is important to note that, even if we allow the case argument of a symbol to be a list (of functions it can or it cannot be), we do not describe ipso facto the complement of the language expressed by the rules. We only add an argument that plays the role of a restriction. The approach presented here can be extended to all the other features and, thus, we can use GG rules, for instance, for the reconstitution of quasi-elliptical constructions in a sentence.

7_ CONCLUSION :

In this paper, we have breifly presented a complete system for the description of features and features assignment functions in logic-based grammars. The tools defined here are presented in the spirit of a declarative programming language. We have also shown how features are used in long distance dependency relations and how they can limit the expansion of gaps in Gapping Grammars.

We think that this formalism is adaptable to other grammar systems, different of those presented here, with minor adaptations. Unification on terms that contain logical variables is the basic operation, from which all the operations we have described may be defined. Once this level of specification defined, the next stage is to develop efficient mechanisms for the execution of the grammar rules.

REFERENCES

[Colmerauer 78] A. COLMERAUER Metamorphosis Grammars. In Bolc Edt Natural language communication with computers. Springer Verlag.

[Dahl 77] V. DAHL Un systéme déductif d'interrogation de banques de données en espagnol. Thése Université , Marseille-Luminy. GIA.

[Dahl and Abramson 84] V. DAHL, H. ABRAMSON On Gapping Grammars. Proc of the 2nd logic programming conf. Uppsala Univ.

[Dahl 84] V. DAHL More on Gapping Grammars. Conf. on Fifth Generation Computer systems 1984, Tokyo.

[Gazdar 82] G. GAZDAR Phrase Structure Grammars, in *"The nature of syntactic representation"* Jacobson and Pullum Edts, D. Reidel Pub.Co.

[Gazdar and al. 82] G. GAZDAR, G.K. PULLUM Generalized Phrase Structure Grammar : a Theoretical Synopsis. Research Report, Univ Of Sussex

[Gross 84] M. GROSS Lexicon Grammars, Proceedings of COLING-84.

[Jacobson 82] P. JACOBSON Evidence for gaps. In P. JACOBSON and G.K. PULLUM Edts *"The nature of syntactic representation " Jacobson and Pullum Edts.* D Reidel Pub.Co.

[Kaplan 82] KAPLAN R. M. LFG : A formal system for grammatical representation. In The mental representation of grammatical relations, J. Bresnan Edt. MIT Press.

[Karttunen 84] L. KARTTUNEN Features and Values. In Proceedings COLING-84.

[Kay 85] M. KAY Unification in Grammar. In *"Natural language understanding and logic programming. "* V. DAHL and P. SAINT-DIZIER Edts. North-Holland pub. co.

[Le Ny 80] J.F. LE NY La sémantique psychologique. PUF, Paris.

[Miyoshi and Furukawa 85] H. MIYOSHI and K. FURUKAWA Object oriented parser for the logic programming language ESP. In "Natural language understanding and logic programming. V. DAHL and P. ST-DIZIER Edts. North Holland Pub. Co.

[Pereira and Warren 80] F. PEREIRA, D. WARREN Definite clause grammars for natural language analysis. A survey of the formalism and a comparison with ATN. Artificial intelligence no 13.

[Pereira 83] F. PEREIRA Logic for natural language analysis. SRI international technical note no 275.

[Pereira and al. 84] F. PEREIRA, S. M. SHIEBER The Semantics of Grammar Formalisms Seens as Computer Languages. In Proc. COLING-84.

[Saint-Dizier 85] P. SAINT-DIZIER Long distance dependency constraints in Gapping Grammars. Research report INRIA-IRISA Univ. of Rennes 1.

[Shieber and al. 83] S.H. SHIEBER, H. USZKOREIT, F. PEREIRA, J. ROBINSON, M. TYSON The formalism and implementation of PATRII. SRI report 1894.

[Uehara and al. 85] K. UEHARA, R. OCHITANI, O. MIKAMI, J. TOYODA An integrated parser for texte understanding: viewing parsing as passing messages among actors. in "Natural language understanding and logic programming. " V. DAHL and P. ST-DIZIER Edts. North Holland Pub. Co.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique