An LFG parser for Basque (II)

JOSEBA ABAITUA
(University of Manchester.
Institute of Science and Technology)

3. Further structures

This third chapter is an extension of the previous two chapters. In the first place, the periphrastic conjugation of verbs is analysed. The formalization of this part of the grammar is essential for any system aimming at parsing common sentences of Basque, as in this language most verbs are conjugated periphrastically. From this point of view, it will be shown that an LFG approach presents an obvious advantage, that of its simplicity. Finally, in the second part of the chapter, an attempt at sketching an LFG version of facts related with the focus position which was debated in the prior chapter is presented.

3.1. *Auxiliary Verbs*

This section outlines an analysis of Basque auxiliary verbs within the LFG framework. First, a few comments from Wilbur and Rebuschi are offered, seeking to display the complexity that a formalization of the periphrastic conjugation of verbs in Basque might lead to. The status of these auxiliary verbs, on the other hand, is rather transparent, as is revealed by the exposition below, taken from Trask. The approach used here to these auxiliaries is basically the same as that of Falk's for *do* in English (Falk, 1983: 504).

Wilbur (1979: 30) refers to verb complex (VC) as a category which simplifies the description of the behaviour of the Basque verb. He comments that «the greater number of verbs are composed minimally of two elements: a non finite form of the lexical verb, plus an auxiliary, or finite form, that incorporates pronominal indications of ergative, absolutive and dative noun phrases». Rebuschi (1983: 39) likes the term 'verb complex' because it is useful for practical and descriptive purposes: «For the moment, let us note that the main verb and the finite auxiliary form a block, called 'verb complex'», although, he admits, «I am not sure whether it can have theoretical status». Rebuschi adopts, with a few modifications, Goenaga's (1980) formalization for a verb complex rule, as in (1):

(1) VC → V + AUX
AUX → (ASP)INFL
ASP → Perfective
Imperfective
Prospective

Rules in (1) are glossed by Rebuschi as follows: «AUX = auxiliary; ASP = aspect; INFL = inflection: tense and person makers. Synthetic forms are obtained when ASP is not selected; i.e. when no affix separates V, the main verb here, from INFL it can be directly conjugated. But if ASP is selected, INFL cannot be attached to it, as an auxiliary has to be introduced as a tense (etc.) carrier (it is a form of 'be' if there is no argument in the ergative case, and a form of 'have' otherwise, i.e. the selection of either auxiliary v. that of the other has nothing to do with aspect, contrary to the situation in English, c.p. have (+EN) and be (+ING)».

The general question of how auxiliaries can be identified has been adressed by Steele (1978), Akmajian, Steele and Wasow (1979), and Steele et al (1981). They propose that auxiliaries can be identified by the following criterion:

> i) Auxiliaries typically cover a certain semantic range, including the expression of tense, aspectual and mood distinction, as well as the modal notions of necessity, obligation, permission, ability, and others.

Moreover, auxiliaries generally observe the following properties:

> ii) Auxiliaries frequently carry marks of subject (and possibly object) agreement in preference to main verbs.
>
> iii) Auxiliaries often bear reduced stress, and may become clitized.
>
> iv) Auxiliaries typically occupy a fixed position in the sentence.

Following Trask (1984: 214-219), we can state that these four characteristics clearly identify the class of items traditionally known in Basque as auxiliary verbs. These are the various forms of *izan* (whose commonest form is *da 'is'*), which, apart from its auxiliary uses, also functions as the ordinary copular verb 'to be'; and the various forms of the (hypothetical) verb **edun* (whose commonest form is *du* 'has'), which also functions as the ordinary verb 'to have', with its suppletive citation forms *ukan* and *izan*. These two verbs, when used as auxiliaries, exhibit all the properties i) to iv):

> i) Semantics: They mark a preterite/non-preterite tense distinction, as well as several mood contrasts: indicative/subjunctive/potential/ contingent. They do not mark, however, aspectual contrasts, as these depend on the participial (or infinitival) choice of the non-finite form of the verb.

ii) Agreement: They are, when present, the sole bearers of subject and object agreement.

iii) Phonological reduction: For the non-standard dialects of Basque, the auxiliaries undergo a variety of phonological reductions, which vary from dialect to dialect.

iv) Position: In affirmative sentences, the auxiliaries must immediately follow the main verb, and in negative sentences they must immediately follow the negative particle.

This is to say that for a common verb in Basque which is, like *ekarri dio* 'he has brought it to im', conjugated periphrastically, the information about tense, mood and aspect is shared by both elements, the non-finite verb *ekarri*, and the auxiliary *dio*.

The non-finite verb bears aspect marks which differ depending on which of the following alternative participial forms is selected:

(2) a. present participle or imperfective: *ekartzen*, (↑ PERF) = —
b. past participle or perfective: *ekarri*, (↑ PERF) = +.
c. future participle or prospective: *ekarriko*, (↑ FUT) = +.

These features will combine with tense marks introduced by the auxiliaries:

(3) a. present: *dio*, (↑ PRET) = —.
b. preterit: *zion*, (↑ PRET) = +.

Which will yield the eventual 'tense' for a sentence, as follows:

(4) [(PERF) = —, (PRET) = —] → PRESENT
[(PERF) = —, (PRET) = +] → IMPERFECT
[(PERF) = +, (PRET) = —] → PERFECT
[(PERF) = +, (PRET) = +] → PRETERIT
[(FUT) = +, (PRET) = —] → FUTURE
[(FUT) = +, (PRET) = +] → HYPOTHETICAL

which are all the six common tenses described in most of the grammars for Basque.

Auxiliaries also mark several mood contrasts, so their various lexical entries could be annotated with features such as:

(5) a. indicative: *dio*, (↑ IND) = +.
b. subjunctive: *dièzaion*, (↑ SBJUN) = +.
c. potential: *diezaioke*, (↑ POT) = +.

Subjunctive and potential auxiliaries do not combine with participial non-finite verbs (i.e. (2) a. to c.), but with infinitives, like *ekar*, (↑ INFIN) = +. There should be, therefore, constraining equations on the non-finite verbs, such as (↑ IND) = a^+ in all the participial

forms, and (↑ IND) = c^- in the infinitives; and similarly, subjunctive and potential auxiliaries could be annotated with constraining equations, such as (↑ INFIN) = c^+, to ensure that only the rigth combination of non-finite verbs and auxiliaries will take place, that is, indicative auxiliaries with participial forms, and subjunctive or potential auxiliaries with infinitives.

Unlike tense, aspect and mood, lexical semantic information is not shared by these two elements. In the periphrastic conjugation of verbs in Basque, the lexical semantic value of the verb, together with its predicate argument structure, is introduced solely by the non-finite form:

(6) *ekarri* : V, (↑ PRED) = 'bring <(SUBJ) (OBJ) (OBJ2)>'

Auxiliaries, on the other hand, have no (↑ PRED) feature, that is, they define no clause nucleus, but they have a feature (↑ AUX) for which the values 'du' or 'da' can be given.

(7) *dio* : AUX, (↑ AUX) = 'du'.

Besides tense and mood, the predicate argument structure of the lexical form of the verb determines for the most part the choice of the auxiliary. (Other relations within the sentence are also influential, i.e. number and person agreement between finite form and nominal arguments, see below). The choice of the auxiliary is determined by the non-finite lexical form according to the arguments it subcategorizes for, i.e. according to whether (SUBJ), (OBJ), or OBJ2) are in the verb's lexical form, as we know that in Basque verbs agree not only with their subject, but also with their direct object (if any), or second object (if any). All verbs that have (OBJ), or (SCOMP), in their lexical form (wich implies the subject to be marked with the ERG case), require the auxiliary to be the transitive 'du'. Otherwise the required auxiliary is the intransitive 'da' (and the subject is marked with the ABS case) [1]. So, there should be a redundancy rule such as (8):

(8) «If a verb's lexical form contains (↑ SUBJ CASE) = ERG, then add (↑ AUX) = 'du'; else, add (↑ AUX) = 'da'».

Similarly, whenever (OBJ2) is present in a verb's lexical form, both 'du' and 'da' inflect in agreement with this argument marked dative. (See examples (9) a. and (9) e.).

Moreover, auxiliaries are sensitive to number and person of the arguments they agree with, as shown by the following examples:

(1) There are some exceptions to this general fact. Certain exceptional Basque verbs require the auxiliary to be the transitive 'du' even though they can take no (OBJ). These verbs are called «deponent» by Lafitte (1944: 189), and Trask (1981, 1984: 84-87) has reviewed them, making important considerations. Following Trask, we will treat these verbs as exceptions. In consequence, auxiliaries combined with these verbs will not be matched by general redundacy rules, such as «if a verb's lexical entry contains (OBJ); then add, (↑ SUBJ CASE) = ERG, and (↑ OBJ CASE) = ABS», but by an equation introduced by their lexical entry, merely (↑ SUBJ CASE) = ERG.

(9) a. *dizkizugu* [(SUBJ,PL,1), (OBJ,PL,3), (OBJ2,SING,2)]
b. *zaitugu* [(SUBJ,PL,1), (OBJ,PL,2)]
c. *ditugu* [(SUBJ,PL,1), (OBJ,PL,3)]
d. *gara* [(SUBJ,PL,1)]
e. *gatzaizu* [(SUBJ,PL,1) (OBJ2,SING,2)]

When all three main arguments appear in the verb's lexical form, as in (9 a.), the (OBJ) argument can only stand for third person objects, this being the only restriction in the conjugation. As example (9 b.) shows, such a restriction is not imposed when (OBJ2) is not present. Moreover, neither (SUBJ), nor (OBJ2) ever undergo this restriction. So, if we had the auxiliary 'dizkiguzue', the information available in its lexical entry would be as complete as follows (6):

(10) *dizkiguzue* : AUX, (↑ AUX) = du, (↑ PRET) = —,
(↑ IND) = +, (↑ INFIN) = c—,
(↑ SUBJ CASE) = c ERG, (↑ SUBJ NUM) = PL,
(↑ SUBJ PERS) = 2,
(↑ OBJ CASE) = c ABS, (↑ OBJ NUM) = PL,
(↑ OBJ PERS) 3,
(↑ OBJ2 CASE) = c DAT, (↑ OBJ2 NUM) =PL,
(↑ OBJ2 PERS) = 1.

The equations for case are constraining equations, since the information about cases is not introduced by the auxiliary, but by means of redundancy rules on the node of the lexical form (and are subject to alterations). The redundancy rule proposed in section (2.3) was as follows:

(11) «If a verb's lexical form contains (OBJ), then add (↑ SUBJ CASE) = ERG, and (↑ OBJ CASE) = ABS, else add (↑ SUBJ CASE) = ABS. Whenever it contains (OBJ2), add (↑ OBJ2 CASE) = DAT». (See section 2.3.).

AUX is regarded as a minor category, i.e. a category with no projection (according to X-bar theory), but which like V, head of the VP category, carries the ↑ = ↓ equation (as in PSRs (14)), and hence transmits the functional information encoded in the lexical item that it dominates (e.g. (10)).

For standard Basque, which is the variety we are most concerned with, the formalization of the auxiliary has recently been achieved by the Basque Academy of Language, Euskaltzaindia (1973). Almost three thousand forms are brought together there, and is our claim that all of them could easily be represented following the patterns described in this section. Is obvious, however, that in further research the morphological component of the theory should also be able to deal with auxiliaries, as the conjugation of auxiliary verbs in Basque obeys quite a regular pattern.

Thus, for a sentence like (12):

(12) *Apurtu dizkiguzue lehioak*
Break Aux-you-us-it windows-ABS
'You have broken us the windows'

With the following entries:

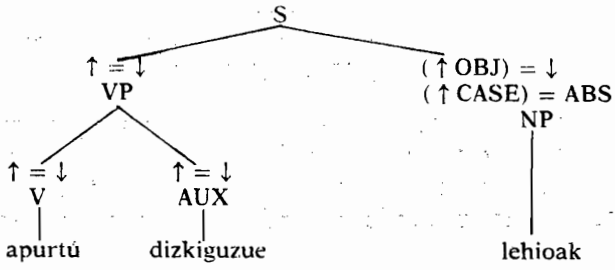
(13) *apurtu* : V, (↑ PRED) = 'break (SUBJ) (OBJ) (OBJ2)',
(↑ PERF) = +, (↑ IND) = c +. (↑ INFIN) = —.
lehioak : N, (↑ PRED) = 'window', (↑ CASE) =ABS,
(↑ NUM) = PL, (↑ PERS) = 3.

And the PSRs a sin (14):

(14)
$$\begin{array}{lcccc} S \rightarrow & XP^* & VP & XP^* \\ & & \uparrow = \downarrow & \\ VP \rightarrow & (XP) & V & AUX \\ & & \uparrow = \downarrow & \uparrow = \downarrow \end{array}$$

The instantiation of both (13) and (14) would yield the following c- and f-structures (15) and (16):

(15)



(16)

SUBJ	CASE	ERG
	NUM	PL
	PERS	2
	PRED	'PRO'
OBJ	CASE	ABS
	NUM	PL
	PERS	3
	PRED	'window'
OBJ2	CASE	DAT
	NUM	PL
	PERS	1
	PRED	'PRO'
AUX	DU	
PRET	—	
PERF	+	
IND	+	
INFIN	+	
TENSE	PERFECT	
PRED	'break <(SUBJ) (OBJ) (OBJ2)>'	

As shown by (16), when any of the arguments in the verb's lexical form is absent in the surface structure of a sentence, it will still be reflected in its f-structure due to the information contained in the auxiliary's lexical entry (e.g. (OBJ2) and (SUBJ) in (16)). Such representations of missing arguments can be regarded as anaphoras.

From (16) we also learn that categories such as ASP and INFL are unnecesary to convey the information that the verb contains in relation to the whole sentence. Moreover, the approach which groups together ASP and INFL under the AUX category has another obvious inconvenience. In Basque, as stated previously, the aspectual information is not provided by the auxiliary, but rather by the non-finite verb, according to whether it is perfective, imperfective or future participle.

Besides this, it has been shown that there is no need to postulate a category such as VC; or in other words, anything that covers just V and AUX, and not constituents in focus. The VP category is sufficient for descriptive purposes and more appropriate from the linguistic point of view. A VP category covering V, AUX and constituents in focus seems, therefore, well established in Basque.

3.2. *Syntactic Binding*

This section presents an analysis of constituent control (syntactic binding) within the scheme developed by Kaplan and Bresnan (1982: 231-263) and Zaenen (1983: 469-504). We will see that the principles of LFG provide a reasonable account of syntactic binding phenomena, although certain modifications of Zaenen's proposals are suggested to accommodate some features of Basque syntax.

3.2.1. *Syntactic Binding and the Focus Position*

As sketched above, in Basque there is a structural position for focused elements which is defined in terms of the verb. Constituents appearing in this fixed preverbal position when focused would be generated otherwise somewhere else in the c-structure outside the VP node. To acknowledge this, we will postulate that focused constituents are echoed outside the VP node by a trace-like gap. Also, we will adopt Zaenen's definition of binding domain, that is, all clauses dominating the bindee and not dominating the binder will be in the binding domain. In LFG, syntactic binding is effected through the instantiation of the so-called linking equations. These employ the metavariables $\downarrow$ and $\uparrow$, which would be attached onto our proposed PSRs according to (1):

(1)

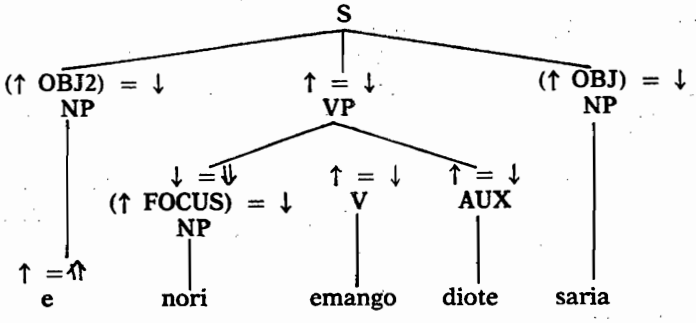
VP →	XP	V	AUX
	$(\uparrow \text{FOCUS} = \downarrow)$	$\uparrow = \downarrow$	$\uparrow = \downarrow$
	$\downarrow = \Downarrow$		
XP →	$\uparrow = \Downarrow$		
	e		

In simplified terms, the presence of ⇓ indicates that the variable at the node carrying the ⇓ must be set equal to the node carrying the ⇑ (see below for a better definition). So if we had sentence like (2):

(2) *Nori emango diote saria*
Who-DAT give Aux-they-him-it prize-ABS
'To whom will they give the prize?'

Its c-structure would be as in (3):

(3)



(Notice that the location of the 'e' is absolutely arbitrary, as long as it is out of the VP node, in accordance with the nonconfigurational generation of major projections in Basque, sketched above).

The instantiation of the functional equations in (3) will yield a f-structure where OBJ2 and FOCUS share the same value.

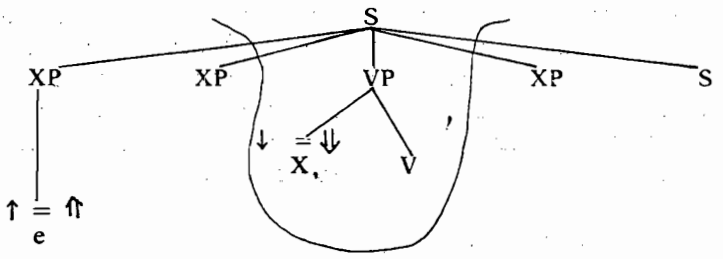
There are, however, some restrictions with regard to the constituent control domain. Zaenen states the following two constraints:

i) The first one is that the binder must c-command the bindee (X c-commands Y iff the first branching node dominating X dominates Y, and X does not dominate Y, nor Y, X). Moreover, only a projection of V can be the right sister of a binder.

ii) The second constraint states that all S-nodes are bounding nodes unless stipulated otherwise.

No immediate problem arises with this second constraint. S nodes are bounding nodes in Basque as well (cf. Azkarate et al, Ortiz de Urbina and Rebuschi, presented above). However, if we observe the c-structure in (3), we notice that, on the one hand, it is not a projection of V, but rather V itself which is the right sister of the binder, and on the other hand, and more troublesome, in no way can the binder c-command the bindee. The first branching node dominating the focus node is VP, and VP is sister of the node dominating the bindee, as in diagram (4):

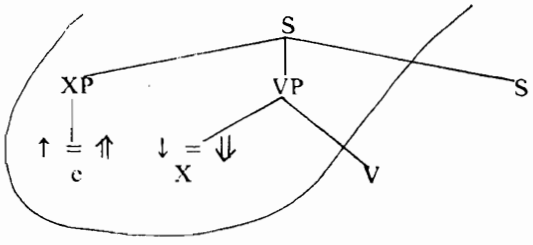
(4)



Zaenen's first constraint on constituent control is much too restrictive for the facts of Basque[2]. The need for focus to be a daughter of the VP node should be clear by now. In contrast with the general behaviour of maximal projections in Basque, no intervening element can be placed between focused constituents (e.g. wh-words) and the verb. It seems pointless, on the other hand, to propose the gap to be generated somewhere under the VP node (so it could be c-commanded by the binder); it is more natural to assume that it behaves like any other constituent in the sentence.

It seems as if there were no other alternative but to slacken Zaenen's constraint (without contradicting Kaplan and Bresnan). It could be proposed that (at least for Basque) it is enough for the binder to *command* (cf. Langacker, 1969: 167) the bindee; i.e. X commands Y iff the first S node dominating X dominates Y, [and X does not dominate Y nor Y,X], as in diagram (5).

(5)



That constituent control phenomena in Basque are somehow different to English and other languages appears to be logical if one considers the characteristics of Basque syntax and the nature of syntactic binding. Syntactic binding (or constituent control) is, as presented by Bresnan, particularly concerned with phenomena depending upon configurational characteristics; it is in other words sensitive to constituent configurations (cf. Bresnan, 1982: 231-232). For a language such as Basque, whose syntax allows an arbitrary order of major constituents in the configuration of a sentence, the conditions on which control phenomena occur should be expected to be different. It seems

(2) As reviewed in the previous chapter (2.2), the same phenomena is responsable for the problems encountered by Ortiz de Urbina's approach within a GB framework.

as if, for constituent control, Basque ignores the level at which maximal projections operate, for it is irrelevant with regard to constituent configurations. Still, syntactic binding phenomena occur in Basque, and they are manifested at other levels in which configurations matter, e.g. under the VP node, where the focus position is located, or within relative and complement clauses.

3.2.2. *Subjacent Binding Domains*

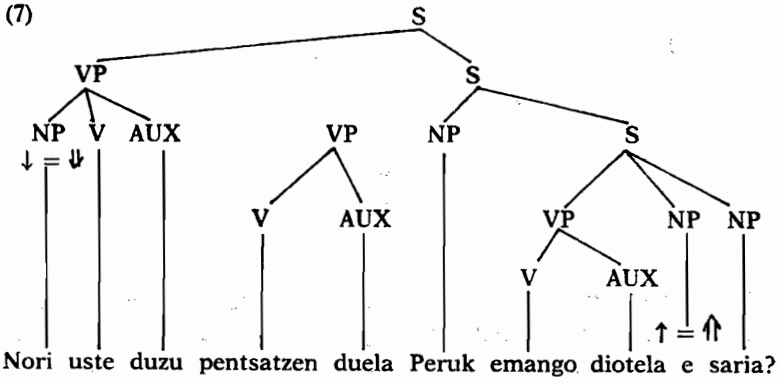
In Basque, as in English and other languages, the binding domain may have to be extended to subjacent S nodes. This is the case of the phenomena described by Azkarate et al, and Ortiz de Urbina as «successive cyclic movement of question words extraction» (reviewed in a previous chapter). The problem is to explain why, when a constituent of an embedded clause is questioned and appears in the focus position of the main clause, all clauses in the binding domain must (in some dialects) be verb initial.

The following example from Ortiz de Urbina illustrates this (6):

(6) *Nori uste duzu [pentsatzen duela Peruk [emango diotela saria?*
Who-DAT think Aux-you [believe Aux-he Peter-ERG [give Aux-they-him-it-COMP prize-ABS?
'To whom do you think Peter believes they will give the prize?'

The c-structure of this would be something like (7):

(7)



The three main problems that this structure presents can be stated as follows:

(a) How do we make subjacent S nodes accessible for the binder (to control the bindee 'e'), since all S nodes are bounding nodes?

(b) How do we ensure that once the binding domain has been extended, the only possible order in intervening clauses is verb initial?

(c) How will we account for this only in the circumstances described above, i.e. only when the constituent in the focus position of the main clause is a constituent of an embedded clause?

According to Zaenen (:481), Ss can be made accessible for binders by PARs that introduce $\Uparrow = \Downarrow$ equations. These link two subjacent binding domains. In English these equations would be introduced on a complementizer. Moreover, S nodes to the right of a COMP node with the equation $\Downarrow$ are root nodes (see below). However, this seems to forget the case of empty COMP nodes, in which, despite the absence of a complementizer, the binding domain needs to be extended (8) [3]:

(8) *The girl wondered who the nurse claimed the boy saw.*

If we adopt Kaplan and Bresnan's (Id: 253) alternative S' rule (9), the linking scheme will let the dependency go through in (8).

(9) $S' \rightarrow$ (that) $\overset{\uparrow = \downarrow}{\underset{\Uparrow = \Downarrow}{(\mathrm{S})}}$

(Where (S) means to be a bounding node). Zaenen's definition of root node does not apply in the case of (9). (S) is not the right sister of a COMP node with the annotation $\Downarrow$; rather, the annotation is on the S node itself. Moreover, the complementizer *that* has to introduce the equation $\Uparrow = \Downarrow$ in some occasions, but not in others, e.g. (10) [Zaenen's example (2)]:

(10) *I think that Bill said that Mary thought that John saw Mary.*

In Basque, the complementizer is not a left sister (nor a right sister) of S, but rather a suffix on the verbal auxiliary [4]. Besides this, it is not the insertion of a complementizer which is relevant to syntactic domain (although only complement clauses can undergo the binding phenomena described above). In other words, in compound sentences with complement clauses there may be no need to extend any binding domain at all. Now there is no constituent which is focused, because now it forms part of the main clause.

I will assume, then, that the linking schema $\Uparrow = \Downarrow$ will appear in an alternative S rule similar to (9). The S bearing such an equation will be regarded as a root node. Following Zaenen, all root nodes of a control domain will also be annotated with the constraining equation

(3) Falk (1983) examines the Subject Dependency Problem, known in the Transformationalist literature as the «*that*-trace phenomenon». This is a typical case of long distance dependency, in which a subject gap may not be adjacent to an overt complementizer. In other words, a case in which binding domains cannot be right sisters of a COMP node. Falk's example:

(1) **Who do you believe that ___ saw me?*

(4) The aim of this chapter is not to analyse extensively the behaviour of complement clauses in Basque, but to investigate rather the implications that the proposed focus position in Basque also has with respect to compound sentences, and how to resolve them. Complement clauses have been studied in more detail by Goenaga (1984).

(↓ BND) = $_c$ +, conveying that the S node is within a binding domain.

For the verb initial clauses we need a rule such as (11):

$$\text{(11)} \quad \text{VP} \rightarrow \underset{(\uparrow \text{BND}) = +}{\text{V}} \quad \text{AUX}$$

This will guarantee that within a root node, marked with the feature BND, only the expansion of VP as in (11) will be possible. We maintain Zaenen's constraint (:494) in that the BND feature (like the inflectional features) can be introduced with lexical categories only.

On the other hand, root S nodes will be expanded as in rule (12), when they contain a complement clause [5]:

$$\text{(12)} \quad \text{S} \rightarrow \underset{(\downarrow \text{BND}) =_c +}{\text{VP}} \quad \text{XP*}$$

The number of verbs that take complement arguments is relatively small. They are verbs like *esan* 'to say', *uste* 'to think', and similar verbs. If we take *esan*, for example, we can say that it subcategorizes for SUBJ, optionally for OBJ2, and alternatively for OBJ or SCOMP. Subject NPs need to be ERG, and they can be placed in the focus position, just like any other constituent.

Compare the following examples:

(13) *Nork esan du hori?*
Who-ERG say Aux-he-it that-ABS?
'Who has said that?'

(14) *Nork esan du bihar etorriko dela?*
Who-ERG say Aux-he-it tomorrow come Aux-he-COMP?
'Who has said that he will come tomorrow?'

In (14) it is clear that the focused constituent *nork*, in the ERG case, could not be part of the complement clause, because *etorri* 'to come' is an intransitive verb. However, problems will arise in cases such as (15) [6]:

(15) *Nork esan du abestuko duela bihar?*
Who-ERG say Aux-he-it sing Aux-he-it-COMP tomorrow?
'Who has said that he will sing tomorrow?'
'Who has he said will sing tomorrow?'

(5) An alternative strategy can be expressed by the following rules:

$$\text{(11')} \quad \text{VP} \rightarrow \left(\underset{(\uparrow \text{BND}) = \text{c} -}{\text{XP}}\right) \quad \text{V} \quad \text{AUX}$$

$$\text{(12')} \quad \text{S} \rightarrow \left(\underset{(\uparrow \text{BND}) = \text{c} -}{\text{XP*}}\right) \quad \text{VP} \quad \text{XP*}$$

Initially, all S nodes would be annotated with BND's value —, but they could be optionally annotated with ⇑ = ⇓ and BND's value +. For this latter choice, XPs preceding V and VP would undergo a clash in their expansion, because of (↓ BND) = + on the S node, and they may need to be null. The sole disadvantage of this approach is that the BND feature would not be introduced by lexical categories, as proposed by Zaenen, but would be annotated to the structural category S.

(6) Notice that *abestu* 'to sing' is one of those exceptional verbs that take ergative subject (and therefore transitive auxiliary) without subcategorizing for (OBJ). In this case, (OBJ) is implicitly understood as 'to sing *a song*').

(I) *abestu:* V, (↑ PRED) = 'sing <(SUBJ)>',
(↑ SUBJ CASE) = ERG.

In (15) *nork* could be part of either clause. There are cases in English where a similar problem arises:

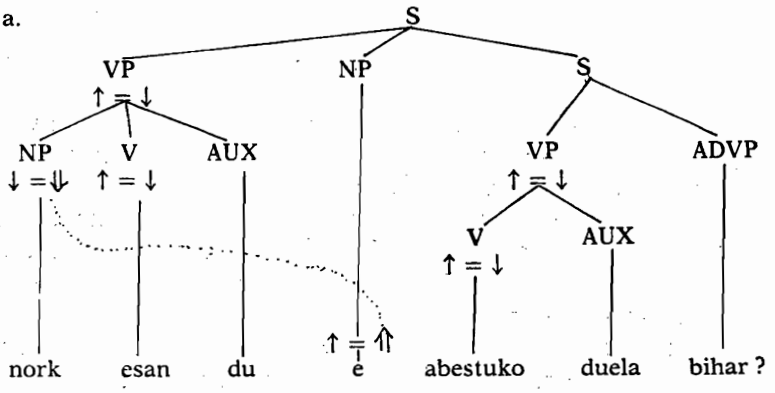
(16) *To whom did you say they would award the prize?*

(17) *To whom did you say they would give the prize?*

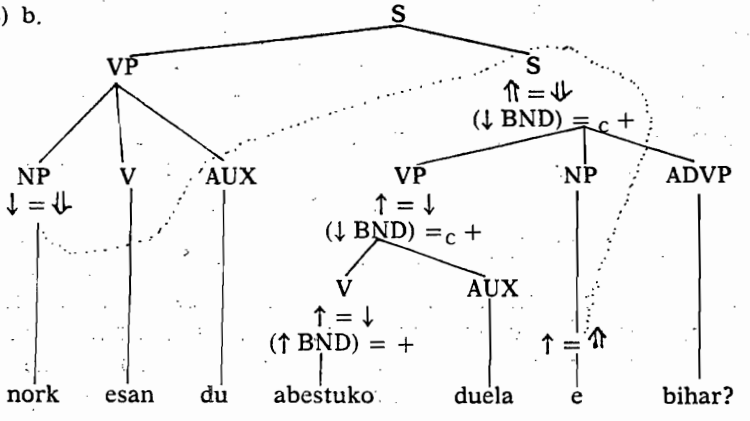
Both these examples are ambiguous, although the reading where *to whom* is an argument of *say* is probably easier to see in (16) than in (17), since *award* has a less strong requirement for a recipient argument than *give* has.

We could say that, when a focused element appears in the focus position of verbs like e.g. *esan*, those which take complement clauses, the expansion of S as a root node will be instantiated alternatively. This is to say that two c-structures would be possible for sentences like (15), one without the expansion of the binding domain, regarding the bindee as part of the main clause, and another extending the binding domain to the subjacent clause. These two c-structures for (15) would be as follows:

(18) a.



(18) b.



From which the following two f-structures would be obtained:

(19) a.

$$\begin{bmatrix} \text{FOCUS} & \text{PRED} & \text{'who'} \\ \text{SUBJ} & & \\ \text{PRED} & \text{'say } \langle(\text{SUBJ}), (\text{SCOMP})\rangle\text{'} & \\ \text{SCOMP} & \begin{bmatrix} \text{SUBJ} & \text{PRED} & \text{'PRO'} \\ \text{PRED} & \text{'sing } \langle(\text{SUBJ})\rangle\text{'} & \\ \text{ADJUNCT} & \text{PRED} & \text{'tomorrow'} \end{bmatrix} & \end{bmatrix}$$

(19) b.

$$\begin{bmatrix} \text{FOCUS} & \text{PRED} & \text{'who'} \\ \text{SUBJ} & \text{PRED} & \text{'PRO'} \\ \text{PRED} & \text{'say } \langle(\text{SUBJ}), (\text{SCOMP})\rangle\text{'} & \\ \text{SCOMP} & \begin{bmatrix} \text{SUBJ} & & \\ \text{PRED} & \text{'sing } \langle(\text{SUBJ})\rangle\text{'} & \\ \text{ADJUNCT} & \text{PRED} & \text{'tomorrow'} \end{bmatrix} & \end{bmatrix}$$

Notice that the coherence condition plays an essential role in the approach here presented. For a focused element to be a governable grammatical function inside a clause, it has to be governed by its verb's predicate argument structure. Negative existential constraints (Kaplan and Bresnan, Id: 212) will rule out function assignments that do not suit. A —(↑OBJ2) notation on the lexical entry of the verb *uste*, for example, will ensure that no wh-word marked DAT ever gets a grammatical role in its clause nucleus. The uniqueness condition will also prevent wrong binding relations from being set up. No focused constituent could be bound inside a clause for which there already is an element filling the same relational function, that is, an element which shares its same case-value. (Remember that in Basque grammatical functions are not determined by the arrangement of c-structure nodes, but by case and agreement features).

Furthermore, binding domains are marked in Basque syntactically, in a way such that a clause in a binding domain must be verb initial. Hence, whenever a complement clause does not meet this condition, alternative solutions like (19 b.) will automatically be disregarded. The feature BND accounts for this fact. When a focused constituent has not been bound in the main clause of a compound sentence with complement clauses, these will be optionally BND = +, and therefore also accessible for the binder by means of the linking schemata $\Uparrow = \Downarrow$. Nodes constrained with BND's value + are committed to

fulfil the conditions stated above, and represented by rules (11) and (12).

In short, the assumption made here is as follows: Whenever an $\Downarrow$ long-distance metavariable has not been linked to its corresponding $\Uparrow$ in the same minimal S, subjacent S nodes are accessible by the $\Uparrow = \Downarrow$ schemata, and pertinent binding conditions are invoked.

This approach to long-distance dependencies in Basque, does not conflict, unlike Ortiz de Urbina's approach, with a non-configurational description of major constituents. On the other hand, the modifications on Zaenen's scheme to accommodate these features of Basque, seem to fit adequately with the LFG framework.

4. Implementation in prolog

This chapter describes the implementation of the grammatical analysis sketched in the prior two chapters. The first section displays the connections between the LFG formalism and the language in which the parser has been implemented, Prolog. Some LFG implementations have been written in Prolog before, viz. Frey and Reyle (1983) and Yasukawa (1984). The program here described, however, is for the most part based upon Peter Whitelock and Brian Chandler's implementation for the English-Japanese MT system under development at CCL. The parser for Basque is described in the second section.

4.1. *Prolog, DCCs and LFG*

The programming language chosen for writing the LFG parser for Basque is Prolog. Prolog has been found to be particularly suitable for natural language processing, which is why the number of grammar formalisms being implemented in this language is increasing rapidly.

Prolog is a logic programming language based upon the early work on theorem proving of Kowalski (1971), Green (1969) and Robinson (1965), among others. Prolog is the practical realization of the idea that resolution theorem provers could be used as programming languages. In this respect, Prolog can be referred to as a resolution theorem prover for *definite*, or Horn, clauses. Theorem proving tasks expressed in definite clauses are expressed in a way such that there is a goal clause, or theorem, and a number of implication clauses, or axioms, from which one can satisfy that goal.

In what follows we trace briefly the connection between Prolog and logic, or more specifically, first order predicate calculus. Then we turn to an explanation of how programs can be executed in Prolog. The section ends with a reference to Prolog's suitability for LFG.

4.1.1. *Prolog and Logic*

Prolog can be roughly considered as a definite clause interpreter. Clause form is a particular way of expressing predicates in logic. A set

of formulae is in clausal form when each formula is a clause. A clause can be defined as a disjunction of literals, e.g. (1):

(1) $P_1 \vee \ldots \vee P_n$

where 'v' is a notation for disjunction, and each literal P represents either a proposition or a negated proposition. Kowalski provided a variant form for clauses which proved to be more efficient in resolutions proofs. A clause like (2),

(2) $\neg P_1 \vee \ldots \vee \neg P_m \vee Q_1 \vee \ldots \vee Q_n$

where —P are negated propositions, Q propositions, and m and n are $\geqslant 0$, is in the Kowalski form when it is written like (3):

(3) $P_1 \,\&\, \ldots \,\&\, P_m \rightarrow Q_1 \vee \ldots \vee Q_n$

That is, all unnegated literals are collected in a disjunction of propositions on the right of an implication arrow, called the consequent, and all negated literals are collected in a conjunction to the left of the arrow, called the antecedent.

Definite clauses are a restricted set of the Kowalski form, i.e. they are those clauses with at most one unnegated literal:

(4) a. Empty clause.
b. Assertion clause: $\rightarrow P_o$
c. Goal clause: $P_1 \,\&\, \ldots \,\&\, P_n \rightarrow$
d. Implication clause: $P_1 \,\&\, \ldots \,\&\, P_n \rightarrow P_o$

The Prolog notation for definite clauses differs slightly from the one presented above. In Prolog the antecedent is written to the right of the implication arrow (which is written as ':—', and the logical & as ','), and the consequent to the left. Prolog programs consist of sets of procedures, each of which define a logical predicate. A predicate, or principal functor of a literal, consists of a sequence of statements like (5):

(5) P_o :— P_1 , P_2 , ... , P_n

which can be read either declaratively: «P_o is true if P_1 and P_2 and ... and P_n are also true», or procedurally: «as to satisfy goal P_o, satisfy P_1, and P_2, and ... and P_n».

In (5) (which corresponds to (4) d.), the antecedent constitutes the *body* of a procedure for calculating P_o. If the *head* P_o were on its own, it would be called a unit clause, or fact, in Prolog (which corresponds to (4) b.). A clause without P_o, is a goal clause (like (4) c.), and is the directive whereby a Prolog program is invoked for execution.

To execute a Prolog program, the system searches for the first clause whose head matches or unifies with the goal clause. The unification process finds the most general common instances between two terms.

If a match is found, the unified clause instance is then activated by executing in turn, from left to right, each of the goals, if any, in its body. If the called clause fails to find a match for a goal, the system backtracks, i.e. it rejects the most recently activated clause and undoes any substitution made by the match with the head of the clause; next it reconsiders the original goal which activated the rejected clause, and tries to find a subsequent clause which also matches the goal. When the goal has been satisfied, that is to say that the theorem we wanted to prove is true, the system returns 'yes', and it returns 'no' otherwise.

Prolog has additional predicates which are not satisfied by resolution, but are specifically evaluated by the computer and then deleted. For example, the predicate «write» will cause its parameter to be printed, and will then be deleted from the clause.

In Prolog pattern-matching and unification perform the same operations as selector and constructor functions in other programming languages, such as *car, cdr* and *cons* in Lisp. Prolog is non-deterministic. With the use of backtracking a procedure can produce alternative results. A particular attraction of Prolog is that programs and data are identical in form. A program consisting solely of unit clauses, or facts, is closer to an array than to a procedure in a conventional programming language. This characteristic is particularly important for Prolog in natural language processing because grammar and dictionaries can be seen as a declarative description of a language, or as a procedure mechanism for parsing or generating a string.

4.1.2. *Prolog's representation of context Free Grammars*

Context free rules can be translated into Prolog as definite clauses. The parsing of an input string by a grammar is equivalent to providing that the string is a deducible theorem from a given set of axioms, i.e. the grammar. This description of a language as a set of Prolog clauses can be executed as a program, in which an input string or sentence is the goal clause to satisfy. Each context free grammar rule or rewrite rule describes the possible forms for a non-terminal symbol by a sequence of terminals and non-terminals. Non-terminals are expressed in Prolog as literals, and terminals as lists [1]. An example of CFG rules is given in (6):

(6) sentence → noun–phrase, verb–phrase.
noun–phrase → determiner, noun.
determiner → [the].

(1) Lists are an important data structure in Prolog. They are represented in square brackets, e.g. [a, b. c]. The notation [X | Y] unifies the variable X (distinguished by an initial capital letter, as opposed to constants distinguished by initial lowercase letters) with the head of the list (i.e. the first argument), and variable Y with the tail of the list (i.e. a list with all remaining elements). When two variables are unified or shared, this means that as soon as one receives a value, the other is automatically instantiated to this same value.

This notation is recognized by the Prolog interpreter which translates it into ordinary Prolog definite clauses. The translation involves the replacement of each non-terminal with a predicate of arity two (where arity means the number of arguments associated with the predicate, or principal functor). The new arguments of the predicate indicate the starting point of that non-terminal in the input string, and what remains after it has been replaced. In (7) is shown a translation of the contex free rules in (6) into Prolog's definite clauses.

```
(7) sentence (SO,S)      :— noun–phrase  (SO,S1),
                            verb–phrase  (S1,S).
    noun–phrase (SO,S)   :— determiner  (SO,S1),
                            noun  (S1,S).
    determiner ([the|S],S).
```

Terminal symbols, i.e. words, can be represented as the head of a list which stands for the starting point of the terminal in the input string. Its tail shares the value with the second argument.

Example (8) shows how a sentence can be recognized having numbers attached to the points in the string, and is translated into a set of unit clauses:

```
(8) [1 the, 2 bird, 3 sings 4]
    determiner ([the|2],2).
    noun ([bird|3],3).
    verb ([sings|4],4).
```

The goal to be satisfied for the input string is:

```
(9) ? sentence (1,4).
```

This question will return 'yes' or 'no' depending on whether the string is a sentence of the language described by the rules of the grammar.

4.1.3. *Definite Clause Grammars and LFG*

Definite Clause Grammars (DCG) correspond to context free grammars in that the grammar rules have non-terminal symbols on the left-hand side of the rule. DCGs, however, do not share with CFGs their rank in the Chomsky hierarchy, as DCGs can be easily extended. This ability is gained by inserting arguments into non-terminal symbols of CFGs. So, by dropping all non-terminal arguments of a DCG one gets its CFG skeleton.

The advantage of using DCGs as Prolog programs lies in that:

i) Arguments can be added to the non-terminal symbols to provide supplementary linguistic information, for example, to ensure syntactic agreement or to transport context sensitive information.

ii) Compound terms can be used to build structures during the parsing process, either to represent the meaning of a sentence, its functional description, or its syntactic configuration. These structures are built solely by the unification process which takes place during Prolog's execution.

iii) Additional functions or conditions in functions can govern the application of the rules, since Prolog permits extra predicates (or procedure calls) to be included on the right hand side of the rule.

All these three advantages are crucial when implementing an LFG formalism. On the one hand, it is important to employ a mechanism which provides the possibility of context sensitivity, as lexical functional languages are included within the set of context sensitive languages [2] (cf. Kaplan & Bresnan, 1982: 259).

On the other hand, all steps of the parsing process of a sentence according to the LFG formalism can be performed simultaneously. That is, c-structures can be passed around and completed as the parsing process executes, and, also, the resolution of the functional equations into f-structures can be performed during the same process of recognition of the input string.

As previously stated, Prolog performs a built-in unification procedure when resolving clauses to prove a goal. There is a problem with this strategy, which is that unification might cause unrestricted DCGs to have a Turing machine power, or, in other words, that DCGs are solely semidecidable, and general proof procedures may not terminate (cf. Pereira & Warren, 1983: 142). However, by using an LFG formalism as a representation for the linguistic knowledge translated into DCGs, stability can be obtained. This is because LFG embodies an offline parsing algorithm, i.e. it has a mechanism which makes constraints and imposes conditions on any resulting output of a context-free algorithm (these are, for example, the functional locality [3] and the functional well formedness conditions). This feature of an LFG formalism is reason enough for its suitability for describing a natural language, and a very good reason for us to write a parser based upon that description.

4.2. *Description of the program*

This section describes an implementation in Prolog of the LFG for Basque sketched in previous chapters. The program attempts to provide a practical realization of most of the linguistic issues investigated, which are to be seen as the theoretical background on which

(2) This is because both LFG functional composition, which permits f-structures to encode a wide range of tree properties, and LFG equality predicate (or unification), which enforces a match between the properties encoded from different nodes.

(3) The functional locality principle disallows paths in rules longer than two features.

the program is founded. The translation into Prolog's definite clauses is aimed at representing the linguistic information in the most accurate way, and it will be shown that the implementation follows quite straightforwardly from the theory.

The parser is, however, in its early stage, both linguistically and computationally. As stated previously, the subset of grammar rules implemented is confined to the analysis of structures described throughout this work, i.e. configuration of major constituents within the scope of S, their functional encoding, and long-distance dependency's phenomena, which include subjacent complement domains. On the other hand, it is accepted that the program, although it provides a feasible ground for implementing linguistic information, leaves several questions to be investigated. Among the most relevant are an account of constraining equations, economization of the lexicon, and a device to implement the completeness condition.

All the same, the parser here presented fulfils its main purpose as it demostrates how an LFG formalism describing Basque can be efficiently implemented in Prolog, providing further support to both the theory and the particular computational choice.

4.2.1. *Context free analysis of c-structures*

The initial set of rules described in chapter 2 can be written in Prolog as follows (10):

```
(10) sentence → x–phrase–star,
                verb–phrase,
                x–phrase–star,
                sentence.
```

The predicate *sentence* is hence defined recursively. This is needed for compound sentences, in which, for example, an embedded clause functions as a complement of the main clause. Because all non-terminals in the body of this clause, i.e. XP*, VP and S, are optional, they have to be provided with a possible empty expansion. This can be achieved in the following two ways (11):

```
(11) a. x–phrase–star → [ ].
     b. (null; sentence).
```

where ';' indicates disjunction, and *null* is a predicate which does not consume any word from the input string, as defined in (12):

```
(12) null (S,S).
```

The choice (11 a.) is used as a termination boundary condition for the recursively defined predicate *x phrase star* (13), and is preferred to the form (11 b.) with the view of avoiding complex clauses (see 4.2.4.).

(13) x–phrase–star → noun–phrase,
x–phrase–star.

The expansion rule for the VP category is as in (14):

(14) verb–phrase → x–phrase,
verb,
auxiliary.

The constituent preceding the verb has been defined as the focused element for a sentence (see section 2.2). It need not be present, however, and therefore it is optionally null. Because our analysis does not yet consider the possibility of more than one constituent being focused, XP will be either rewritten as NP, or any other appropiate category, and it is not defined recursively; so it is defined as *focus phrase* in the program as distinct from x-phrase-star.

NPs rewrite for the moment into nouns solely (15):

(15) noun-phrase → noun.

Terminal symbols are kept as facts, or unit clauses, as in (16 b.):

(16) a. noun → [neska].
b. noun([neska|S],S).

Both expressions are equivalent, the second one being the compiled form of the first.

4.2.2. *Building c-structures*

As stated previously, the construction of tree structures can be carried out during the parsing process. This is achieved by the introduction of arguments on the predicates, e.g. (17):

(17) noun (n(neska), [neska|S],S).

For non-terminal symbols, the parameter of this new argument, defined as a compound term, will be a variable, like (18), which will be successively instantiated as the parsing precedes.

(18) noun-phrase(np(NP), SO,S :— noun(NP, SO,S).

For 'sentence (C,SO,S)' (19), after unification [4] throughout the procedure, the value of the variable 'C' will be as in (20).

(19) ? sentence(C,[neskak,zer,dakar],[]).

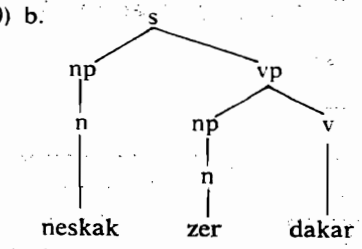
(20) a. C = s(np(n(neskak), vp(np(n(zer)), v(dakar))).

's' in (20) will be a functor with arity 'x', where x is the number of

(4) DCG parsing algorithms are based upon unification, which is the basic rule of inference for definite clauses. Unification succeeds when a substitution has been found for a goal clause which matches the head of either an implication or an assertion clause. By means of unification both clauses become identical.

major constituents a sentence has, in this case x = 2. The value of 'C' in (20) can also be represented as a tree (20 b.):

(20) b.



Because XP* has been defined recursively, permitting more than one XP to be analysed, it is convenient to represent all occurrences as a list, so no dependency relation will hold among them. In other words, the list notation will allow XPs to be represented as sister nodes. This list will be empty if XP* is null, otherwise each NP parsed in the expansion of XP* will be instantiated at the head of the list, and an optional XP* at its tail, as in (21):

```
(21) x–phrase–star( [NP|XP]) →
                              noun–phrase(NP),
                              x–phrase–star(XP).
```

A side effect of this approach is that the whole sentence needs to be represented as a list, since VP nodes as well are sister nodes of NPs. 's' then will become (in the program) a functor of arity 1, whose sole argument is a list. We therefore need the *append* predicate to append lists together during the parsing process. The result of this will not be very different from (20):

```
(22) C = s( [ np( n(neskak), vp( np( n(zer)), v(dakar)) ]).
```

4.2.3. *Building f-structures*

The functional structure of a sentence encodes its meaningful grammatical relations, and is a crucial part of LFG's description of a sentence. F-structures represent their information as a set of ordered pairs, each of which consists of an attribute and a specification of that attribute's value for this sentence. An attribute is the name of a grammatical function or feature, e.g. SUBJ, PRED, VCOMP, NUM, CASE, etc. There are three primitive types of values:

i) simple symbols, atoms or integers in Prolog's notation.
ii) semantic predicates, like sem(X), where X is a predicate.
iii) subsidiary f-structures, or sets of ordered pairs representing complexes of internal functions (see below).

F-structures, similarly to c-structures, can be built during the parsing process. Again, what is needed is an additional argument to

transport the functional information obtained during the unification resolution in the execution of the program.

This new argument is conceived as a list. All its arguments being the values, or sets of values, assigned to the different functions and features (attributes), annotated to the syntactic nodes, or to lexical entries. Attributes will be arguments of a built-in list [5], a limited list which comprises all possible attributes chosen for the description of a given language. The final f-structure for a sentence will be accomplished by unification of both these two lists.

By means of a function assigning predicate, *apply*, the parser makes correspondences between values and attributes, as they are instantiated throughout the process. This predicate is defined as follows (23) [6].

```
(23)  apply( [Val|_],Att,Val,[Att|_] ) :- !.
      apply( [Val|_],Att,Val2,[Att|_]) :- !,fail.
      apply( [_|Valtail],Att,Val,[_|Atttail] ) :-
                        apply(Valtail,Att,Val,Atttail).
```

If a pattern match succeeds in the first clause, either the variable Val in the list was already instantiated and has the same value as the matched value, or it was uninstantiated, in which case it is unified with the matched value and returns the predicate true. Otherwise, it will try the second clause, and fail, making the system backtrack. This predicate is defined recursively, as it deals with lists, and its arity is 4.

The first argument operates as an identifier variable. Given that each syntactic node has a unique identifier variable (Id-variables), it is possible to identify each node with one f-structure. The Id-variables stand for LFG's ↑ and ↓ immediate domination metavariables. This first argument stands for ↑.

The second argument acts as a variable which will be instantiated to the appropiate attribute at the time the predicate is being matched.

The third argument stands either for the metavariable ↓, or a value, which is then the instance of the argument.

This will be clearer in an example (24):

```
(24)  a.  verb_phrase( IdVP) →
                x_phrase( IdXP),
                      {apply( IdVP,focus,IdXP)},
                verb(IdVP).
```

(5) This is just an arbitrary choice, as in fact attributes could be incorporated into the list as they were encountered during the parsing proces; i.e. this definite set of attributes could as well have been defined as an open ended list.

(6) The *cut* '!' symbol is a facility provided by Prolog for specifying control information. It is inserted in the program just like a goal, but is not to be regarded as part of the logic of the program, and should be ignored as far as the declarative semantics is concerned. The cut as a goal always succeeds, and commits the system to all choices made since the parent goal was invoked, and causes other alternatives to be discarded. One of the main effects of the cut is that it permits programs to operate faster.

```
b. noun( IdNP,[gizonek|S],S :—
                apply( IdNP,case,erg).
```

(Note that the apply predicate has been defined as a functor of arity 4, and that in these examples its arity is 3. This is done for purely practical reasons with the view of clarity. When apply is called, a new argument, i.e. the built-in list of attributes, is automatically added as the fourth argument of apply).

The two clauses in (24) correspond to the following LFG rules:

(25) a. VP → XP V
(↑ FOCUS) = ↓ ↑ = ↓
b. *gizonek* : N, (↑ CASE) = ERG.

In what follows we describe how the program deals with the set of equations proposed in section 2.3. for functional encoding of SUBJ, OBJ and OBJ2 noun phrases.

In non-configurational encoding the basic principle is to associate pairs of function-assigning and feature-assigning equations of the form given in (26):

(26) (↓ F) = v
(↑ G) = ↓

The program here described deals with this pair of equations by combining them into a new predicate *np-apply* which will be called whenever an NP needs to be assigned its grammatical function, i.e. during the expansion of XP*.

So, for the following schemata (27):

(27) a. (↓ CASE) = ABS
(↑ SUBJ) = ↓

b. (↓ CASE) = ABS
(↑ OBJ) = ↓

The Prolog notation is as in (28):

```
(28) a. np–apply( IdS,IdNP) :—
                apply(IdNP,case,abs),
                apply(IdS,subj,IdNP).
     b. np–apply( IdS,IdNP). :—
                apply(IdNP,case,abs),
                apply(IdS,obj,IdNP).
```

There is a problem with this realization of (27), which is that the program will be doubtful about which function to apply when the encountered NP is marked ABS. The parser will produce undesired answers, if not just wrong ones. There is an easy way to prevent this. Verbs that subcategorize for OBJ, i.e. transitive verbs in Basque, require a subject marked ergative. So we could state a new condition for (28 b.), that is, for a sentence to have an OBJ, it also needs an ergative subject. We can now redefine (28 b.) as follows:

```
(29)  np-apply( IdS,IdNP) :—
                apply(IdNP,case,abs),
                apply(IdS,subj,IdX),
                apply(IdX,case,erg),
                apply(IdS,obj,IdNP).
```

Only when all the clauses in the body of (29) are satisfied, will the current NP be assigned the OBJ function.

In addition to these predicates, we have sketched a function assignment procedure for postpositional phrases (which in Basque correspond to all nominal phrases with inflections other than ABS, ERG or DAT). For the general equation (30) [7]:

(30) $(\downarrow\ (\uparrow \text{PCASE})) = \downarrow$

The following form has been employed:

```
(31)  apply( IdPP,case,Att),
      apply( Ids,Att,IdPP),
```

Both Att variables will share their value, and therefore the grammatical function assigned to the sentence will be the same specific case mark of that PP.

The LFG functional component not only encodes the meaningful grammatical relations of a sentence, or the sufficient information for the semantic component, but acts as a filter on the output of the c-structure component. From the three conditions on functional well-formedness (see section 2.3.), the uniqueness condition is considered to be the most important (cf. Kaplan and Bresnan, Id: 204) [8]. It ensures that the assignment of grammatical functions and features to lexical items and c-structure configurations are globally consistent. Parsing with Prolog, this condition is fulfilled by the unification mechanism, as a variable cannot be instantiated to more than one value. When two variables are unified, their values have to match when they are instantiated, or otherwise unification will fail.

4.2.4. *Long-distance dependencies*

This part of the program deserves a special mention if only

(7) This procedure could also be used instead of *np-apply* (see above). ERG case should be annotated as SUBJ case, and so on. One minor problem will occur with ABS cases, as they are used for subject as well as for object. This problem can be resolved adding —(↑ OBJ) negative existential constraints into intransitive verbs' lexical entries.
So in a sentence like:

neska joan da
the girl-ABS go Aux-she
'the girl has gone'

neska would never be regarded as the OBJ of *etorri* but as SUBJ. (We thank Ron Kaplan for making this comment).

(8) The uniqueness condition is in fact what makes an f-structure a function. That is, a function whose domains are attribute names, whose range are attribute values and for which each name has no more than one value. (I thank Pete Whitelock for making this comment).

because no genuine implementation of LFG constituent control has been reported yet in the literature. An attempt at doing so for a fairly non-configurational language like Basque holds even more interest.

As reviewed in the last chapter, long-distance dependencies occur in Basque when a constituent in the focus position under the VP node is binding a gap somewhere else in the c-structure. When the binding relation is extended to subjacent complement clauses, these have to be verb initial, which means that long-distance dependencies influence the configuration of intervening sentences within a binding domain. As previously stated, the three main problems to solve are:

i) How to make sentential nodes accessible for the binder, when a gap is expected inside its structure?

ii) How to constrain these sentential nodes so they are verb initial, i.e. there are no XP* preceding VP, and the focus position is null?

iii) How alternative solutions can be given for the ambiguous sentences described in section 3.3.?

Pereira (1981) presents a solution for a similar problem that occurs in languages like English, French, Spanish and Portuguese. Pereira refers to syntactic binding phenomena as left extraposition, described in the following way: «Left extraposition occurs in a natural language sentence when a subconstituent of some constituent is missing, and some other constituent, to the left of the incomplete one, represents the missing constituent in some way». The method he proposes for dealing with this problem is analogous to the introduction of derived rules by Gazdar (1982). Pereira's proposal comprises the insertion of additional arguments to all non-terminals from which a constituent might be extraposed. These will be sensitive to whether a trace of the extraposed constituent has to be or has been replaced (see below).

For Basque, within the LFG framework, the required correspondence is given by the f-structure associated with a constituent in the focus position. When the gap's node is reached in the parsing process, the previously mentioned f-structure will be assigned to the node corresponding to the gap. So the f-structure of a constituent in focus and that of its gap share the same value. Note that in Basque it is not the structural location of the gap, but rather the binder's case marking, which determines its grammatical function within the sentence.

Pereira's proposed additional arguments are represented in our approach as lists. When no f-structure needs to be carried across, these lists will be null. Otherwise, the transported f-structure will be represented at the head of the list.

Figure (32) shows how a list can be instantiated to the binder's f-structure [9]:

```
(32) verb–phrase( IdVP,E0,E1) →
                    x–phrase(IdXP),
                        equal([IdXP],E1),
                    verb(IdVP).
```

This resembles LFG equation (33):

(33) VP → XP V
$\downarrow = \Downarrow$ $\uparrow = \downarrow$

The bounded domination metavariable $\Downarrow$ being the argument E1. An expansion of XP* as an empty element is written in the program as in (34).

```
(34) x–phrase–star( IdS,[IdXP;E2]E2) → [ ],
                        {focus–apply( IdS,IdXP) },
                    x–phrase–star( IdS,E2,E2).
```

Which corresponds to LFG rule (35):

(35) XP → e
$\uparrow = \Uparrow$

where the grammatical function encoding is performed be the predicate *focus apply*, in an analoguous way to (29) and (31). The head of the list in the second argument above, IdXP, is the notation for the bounded metavariable $\Uparrow$. The tail of the list will be null, conveying that the f-structure has been assigned to its appropriate gap node, and hence no further f-structure needs to be transported across phrases. This is clearer in the case of subjacent complement clauses. In such occurrences, when VPs and XP*, in a binding domain, have to pass over the f-structure in question, without performing any assigning operation, so that the f-structure can reach the gap. For example (36):

```
(36) verb–phrase( IdVP,[IdXP],[IdXP]).
```

When this happens, there cannot be any constituent in the focus position, i.e. the expansion of VP has to be constrained somehow so verb comes first. In other words, the condition is that the optional XP is null. This is achieved in the program by adding an extra clause to the body of VP's predicate: «For XP to rewrite, E1 has to be [ ], or otherwise it will be null (see (37))».

Similarly an extra argument is needed in the predicate of sentence, revealing whether an f-structure needs to be passed through or

(9) In this approach, long distance dependencies are encoded in the c-structure, by means of the two extra-arguments E0 and E1, but it could alternatively be encoded in f-structure, employing the predicate *apply* in a way similar to the following:

```
apply (IdVP, focus_in, [E0]),
apply (IdVP, focus_out, [IdXP]),
```

in (32) for example in the body of the VP clause. (We thank Stuart Shieber for making this comment).

not. The initial sentence predicate will be instantiated to [], but subjacent sentences can be instantiated to the binder's f-structure, if this has not yet been assigned to a gap in the sentence. When this occurs, these sentential nodes will be regarded as root-nodes of a binding domain, and their argument carrying the f-structure will correspond to LFG's $\uparrow = \downarrow$ linking schema. As we know, in such a case no XP* can be reported before VP. So, the condition for XP* to rewrite in pre-VP position will be this argument, E0, to be null, or otherwise XP* will be null. All this is shown clearly in figure (37):

```
(37)  sentence0( IdS) → sentence( IdS,[ ]).
      sentence( IdS,EO) →
                  (x–phrase–star(LdS,EO,EO), EO=[ ]; null),
                  verb–phrase(IdS,EO,E1),!,
                  x–phrase–star(IdS,E1,E2),
                  (sentence(IdSCOMP,E2),
                        {apply(IdS,scomp,IdSCOMP)}; null, EO=[ ].
```

The program is non-deterministic: when it is called to fail after it has placed the gap's f-structure in the first clause, it backtracks and gives all the other possible solutions among the embedded clauses, if any.

5. Conclusions

The main purpose of the project has been achieved. We have demonstrated how an LFG formalism describing Basque can be efficiently implemented in Prolog. This provides further support to both the theory and the particular computational choice. The LFG formula for Basque is translated into Prolog's definite clauses in an accurate way. The implementation follows straightforwardly from the theory.

The parser accepts all the structures for Basque described as grammatical by the grammar as it currently exists. It encodes grammatical functions in a non-configurational way, i.e. not by means of dominance and precedence but by case marking on the nominal phrases and, complementarily, by verb agreement features.

The parser accounts for the focus structural position and the long distance dependencies related to it. It also deals comfortably with periphrastic verbs.

The parser produces c- and f-structures of the input sentences, which are the two stage model of syntactic description in LFG. C-structures are context free analyses of the surface structure of a sentence. F-structures are a computation of the grammatical relations of the sentence. The lexicon functions as a primary knowledge base and makes the parser specially versatile from the computational point of view. The parser as it stands now is easily extendable.

Well-formedness conditions and functional locality are the constraints that make LFG embody an off-line parsing algorithm, which is desirable for decidability. Well-formedness conditions are accounted for as follows:

i) Uniqueness: It is fulfilled by Prolog's unification.

ii) Completeness: Achieved by the comprehensive relational information lexical entries contain in Basque; both nominal and verbal entries.

iii) Coherence: This condition still needs to be implemented, i.e. constraining equations and negative existential constraints need to be investigated.

Functional locality is not needed for the moment, since all rules implemented so far deal with no more than one feature and no mechanism to extend them has been employed.

The grammatical coverage should be extended. Some of the most interesting structures to formalize would be the following:

i) Negative constructions, in which constituent order inside the VP node changes drastically.

ii) Relative clauses (and other embedded constructions) where the inversion of constituents and long distance dependencies occur.

iii) The internal configuration of NPs and PPs.

iv) Stative and adjective phrases.

These structures have been analysed in the literature within other frameworks, and these analyses provide a good grounding for an LFG formalization.

The lexical component needs to be extended as well. As it now stands, it is redundant and clumsy, or in other words, it is too uneconomical. Inflectional morphology in the word formation component, for dealing with both nominal and verbal inflection, and lexical redundancy rules to account for alterations on the functional argument correspondences need to be investigated.

Finally, improvements in the Prolog program also need to be made to increase its efficiency and user friendliness. The parser however proves, so far, to be a suitable base for a future extendable NLP system dealing with Basque.

REFERENCES

ABAITUA, J. (in preparation), *Lexical Rules for Basque.*

———, & TRASK, R. L. (forthcoming), «Accusativity in Basque».

AKMAJIAN, A., STEELE, S. & WASOW, T. (1979). «The Category AUX in Universal Grammar», *Linguistic Inquiry* 10, 1-64.

ALTUBE, S. (1929; reprinted 1975). *Erderismos.* Bilbao: Euskaltzaindia.

ANDERSON. J. M. (1977). *On case grammar: prolegomena to theory of grammatical relations.* London: Croom Helm.

ANDERSON, S. (1976). «On the notion of subject in Ergative languages», in C.N. Li (ed.), 1-26.

ANDREWS, A. D. (1982). «Case in Modern Icelandic». In Bresnan (ed.), (1982), 427-503.

AZKARATE, M., FARWELL, D., ORTIZ DE URBINA, J. & SALTARELLI, M. (1981). «Word order and Wh-movement in Basque». *Proceedings of the 12th Annual Meeting of the North Eastern Linguistics Society.* Cambridge Mass.

BAKER, M. (1983). «Object, Thems, and Lexical Rules in Italian». In L. Levin, M. Rappaport and A. Zaenen (eds.), (1983), 1-45, *Papers in Lexical Functional Grammar.* Indiana, Bloomington: Indiana University Linguistics Club.

BOSSONG, G. (1985). «Ergativity in Basque». *Linguistics* 22, 341-392.

BRESNAN, J. (1976). «The form and function of transformations». *Linguistic Inquiry* 7, 3-40.

———, (ed.) (1982). *The Mental Representation of Grammatical Relations.* MIT Press.

BRETTSCHNEIDER, G. (1979). «Typological Characteristics of Basque». In F. Plank (ed.) (1979), 371-384.

BURZIO, L. (1981). *Intransitive Verbs and Italian Auxiliaries,* Ph. D. dissertation, MIT.

CHOMSKY, N. (1957). *Syntactic Structures.* The Hague, Mouton.

———, (1970). «Deep Structure, Surface Structure, and Semantic Interpretation». In D. D. Steinberg & L. A. Jakobivits. (1971), 183-216. *Semantics: An Interdisciplinary Reader in Philosophy, Linguistics and Psycology.* Cambridge: University Press.

———, (1981). *Lectures on Government and Binding.* Dordrecht: Foris.

———, (1982). *Some concepts and consequences of the theory of Government and Binding.* MIT Press.

CONTRERAS, H. (1976). *A Theory of Word Order.* Amsterdam: North-Holland.

COMRIE, B. (1978). «Ergativity». In W. P. Lehman (ed.), (1978), 329-394, *Syntactic Typology.* Sussex: The Harvest Press.

———, (1983). «Universals: form and functions». *Linguistics* 21.

COOREMAN, A., FOX, B., & GIVON, T. (1984). «The Discourse Definition of Ergativity». *Studies in Language* 8.

DIXON, R. M. W. (1979). «Ergativity». *Language* 55, 59-138.

DONZEAUD, F. (1972). «The Expression of Focus in Basque». *ASJU* VI, 35-45.

DOWTY, D. (1982). «Grammatical Relations and Montague Grammar». In P. Jacobson and G. K. Pullum (eds.), 79-103.

ENTWISTLE, W. J. (1936). «On the Passivity of the Basque Verb.» *Medium Aevum* 5, 104-114.

EUSKALTZAINDIA. (1973). «Aditz Laguntzaile Batua». (Separata). *Euskera* XVII. Bilbao.

———, (1979). «Bergarako biltzar ondoko erabakiak». *Euskera.* Bilbao.

FALK, Y. N. (1983). «Subjects and Long-Distance Dependencies». *Linguistic Inquiry* 12, 245-270.

———, (1984), «The English Auxiliary System. A Lexical Functional Analysis». *Language* 60, 483-509.

FREY, W. & REYLE, U. (1983). «A Prolog implementation of LFG as a base for a natural language processing system». *Proceedings of the first Conference of the European Chapter of the Association for Computational Linguistics* 52-57.

GAZDAR, G. (1982). «Phrase Structure Grammar». In P. Jacobson & G. K. Pullum (eds.), 131-186.

GAVEL, H. (1930). «Quelques observations sur la passivité du verbe basque». *RIEV* 21, 1-14.

GEGGUS, J. (1983). «Multiple Agreement: The Case of Basque». Manuscript.

GOENAGA, P. (1980). *Gramatika Bideetan*[2]. San Sebastián: Erein.

———, (1984). *Euskal Sintaxia: Konplementazioa eta Nominalizazioa.* Ph. D. Dissertation. Vitoria-Gasteiz: Euskal Herriko Unibertsitatea.

GREEN, C. (1969). «Application of Theorem Proving to Problem Solving». *IJCAI* 1.

———, (1976). «How free is word order in Spanish». In M. Harris (ed.). *Romance Syntax: Syncronic and Diachronic Perspectives.* Salford: University.

HEATH, J. (1977). «Remarks on Basque Verbal Morphology». In W. A. DOUGLAS, R. W. ETULAIN & W. H. JACOBSEN (eds.), *Anglo-American Contributions to Basque Studies. Essays in Honor of Jon Bilbao.* Desert Research: Institute Publications in Social Sciences 13, 193-201.

HOEKSTRA, T. (1984). *Transitivity: Grammatical Relations in Government and Binding Theory.* Dordrecht: Foris.

HORROCKS, G. (1983). «The Order of Constituents in Modern Greek». In G. Gazdar, E. Klein & G. K. Pullum (eds.), *Order, Concord and Constituency.* 95-112. Dordrecht: Foris.

JACKENDOFF, R. S. (1972). *Semantic Interpretation in Generative Grammar.* MIT Press.

———, (1977). $\overline{X}$ *Syntax: A Study of Phrase Structure (Syntax)* Linguistic Inquiry Monograph 2.

JACOBSON, P. & PULLUM G. (eds.), (1982), *The Nature of Syntactic Representations.* Dordrecht.

JELINEK, E. (1984). «Empty Categories, Case and Configurationality». *NLLT* 2, 39-76.

KAPLAN, R. M. (1975). «On process models for sentence comprehension». In D. Norman & D. Rumelhart, (eds.). *Explorations in Cognition.* San Francisco: Freeman.

———, & BRESNAN, J. (1982). «Lexical Functional Grammar: A Formal System of Grammatical Representation». In J. Bresnan (ed.), 174-281.

KEENAN, E. L. (1976). «Towards a Universal Definition of 'Subject'. In C. N. Li (ed.). 303-334.

KISS, K. (1981). «Structural Relations in Hungarian, a Free Word Order Language». *Linguistic Inquiry* 12, 185-213.

KOWALSKY, R. A. & KUEHNER, D. (1971). «Linear Resolution with selection function». *Artificial Intelligence* 2, 227-260.

LAFITTE, P. (1931). «Pour ou contre la passivité du verbe basque?». *Gure Herria,* mai-juin, 263.

———, ([1944], 1979). *Grammaire basque: Dialecte navarro-labourdin litteraire.* San Sebastián: Elkar.

LANGACKER, R. W. (1969). «On Pronominalization and the Chain of Command». In D. A. Reibel & S. Schane (eds.), 160-186. *Modern Studies in English.* Prentice Hall, Englewood Cliffs, N. J.

LEVIN, B. (1983). «Unaccusative Verbs in Basque». Paper presented at *NELS*, 13.
LEVIN, L., RAPPAPORT, M. & ZAENEN, A. (eds.), (1983). *Papers in Lexical Functional Grammar*. Indiana Bloomington: Indiana University Lingustic Club.
LI, C. N. (ed.), (1976). *Subject and Topic*, New York: Academy Press.
MARTINET, A. (1958). «La construction ergative». *Journal de Psychologie Normal et Pathologique* (juillet-septembre).
———, (1962). «Le sujet comme fonction linguistique et l'analyse syntaxique du basque». *Bulletin de la Societé Linguistique* 57, 73-82. Paris.
MICHELENA, L. (1977). «Notas sobre compuestos verbales vascos». *Revista de Dialectología y Tradiciones Populares* XXXIII, 245-271.
———, (1978). «Miscelánea filológica vasca: relato y orden de palabras». *FLV* 29 204-235.
———, (1981). «Galdegaia eta Mintzagaia euskaraz». In *Euskal Linguistika eta Literatura: Bide Berriak*. 57-81. Universidad de Deusto.
MOHANAN, K. (1982). «Grammatical Relations and Clause Structure». In Bresnan (ed.), (1982), 504-589.
———, (1983). «Move NP or Lexical Rules? Evidence from Malayalam Causativization», In L. Levin *et al*, (eds.), (1983), 47-111.
NAERT, P. (1956). «Le verbe basque est il passif?». *Studia Linguistica* 10, 45-49.
ORTIZ DE URBINA, J. M. (1983). «Empty Categories and Focus in Basque». *Studies in Linguistic Science* 13, 1.
PEREIRA, F. C. N. (1981). «Extraposition Grammars». *American Journal of Computational Linguistics* 7, 4, 243-256.
———, & WARREN, D. H. D. (1980). «Definite Clause Grammar for Language Analysis: A Survey of the Formalism and a Comparison with Augmented Transition Networks». *Artificial Intelligence* 13, 213-278.
———, (1983). «Parsing as a Deduction». *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, 137-144. Boston, Massachusetts.
PERLMUTTER, D. M. & POSTAL, P. M. (1974). *Lectures on Relational Grammar*. Summer Linguistic Institute of Linguistic Society of America. Amherst: University of Massachusetts.
PERLMUTTER, D. M. (1982). «Syntactic Representation, Syntactic Levels, and the notion of Subject». In Jacobson *et al*, (eds.), (1982), 283-340.
PINKER, S. (1982). «Lexical Interpretative Grammars». In Bresnan (ed.), (1982), 655-726.
PLANK, F. (ed.), (1979). *Ergativity: Towards a Theory of Grammatical Relations*. New York: Academic Press.
REBUSCHI, G. (1978). «Cas et fonction sujet en basque». *Verbum: Revue de linguistique publiée par l'Universite de Nancy* II, 1, 69-98.
———, (1983). «A note on focalization in Basque». *Journal of Basque Studies* 4, 2, 29-42. Indiana: University of Pennsylvania and Society of Basque Studies in America.
———, (1984). «Positions, configurations, et classes syntaxiques. Aspects de la construction de la phrase basque. *Communication-Congres «Arturo Campion»*. Pamplona-Iruñea: Euskaltzaindia.
———, (1985). «Theorie du liage et langues nonconfigurationnelles: quelques données du basque navarro-labourdin». Université de Nancy.
RIJK, R. P. DE (1969). «Is Basque an S.O.V. Language?». *FLV* 3, 319-351.
———, (1978). «Topic Fronting, Focus Positioning, and the Nature of the Verb Phrase in Basque». In F. Jansen (ed.), *Studies on Fronting*, 81-112. Lisse: Peter de Ridder Press.
ROBINSON, J. A. (1965). «A machine-oriented logic based on the resolution principle». *Journal of the Association of Computing Machinery* 12, 23-41.

ROTAETXE, K. (1978). «Lingüística-lógica: la construcción ergativa vasca». *Revista Española de Lingüística*, 8, 431-445.
———, (1980). «Ergatiboaren sujetotasunaz». *Euskera* XXV, 417-424.
SCHACHTER, P. (1976). «The Subject in Philippine Languages: Topic, Actor. Actor-Topic, or none of the Above?». In C. N. Li (ed.).
———, (1977). «Reference-Related Properties of Subjects». In Cole and Sadock (eds.). *Grammatical Relation.* New York: Academic Press.
SHIEBER S. M. (1985). «An Introduction to Unification-Based Approaches to Grammar». Paper presented as a Tutorial Session at the 23rd Annual Meeting of the Association for Computational Linguistics. University of Chicago, Illinois.
SLOBIN, D. I. (1982). «Universal and Particular in the Acquisition of Language». In L. Gleitman & E. Wanner (eds.). *Language Acquisition: The State of the Art.* Cambridge: University Press.
STEELE, S. (1978). «The Category AUX as a Language Universal». In J. H. Greenberg (ed.), 7-45. *Universals of Human Language, Vol. 3: Word Structure.* Stanford: University Press.
———, (ed.) (1981). *An Encyclopedia of AUX.* Cambridge: MIT Press.
TCHEKHOFF, C. (1978). *Aux fondements de la syntax: l'ergatif.* Paris: Presses Universitaries de France.
TRASK, R. L. (1981). «Basque Verbal Morphology». In *Euskalarien Nazioarteko Jardunaldiak. Iker 1.* 285-304. Euskaltzaindia (ed.). Bilbao.
———, (1984). *Syncronic and Diachronic Studies in the Grammar of Basque.* Ph. D. Thesis. University of London.
———, (forthcoming). «On the Representation of NP gaps».
UMANDI (pseudonym of A. URRESTARAZU). (1976). *Gramática Vasca.* Tolosa: Kardaberaz.
VAN VALIN, R. D. (1977). «Ergativity and the Universality of Subjects». *Papers from the Regional Meeting of the Chicago Linguistic Society* 13, 689-705.
VILLASANTE. L. (1980). *Sintaxis de la oración simple.* Oñate: Aránzazu.
WILBUR, T. H. (1970). «Ergative and Pseudo-Ergative in Basque». *FLV* 4, 57-66.
———, (1979). *Prolegomena to a Grammar of Basque.* Amsterdam: John Benjamins B.V.
WILLIAMS, E. (1984). «Grammatical Relations». *Linguistic Inquiry* 15, 4, 639-673.
WHITELOCK, P., Johson, R. & BENNETT, P. (1984). *The UMIST Translation System. CCL Report.* University of Manchester Institute of Science and Technology.
YASUKAWA, H. (1984). «LFG system in Prolog». *Proceedings of the 22nd Annual Meeting of the Association for Computational Linguistics,* 358-361. Stanford University, California.
ZAENEN, A. (1983). «On Syntactic Binding». *Linguistic Inquiry* 14, 3, 469-504.
ZAMARRIPA, P. (1928). *Gramática Vasca.* Bermeo: Gaubeka.