HAL
open science

French Unification Categorial Grammars

Karine Kray-Baschung, Gabriel G. Bès, Thierry Guillotin

To cite this version:

Karine Kray-Baschung, Gabriel G. Bès, Thierry Guillotin. French Unification Categorial Grammars. Springer. A Natural Language and Graphics Interface - Results and Perspects from the ACORD Project, Springer, pp.23-46, 1992, Resaerch Reports - ESPRIT - Project 393 ACORD. halshs-00371433

HAL Id: halshs-00371433

https://shs.hal.science/halshs-00371433v1

Submitted on 18 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

CHAPTER 3

French Unification Categorial Grammars

Karine Baschung, CLF
Gabriel G. Bès, CLF
Thierry Guillotin, LdM

1 Introduction

Unification Categorial Grammar (UCG) combines the syntactic insights of Categorial Grammar with the semantic insights of Discourse Representation Theory (DRT, cf. Kamp (1981)). The addition of unification (Shieber et al. (1983)) to these two frameworks allows a simple account of interaction between different linguistics levels. The resulting, computationally efficient, system provides an explicit formal framework for linguistic description, within which large grammar fragments for English have been developed, e.g. Calder et al. (1986), Zeevat et al. (1987), Calder et al. (1988), Zeevat (1988) and Moens et al. (1989). This paper will describe the problems related to the adaptation of the UCG framework to French, illustrated by the two grammar fragments developed during the ACORD project.

2 UCG

UCG embodies several recent trends in linguistics. First, being a categorial grammmar, it is strongly *lexicalist*. In others words, relatively little information is contained in grammar rules ; most information originates in the lexicon. Furthermore, the grammmar incorporates a powerful locality constraint allowing only the concatenation of *adjacent* lexical items. Second, it is strictly declarative ; *unification* is the only operation allowed over grammatical entities. Third, there is a very close relationship between the syntax and semantics of linguistic expressions inasmuch as the different levels of representation - semantic, syntactic and phonological - are built up simultaneously, by the uniform device of monotonic compositional construction. This property is generally dubbed *monostratality*.
Following Pollard (1985), UCG refers to linguistic expressions as *signs*. A sign represents a complex of phonological, syntactic, semantic and order information, each of these linguistic levels having its own definition of well-formedness. This can be represented as follows :

(1) *UCG sign*

Phonology : Category : Semantics : Order

Phonology :
Category :
Semantics :
Order :

We need not concern ourselves here with the *Phonology* field of the sign. Following the categorial tradition, categories can be basic or complex. Basic categories are of the form *Head^Features*

where *Head* is one of the atomic symbols *n* (nouns), *np* (noun phrases) or *s* (sentences) and *Features* is a list of feature values, allowing further specification on these primitive categories. Complex categories are of the form *C/sign*, where C is either atomic or complex and *sign* is a sign, so that departing from traditional CG's, a functor places constraints on the whole sign of the argument rather than on its syntactic category only. The part of a complex category which omits the *Head^Features* information constitutes the *active part* of the sign. The first *accessible sign* in the active part is called the *active sign*, the idea being that e.g. verb valencies are ordered in a list so that each time a valency is consumed, the next sign in the active part becomes the new active sign. Variables are allowed at each level of representation, capturing the notion of imcomplete information. A sign which contains variables can be further specified by the form of unification UCG relies on, i.e. first order term unification, provided as a basic operation in programming langages such as Prolog. The use of variables in UCG syntax makes it heavily polymorphic in the sense that the category identity of a function application may typically depend on the make-up of the argument (e.g. in (4) below). This approach allows to dramatically simplify the set of categories employed by the grammar, while also retaining the fundamental insight of standard categorial grammar, namely that expressions combine as functor and argument.

The semantic representation language used to encode the *Semantics* of a sign is called InL (for *Indexed Language*), and is derived from Discourse Representation Theory (DRT, cf. Kamp (1981)) supplemented by a Davidsonian treatment of verb semantics (Davidson (1967)). The main similarity with the DRT languages lies in the algebraic structure of InL. There are only two connectives for building complex formulas ; an implication that at the same time introduces universal quantification, and a conjunction. The language InL differs in one important respect form the DRT formalism, and thus earns its name ; every formula introduces a designated variable called its *index*, which has a special status. Every sign has an associated ontological type, represented by the *sort* of its index. Subsumption relations hold between certain sorts ; for instance, an index of sort *singular* will unify with an index of sort *object* to yield an index of sort *singular*. Lower case letters represent sorted variables in InL formulas, upper case letters variables over formulas ; the index of an expression appears within square brackets in prenex position. (2) gives example translations of some expressions. :

(2)

Index	*Formula*	*Expression*	*Sort*
[e]	walk (e, X)	walk	event
[x]	student (x)	student	singular object
[x]	[park (y), [x] [in (x, y), man (x)]]	man in a park	singular object
[m]	butter (m)	butter	mass object
[s]	stay (s, X)	stay	state

The *Order* attribute places constraints on the combination rule that may apply to a functor : *pre* on an argument sign Y indicates that the functor X/Y must precede the argument, while *post* indicates that the functor must follow the argument.

Using terms and term unification, the forward version[1]of functional application can then be stated as follows, where upper letters indicate Prolog variables.

[1]Backward application is just the symetrical rules of (3) where the argument precedes the functor and *pre* becomes *post*.

(3) *Forward Application*

Functor	*Argument*
PhonologyF	
: CategoryF / PhonologyA	PhonologyA
: CategoryA	: CategoryA
: SemanticsA	: SemanticsA
: pre	: pre
: SemanticsF	
: OrderF	
⇒	*Result*
	PhonologyF PhonologyA
	: CategoryF
	: SemanticsF
	: OrderF

In effect, the rule states that the active part of the functor sign term unifies with the argument sign. The *Result* is a sign identical to the functor sign, thus effecting a very strict kind of Head Feature Convention, but where the complex category is stripped of its active part and where variables shared by the active part of the functor and the rest of the functor sign may have become the result of the active part unifying with the argument. The resulting phonology consists of the phonology of the functor followed by that of the argument. An illustrative combination is given in (4) below for the sentence *Jean marche* where lines represent the information flow determining ordering : shared variables ensure that *pre* in a verb valency constrains the functor NP that consumes this valency to precede the verb carrying the valency.

(4) Derivation of *Jean marche*

```
jean                                  marche
: C/_                                 : s^[fin]/(_:np:X:pre)
    : (C/(_:np:jean':O)               : [e] marche' (e,X)
    : S                               : _
    : O
: S
:_
    ⇒                                 jean marche
                                      : s^[fin]
                                      : [e] marche' (e, jean')
                                      : _
```

(4) states that a functor may place restrictions on any of the dimensions of its argument. Likewise it will determine the role that the information expressed by its argument plays in the resulting expression. A UCG sign thus represents a complex of constraints along several dimensions. The introducion of signs to the right of the categorial means that the semantic combination is subsumed within a generalized functional application, and that the necessity of constructing specialized functors in the semantics (e.g. via lambda-abstraction) simply disppears.

3 French Linguistic Observations

Word order in French is characterised by three main facts. First, the positioning -left or right- of a particular argument with respect to the verb is relatively free. As illustrated in (5), the subject can appear to the left (5a) or to the right (5b,c) of the verb, or between the auxiliary and

the verb (5d), depending on the morphological class of the NP and on the type of the sentence (declarative, subject-auxiliary inversion, wh-question, etc).

(5) (a) *Sam* lit un livre.
(b) Lit-*il* un livre ?
(c) Quel livre lit *Sam*?
(d) A-t-*il* lu un livre ?

All other arguments can also appear to the left or to the right of the verb under similar conditions. For example, a lexical non-nominative NP can never be to the left of the verb, but clitics and wh-constituents can.

(6) (a) *Le livre lit Sam? (with *Le livre* = Object)
(b) Quel livre lit Sam ? (with *Quel livre* = Object)
(c) Sam le lit.

Second, there seems to be no clear regularities governing the relative ordering of a sequence of arguments. That is, assuming that only adjacent constituents may combine and taking the combinations left-to-right, the combination pattern varies as indicated below of each example in (7). Here again, the permissible distributions are influenced by factors such as the morphological class of the constituents and the verb mood.

(7) (a) Sam offre *à* Luce un livre
[Subj, IObj, Obj]
(b) Sam offre un livre *à* Luce
[Subj, Obj, IObj]
(c) Le lui donne-t-il ?
[Obj, IObj, Subj]
(d) Se le donne-t-il ?
[IObj, Obj, Subj]

Third, cooccurrence restrictions hold between constituents. For example, clitics constrain the positioning and the class of other arguments as illustrated in (8)[2]:

(8) (a) Sam le lui donne
(b) Sam *lui* en donne
(c) Sam *lui* donne un livre
(d) *Sam *lui* le donne
(e) *Sam *lui* y donne

Since the ordering and the positioning of verb arguments in French are very flexible, the rigid ordering forced by the UGC active list and the fixed positioning resulting from the *Order* attribute are rather inadequate. In UCG, the verb typically encodes the notion of a canonical ordering of the verb arguments, but empirical evidence like clitics or interrogatives does not support the notion of a canonical order for French.

4 Extensions to UCG

The UCG formalism and its computational environment have been developped at the Edinburgh Centre for Cognitive Science by Calder et al. (1986). The PimPLE software comprises a Prolog implementation of the PATR-II linguistic environment (including facilities for defining

[2]In italics : the word whose cooccurrence restriction is violated (starred sentences) or obeyed (non stared sentences). For instance, (8d) is starred because *lui* may not be followed by *le*.

templates, lexical rules end path-equations as in PATR-II), and a shift-reduce parser.
Below are presented two French UCG grammars that account for the facts presented in section 3. The first one, dubbed FDP (*French Dialogue Parser*) is the analyser developed at the Laboratoires de Marcoussis (§4.1), whereas the second one, FG (*French Grammar*) has been implemented at the University Blaise Pascal Clermont II (§4.2).

4.1 French Dialogue Parser

As in UCG the main building blocs of the FDP grammar are *grammar rules, lexical entries* and *lexical rules*. Lexical rules operate transformations on signs, grammar rules produce new signs from signs that they combine. Lexical entries constitute the basis of the grammar. They are built up from abstract generic *templates* using PI*m*PLE and further specified in each specified lexical entry. Considering the fact that French morphology is too rich to allow the storage of one lexical entry for each inflected form, only the basic form of the word was stored and a morphological component was introduced into the system. This way, full lexical entries are constructed dynamicallly during the parse. A word of the sentence is analysed morphologically; values for person, gender and number resulting from this analysis are used to instantiate the corresponding features in the basic lexical entry, the value for the tense feature is used to spawn the corresponding lexical rule which will modify the lexical entry obtained by the previous instantiations.

4.1.1 The Sign model

The UGC sign model for FDP obeys the following BNF :

$$Sign \rightarrow Category : Mtrans : Semantics : Order : Phonology$$

Category

$Category \rightarrow Head_{[Mode,Agreement]} \parallel Category\ /\ Sign$
$Head \rightarrow sent \parallel noun \parallel np \parallel pp$
$Mode \rightarrow (fin \parallel inf \parallel pas \parallel psp \parallel cfin) \parallel (subj \parallel obj \parallel a)$
$Agreement \rightarrow Gender : Number : Person$
$Gender \rightarrow masc \parallel fem$
$Number \rightarrow sg \parallel pl$
$Person \rightarrow 1 \parallel 2 \parallel 3$

Category can be either of simple type ($Head_{Mode,Agreement}$) or of complex type (*Category / Sign*).

Mtrans

$Mtrans \rightarrow [Wh,\ Clitic_order,\ Neg]$
$Wh \rightarrow que \parallel nque \parallel wh \parallel whs \parallel inv2$
$Clitic_order \rightarrow v \parallel le \parallel lui \parallel en \parallel y \parallel te \parallel me \parallel se$
$Neg \rightarrow nneg \parallel ne \parallel neg1 \parallel neg2$

The **Mtrans** field rules out different problems concerning the order of constituents. **Wh** is devoted to the compatibility between "wh-constituents" in a sentence. **Clitic_order** deals with the possible order of clitics, which obeys the following transition matrix[3]:

[3]Where G = grammatical, * = non grammatical, ø= impossible.

↗	prod[acc]	prod[dat]	pro[acc]	pro[dat]	y	en[acc]	en[de]	se[acc]	se[dat]
prod[acc]	ϕ	*	ϕ	*	G	ϕ	G	ϕ	*
prod[dat]	*	ϕ	G	ϕ	ϕ	G	G	*	ϕ
pro[acc]	ϕ	*	ϕ	G	G	ϕ	G	ϕ	*
pro[dat]	*	ϕ	*	ϕ	ϕ	G	G	*	ϕ
y	*	ϕ	*	ϕ	ϕ	G	*	*	*
en[acc]	ϕ	*	ϕ	*	*	ϕ	*	ϕ	*
en[de]	*	*	*	*	*	*	ϕ	*	*
se[acc]	ϕ	*	ϕ	*	G	ϕ	G	ϕ	*
se[dat]	*	ϕ	G	ϕ	ϕ	G	G	*	ϕ

Where :
prod[acc] recovers *me, te, nous, vous* in accustive position
prod[dat] recovers *me, te, nous, vous* in dative position
pro[acc] recovers *le, la les* in accusative position
pro[dat] recovers *lui, leur.*

Neg does the same for the different ways to open/close a negative constituent in French. *nneg* corresponds to a non negative constituent, *ne* corresponds to the opening of the negation (the necessary "ne" in formal written standard French), *neg2* corresponds to a first way to close the negative ("pas") and *neg1* is another way to close the negation ("aucun", "rien", "personne", "jamais", ...).
The following matrix states the (in)compatibilities between them :

↗	nneg	ne	neg1	neg2
nneg	G	G	G	G
ne	G	*	*	*
neg1	*	G	G	*
neg2	*	G	*	*

Semantics

Semantics → Index : Predicate : [Arg1, ..., Argn]
Index → atomic value
Predicate → atomic value
Argn → Index || Semantics

Order

Order → pre || post || opt || jump

Phonology

Phonology → "the phonology"

So a sign of the FDP system looks like :

Category : Head^[Feat, Agree] / Signi .../ SignN
Mtrans : [Wh, Clitic_order, Neg]
Semantics : (Index : Predicate : [Argi, ..., ArgN])
Order : A_Order
Phonology : A_Phonology

which will be presented in the following attribute-value pairs structure :

$$\begin{bmatrix} cat : Head_{[Mode,\ Agree]} \ / \ Sign1 \ / \ \ldots \ / \ SignN \\ mtr : [Wh,\ Clitic_Order,\ Neg] \\ sem : \begin{bmatrix} index : Index \\ pred : Predicate \\ args : [Arg1, \ldots, ArgN] \end{bmatrix} \\ ord : Order \\ pho : Phonology \end{bmatrix}$$

Hereafter are some sample lexical entries, for a common noun :

$$\text{camion} \begin{bmatrix} cat : noun_{[(masc:sg:3)]} \\ mtr : \\ sem : \begin{bmatrix} index : \boxed{I} \\ pred : truck \\ args : [] \end{bmatrix} \\ ord : \\ pho : camion \end{bmatrix}$$

An intersective adjective :

$$\text{rouge} \begin{bmatrix} cat : noun_{\boxed{Ag}} / \begin{bmatrix} cat : noun_{\boxed{Ag}} \\ mtr : \boxed{M} \\ sem : \boxed{S} \begin{bmatrix} index : \boxed{I} \\ pred : \boxed{P} \\ args : \boxed{A} \end{bmatrix} \\ ord : post \end{bmatrix} \\ mtr : \boxed{M} \\ sem : \begin{bmatrix} index : \boxed{I} \\ pred : and \\ args : \left[\begin{bmatrix} index : \boxed{I} \\ pred : red \\ args : [] \end{bmatrix}, \boxed{S} \right] \end{bmatrix} \\ ord : \\ pho : rouge \end{bmatrix}$$

A transitive verb :

$$\text{conduit}\left[\begin{array}{l} cat : sent_{[fin,(Gc:sg:3)]} \Big/ \left[\begin{array}{l} cat : np_{[subj,(Gc:sg:3)]} \\ mtr : \\ sem : \boxed{I} \\ ord : pre \\ pho : \end{array}\right] \Big/ \left[\begin{array}{l} cat : np_{[obj]} \\ mtr : \\ sem : \boxed{J} \\ ord : post \\ pho : \end{array}\right] \\ mtr : [nneg, v, nque] \\ sem : \left[\begin{array}{l} index : \boxed{K} \\ pred : give \\ args : [\boxed{I}, \boxed{J}] \end{array}\right] \\ ord : \\ pho : conduit \end{array}\right]$$

A proper noun :

$$\left[\begin{array}{l} cat : \boxed{C} \Big/ \left[\begin{array}{l} cat : \boxed{C} \Big/ \left[\begin{array}{l} cat : np \\ sem : jean \\ ord : \boxed{O} \end{array}\right] \\ sem : \boxed{S} \\ ord : \boxed{O} \end{array}\right] \\ sem : \boxed{S} \\ pho : jean \end{array}\right]$$

4.1.2 Type Raising

UCG is a monostratal framework which allows us to build the semantics of a sentence while it builds the possible derivation corresponding to that sentence. In order to assign the quantifiers their appropriate restrictor and scope, all nominal and prepositional phrases are "type-raised". A Type-Raising rule could be the following :

$$X \rightarrow C/(C/(X : O)) : O$$

So NPs do not have the form :

$$\left[\begin{array}{l} cat : np \\ sem : \boxed{S1} \end{array}\right]$$

but obey the following general pattern :

$$\left[\begin{array}{l} cat : \boxed{C} \Big/ \left[\begin{array}{l} cat : \boxed{C} \Big/ \left[\begin{array}{l} cat : np \\ sem : \boxed{S1} \\ ord : \boxed{O} \end{array}\right] \\ sem : \boxed{S} \\ ord : \boxed{O} \end{array}\right] \\ sem : \boxed{Sem} \end{array}\right]$$

This does not really change for proper names for which the resulting semantics **Sem** would be S. But for universally quantified expressions this will allow the assignment of the correct restrictor and scope of the expression as in "chaque camion" :

$$\begin{bmatrix} cat : \boxed{c} / \begin{bmatrix} cat : \boxed{c} / \begin{bmatrix} cat : np \\ sem : \boxed{I} \\ ord : \boxed{o} \end{bmatrix} \\ sem : \boxed{s} \\ ord : \boxed{o} \end{bmatrix} \\ sem : \begin{bmatrix} index : state \\ pred :\Longrightarrow \\ args : [\begin{bmatrix} index : \boxed{I} \\ pred : truck \\ args : [] \end{bmatrix}, \boxed{s}] \end{bmatrix} \\ pho : [> apply,\ chaque,\ camion] \end{bmatrix}$$

The semantic reason to raise the type of NPs and PPs is not a unique one. Clitics exhibit independently another reason to type-raise NPs and PPs : they capture (on their left) an argument which is subcategorised for them on its right, as it is for example the case with the clitic "le" :

$$\begin{bmatrix} cat : \boxed{c} / \begin{bmatrix} cat : \boxed{c} / \begin{bmatrix} cat : np_{[obj]} \\ sem : \boxed{s_1} \\ ord : post \end{bmatrix} \\ mtr : [\boxed{wh}, (v||en||y||lui), nneg] \\ sem : \boxed{s} \\ ord : \boxed{p}re \end{bmatrix} \\ mtr : [\boxed{wh}, le, nneg] \\ sem : \boxed{s} \\ pho : le \end{bmatrix}$$

4.1.3 Lexical Rules

Lexical rules operate transformations on the root signs with regard to the result of a morphological analysis. They take the root sign found in the lexicon for a particular word and deliver the modified sign corresponding to the particular morphology of that word. Passivisation is an example of such a transformation.

Passive

$$\left[\begin{array}{l} cat : sent_{[fin,\boxed{4}]} \Big/ \left[\begin{array}{l} cat : np_{[subj,\boxed{4}]} \\ sem : \boxed{5} \\ ord : pre \end{array}\right] \Big/ \left[\begin{array}{l} cat : np_{[obj]} \\ sem : \boxed{6} \\ ord : post \end{array}\right] \\ mtr : [nneg, v, nque] \\ sem : \boxed{10} \left[\begin{array}{l} index : \boxed{1} \\ pred : \boxed{9} \\ args : \langle\boxed{5}, \boxed{6}\rangle \end{array}\right] \\ ord : \boxed{2} \\ pho : \boxed{3} \end{array}\right]$$

$$\Rightarrow$$

$$\left[\begin{array}{l} cat : sent_{[pas,\boxed{4}]} \Big/ \left[\begin{array}{l} cat : np_{[subj,\boxed{4}]} \\ sem : \boxed{5} \\ ord : pre \end{array}\right] \Big/ \left[\begin{array}{l} cat : pp_{[par]} \\ sem : \boxed{5} \\ ord : post \end{array}\right] \\ mtr : [nneg, v, nque] \\ sem : \boxed{10} \\ ord : \boxed{2} \\ pho : \boxed{3} \end{array}\right]$$

So the sign for "donnée" would be :

$$\left[\begin{array}{l} cat : sent_{[pas,\boxed{Ag}(fem:pl:3)]} \Big/ \left[\begin{array}{l} cat : np_{[subj,\boxed{Ag}]} \\ sem : \boxed{P} \\ ord : pre \end{array}\right] \Big/ \left[\begin{array}{l} cat : pp_{[par]} \\ sem : \boxed{A} \\ ord : post \end{array}\right] \\ mtr : [nneg, v, nque] \\ sem : \left[\begin{array}{l} index : \boxed{I} \\ pred : give \\ args : \langle\boxed{A}, \boxed{P}\rangle \end{array}\right] \\ pho : donne'es \end{array}\right]$$

4.1.4 Grammar Rules

Binary Rules

The binary grammar rules are basically concatenation rules as in traditionnal categorial grammar (functional backward and forward applications).

The rules are presented in the following way :

Name of the rule : *Mother* $\triangleright$ *Daughter1, Daughter2.*

Forward Apply

$$\begin{bmatrix} cat : \boxed{C} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [> apply, \boxed{W1}, \boxed{W2}] \end{bmatrix} \triangleright \begin{bmatrix} cat : \boxed{C} / \boxed{Active} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : \boxed{W1} \end{bmatrix}, \boxed{Active} \begin{bmatrix} ord : pre \\ pho : \boxed{W2} \end{bmatrix}$$

Backward Apply

$$\begin{bmatrix} cat : \boxed{C} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [< apply, \boxed{W1}, \boxed{W2}] \end{bmatrix} \triangleright \boxed{Active} \begin{bmatrix} ord : post \\ pho : \boxed{W1} \end{bmatrix}, \begin{bmatrix} cat : \boxed{C} / \boxed{Active} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : \boxed{W2} \end{bmatrix}$$

According to the lexical entries already introduced we can illustrate the derivation of "camion rouge" using Backward Apply where the sign for "camion" is the argument of the rule and the sign for "rouge" is the functor of the rule, leading to the following resulting sign as mother of the rule :

$$\begin{bmatrix} cat : noun_{[(masc:sg:_)]} \\ mtr : \boxed{M} \\ sem : \begin{bmatrix} index : \boxed{I} \\ pred : and \\ args : [\begin{bmatrix} index : \boxed{I} \\ pred : red \\ args : [] \end{bmatrix}, \begin{bmatrix} index : \boxed{I} \\ pred : truck \\ args : [] \end{bmatrix}] \end{bmatrix} \\ ord : \\ pho : [< apply, camion, rouge] \end{bmatrix}$$

Apart from the Binary Functional Application (Forward and Backward) rule, a **Forward Gap Rule** handles "movement" phenomena. It can be seen as a particular use of Functional Composition (Steedman) restricted so as to mimic a Gap Threading.

Forward Gap

$$\begin{bmatrix} cat : \boxed{C} / \begin{bmatrix} cat : \boxed{C2} (np||pp) \\ mtr : \boxed{M2} \\ sem : \boxed{S2} \\ ord : pre \\ pho : \boxed{W2} (\neg lex) \end{bmatrix} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [> gap, \boxed{W}, \boxed{W1}] \end{bmatrix} \triangleright \begin{bmatrix} cat : \boxed{C} / \begin{bmatrix} cat : \boxed{C1} \\ mtr : \boxed{M1} \\ sem : \boxed{S1} \\ ord : pre \\ pho : \boxed{W1} \end{bmatrix} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : \boxed{W} \end{bmatrix}, \begin{bmatrix} cat : \boxed{C1} / \begin{bmatrix} cat : \boxed{C2} \\ mtr : \boxed{M2} \\ sem : \boxed{S2} \\ ord : \\ pho : \boxed{W2} \end{bmatrix} \\ mtr : \boxed{M1} \\ sem : \boxed{S1} \\ ord : pre \\ pho : \boxed{W1} \end{bmatrix}$$

This rule will be implied in the resolution of all linguistic phenomena which include constituent movement such as wh-movement or (non-subject) relative-clauses.

Unary Rules

Several unary rules handle optionality of arguments (**Delopt**), order of arguments (**Jump**), and addition of prepositional "adjunct" arguments on nouns, verbs and sentences fragments (**Ppn**, **Ppv**, **Pps**).

Delopt

$$\begin{bmatrix} cat : \boxed{C} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [delopt, \boxed{W}] \end{bmatrix} \triangleright \begin{bmatrix} cat : \boxed{C} / \begin{bmatrix} cat : (np||pp) \\ sem : unknown \\ ord : opt \end{bmatrix} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : \boxed{W} \end{bmatrix}$$

Jump

$$\begin{bmatrix} cat : \boxed{C} / \begin{bmatrix} cat : \boxed{C1} \\ mtr : \boxed{M1} \\ sem : \boxed{S1} \\ ord : post \\ pho : \boxed{W1} \end{bmatrix} / \boxed{Act} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [jump, \boxed{W}] \end{bmatrix} \triangleright \begin{bmatrix} cat : \boxed{C} / \boxed{Act} / \begin{bmatrix} cat : \boxed{C1} (np||pp) \\ mtr : \boxed{M1} \\ sem : \boxed{S1} \\ ord : jump \\ pho : \boxed{W1} \end{bmatrix} \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [jump, \boxed{W}] \end{bmatrix}$$

The **Delopt** and **Jump** rules could be considered as lexical rules transforming (once) the basic signs. The **Jump** is presented here as a unary rule, but in fact it could be a ternary rule involving the verb and its two arguments in reversed order (X → X/A1/A2, A1, A2) which would be more efficient with respect to parsing.

The Ppv, Ppn and Pps rules state that verbs, nouns and sentences can be modified by prepositional adjunct constituents. These rules could be extended to any kind of adjunct constituents.

They are balanced with the fact that prepositional adjunct constituents are ambiguous modifiers with respect to their attachment, as follows :

Ppv

$$\left[\begin{array}{l} cat : \boxed{C} \,/\, \left[\begin{array}{l} cat : pp_{[adj]} \\ sem : \boxed{I} \\ ord : post \end{array}\right] \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [ppv, \boxed{W}] \end{array}\right] \triangleright \left[\begin{array}{l} cat : \boxed{C} \left[sent_{[(fin \| inf \| imp)]} / \left[\begin{array}{l} cat : np_{[subj]} \\ ord : pre \end{array}\right] \right] \\ mtr : \boxed{M} \\ sem : \boxed{S} \left[\begin{array}{l} index : \boxed{I} \\ pred : \boxed{P} \\ args : \boxed{A} \end{array}\right] \\ ord : \boxed{O} \\ pho : \boxed{W} \end{array}\right]$$

Ppv

$$\left[\begin{array}{l} cat : \boxed{C} \,/\, \left[\begin{array}{l} cat : pp_{[adj]} \\ sem : \boxed{I} \\ ord : post \end{array}\right] \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [ppv, \boxed{W}] \end{array}\right] \triangleright \left[\begin{array}{l} cat : \boxed{C} \left[sent_{[(fin \| inf \| imp)]} / \left[\begin{array}{l} cat : np_{[subj]} \\ ord : pre \end{array}\right] / \left[cat : (np \| pp) \right] \right] \\ mtr : \boxed{M} \\ sem : \boxed{S} \left[\begin{array}{l} index : \boxed{I} \\ pred : \boxed{P} \\ args : \end{array}\right] \boxed{A} \\ ord : \boxed{O} \\ pho : \boxed{W} \end{array}\right]$$

Ppn

$$\left[\begin{array}{l} cat : \boxed{C} \,/\, \left[\begin{array}{l} cat : pp_{[adj]} \\ sem : \boxed{I} \\ ord : post \\ pho : lex \end{array}\right] \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [ppn, \boxed{W}] \end{array}\right] \triangleright \left[\begin{array}{l} cat : \boxed{C}\ noun \\ mtr : \boxed{M} \\ sem : \boxed{S} \left[\begin{array}{l} index : \boxed{I} \\ pred : \boxed{P} \\ args : \boxed{A} \end{array}\right] \\ ord : \boxed{O} \\ pho : \boxed{W} \end{array}\right]$$

Pps

$$\left[\begin{array}{l} cat : \boxed{C} \Big/ \left[\begin{array}{l} cat : pp_{[adj]} \\ sem : \boxed{I} \\ ord : pre \\ pho : lex \end{array}\right] \Big/ \left[\begin{array}{l} ord : post \\ pho : comma \end{array}\right] \\ mtr : \boxed{M} \\ sem : \boxed{S} \\ ord : \boxed{O} \\ pho : [pps, \boxed{W}] \end{array}\right] \triangleright \left[\begin{array}{l} cat : \boxed{C}\ sent_{[fin]} \\ mtr : \boxed{M} \\ sem : \boxed{S} \left[\begin{array}{l} index : \boxed{I} \\ pred : \boxed{P} \\ args : \boxed{A} \end{array}\right] \\ ord : \boxed{O} \\ pho : \boxed{W} \end{array}\right]$$

4.1.5 Extensions to Constraints Handling

The PImPLE environment has been modified in several ways to support the FDP Grammar design. The main extension concerns the handling of constraints. PImPLE was originally able to handle negative and disjunctive constraints. We have introduced a new facility which concerns *co-relational constraints.* This facility allows us to express the relations between different fields of the sign as the following example illustrates, which co-relates the semantics of the nouns to its morphological agreement :

$$\left[\begin{array}{l} cat : np_{[\neg\ (masc: \boxed{Nu} \neq 3)]} \Big/ \left[\begin{array}{l} cat : noun_{[\neg\ (masc: \boxed{Nu} \neq 3)]} \\ sem : \boxed{S} \end{array}\right] \\ sem : \boxed{R_s} \\ crc : [\boxed{Nu}, \boxed{R_s}] = [sg,\ singular(\boxed{S})] \parallel [pl,\ plural(\boxed{S})] \end{array}\right]$$

These complex constraints basically allow us to express disjunction over sets of variables. For a set of variables S_2 ($[Var'_1, Var'_2, \ldots, Var'_n]$) where the Var_i are dependent on each other, those variables are said to be *co-relationnally* constrained. According to the value V_i will admit, the other variables of the set will take *co-relationally* different values. They obey the following syntax :

$$[Var_1, Var_2, \ldots, Var_n] = [x_1, x_2, \ldots, x_n] \parallel [y_1, y_2, \ldots, y_n] \parallel \ldots$$

where V_i is a PROLOG variable, and x_i either a PROLOG term or a negative or disjunctive constraint which Var_i handles, where $\parallel$ is the disjunctive operator. They are treated in the following way : if a variable V_i in a set S takes a particular value in the disjunction of possible values for it then the whole co-relational constraints expression will be reduced logically following the subset of values S has taken (or can take) and the remaining disjunction will be calculated. For example, on the constraint :

$$[V_1\ V_2\ V_3] = [a, b, c] \parallel [a, e, f] \parallel [g, h, i]$$

if only V_1 gets instantiated (to a) then the expression will be reduced and will become :

$$[V_2, V_3] = [b, c] \parallel [e, f]$$

4.1.6 Grammar Coverage

Given the main goal of the ACORD project, considerable effort has been put into the description of interrogative structures in French. A general, theory independent, description is to be found in Deliverable T2.3 (*Contextual Phenomena in Dialogue*). This has been reformulated according to the UCG framework. The actual grammar covers yes/no questions and wh-questions. Both types of question require the treatment of movement phenomena, i.e. verb/subject inversion or wh-movement. Other structures that can be parsed cover : nominal and adjectival phrases (nouns, personal pronouns, determiners, intersective adjectives, interrogative nominal structures), prepositions (case marking prepositions, verbal prepositions, including interrogative verbal forms), verbal base forms (with nouns phrase arguments, with prepositional phrase arguments, including interrogative verbal forms), relative clauses (on noun phrases arguments of verbs, on prepositional adjuncts of verbs), auxiliaries (with passive participle, with past participle), adjuncts (noun modifiers, verb phrase modifiers, sentence modifiers).

4.2 French Grammar

In FG, UCG has been modified in two ways. Firstly, the active part of a verb category is represented as a set rather than a list. Secondly, a feature system is introduced which embodies the interactions of the different elements conditioning word order.

4.2.1 Sign Structure and Combination Rule : from an Active List to an Active Set

To accomodate the analysis, the sign structure and the combination rule had to be slightly modified. In FG (cf. Bès et al. (1989), Gardent et al. (1989) and Baschung (1990)), a sign is as follows.

(9) *FG sign*

Phonology :
Category :
Valencies :
Features :
Rule_sensitivity :
Semantics :
Optionality :

Semantics and Phonology are as in UCG. The *Category* attribute differs from UCG in that (i) there are no *Features* associated with the *Head* and (ii) the active part of a verb is viewed as a set rather than as a list (§4.3.3), which constitutes the *valencies* field of the sign. There is also an *Optionality* field for order[4]. The features are accessible independently from the *Category* and to be found both in the *Features* and *Rule_sensitivity* fields. Only those relevant to order constraints are mentioned here, i.e. case (represented via the variable *Case*, the verb form (*VForm*, the morphological class of NPs – lexical (*lex*), clitic (*le, lui...*) or interrogative (*wh*) – or of verbs, and the class of the last concatenated element to the left (*ImF*) or the right (*ImB*). The latter features are updated by the combination rule. For instance, the sign associated with *Sam lui a donné un livre* will have *lex* as values for ImB and ImF whereas *lui a donné un livre* has *lui* and *lex* respectively. The *Features* attribute can be represented as in (10) below, where the same feature may occupy a different position in the feature list of different linguistic units, e.g. feature list of verb valencies or active signs of NPs (valency features) vs. feature list of verb

[4]In the rest of this paper, the *Semantics* and the *Optionality* attributes will be omitted since they have no role to play in our treatment of word order while *Phonology* will only be represented when relevant.

signs or NPs (sign features).

The features attribute in :

(10)	(a)	Valency features	[Case, Class, ImF, ImB]
	(b)	Sign features	[Class, (ImF : ImB), VForm]

As illustrated in (11), the *Rule_sensitivity* attribute has two parts, one for when the functor combines forwards (**fc**), the other for when it combines backwards.

(11) *Sign features within the Rule-sensitivity field of an active sign :*

CdtsF⇒ fc ⇒ ResFeatF, CdtsB ⇒ bc ⇒ ResFeatB

where *Cdts* and *Resfeat* are lists of feature values whose order and content are independent from those of the *Features* attribute. The intuition behind this is that functors (i.e. type-raised NPs) are two-sided i.e., they can combine to the left and to the right but under different conditions and with different results. The features in *Cdts* place constraints on the features of the argument while the features in *Resfeat* are inherited by the resulting sign. These effects are obtained by the unification of shared variables in the rules of combination. Omitting *Semantics* and *Optionality* attributes, the forward combination rule is as follows.

Forward Combination (FC)

Functor	*Argument*
PhonologyF	
CategoryF /PhonologyA:	PhonologyA:
CategoryA:	CategoryA:
ValenciesA:	ValenciesA':
[ClassA, ...] :	[ClassA, (ImF: ImB), VForm]
([(ImF: ImB), VForm] ⇒ fc ⇒	:_
[(ClassF:_), VForm'], _)	
:_:_:_	

{combine(ValenciesA, ValenciesA', ValenciesR)}

⇒ *Result*
PhonologyF PhonologyA:
CategoryF:
ValenciesR:
[ClassA, (ClassF: ImB), VForm']:_

The resultant category is the category of the functor (*CategoryF*). The rule requires that the valencies of the active sign of the functor (*ValenciesA*) and the valencies of the argument (*ValenciesA'*) be combined to get the resultant valencies (*ValenciesR*). The notion of combination relies on the idea that the valencies are forming a set rather than a list. More precisely, given a type raised NP *NP1* with category $C/C[NP_i]$ where NP_i is a valency sign, and a verb *V1* with category *s: ValSet* where *ValSet* is a set of valency signs, *NP1* combines with *V1* to yield *V2* iff NP_i unifies with some NP-valency sign in the active set *ValSet* of the verb. *V2* is identical to *V1* except that the unifying NP_i valency sign has been removed from the active set and that some feature in *V1* will have been instantiated by the rule. Forward combination further requires that the two features in the condition list to fc unify with the *ImF* and *VForm* features of the

argument (the features conditioning bc are ignored in that case).
Finally the rule requires that the class of the result be the class of the argument (*ClassA*), that the information about the last right-concatenated functor (*ImB*) be transmitted without change from the argument, and that the class of the functor become the immediate information on the left of the result (*ClassF*). The resulting features determined by the order of combination (e.g. *VForm'*) are inherited from the *ResFeatF* (see (11)). The percolation of features by rule is crucial to our treatment of word order. It is illustrated by (12) below where the sign associated with the clitic *le* combines with the sign for *regarde* to yield a new sign *le regarde.*

(12) *Derivation of "le regarde"*

le:
C/(C: {(np:_:[obj...]:_) ...}
:[verb ...]
:([i or lui, ind] ⇒ fc ⇒ [le, ind], [i, imp] ⇒ bc ⇒ [le, imp])
:{}
:[le ...]
:_

regarde:
s: {(np:_:[obj ...]:_), (np:_:[nom ...]:_), (np:_ :[mod ...]:_)}
:[verb, (i: i), ind]
:_

le regarde :
s:{(np:_:[nom ...]:_), (np:_:[mod ...]:_)}
:[verb, (le: i), ind]
:_

When *le* is used as a forward functor, the conditions on fc require that the argument (i.e. the verb) bears for the feature ImF the value *lui or i* (where *i* stands for *initial* state) thus requiring that the verb has not combined with anything on the left. When it combines by BC, the conditions on bc ensure that the argument has not conbined with anything on its right and that it has mood *imperative.* In this way, all sentences in (13) are parsed appropriately.

(13) (a) Il le lui donne.
(b) *Il lui le donne.
(c) Donne le lui.
(d) *Donne lui le.

The backward combination rule (BC) functions like FC except for two things. First, the argument must be to the left of the functor and second, the condition field considered is that of bc rather than of fc. There is also a deletion rule to eliminate optional valencies. No additional rule is needed.

4.2.2 Expressing the Variables Underlying Word Order Constraints

In FG, there are no *post* and *pre* primitive values associated with specific verb valencies. Instead, features interact with combination rules to enforce the constraints on word order. For instance, a lexical NP can be subject or object. If it is subject and it is to the left of the verb, it cannot be immediately followed by a wh-constituent. If it is subject and it is placed to the right of the verb, it must be immediately adjacent to it. These constraints can be stated using unification along the following lines. A verb valency is of the form

(14) (np:_ :[...X, Y ...] :_)

where X and Y are either the anonymous variable or a constant. They state constraints, among others, on possible values of *ImF* and *ImB* features of the verb. Recall that a valency is a sign which is a member of a set in the *Valencies* attribute of a verbal sign. The active sign of a type raised NP is of the form :

(15) C:{(np:_ :[...V1, V2 ...]:_) ...}
:[verb ...]
:([V1 ...] ⇒ fc ⇒ Z, [V2 ...] ⇒ bc ⇒ W)

By rule, V1 and V2 in the *Valencies* attribute of (15) must unify with X and Y, respectively, in the verb valency (14). Being shared variables, they transmit the information to the *Conditions* on concatenation by FC (fc) and BC (bc) respectively. Furthermore, V1 and V2 in the *Rule-sensitivity* attribute of the functor must unify, by rule, with some specified features in the verb *Features* attribute represented in (16).

(16) [verb, (ImF : ImB) ...]

The flow of information between (14), (15) and (16) is represented in (17), where (17a), (17b), and (17c) correspond to (14), (15) and (16) respectively. (17a) and (17c), which express the *Valencies* and the *Features* attributes of the same verbal sign, have been dissociated for the sake of clarity.

(17) *Flow of information between functor and argument*

(a) (np:_ :[...X, Y ...]:_)

(b) C:{(np:_ :[...V1, V2 ...]:_) ...}:[verb ...] — via FC

:([V1 ...] ⇒ fc ⇒ Z, [V2 ...] ⇒ bc ⇒ W) — via BC

(c) [verb, (ImF: ImB)...]

For example, assume the nominative valency (18a) in the verbal sign *téléphone à la fille*, whose *Features* attribute is as in (18c), and the lexical sign *Jean* (18b).

(18) *Flow of information between "Jean" and "téléphone à la fille"*

(a) (np:_ :[nom ...~wh, i ...]:_)

(b) C:{(np:_ :[...nom or obj ...V1, V2 ...]:_) ...}:[verb ...]

:([V1 ...] ⇒ fc ⇒ Z, [V2 ...] ⇒ bc ⇒ W)

(c) [verb, (i: lex) ...]

The concatenation by FC is allowed (~*wh* is compatible with *i*), the requirement extracted from the verbal valency being that the last left-concatenated contituent with the verb is not a wh-constituent. But a concatenation by BC will fail (*i* does not unify with *lex*). Thus the examples in (19), among others, are correctly recognised.

(19) (a) Jean téléphone à la fille.
(b) *Téléphone à la fille Jean.
(c) *Jean à quelle fille téléphone ?
(d) A quelle fille téléphone Jean ?

4.2.3 Combination Procedure

In the combination rules, there is no direct unification (i.e. Prolog's term unification) between the set of valencies of the functor's active sign and the set of valencies of the argument. The combination procedure deals precisely with the relationship between sets of valencies, and it can be stated in the following general form :

(20) $(V_A - V_{FAS} \cup V_F = V_R)$
where V_R is the valencies set of the result
V_A is the valencies set of the argument
V_{FAS} is the valencies set of the functor's active sign
V_F is the valencies set of the functor

It is easy to verify that the derivation of *le regarde* in (12) corresponds to the following operation[5]

(21) $(\{obj, nom, mod\} - \{obj\}) \cup \{\} = \{nom, mod\}$

In other words, for the two signs to be concatenated, a valency of the argument must be consumed. In other cases, no valency is consumed e.g. when a verb (which is an argument) concatenates with the negative particle *ne*, whose sign may be schematized as follows :

(22) ne:
C/(C: {}
: [verb ...]
: ...)
: {}
: [ne ...]
:_

The derivation of *ne regarde* corresponds thus to (23), where the valencies set of the verb remains unchanged :

(23) $(\{obj, nom, mod\} - \{\}) \cup \{\} = \{obj, nom, mod\}$

Another case is illustrated by the reflexive *se*, which must consume an object valency and agree with the nominative valency :

(24) se:
C/(C : {(nom, G:N:p3), (obj, G:N:p3)}
: [verb ...]
: ...)
: {(nom, G:N:p3)}
: [se ...]
:_

The derivation of *se regarde* corresponds to (25) :

[5] Herafter, only the relevant information w.r.t. valencies will be explicitly stated. A set {obj, nom, mod} is thus short for {(np :_ : [obj ...] :_), (np :_ : [nom ...] :_), (np :_ : [mod ...] :_)}. In FG, an optional *mod* valency (modifier) is systematically introduced in the valencies set of each verb.

(25) ({obj, nom, mod} - {(obj, G:N:p3), (nom, G:N:p3)}) ∪ {(nom:G:N:p3)}
= {(nom, G:N:p3), mod}

where two valencies are to be consumed. The combination procedure unifies these valencies with the *obj* and *nom* valencies of the valencies set of the argument. The latter are then removed from the verb valencies set, whereas the functor valencies set reintroduces the instantiated *nom* valency in the valencies set of the result. Both consuming and reintroducing the *nom* valency has therefore the effect of unifying the values for agreement of the *nom* and *obj* valencies of the verb.
The two generic operators - and ∪ of (20) are sensitive to the kind of objects they apply to. This polymorphism relies on a typology of objects and on definitions such as (26), where the symbols -' and ∪' stand for unusual subtraction and set union, respectively.

(26) If B = {x, *y, z} (i.e. B is of type T*),
then A -' B = (A - x) iff ∀ v ∈ (A_ x), v unifies with y
else A-'B = ⊥

Informally, the symbol * allows some additional verification on the valencies set : for two signs to be concatenated, one or more constraints have to be checked, which may instantiate new values on the valencies set of the result. (27) presents for instance the (simplified) sign for the complementizer *que* :

(27) que:
C/ (C : {(s : {npX, ~nom, wh} ...) ...} ...)
/(s : {(npX, ~nom, wh), (*np, wh, op)} ...)
:{(npX, ~nom, wh)}
: [comp ...]
: _

Concatenating *que* with *regarde* involves the operation in (26) :

(28) $(V_A - V_{FAS}) \cup V_F = V_R$

(a) ({obj, mod} - {(obj, wh)}) ∪ {(obj, wh)} = {(obj, wh), (mod, wh)} [6]

(b) ({mod} - {(mod, wh)}) ∪ {(mod, wh)} = {(mod, wh)}

In order to combine *que* with *regarde*, the combination procedure verifies that each verb valency (apart from that which is to be consumed) unifies with the **np* valency : from a linguistic point of view, this means that either an optional valency or a single non optional valency may be extracted from a finite embedded clause, that extraction marks the remaining verb valencies as being interrogative (*wh*) and that it also verifies that all the obligatory valencies have already been consumed. This process is only possible for the concatenation of *que* with *NP[nom] regarde* ((28a)) and with *NP[nom] regarde NP[obj]* ((28b)), which in turn become functors w.r.t the matrix verb, e.g. *promet*, where they consume the (s : np, {~nom} ...) valency :

(29) promet :

s:{(np:_ : [à ...] :_), (np: _ : [nom ...] :_), (np :_ : [mod ...] :_), (s:{np, ~ nom : [obj...] : _)}
:[verb ...]
:_

[6] Independently motivated principles of the grammar prevent two (or more) wh-elements to be fronted at the begining of the sentence.

We obtain thus :

(30) Jean promet que Marie regarde
Jean promet que Marie regarde le tableau
Quel tableau Jean promet que Marie regarde ?
*Quel tableau Jean promet que regarde ?
*Qui quel tableau Jean promet regarde ?
...

For further details about this combination procedure, see Baschung (1990).

4.2.4 Efficient Parsing

Because the subcategorisation information is represented as a set rather than as a list, there is no constraint on the order in which each valency is consumed. This raises a problem with respect to parsing which is that for any triplet X, Y, Z where Y is a verb and X and Z are arguments to this verb, there will often be two possible derivations i.e. (XY)Z and X(YZ). The problem of spurious parses is a well-known one in extensions of pure categorial grammar. It derives either from using other rules or combinators for derivation than just functional application (Pareschi and Steedman (1987), Wittenburg (1987), Moortgat (1987), Morrill (1988)) or from having unordered set valencies (Karttunen (1986)), the latter case being that of FG. The solution we offer is to augment a shift-reduce parser with a heuristic whose essential content is that no same functor may consume twice the same valency on the same predicate unless the functor combines with two edges that stretch over the same region. This ensures that for all semantically unambiguous sentences, only one parse is output. To ensure that a parse is always output whenever there is one, that is to ensure that the parser is complete, the heuristic only applies to a restricted set of edge pairs and the chart is organized as a queue. Coupled with the partial associativity of FG, this strategy guarantees that the parser is complete. Consider the derivation in (31), avoiding spurious parses :

```
(31)  Jean          aime          Marie
0 - Ed1 - 1 - Ed2 - 2 - Ed3 - - 3
0 ———————— Ed4 ———————— 2                    Ed4 = Ed1 (Ed2, pl, subj)
0 ———————— Ed5 ———————— 2                    *Ed5 = Ed1 (Ed2, pl, obj)
          1 ———————— Ed6 ———————— 3          Ed6 = Ed3 (Ed2, pl, obj)
          1 ———————— Ed7 ———————— 3          Ed7 = Ed3 (Ed2, pl, subj)
0 ———————————— Ed8 ———————————— 3            Ed8 = Ed1 (Ed6, pl, subj)
0 ———————————— Ed9 ———————————— 3            Ed9 = Ed3 (Ed4, pl, obj)
0 ———————————— Ed10 ——————————— 3            *Ed10 = Ed1 (Ed7, pl, obj)
```

where Ed4 = Ed1 (Ed2, pl, subj) indicates that the edge Ed1 reduces with Ed2 by consuming the subject valency of the edge Ed2 with predicate p1.
Ed5 and Ed10 are ruled out by the grammar since in French no lexical (as opposed to clitics and wh-NP) object NP may appear to the left of the verb. Ed9 is ruled out by the heuristic since Ed3 has already consumed the object valency of the predicate p1 thus yielding Ed6. Note also that Ed1 may consume twice the subject valency of p1 thus yielding Ed4 and Ed8 since the heuristic does not apply to pairs of edges labelled with signs of type f1 and f0 respectively.

4.2.5 Implementation and Coverage

FG handles the core french data w.r.t. linearity i.e. declarative, interrogative and negative sentences in all moods, with simple or compound verb forms : yes/no questions, wh-constituents, in-

terrogative inversion, clitic placement (incorporating reflexives), auxiliaries (incorporating agreement between the participle and its fronted NP object), passives, finite and infinitival embedded classes (incorporating some relative clauses), phrase or sentence negation. Unbounded dependencies are lexically handled, without using gap threading like in standard UCG or functional composition like in CCG (cf. Steedman (1986)) or in FDP. Furthermore, some non local order constraints are accounted for without problem, like the distribution of negative particles, or the necessity for an inverted sentence (with a lexical NP subject on the right of the verb) to incorporate a wh-element on the left of the verb.

4.3 Conclusion

In extending UCG to allow the treatment of unbounded dependency constructions and partially free word order, heavy use is made of unary rules (Wittenburg (1986), Calder et al. (1986)). That is why the basic insights of FG form an interesting synthesis of the main features of generalized phrase structure grammar and categorial grammar. As in GPSG, in FG verbal valencies are not ordered, without using LP-statements. Contrary to GPSG (and as in standard UCG) there is no need for specific feature instantiation mechanisms, the information being circulated via the grammar rules by means of unification. Contrary to UCG the number of grammar rules are kept down to three in FG (two combination rules and one deletion rule interveening once at the end of the parse). The combination procedure is defined by a finite number of operations between sets of valencies. On the other hand, the grammar does not overgenerate i.e. it accepts all and only the grammatical sentences. This result is due both to the feature mechanism which restricts the possible combinations and to the systematic elimination of spurious ambiguity as well as spurious *lexical* ambiguity (i.e. ambiguity not motivated by actual categorial or semantical differences but allowing for the placement of the word in all its correct positions in a sentence). The system described in §1 is deficient in some other respects. For example, requiring coincidence between the phonological, syntactic and semantic functors may be too strict. The problem of quantifier scoping is a case in point, because it is the order of concatenation which determines (sometimes incorrectly) the scope of quantifiers. In FG, the *Semantics* part of the sign contains a non-standard InL formula, dubbed InL'. Different derivations of a string may yield a sentence sign whose InL' formulae are formally different, in that the order of their sub-formulae are different, but the set of their sub-formulae are equal. Furthermore, sub-formulae are so built that formulae differing in the ordering of their sub-formulae can in principle be translated to a semantically equivalent representation in a first order predicate logic. This is because: (i) in InL', the scope of scoping operators is left undefined ; (ii) shared variables express the relation between determiner and restrictor, and between scoping operators and their semantic arguments; (iii) the grammar places constants (i.e. proper nouns) in the specified place of the argumental list of the predicate. For instance, FG associates to (32) the InL' formulae in (33a) and (33b) :

(32) Un garçon présente Marie *à* une fille

(33) (a) [E] [ind(X) & garçon(X) & ind(Y) & fille(Y) & présenter (E, X, marie, Y)]

(b) [E] [ind(Y) & fille(Y) & ind(X) & garçon(X) & présenter (E, X, marie, Y)]

While a scoping operator of a sentence constituent is related to its argument by the index of a noun (as in the above (33)), the relation between the argument of a scoping operator and the verbal unit is expressed by the index of the verb. For instance, the negative version of (32) will incorporate the sub-formula *neg (E)*. In InL' formulae, determiners immediately precede their restrictors. In formally different InL' formulae, only the ordering of scoping operators sub-formulae can differ, but this can be shown to be irrelevant with regard to the semantics. In French, scope ambiguity is the same for members of each of the following pairs, while the ordering of the corresponding semantic sub-formulae, thanks to concatenation of adjacent signs, is inescapably different.

(34) (a) Jacques avait donné un livre (a) à tous les étudiants (b)
(a') Jacques avait donné à tous les étudiants (b) un livre (a)
(b) Un livre a été commandé par chaque étudiant(a) à une librairie (b)
(b') Un livre a été commandé à une librairie (b) par chaque étudiant (a)

At the grammatical level (i.e. leaving aside pragmatic considerations), the translation of an InL' formula to a scoped logical formula can be determined by the specific scoping operator involved (indicated in the sub-formula) and by its relation to its semantic argument (indicated by shared variables). This translation must introduce the adequate quantifiers, determine their scope and interpret the '&' separator as either $\wedge$ or $\rightarrow$, as well as introduce $\perp$ in negative forms. For instance, the InL' formulae in (Y) translate[5] to :

(35) $\exists$E, $\exists$X, $\exists$Y (garçon (X) $\wedge$ fille (Y) $\wedge$ présenter (E, X marie, Y)).

We assume here the possibility of this translation without saying any more on it. Since this translation procedure cannot be defined on the basis of the order of the sub-formulae corresponding to the scoping operators, InL' formulae which differ only w.r.t. the order of their sub-formulae are said to be semantically equivalent.
In any case, it has been tentatively demonstrated that UCG is a formally tractable formalism within which a great deal of linguistic data may be expressed and implemented.

References

Bar-Hillel, Y. (1953) A quasi-arithmetical notation for syntactic description, *Language*, 29, 47-58.

Baschung, K. (1990) Grammaires d'unification à traits et contrôle des infinitives, Thèse de Doctorat, Université Blaise Pascal Clermont II. [To be published by Editions ADOSA, Clermont-Ferrand].

Baschung, K., Bès, G.G., Corluy, A. & Guillotin, T. (1986) Auxiliaries and Clitics in French UCG Grammar, *Proceeding of the Third European Chapter of the Association for Computational Linguistics*, Copenhague, 173 - 178.

Bès, G.G. & Gardent, C. (1989) French Order without Order, *Proceedings of the fourth European Chapter of the Association for Computational Linguistics*, Manchester, 249-255

Calder, J., Klein, E., Moens, M. & Reape, M. (1988) Global Constraints in Unification Grammars, *ACORD Deliverable T1.6*, Centre for Cognitive Science, Edinburgh.

Calder, J., Moens, M. & Zeeavat, H. (1986) A UCG Interpreter, *ACORD Deliverable T2.6*, Centre for Cognitive Science, University of Edinburgh.

Davidson, D. (1967) The logical form of action sentences, in Rescher, N. (ed.) *The Logic of Decision and Action*, Pittsburgh, University of Pittsburgh Press.

Gardent, C., Bès, G.G., Jurie, P.F. & Baschung, K. (1989) Efficient Parsing for French, *Proceedings of the Conference of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, 280-287.

[5] In (34) $\exists$ can be paraphrased as "There exists an event".

Kamp, H. (1981) A Theory of Truth and Semantic Representation, in Groenendijk J.A.G., T.M.V. Jansen and M.B.J. Stokhof (eds.) *Formal Methods in the Study of Language*, Vol. 136, 277-322, Amsterdam : Mathematical Centre Tracts.

Karttunen, L. (1986) *Radical Lexicalism*, Report CSLI-86-68, Center for the Sudy of Language and Information, Paper presented at the Conference on Alternative Conceptions of Phrase Structure, July 1986, New-York.

Moens, M., Calder, J., Klein, E., Reape, M. & Zeevat, H. (1989) Expressing Generalisations in Unification-based Grammar Formalisms, UMIST, Manchester, *Proceedings of the Fourth ACL European Chapter Conference*, 174-181.

Pollard, C. (1985) *Lectures on HPSG*, Unpublished lecture notes, CSLI, Stanford University

Steedman, M.J. (1986) Incremental Interpretation in Dialogue, *ACORD Deliverable T2.4*, Centre for Cognitive Science, University of Edinburgh.

Uszkoreit, H. (1987) Word Order and Constituent Structure in German, CSLI *Lectures notes* 8, Stanford.

Zeevat, H., Klein, E. & Calder, J. (1987) Unification Categorial Grammar, in Haddock N.J., E. Klein & G. Morill (eds.), *Categorial Grammar, Unification Grammar and Parsing*, Edinburgh Working Papers in Cognitive Science 1, Centre for Cognitive Science, Edinburgh, 195-222.

Wittenburg, K. (1986) *Natural Language Parsing with Combinatory Categorial Grammar in a Graph-Unification-Based Formalism*, PhD Thesis, Department of Linguistics, University of Texas.