XLFG-an LFG parsing scheme for French

∗Lionel Clément, ⋄Alexandra Kinyon
∗UFRL - University of Paris 7 ⋄IRCS - University of Pennsylvania
Paris, FRANCE Philadelphia, USA
clement@linguist.jussieu.fr kinyon@linc.cis.upenn.edu
http://talana.linguist.jussieu.fr/~lionel
http://www.cis.upenn.edu/~kinyon

Proceeding of the LFG01 Conference
University of Hong Kong, Hong Kong
Miriam Butt and Tracy Holloway King (Editors)
2001
CSLI Publications
http://csli-publications.standford.edu/

Abstract

We present XLFG, a new parser for LFG developed by Lionel Clément (Clément 2000), (Clément 2001b). This parser is freely available for pedagogical as well as for research purpose. In this paper, we explain how the tool has been used to develop and evaluate a medium-size LFG for French. We also show how we have exploited the similarities between the LFG and the Tree Adjoining Grammar Frameworks in order to integrate into XLFG a rule-based parse-ranker for French.

Introduction

Several parsers have already been implemented for Lexical Functional Grammars (LFG), for example (Kaplan and Maxwell 1994), (Andrews 1990). Some of these parsers have even been developed for French (Zweigenbaum 1991) (Vappillon 1997). However, these tools are sometimes not publicly available, or not flexible enough to implement additions to the "base" LFG formalism (e.g. the source code may not be available). This is why we have decided to implement XLFG, a new parser for LFG. Our goal was to provide a tool that is user-friendly enough to be used in a pedagogical environment, reasonably fast, portable to most Operating Systems, and flexible enough so that additions could be made to the base LFG framework in order to test new linguistic

hypothesis. In the first part of this paper, we provide an overview of XLFG, we explain how XLFG was used to develop an LFG medium-size grammar for French, and how it can be used for pedagogical purpose. In a second part, we explain how XLFG has been used to investigate the relationship between the Tree-Adjoining Grammars (TAG) framework, and the LFG framework. More specifically, we exploit the similarities between TAG "derivation trees" and LFG "functional structures" to incorporate into XLFG a parse-ranker for disambiguation purpose. Finally, we present a preliminary evaluation of the tool on the TSNLP test-suite for French. We assume the reader is familiar with the LFG framework and refer to (Bresnan 1982), (Kaplan 1989) for an introduction.

1 Overview of XLFG

The XLFG project started in 1996. XLFG, a wide-coverage parser for lexical functional grammars (LFG), is written in C, using an LR algorithm (Knuth 1965). We have tried to put an emphasis on the user-friendliness, thanks to a Graphical User Interface (GUI) developed in Tcl/Tk and to provide a fast and ergonomical way to encode and test LFG grammars (Clément 2000). XLFG is highly portable and is publicly available for academic and research purpose. Both the program and the source code can be downloaded on the www for most platforms[1].
The input of the parser consists in 3 files:

- A template file
- A lexicon file
- A rule file

For each sentence parsed, the output of the parser consists in:

- One or several couples of constituent and functional structure(s)

Moreover, XLFG can be used both online and offline. When used online, a constituent structure and an functional structure are displayed in the GUI[2]. One can see on figure 1 an example of output when parsing the sentence *Le chien aboie (The dog barks)*.

When used offline, one can obtain different types of output (e.g. a Latex file), which may be further postprocessed (for example to create figures, to input to a parse-ranker, to build a Treebank ...).

At this point, the following items have been implemented:

- Computation of constituent and functional structures
- Functional descriptions

[1] http//talana.linguist.jussieu.fr/~lionel/xlfg/

[2] When a sentence is ambiguous, several constituent and functional structures are then displayed.

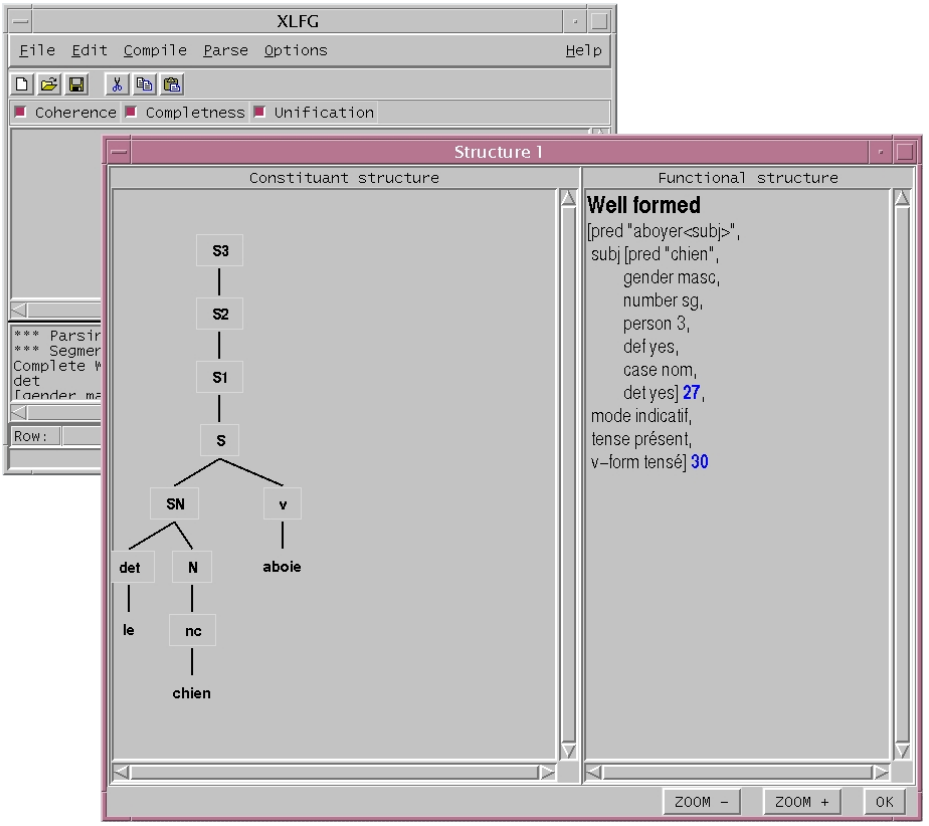


Figure 1: Sample output *Le chien aboie (the dog barks).*

  - unification (with “re-entrance”)
  - defining equations
  - constraining equations
  - set of functional structures
  - existential constraint
- Lexical rules
- Functional uncertainty

The goal of this work twofold:

- XLFG may be used as a pedagogical tool
- XLFG can easily be augmented for research purpose

1.1 XLFG: a pedagogical tool

Since XLFG can be used as a pedagogical tool to learn or teach the LFG framework, it was important to scrupulously follow the base LFG model as it was defined in the 80's in (Bresnan 1982), (Kaplan 1989).

XLFG can be used to teach the LFG framework to linguistics students "hands-on".

It has been used at the University of Paris 7 to teach LFG to MA students in computational linguistics. It has also been used at the University of Provence in an introductory course to syntactic formalisms. XLFG may be used during lab sessions (often on Unix machines), but also for homework assignments after being downloaded by students (who often use Microsoft Windows). Therefore, portability was important.

In addition, it is possible for lecturers to establish an online connexion using a client-server model. The server provides a large, robust wide-coverage lexicon. Thus, the lecturer and the students can focus on the syntactic phenomena under investigation without having to deal with lexical coverage issues.

To emphasize the pedagogical aspect, XLFG also parses ungrammatical sentences, as long as a "good" constituent structure is found (i.e. sentences with a good constituent structure, but possibly an ill-formed functional structure). It then expicitely shows in the output why the functional structure is ill-formed (i.e. uniqueness, coherence and/or completeness, as well as unification problems). Some examples are shown on figures 2,3 and 4).

In addition, it is possible to relax conditions on constraint equations as well as on existential equations.

1.2 Application to French

XLFG was used to develop and test a grammar for French, similarly to what has been achieved by (Abeillé et al. 2000a) for Tree Adjoining Grammars. The following syntactic phenomena are currently handled: clitics, passive, sentential complements, infinitival complements (control and raising verbs), some idiomatic expressions, some comparatives and coordinations (including gapping phenomena), compound tenses, negation (both for finite and infinitival clauses), some long distance dependencies.

Our grammar, which is hand-crafted, currently comprises 78 syntactic rules. Our lexicon currently comprises 450,000 inflected forms.

The following examples show how the lexicon is encoded (inflected forms) some syntactic rules for XLFG. The inflected forms of the lexicon are computed from morphological tables and lexical properties on lemmas (such as sub-categorization, lexical transformations, functional descriptions of raising verbs, etc.).

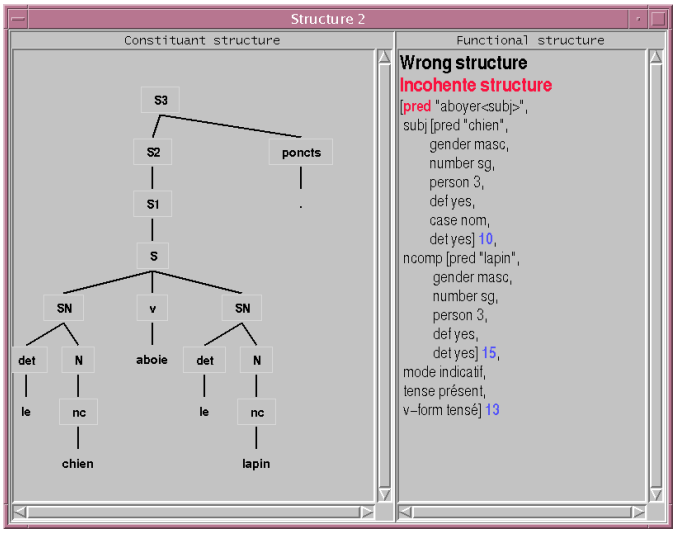


Figure 2: An incoherent structure **Le chien aboie le lapin (*The dog barks the rabbit).*

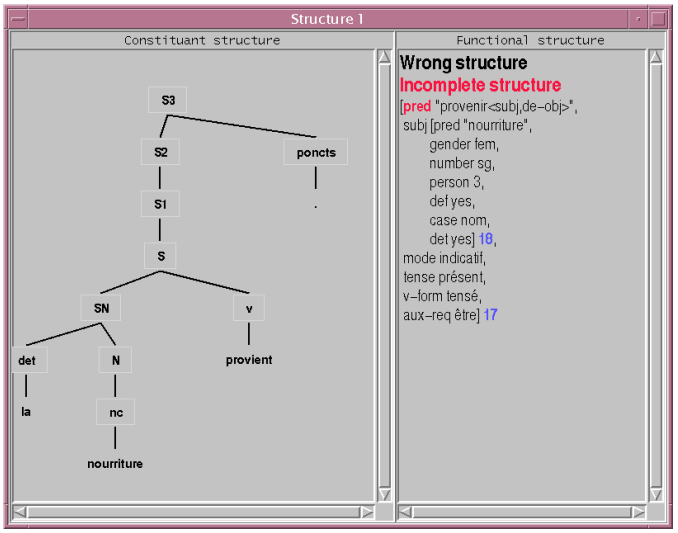


Figure 3: An incomplete structure **La nourriture provient (*The food comes from).*

Extract of the French lexicon (inflected forms)[3]

```
saisonné v [pred="saisonner<subj>",@Kms,@avoir,@active]  ;
```

[3]In features, @Kms, @Kfs, etc. are abbreviations. For agreement features, @Kms stands for "v-form=past-p, gender=masc, number=sg"; For auxiliary verbs, @avoir stands for "aux-req = avoir,

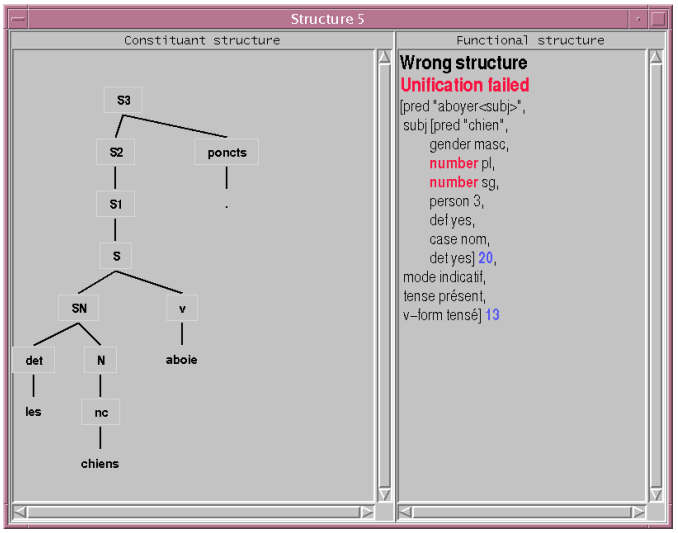


Figure 4: A structure where unification fails because of agreement **Les chiens aboie (*The dogs barks).*

```
saisonnée v [pred="saisonner<subj>",@Kfs,@avoir,@active]  ;
saisonnées v [pred="saisonner<subj>",@Kfp,@avoir,@active]  ;
saisonnés v [pred="saisonner<subj>",@Kmp,@avoir,@active]  ;
saisons nc [pred="saison",@fp] ;
saisîmes v [pred="saisir<subj,(obj)>",@Jp1]  ;
saisîmes v [pred="saisir<subj,(de-obj)>obj",@Jp1] @Pron ;
saisît v [pred="saisir<subj,obj>",@Ts3]  ;
saisît v [pred="saisir<subj,(de-obj)>obj",@Ts3] @Pron ;
saisîtes v [pred="saisir<subj,obj>",@Jp2]  ;
saisîtes v [pred="saisir<subj,(de-obj)>obj",@Jp2] @Pron ;
sait v [pred="savoir<subj,(obj/comp/vcomp)>",@Ps3] @CompInd @Ctrl-
Subj ;
saké nc [pred="saké",@ms] ;
sakés nc [pred="saké",@mp] ;
sala v [pred="saler<subj,obj>",@Js3]  ;
salace a [pred="salace<subj>",@s]  ;
salaces a [pred="salace<subj>",@p]  ;
```

For non feature abbreviations, @Pron stands for the functional description: "(↑ obj case =c rfl)", @CtrlSubj for "(↑ vcomp subj = ↑ subj)" for subject raising of control verbs.

Extract of the rules for a French grammar[4].

```
// sample: Wh extraction
// A quelle fille Jean pense-t-il~? (Which girl does J think about)
S3 -> SP S2 (poncts);
($$ top = $1)
($1 qu =c +)
($$ = $2)
($$ (($1 pcas)-obj) = $1)
($$ = $3)
(($$ mode) = indicative/conditional);

// sample: ‘‘surcomposed past’’
// Jean a eu déjeuné (J has had luch)
// Jean n’a pas eu moins déjeuné
_VERB -> (_CLITIC) aux (claff) (cln) (advneg) aux (SADV) v;
($$ tense = $2 tense)
($$ mode = $2 mode)
($$ subj number = $2 number)
($$ subj person = $2 person)
($2 form-aux = avoir)
($$ subj = $2 subj)
($$ = $3)
($$ subj = $4)
($5 neg = $$ neg)
($5 < ($$ adjunct))
($6 form-aux = avoir)
($6 v-form = past-p)
($$ = $8)
($$ aux-req = $6 form-aux)
($$ aux-req = $6 form-aux)
($$ aspect = perfect);
```

The previous two syntactic rules show how one can encode rewriting rules as well as functional descriptions.

Although the goal of XLFG is not to develop a robust and fast parser, but rather to develop LFG grammars and test linguistic hypothesis in a user-friendly and portable environment, the system is nonetheless fast. For instance, a sentence such as *L’europe espère que le marché national améliorera l’influence de la Commission. (Europe hopes that the national market will improve the influence of the commission)* was parsed in 2 seconds on a Super Sparc, yielding 73 analyses.

[4]In this rules, $$ is the notation for $\uparrow$, and $\$n$ for $\downarrow$ of the n^{th} term. $<$ is the notation for $\in$. We mark $(\uparrow x = y)$ for $(\uparrow x) = y$

As always when parsing, a trade-off must be found between trying to be wide-coverage and obtaining an acceptable syntactic ambiguity rate.

To increase the coverage, we are currently adding more rules to the grammar, as well as more entries to the lexicon. Moreover, to deal with certain specific syntactic phenomena such as ellipsis (e.g. *Jean mange des pommes et Marie des cerises (John eats apples and M. cherries)*, *Jean est plombier et fier de l'être (John is a plumber and proud of it)*), we have augmented the LFG formalism with the notion of *lexical capture*, and have implemented this feature into XLFG. Due to space limitations, we do not develop this notion here but refer to (Clément 1996) and future publications for more details.

To limit the ambiguity rate, we needed to integrate a disambiguation strategy into XLFG. We chose to depart from methods that have been traditionally used for LFG, such as Optimality Theory (Frank et al. 2000), or solutions based on probabily models (Riezler et al. 2000). Instead, we have observed that LFG functional structures and TAG so-called "derivation trees" are intuitively very similar (modulo re-entrant features)[5].

This point is developed in the next section.

2 Adapting TAG based disambiguation principles to LFG

2.1 Why not use a probabilistic disambiguation model ?

In order to disambiguate (LFG parses, as well as parses from other frameworks), one can resort to probabilistic models. However, we chose not to, both for theoretical as well as for practical reasons.

- From a practical point of view, probabilistic models are not language nor domain independent, they are costly because one needs large training data (i.e. treebank). Unfortunately, such training data is not available for most languages other than English. For French, there is no treebank to train on (although one is currently being built by (Abeillé et al. 2000b), (Clément 2001a)). Therefore, we could not resort to a probabilistic disambiguation model in order to disambiguate.

[5] It is also interesting to note that both functional structures and derivation trees are in turn quite close the notion of dependency tree, but we don't develop this point here.

- From a theoretical point of view, probabilistic models have no "explanatory power"(they may work, but this does not say anything about why it works). In human sentence processing, the importance of Lexical preferences is widely accepted (e.g. (Trueswell 1996)); For instance, a given verb may prefer to attach a PP as an argument. But there is little data available regardings these human preferences. And finding frequency effects in language comprehension does not automatically condemn all structural approaches: If one considers the sentence *John put the book that you were reading in the library*, although *put NP1 in NP2* is obviously a common subcategorization frame for *put*, the sentence nonetheless seems incomplete, although it is syntactically well-formed.

Therefore, since we were not convinced that probabilistic disambiguation methods were sufficient from a theoretical point of view, and since we did not have any data to train a probabilistic disambiguation model on, we had to turn to rule-based methods[6].

2.2 Why not express preference rules on C-structures ?

Traditionnally, rule-based disambiguation principles have been formulated on the shape of constituent trees. For instance (Kimball 1973) formulated the right association principle, which allows to retrieve the correct attachment in a sentence such as *Tom said that Joe left yesterday* (were yesterday attaches to left rather than to Tom). Similarly, (Frazier and Fodor 1978) formulated a minimal attachement principle which states that in case where several constituent structures are associated to a given sentence, the constituent tree with the fewest number of nodes should be favored. For a sentence such as *Tom bought the flowers for Sue*, this principle allows one to retrieve the preferred attachment, where *for Sue* attaches to *bought* rather than to *flowers*. However, as argued in (Kinyon 1999) this type of principles, formulated on constituent trees were deemed unsatisfactory for several reasons:

[6]We do not of course reject the idea of lexical preference. In fact, in addition to structural preferences (see infra), we have added some weights to lexical items to model some lexical preferences.

- Principles such as Minimal Attachment make some assumption about how a constituent tree should be built[7].
- The interaction between these principles is unclear.
- It is difficult to integrate semantic and/or pragmatic information into such principles.
- These principles do not establish a distinction between argument and modifier attachment.
- Counter-examples are easy to find: e.g. the validity of right association is debated for Spanish (Cuetos and Mitchell 1988) and Dutch (Brysbaert and Mitchell 1996).

Therefore, we did not want to express disambiguation principles in terms of constituent structure.

On the other hand, some widely accepted disambiguation principles may not be expressed on constituent tree, but are easy to express on "dependency-like" structures:

- Prefer the idiomatic interpretation of a sentence over its literal interpretation.
- Prefer to attach an element as an argument rather than as a modifier.
- Prefer to attach an argument to its closest potential governor.

Principle 1 allows the idiomatic interpretation of a sentence such as *John breaks the ice* to be preferred over its literal interpretation.
Principle 2 allows one to prefer to attach *to be honest* as argument of *prefer*, rather than as a sentence modifier in a sentence such as *John prefers his daughter to be honest*[8].
Principle 3 allows to prefer to attach *of the demonstration* as argument of *organizer*, rather than as an argument of *suspect* in a sentence such as *John suspects the organizer of the demonstration.*

These disambiguation principles were succesfully used to implement a parse-ranker for French (Kinyon 2000) for TAGs, expressed on the number of nodes in TAG "derivation trees", and thus we wanted to see if this was transposable to LFG, and expressable on "f-structures".

[7]Most of these principles were formulated within an X-bar type of linguistic framework. And principles such as Minimal attachment rely for a large part on some X-bar characteristic concerning the number of nodes a given constituent tree will have.

[8]The modifier reading being identical to *To be honest, John prefers his daughter.*

2.3 Similarities between TAG and LFG

The similarity between LFG and TAG was to the best of our knowledge first investigated in (Kameyama 1986). Surprisingly, very little has been written on this topic since. We will not provide an introduction to TAGs here (and refer to (Joshi 1994) and (Abeillé and Rambow 2000) for an overview). For our purpose, one just needs to know that in TAG, the elementary structures, called "elementary" trees, combine via 2 operations: substitution (for "initial" elementary trees) and adjunction (for "auxiliary" elementary trees). Figure 5 shows how one can parse the sentence *Jean ne voit pas Marie (John does not see Mary)* using a Lexicalized TAG consisting in four elementary trees[9]. In this example, the initial trees "α-Jean" and "α-Marie" were substituted respectively as subject and object in the initial tree "α-voit", and the auxiliary tree "β-ne-pas" was adjoined as a modifier into the initial tree "α-voit". The result one obtains from combining the trees consists in:

- A **derived** tree, which is a constituent structure.
- A **derivation** tree, close to a dependency structure, which is built by keeping the historic of operations (i.e. substitution and adjunction) performed to obtain the derived tree.

The important thing to note is that both LFG and TAG yield a constituent structure (categorial structure for LFG, derived tree for TAG) and a *dependency-like structure* (functional structure for LFG, derivation tree for TAG)[10]. So the derivation tree from figure 6(a) can be seen as equivalent to the F-structure on figure 6(b). This similarity between TAG derivation trees and LFG functional structure does not of course extend to the case of reentrant features, which turn f-structures into a graph rather than a tree, but reentrant features are in practice limited to bounded phenomena such as control[11].

[9] Agrement is dealth with by associating features to nodes in elementary trees (not shown on figure for readability). α denote inital trees, β auxiliary trees, plain lines represent an adjunction (i.e. the insertion of an tree into another tree), dotted line a substitution.

[10] Other similarities exist, for instance (Rambow 1996) noted the similarity between *d-links* used in some formalisms related to TAGs, and the LFG notion of functional uncertainty but this is beyond the scope of our work.

[11] So turning LFG functional structures into TAG derivation tree may imply a "loss" of information, but this information can be encoded using co-indices on derivation tree leaves.

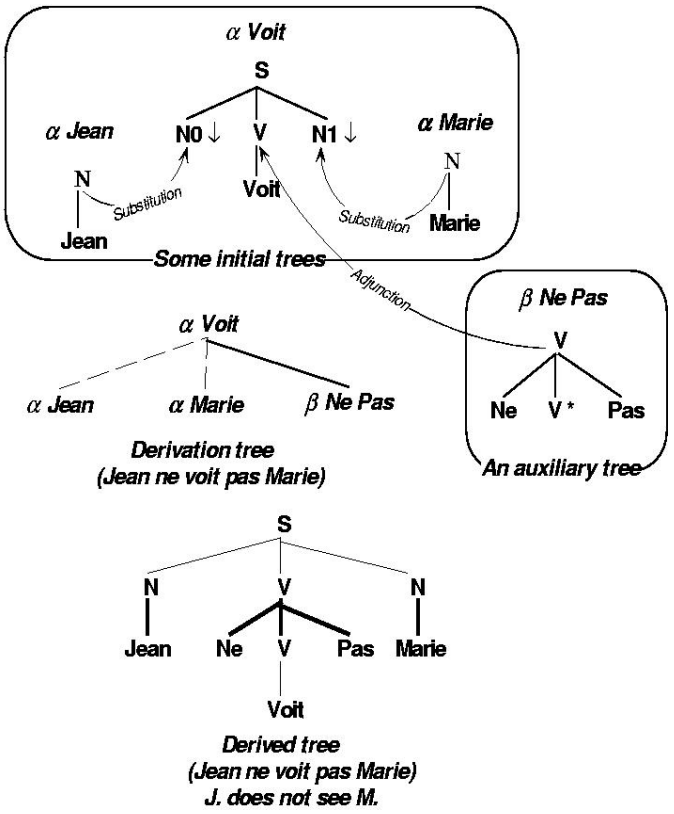
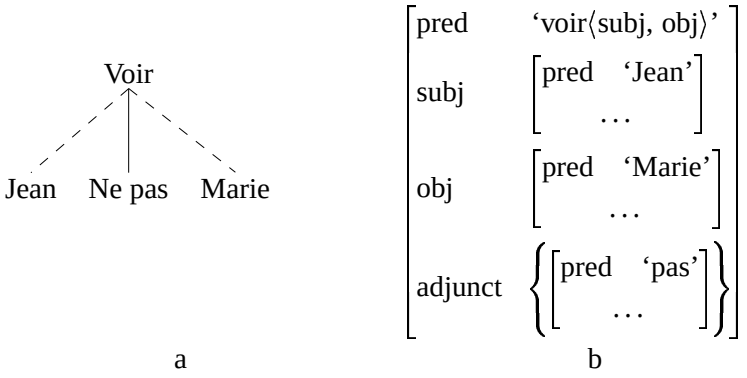


Figure 5: Jean ne voit pas Marie (*John doesn't see Mary.*).

2.4 Expressing disambiguation principles in terms of f-structure

Principle 1, which says that the idiomatic interpretation of a sentences should be favored over its literal interpretation, was modeled in TAGs by preferring the derivation tree (i.e. the dependency-like structure) with the fewest number of nodes. To implement Principle 1 into XLFG, we only had to make sure that the functional structure which resorts to the most constrained lexical item would be favored (here *break1* which constrains its object to be *ice*, rather than *break2* which is unconstrained for its object) (cf fig 7).

Principle 2, which says that the attachment of an element as an argument rather than as a modifier should be preferred was expressed on TAG derivation trees by preferring the derivation trees with the fewest



a b

Figure 6: Derivation tree and equivalent F-structure for *Jean ne voit pas Marie (John doesn't see Mary).*

number of β-tree nodes[12]. This principle was also easy to encode on LFG functional structure, simply by counting the number of local arguments functions of a predicator (i.e. *subj, obj, à-obj, de-obj, ...*), and favoring the functional structure with the highest count (cf fig. 8).

Finally, Principle 3 which states that one should prefer to attach arguments as close as possible to their governor, was implemented in TAGs by computing on each derivation tree the sum of the distances between each node in the tree and their son. This allows one to prefer *to Paul* as an argument of *give* rather than as an argument of *says* in example 1a, since the distance between *to Paul* and *give* is shorter than the distance between *to Paul* and *say*. Whereas,conversely, in sentence 1b, *to whom* will be argument of say, rather than argument of *give*, for the same reason.

(1) a. John says that Mary gives flowers to Paul.

b. To whom does John say that Mary gives flowers.

This principle was adapted to LFG by computing the sum on the functional structures, seen as dependency-like structures: For each node in a given structure, one computes the sum of the distances between the predicator and all of its daughters, according the linear order of

[12]TAG β trees (i.e. auxiliary trees), are mainly used to encode modifiers, whereas TAG α trees (i.e. initial trees), are mainly used to encode arguments.

the sentence[13]. The functional structure with the lowest sum will be favored (cf fig. 9).

As was done for TAGs, the 3 principles are applied in a sequential order. Thus there are no conflicts. The parser computes all possible functional structures for a given sentence. It then applies the 3 principles and outputs the n-best functional structures according to these principles, along with their corresponding constituent structure.

In addition, we do not commit exclusively to a rule-based disambiguation system: since a weight is associated to each lexical entry in the lexicon, this allows us, in addition of favoring the idiomatic interpretation of a sentence, to encode lexical or semantic preferences as well, thus allowing to mix rule-based general structural preference principles with more fine-grained statistical lexical preferences in order to obtain the best possible disambiguation. For instance, this may be used to prefer "function words" over"lexical word" (e.g. to prefer *être (to be), avoir(to have)* as auxiliaries rather than full lexical verbs.).

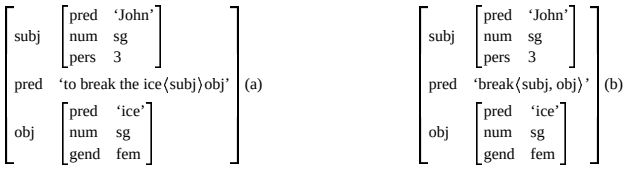


Figure 7: Preferred (a) and dispreferred (b) functional structures for the idiomatic and literal interpretations of *John breaks the ice*.

[13]We take into account only the nodes where *Pred* is associated to a lexical item (and not e.g. to a morpheme or discourse unit).

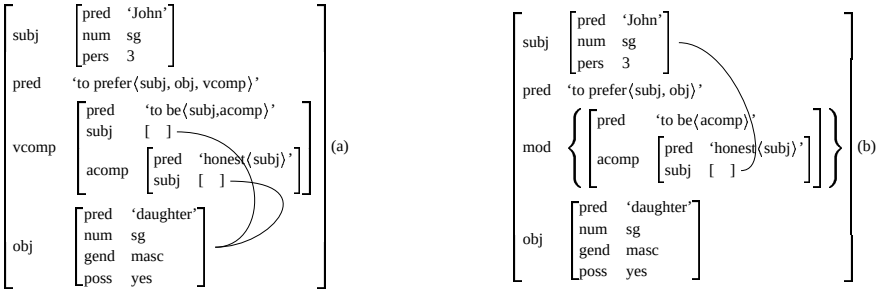


Figure 8: Preferred (a) and dispreferred (b) functional structure for *John prefers his daughter to be honest* with *to be honest* as an argument (a) or as a sentence modifier(b).

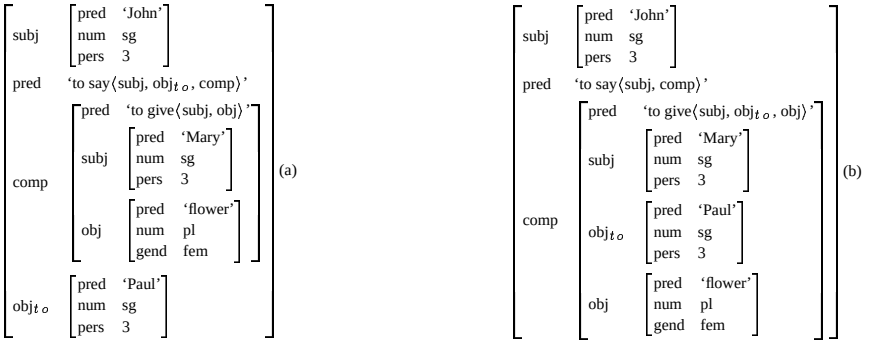


Figure 9: Preferred (a) and dispreferred (b) functional structure for *John says that Mary gives flowers to Paul.*

3 Preliminary Evaluation of XLFG

In order to test both the coverage of XLFG as well as the validity of the rule-based disambiguation principles we have implemented, we used 1,200 grammatical sentences for French from the test suite TSNLP (Estival and Lehman 1997). We parsed these 1,200 sentences with XLFG and our grammar for French (approximately 78 syntactic rules, 450,000 lexical items) without any disambiguation. We developed a tool to browse through all the analysis for each sentence (i.e. the different pairs of possible constituent structure / functional structure assigned to each sentence by the parser). We used the output of the parser and the annotation tool to manually build a gold-standard. We then reparsed the 1,200 sentences, this time with the disambiguation module turned on and compared this output to the gold standard. Without disambiguation, XLFG produced one or more correct parse(s) for 81.5% of the sentences (i.e. 972 of the 1,200 sentences), with an average of 3.36 functional structures per sentence. 80.3% of these sentences were ambiguous (i.e. 781 ambiguous sentences out of 972 parsed sentences). With disambiguation, XLFG still produced one or more correct parse(s) for 81.5% of the sentences, but this time with an average of 1.29 functional structures per sentence. Only 38% of the parsed sentences were now ambiguous (i.e. 373 out of 972 parsed sentences): 408 sentences (i.e. 52.2%) had been totally disambiguated, and had only one functional structure associated to them. And another 40 sentences (i.e. 5%) had been partially disambiguated, meaning that they were associated with fewer functional structures then when they were initially parsed without the disambiguator. More importantly, no correct functional structure was discarded.

The 20% of sentences that did not receive any parse in the first place were due either to unknown lexical items, or to the fact that our grammar did not deal with some syntactic phenomena. TSNLP is a test-suite, so the sentences are built artificially and resort on purpose to a limited lexicon. It would be interesting in the future to enrich both the grammar and lexicon of XLFG to make it more robust and perform a new evaluation on *real text*.

It is interesting to note though that when tested on the same TSNLP sentences XLFG obtained basically the same results as FTAG, a wide coverage grammar for French (Abeillé et al. 2000a) which is semi-automatically generated, and consists in 5,000 elementary trees and approximately 600,000 lexical entries: approx 80% of the sentences

get at least one correct parse. Again, the goal of XLFG is not to be a robust wide-coverage parser, but, however, these results are very encouraging, especially since our LFG grammar contained far fewer rules than its TAG counterpart.

Conclusion

We have presented XLFG, an LFG parser freely available for the research community. We have shown how this tool can be used as a pedagogical tool, as well as to test and implement new linguistic hypothesis. We have so far manually developed a medium-size grammar for French of approximately 78 rules and 450,000 lexical entries. We have also implemented a rule-based disambiguation module which exploits the similarities between LFG functional structures and TAG derivation trees and have presented a preliminary evaluation of the parser and the disambiguator. In the future, we hope to augment the coverage of our grammar for French, and to develop grammars for other languages. In addition, we plan to investigate more systematically the link between the LFG framework and the TAG framework, based on the work initiated by (Kameyama 1986).

References

Abeillé, A., M. Candito, and A. Kinyon. 2000a. The current status of FTAG. In *TAG+5*, Paris.

Abeillé, A., L. Clément, and A. Kinyon. 2000b. Building a Treebank for French. In *Proceedings LREC'2000*, Athens.

Abeillé, A., and O. Rambow (Eds.). 2000. *Tree Adjoining Grammars: Formalism, linguistic analysis and processing.* CSLI publications.

Andrews, A. 1990. Functional closure in LFG. Technical report, The Austrilian National University.

Bresnan, J. 1982. *The mental representation of Grammaticals Relations.* MIT Press.

Brysbaert, M., and D. Mitchell. 1996. Modifier attachment in sentence parsing: Evidence from Dutch. *Quaterly Journal of experimental psychology* 49a:664–695.

Clément, L. 1996. Un analyseur syntaxique pour le français en LFG - quelques pistes pour l'implémentation de la coordination. Master's thesis, University Paris 7.

Clément, L. 2000. XLFG - Une plate-forme de développement de Grammaires Lexicales Fonctionnelles. In *TALN 2000*, Lausanne.

Clément, L. 2001a. *Construction et exploitation d'un corpus syntaxiquement annoté pour le français*. PhD thesis, University of Paris 7.

Clément, L. 2001b. XLFG a parser to learn LFG framework. In *NAACL Demos*, Pittsburgh.

Cuetos, M., and D. Mitchell. 1988. Cross linguistic differences in parsing: restrictions on the use of the late closure strategy in Spanish. In *Cognition 30*, 73–105.

Estival, D., and Lehman. 1997. TSNLP: des jeux de phrases test pour le TALN. *TAL* 8:1.

Frank, A., T. Holloway King, J. Kuhn, and J. Maxwell. 2000. Optimality Theory Style Constraint Ranking in Large-scale LFG Grammars. In *LFG'98*, Australia.

Frazier, L., and J. Fodor. 1978. The sausage machine: a two stage parsing model. In *Cognition 6*.

Joshi, A. 1994. Introduction to tree-adjoining grammars–Preface to the Special Issue on Tree Adjoining Grammars. *Computational Intelligence*.

Kameyama, M. 1986. Characterising Lexical Functional Grammar (LFG) in terms of Tree Adjoining Grammar (TAG). Technical report, Dept. of Computer and Information Science. University of Pennsylvania, Philadelphia.

Kaplan, R. 1989. The formal Architecture of Lexical Functionnal Grammar. In *Journal of Informations Science and Engineering.*

Kaplan, R., and J. Maxwell. 1994. *Grammar Writer's Workbench*. Xerox Corporation. Version 2.0.

Kimball, J. 1973. Seven principles of surface structure parsing in Natural Language. In *Cognition 2*.

Kinyon, A. 1999. Some remarks on the psycholinguistic relevance of LTAGs. In *CLIN'99*, Utrecht.

Kinyon, A. 2000. Are structural principles useful for automatic disambiguation ? In *COGSCI'00*, Philadelphia.

Knuth, D. E. 1965. On the translation of langages from left to right. In *Information and Control*.

Rambow, O. 1996. Word order, clause union, and the formal machinery of syntax. In M. Butt and T. H. King (Eds.), *On-line Proceedings of the First LFG Conference, Rank Xerox, Grenoble, August 26–28, 1996.*

Riezler, S., D. Prescher, J. Kuhn, and M. Johnson. 2000. Lexicalized Stochastic Modeling of ConstraintBased Grammars using Log-Linear Measures and EM Training. In *ACL'00*, Hong Kong.

Trueswell, J. 1996. The role of lexical frequency in syntactic ambiguity resolution. *Journal of Memory and Language* 35:566–585.

Vappillon, J. 1997. Un atelier de Génie Linguistique pour l'analyse syntaxique fondée sur le formalisme LFG. In *JST Francil*, Avignon.

Zweigenbaum, P. 1991. Un analyseur syntaxique pour grammaires lexicales fonctionnelles. *T.A. Informations.*