3

Automating the Generation of a Wide-coverage LFG for French using a MetaGrammar

Lionel Clément and Alexandra Kinyon

[1]In this paper, we explain how the notion of MetaGrammar, which has successfully been used for generating wide-coverage tree adjoining grammars (TAGs) for various languages such as French (Abeillé et al. (1999)) and German (Gerdes (2002)), may be used to generate a wide-coverage Lexical Functional Grammar (LFG) for French. We first introduce the notion of MetaGrammar and present the tools we use to automatically generate, from a compact MetaGrammar hierarchy, a LFG for French: the grammar writer specifies in compact manner syntactic properties that are usually encoded separately in the LFG machinery in the lexicon, in lexical rules and in rewriting rules. From this hierarchy an LFG is automatically generated. We also propose an augmentation of the existing MetaGrammar tool.

3.1 Introduction

Expensive dedicated tools and resources (e.g., grammars, parsers, lexicons) have been developed for a variety of grammar formalisms - which all have the same goal - model the syntactic properties of natural language, but resort to a different machinery to achieve that goal. However, there are some core syntactic phenomena on which a cross-framework

[1]We thank Eric de la Clergerie, Julia Hockenmaier, Aravind Joshi, Owen Rambow, Pierre Boullier and three anonymous reviewers for helpful comments on earlier versions of this work.

Proceedings of FGVienna: The 8th Conference on Formal Grammar.
Gerald Penn (ed.).

(and to some extent a cross-language) consensus exists, such as the notions of subcategorization, valency alternations, and the notion of syntactic function. From a theoretical perspective, a MetaGrammatical level of representation allows one to encode such consensual pieces of syntactic knowledge and to compare different frameworks. From a practical perspective, encoding syntactic phenomena at a MetaGrammatical level, from which grammars in different frameworks can be generated, has several advantages such as portability among grammatical frameworks, increased coherence and consistency in the generated grammars and less need for human intervention in the grammar development process. Here, we focus on the use of MetaGrammars to automate the generation of a wide-coverage LFG for French.

In the first part of this paper, we explain the notion of *MetaGrammar* and present the MetaGrammar tool we have used. In the second part of this paper, we justify the use of a MetaGrammar for generating LFGs and explore several options for doing so. In the third part of this paper we discuss the coverage of the grammar we generate as well as the advantages of a MetaGrammar approach. We assume the reader is familiar with the LFG framework and refer to (Bresnan and Kaplan (1982)) for an introduction.

3.2 What is a MetaGrammar?

The notion of MetaGrammar (MG) was originally presented in (Candito (1996)) to automatically generate a Tree Adjoining Grammar (TAG)[2] for French and Italian.[3] The idea is to add a higher-level of linguistic description to encode, partial descriptions of trees. The general organization for syntactic information is defined in three multiple inheritance networks. Properties are shared among different trees in the grammar thanks to the multiple inheritance mechanism.[4]

The first hierarchy encodes canonical lexical subcategorizations (i.e., transitive, ditransitive), the second encodes valency alternations and redistribution of syntactic functions (e.g., it allows to add an argument for causatives, to erase one for passive with no agents, etc.), and the third encodes the surface realization of arguments (e.g., declares if a direct-object is pronominalized, wh-extracted, etc.).

[2]We do not present the TAG formalism here. Suffice it to say that it is a tree rewriting system, hence a TAG rule is a tree which may be of depth greater than one, unlike LFG rewriting rules.

[3]A Similar MetaGrammar type of organization for TAGs was independently presented in (Xia (2001)) for English.

[4]The main difference with HPSG hierarchical organization (e.g. Flickinger (1987)) is that HPSG type hierarchies are an inherent part of the HPSG framework, whereas the MetaGrammar specifications are framework-independent.

A well-formed TAG elementary tree is generated by inheriting from exactly one initial subcategorization, one redistribution (which can be complex e.g., Passive + Causative), and for each argument, from one realization. From a linguistic perspective, predicate-Argument Co-occurrence Principles are enforced by the fact that the tool automatically crosses the constraints between the three dimensional networks.[5]

For instance, the elementary tree for "Par qui sera accompagnée Marie" (*By whom will Mary be accompanied*) is generated by inheriting from **transitive** in dimension 1, from **passive** in dimension 2 and **subject-nominal-inverted** for its subject and **Wh-questioned-object** for its object in dimension 3.

3.2.1 HyperTags

A grammar rule is associated to each grammar rule which is automatically generated from the MetaGrammar hierarchy. The main idea behind HyperTags, which is a notion first introduced in (Kinyon (2000)), was to keep track, when trees (i.e., TAG grammar rules) are generated from a MetaGrammar hierarchy, of which terminal classes were used for generating the tree. This allows one to obtain a framework-independent feature structure containing the salient syntactic characteristics of each grammar rule.[6] For instance, the verb *give* in *A book was given to May* could be assigned the HyperTag:

$$\begin{bmatrix} \text{Subcat} & \text{Ditransitive} \\ \text{Valency alternations} & \text{Passive with no agent} \\ \text{Argument Realization} & \begin{bmatrix} \text{Subject:} & \text{Canonical NP} \\ \text{Object:} & \text{Not realized} \\ \text{By-Phrase:} & \text{Canonical PP} \end{bmatrix} \end{bmatrix}$$

Although we retain the linguistic insights presented in (Candito (1996)), that is the three dimensions to model syntax (subcategorization, valency alternation, realization of syntactic arguments) we use a different MetaGrammar tool which, although it was also originally designed to generate TAGs, is less framework-dependent and supports the notion of HyperTag.

[5] So, in Candito's implementation, classes cross by default.

[6] The notion of HyperTag was inspired by that of supertags presented in (Srinivas (1997)), which consists in assigning a TAG elementary tree to lexical items, hence enriching traditional POS tagging. However, HyperTags are framework-independent.

3.2.2 The LORIA MetaGrammar compiler

To generate LFGs, we use a modified version of the LORIA MG compiler presented in (Gaiffe et al. (2002)).[7] In the LORIA tool, each class in the MG hierarchy encodes:

- Its Super-class(es)
- A HyperTag which captures the salient linguistic characteristics of that class
- What the class needs and provides
- A set of quasi-nodes (i.e., variables)
- Topological relations between these nodes (*parent, dominates, precedes, equals*).
- A function for each quasi-nodes to decorate the tree (e.g., traditional agreement equations and/or LFG functional equations).

Each class in the hierarchy encodes properties of its own (such as parts of decorated trees, what the class needs and provides, etc.), and also inherits properties from its super-classes. The MG tool automatically crosses terminal classes in the hierarchy[8], looking to create *balanced* classes, that is classes which do not need nor provide any resource. Then for each balanced class, their HyperTags and the structural constraints between their quasi-nodes are unified. If the unification succeeds, one or more <HyperTag, tree> pairs are generated. Figure 1 illustrate how a simple TAG rule is generated with this tool.

Contrary to Candito's MetaGrammar, the LORIA MetaGrammar compiler explicitly describes how the classes cross. There are no general constraints such as **completeness** and **coherence** to construct well-formed objects. So the LORIA tool is more flexible in the sense that there is no explicit notion of *dimension*: it is possible to construct a MetaGrammar in a way which differs from the three-dimensional framework based on functional complementation. Instead, for making classes cross, one must resort to the resource allocation model (i.e. one must explicitly state what each class needs and provides) in order to obtain an adequate crossing between terminal classes in the hierarchy (e.g. to ensure an appropriate complementation, a correct ordering between constituents and other syntactic properties).[9]

[7] The LORIA compiler is freely available on http://www.loria.fr/equipes/led/outils/mgc/mgc.html

[8] Terminal classes are classes which do not have any subclass

[9] Another way to rephrase this is to say that, in Candito's MetaGrammar tool, classes cross by default, whereas with the LORIA tool, the default is for a class not to cross unless one of its resource needs to be satisfied.

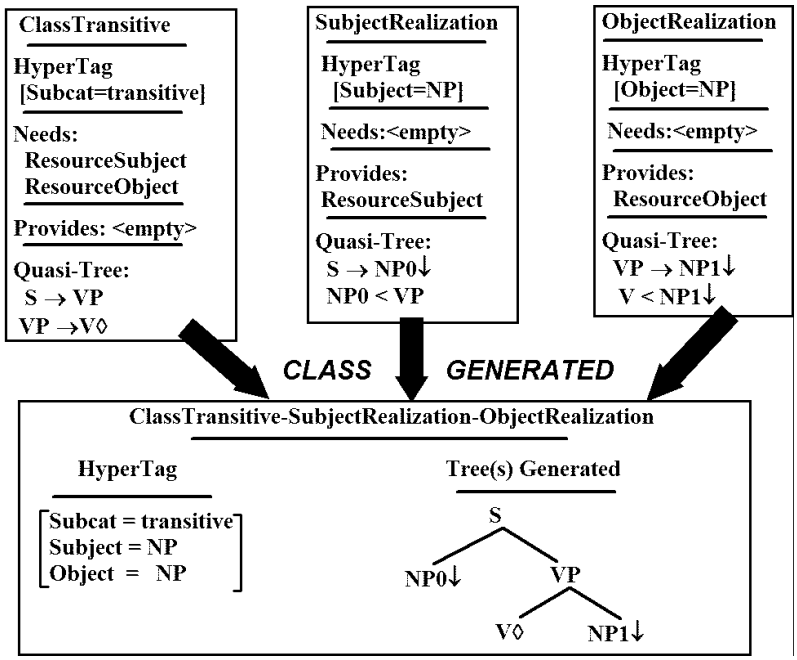

FIGURE 1 Generation of a simple TAG rule → stands for father, < for precedes,◇ for a lexical anchor and ↓ for substitution node in the TAG terminology

3.3 Generating LFGs with a MetaGrammar

3.3.1 Why use a MetaGrammar for LFGs ?

Because TAGs are a tree rewriting system, there are intrinsic redundancies in the rules of a TAG. E.g., all the rules for verbs with a canonical NP subject and a canonical realization of the verb will have a redundant piece of structure
(S NP0↓ (VP (V◇))). This piece of structure will be present not only for each new subcategorization frame (intransitive, transitive, ditransitive,...), but also for all related non-canonical syntactic constructions such as in each grammar rule encoding a Wh-extracted object. This redundancy justifies the use of a MetaGrammar for TAGs. Since LFG rules rely on a context free backbone, it is generally admitted that there is less redundancy in LFG rules than in TAG rules. However, there are still redundancies, at the level of rewriting rules, at the level of functional equations, and at the level of lexical entries. For instance, a rule such as the one shown on figure 2 could be written for the VP expansion

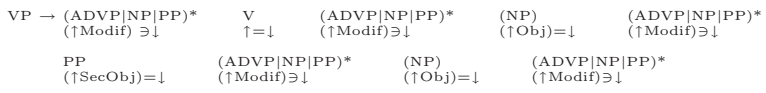

FIGURE 2 Redundant VP expansion for French ditransitives

of a ditransitive verb for French, with the free insertion of a modifier anywhere in the right hand-side of the rule. We see that the modifier is inserted 5 times, with its corresponding functional equation, and the NP for the direct object is repeated twice.

So, in addition of inherent advantages of a MG approach (independently of the grammar framework one chooses to generate), LFG has redundancies which justify the use of a MG for this particular framework. LFG redundancies and a comparison of the MG to the use of *LFG specific* operators such as *shuffle* are discussed in more detail in (Clément and Kinyon (2003a)) and (Clément and Kinyon (2003b)): the main distinction between our MG approach and the use of operators is that the proliferation of operators (e.g. shuffle, ID/LP rules, macros etc.) makes it possible to express the same rule in many different ways, which may be helpful for grammar writing purpose, but not so much for maintenance purpose.[10] By contrast, the MetaGrammar outputs very *homogeneous* grammars: even though different choices can be made at the MetaGrammatical level (e.g. should we resort to VP nodes?, should these VPs be flat or right branching? etc.), once a choice is made, it applies homogeneously throughout the whole generated grammar.

3.3.2 Generating LFGs

We have seen in section 3.2.2 that the MetaGrammar tool we use outputs <HyperTag, tree> pairs. When generating TAGs, *tree* is a grammar rule (i.e., a TAG elementary tree). We extend the approach to LFG by generating <HyperTag, tree> pairs, where *tree* is a constituent structure decorated with functional equations and corresponds to one or more LFG rewriting rules, as shown in figure 5. Note that this decorated tree is obtained in exactly the same manner as its TAG counterpart from fig 1. In addition to generating rewriting rules, our MetaGrammar hierarchy also yields lexical templates.[11] This is illustrated in figure 3 and figure 4: a decorated tree is generated automatically from our MG hierarchy (3). This decorated tree yields one LFG rewriting rule and one lexical template (4) for a French verb such as

[10] An analogy can be made to computer programs written in Perl.

[11] A lexical template encodes a linguistic generalization (such as a subcategorization frame) shared by a set of lexical entries

"éloigner" (*take away from*), which subcategorizes for an NP object and for a PP object introduced by "de". (Ex: "Peter éloigne son enfant de la fenêtre"-*Peter takes his child away from the window*).

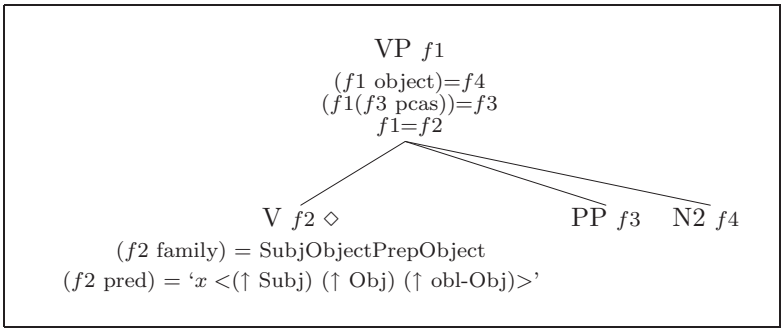

FIGURE 3 Decorated constituent tree for a simple clause

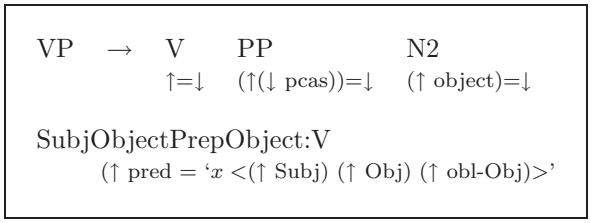

FIGURE 4 LFG Rule and a lexical template

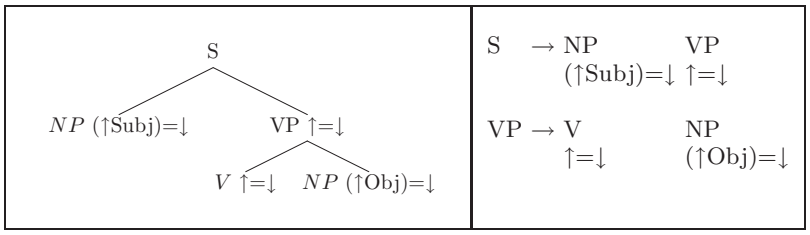

FIGURE 5 Generating a decorated tree which yields two simple LFG rules

As discussed in (Joshi and Vijay-Shanker (1989)), an essential difference between TAGs and LFGs resides in the domain of locality of the grammar rules. The MG does not impose a given domain of locality.

Hence, in order to generate LFG rules with a MG, we have two options. The first option consists in generating *standard* LFG rules, that is trees of depth 1 decorated with functional equations (as shown in fig 3): the root of the tree of depth one corresponds to the left hand-side of an LFG rewriting rule,[12] the children of this root correspond to the right hand-side of the rule. The second option, which is the one we have opted for (illustrated in fig 5 and 6), consists in generating constituent trees which may be of depth superior to one, decorated with feature equations. Namely, we adopt a domain of locality that is at the *clause* level, which has the following advantages:

- It allows for a more natural parallelism between the LFG grammars generated and other approaches.[13]
- It allows for a more natural encoding of syntax at the MetaGrammar level.
- It allows us to generate LFGs without resorting to LFG Lexical Rules.

The trees automatically generated from the MetaGrammar hierarchy, are decorated with LFG functional annotations, as was first proposed by (Kameyama (1986)), and in a way which is similar to that presented in (Frank (2000)). These decorated trees are then decomposed into standard LFG rewriting rules (similarly to the work presented in Hepple and van Genabith (2000)), and into lexical templates. For modifiers, we have implemented right branching adjuncts (e.g. non-flat VPs for free modifier insertion, i.e. unlike the VP rule shown in 2.).[14]

In a first step, for each decorated tree we generate, each subtree of depth one produces one LFG rewriting rule. In a second step, equivalent rules are merged. Equivalent rules are defined by induction: Two rules are equivalent if and only if they differ only on equivalent non-terminal symbols with identical functional equations. The grammar we obtain is then interfaced with a parser.[15]

The *resource model* of the MetaGrammar, based on *needs* and *provides*, allows for a natural encoding and enforcement of LFG coherence, completeness and uniqueness principles: A transitive verb needs exactly

[12] Modulo the slight difference that the root may be decorated, contrary to standard left hand-sides of LFG rules

[13] Although we do not develop this point here, we have used our MetaGrammar hierarchy to generate grammars in frameworks other than LFG, esp. TAGs and Range Concatenation Grammars (Boullier (1998))

[14] As an alternative, (Kinyon (2003)) resorts to flat VPs with a MG hierarchy for free modifier insertion between each constituent in the VP, except the clitics.

[15] We use the freely available XLFG parser described in (Clément and Kinyon (2001)) and have experimented with the LFG parser described in (Kaplan and Maxwell (1996)).

one resource *Subject* and one resource *Object*. Violations result in invalid classes which do not yield any rules. So from that perspective, it makes little sense (apart from practical reasons, such as interfacing the grammar with an existing parser) to force the rules generated to be trees of depth one. Moreover, the same type of resource-sensitivity is used within the LFG community. e.g. in (Dalrymple et al. (1995)) to compute semantic forms using linear logic.

Concerning lexical rules, traditional LFGs encode phrase structure realizations of syntactic functions such as the wh-extraction or pronominalization of an object in phrase structure rules. In the MetaGrammar, these are encoded in the *Argument Realization* dimension (dimension 3 in Candito's terminology). For valency alternations, i.e., when initial syntactic functions are modified, LFG resorts to the additional machinery of lexical rules.[16] However, these valency alternations are encoded directly in the MetaGrammar in the *valency alternation* dimension (dimension 2 in Candito's terminology). Hence, when a rule is generated for a canonical transitive verb, rules are generated not only for all possible argument realization for the subject and direct object (wh-questioned, relativized, cliticized for French etc.), but also for all the valency alternations allowed for the subcategory frame concerned (here, passive with/without agent, causative etc) in a monotonic manner. Therefore, there is no need to generate usual LFG lexical rules, and the absence of lexical rules has not effect on interfacing the grammars we generate with existing LFG parsers.

3.3.3 Yet another MetaGrammar compiler

In order to implement some additions to the MetaGrammar formalism, we have developed our own tool. In our compiler, it is possible to describe topological relations between variables (encoded as logical constraints), even if these variables are not instantiated. This feature helps to describe in an abstract manner some syntactic phenomena such as clitic ordering in French, without worrying about whether a given phenomenon will appear in a grammar rule. More precisely, for clitics, one encodes in a single class ordering constraints for the realization of nominative, accusative, dative, genitive clitics[17], and from these abstract specifications, one obtains several rules with one, two, three or more clitics realized. By contrast, the LORIA MG tool only allows one to formulate constraints on elements which are going to be realized in a rule, so if one writes a class for ordering 3 clitics (e.g. nominative,

[16] Although some versions of LFG dispense with lexical rules altogether.

[17] The ordering rules are quite complex and depend on the person and case of the clitic, as well as on the class of verb to which the clitic will attach

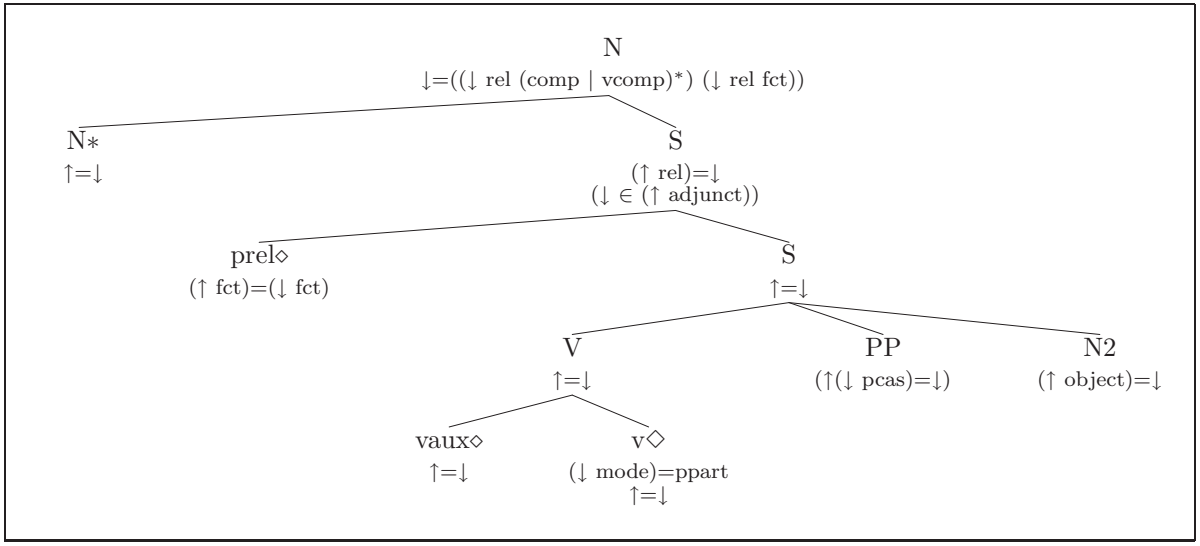

FIGURE 6 Generated tree (partial representation) for a simple relative clause with a ditransitive verb, decorated with LFG functional notation.

accusative, dative), one will generate rules realizing 3 cliticized arguments such as the subject, the object, and the second-object (ex: "Il le lui donne" - lit: *he:nom it:acc him:dat gives*), but will not be able to generate a rule where only two arguments are cliticized (ex: "Il lui donne une pomme" - lit: *He him:dat gives an apple*). This means that with the LORIA tool, one has to encode a class for each possible combination of clitic(s). We have also enriched the MetaGrammar compiler with the notion of *optional resource*. The idea is that, in some cases, one wants a class to be able to cross, or not cross, with some other class. For instance, a verb realization may require a negation, but it also may not. With the LORIA tool, both cases have to be listed e.g. by having a class *standardVerb* and a class *negatedVerb*, Our notion of *optional resource* prevents redundancy at the MetaGrammar level and therefore allows for an even more compact encoding of syntactic knowledge.

3.4 Advantages of a MetaGrammatical level

A first advantage of using a MetaGrammar, as is argued in (Kinyon and Prolo (2002))[18] is that the syntactic phenomena covered are quite systematic: if rules are generated for "transitive-passive-whExtractedByPhrase" (e.g., *By whom was the mouse eaten*), and if the hierarchy includes ditransitive verbs, then the automatic crossing of phenomena ensures that sentences will be generated for "ditransitive-passive-whExtractedByPhrase" (i.e., *By whom was Peter given a present*). All rules for word order variations are automatically generated by underspecifying relations between quasi-nodes in the MetaGrammar hierarchy. For example, the precedence relation between direct object (NP) and second object (PP) is left unspecified to allow both:

"Il donne une pomme à Marie" (*lit: He gives an apple to Mary*)
"Il donne à Marie une pomme" (*lit: He gives to Mary an apple*)

A second advantage of the MetaGrammar is to minimize the need for human intervention in the grammar development process. The grammar writer encodes the linguistic knowledge in a compact manner i.e., the MG hierarchy, and then verifies the validity of the rules generated. If some grammar rules are missing or incorrect, then changes are made directly in the MG hierarchy and never in the generated rules. This ensures a homogeneity not necessarily present with traditional handcrafted grammars. A third advantage is that it is straightforward to obtain from a single hierarchy parallel multi-lingual grammars such as the parallel LFG grammars presented in (Butt et al. (1999)) and (Butt

[18]They also contrast MG and MetaRule approaches: the main difference is that MG are, unlike MetaRules, monotonic.

et al. (2002)), but with an explicit sharing of classes in the MetaGrammar hierarchy: (Kinyon and Rambow (2003b)) have used the tool to generate from a single hierarchy cross-framework and cross-language annotated test-suites, including English and German sentences annotated for F-structure, as well as for constituent and dependency structure.[19]

3.5 Coverage of the grammar

So far, we have implemented a non trivial hierarchy which consists in 189 classes. A fragment of the hierarchy is shown in Figure 7. From this hierarchy, we generate 550 trees decorated with functional annotation, which are then decomposed and compacted into 140 standard LFG rewriting rules. We cover the following syntactic phenomena: 50 verb subcategorization frames (including auxiliaries, modals, sentential and infinitival complements), clitics (and their placement), passives with and without agent, long distance dependencies (relatives, wh-questions, clefts) and a few idiomatic expressions. We handle most of the simple Eurotra sentences taken from (Danlos and Laurens (1991)): We have chosen those sentences because we had an appropriate lexical coverage. The grammar is under constant development, so all the numbers provided in this section are approximations, and prone to change over time. Here are some examples of constructions handled by our grammar:

Relatives

- L'ingénieur qui brigue le poste ... *(lit: The engineer who wants the position...)*
- L'ingénieur à qui l'on propose le poste ...*(lit: The engineer to whom one offers the position ...)*
- L'ingénieur à qui l'on pense proposer le poste *(lit: The engineer to whom one thinks about offering the position to ...)*
- L'ingénieur à qui Marcel pense que la DRH proposera le poste *(lit:The engineer to whom Marcel thinks the manager will offer the position ...)*

Topicalization

- C'est l'ingénieur qui brigue le poste *(lit:It is the engineer who wants the position)*
- C'est l'ingénieur à qui l'on propose le poste *(lit:It is the engineer to whom one offers the position)*
- C'est l'ingénieur à qui l'on pense proposer le poste *(lit:It is the engineer to whom one considers offering the position to)*

[19]The cross-language application of the MG for LFG is further discussed in (Kinyon and Rambow (2003a)).

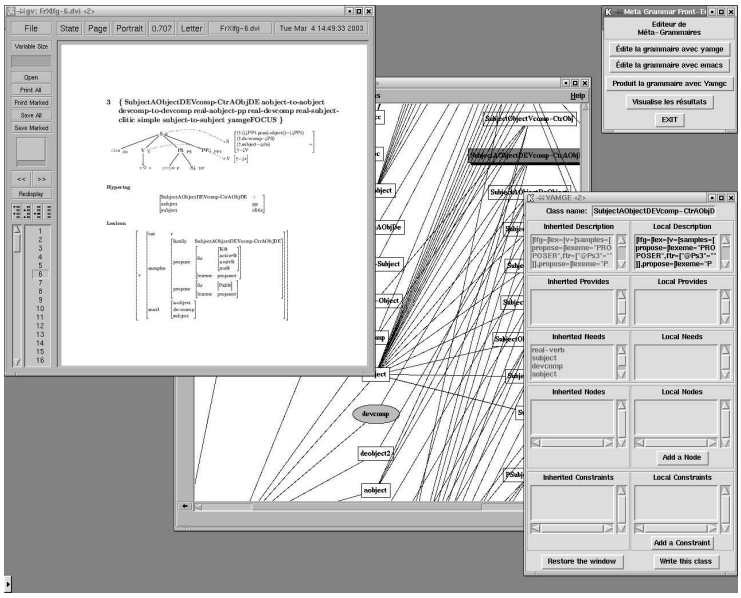

FIGURE 7 Screen capture of a fragment of our MetaGrammar hierarchy

- C'est l'ingénieur à qui Marcel pense que la DRH proposera le poste *(lit:It is the engineer to whom Marcel believes that the manager will offer the position)*

Wh-Questions

- Qui brigue le poste ? *(Who wants the position?)*
- A qui propose-t-on le poste ? *(To whom does one offer the position?)*
- A qui pense-t-on proposer le poste ? *(To whom does one think about offering the position?)*
- A qui Marcel pense-t-il que la DRH proposera le poste ? *(To whom does Marcel think that the manager will offer the position?)*

Clitics, including clitic climbing from argument position, but also from modifier position [20]

- L'ingénieur me le donne *The engineer gives it to me, (lit: The engineer me:dat it:acc gives)*
- L'ingénieur le lui donne *The engineer gives it to him, (lit: The engineer it:acc him:dat gives)*

[20] Clitic climbing, and in general extraction, from modifer positions, is easily handled by LFG, but not by TAGs (ex: *Which castle did you take a picture of*)

- L'ingénieur l'y rencontre *The engineer meets him there, (lit: The engineer him:acc there meets)*
- L'ingénieur l'y a toujours rencontré *The engineer always met him there, (lit: The engineer him:acc there has always met)*
- L'ingénieur le fait travailler *The engineer makes him work, (lit: The engineer him:acc makes to work)*
- L'ingénieur en propose le contenu *The engineer proposes the contents of it, (lit: The engineer of it proposes the content)*

3.6 Conclusion

We have presented a MetaGrammar tool which allows us to automatically generate a French LFG grammar. We keep enriching our hierarchy in order to increase the coverage of our grammars, and plan to add new languages and new formalisms. We are also investigating the automatic acquisition of a MetaGrammar can from a treebank.

References

Abeillé, A., M. Candito, and A. Kinyon. 1999. FTAG: current status and parsing scheme. In *Proceedings of the VEXTAL Conference (Venezia per il Trattamento Automatico delle Lingue*. Venice.

Boullier, P. 1998. Proposal for a natural language processing syntactic backbone. Technical Report, INRIA, France.

Bresnan, J. and R. Kaplan. 1982. Introduction: grammars as mental representations of language. In *The Mental Representation of Grammatical Relations*, pages xvii–lii. Cambridge: MIT Press.

Butt, M., S. Dipper, A. Frank, and T. Holloway-King. 1999. Writing large-scale parallel grammars for English, French, and German. In *Proceedings of the LFG '99 Conference*. CSLI Online Publications.

Butt, M., H. Dyvik, T. H. King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. In *Proceedings of 19th International Conference on Computational Linguistics, Workshop on Grammar Engineering and Evaluation*, pages 1–7. Taipei.

Candito, M. H. 1996. A principle-based hierarchical representation of LTAGs. In *Proceedings of the 16th International Conference on Computational Linguistics*, vol. 1, pages 194–199. Copenhagen.

Clément, L. and A. Kinyon. 2001. XLFG: an LFG parsing scheme for French. In *Proceedings of the LFG '01 Conference*. Hong-Kong. CSLI Online Publications.

Clément, L. and A. Kinyon. 2003a. Generating LFGs with a MetaGrammar. In *Proceedings of the LFG '03 Conference*. Saratoga Springs. CSLI Online Publications.

Clément, L. and A. Kinyon. 2003b. Generating parallel multilingual LFG-TAG grammars with a MetaGrammar. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo.

Dalrymple, M., J. Lamping, F. Pereira, and V. Saraswat. 1995. Linear logic for meaning assembly. In *Proceedings of the Joint COMPULOGNET/ELSNET/EAGLES Workshop on Computational Logic for Natural Language Processing*. Edinburgh.

Danlos, L. and O. Laurens. 1991. Présentation du projet Eurotra et des grammaires d'Eurotra-France. Technical Report, Talana, CNRS Paris 7.

Flickinger, D. 1987. *Lexical rules in the hierarchical lexicon*. Ph.D. thesis, Stanford.

Frank, A. 2000. Automatic F-Structure annotation of treebank trees. In *Proceedings of the LFG '00 Conference*. Berkeley. CSLI Online Publications.

Gaiffe, B., B. Crabbe, and A. Roussanaly. 2002. A new metagrammar compiler. In *Proceedings of the 6th International Workshop on TAG and Related Formalisms*. Venice.

Gerdes, K. 2002. DTAG: attempt to generate a useful TAG for German using a metagrammar. In *Proceedings of the 6th International Workshop on TAG and Related Formalisms*. Venice.

Hepple, M. and J. van Genabith. 2000. Experiments in structure preserving grammar compaction. In *Proceedings of the 1st meeting on Speech Technology Transfer*. Sevilla.

Joshi, A. K. and K. Vijay-Shanker. 1989. Treatment of long distance dependencies in LFG and TAG: Functional uncertainty in LFG is a corollary in TAG. In *Proceedings of the 27th Meeting of the Association for Computational Linguistics*, pages 220–227. Vancouver.

Kameyama, M. 1986. Characterising LFG in terms of TAG. Technical Report, University of Pennsylvania.

Kaplan, R. and J. Maxwell. 1996. LFG grammar writer's workbench. Technical Report, Xerox Corporation. Version 3.1.

Kinyon, A. 2000. Hypertags. In *Proceedings of the 18th International Conference on Computational Linguistics*, vol. 1, pages 446–452. Saarbruecken.

Kinyon, A. 2003. MetaGrammars for efficient development, extraction and generation of parallel grammars. Ph.D. thesis proposal, University of Pennsylvania.

Kinyon, A. and C. Prolo. 2002. A classification of grammar development strategies. In *Proceedings of 19th International Conference on Computational Linguistics, Workshop on Grammar Engineering and Evaluation*. Taipei.

Kinyon, A. and O. Rambow. 2003a. Using a MetaGrammar for parallel multilingual grammar development and documentation. In *Proceedings of the 15th European Summer School in Logic Language and Information, Workshop on multilingual grammar development*. Vienna.

Kinyon, A. and O. Rambow. 2003b. Using the metagrammar to generate cross-language and cross-framework annotated test-suites. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, 4th International Workshop on Linguistically Interpreted Corpora*. Budapest.

Srinivas, B. 1997. *Complexity of lexical descriptions and its relevance for partial parsing*. Ph.D. thesis, University of Pennsylvania.

Xia, F. 2001. *Automatic grammar generation from two perspectives*. Ph.D. thesis, University of Pennsylvania.