Generating LFGs with a MetaGrammar

Lionel Clément
INRIA-Rocquencourt
lionel.clement@inria.fr

Alexandra Kinyon
Computer and Information Science
University of Pennsylvania
kinyon@linc.cis.upenn.edu

Proceedings of the LFG03 conference
University at Albany, State University of New-York
Miriam Butt and Tracy Holloway King (Editors)
2003
CSLI Publications
http://csli-publications.stanford.edu/

Abstract

In this paper, we build up on the work presented in [Clément and Kinyon, 2003b], and [Kinyon and Rambow, 2003b]: we present the notion of MetaGrammar, which was originally successfully used to ease the development and maintenance of large Tree-Adjoining Grammars. We explain how we reuse the notion of MetaGrammar in order to generate LFGs. The main idea is that a compact MetaGrammar hierarchy is hand-crafted, from which large grammars are automatically generated offline. We argue that MetaGrammar hierarchies should abstract as much as possible from any given syntactic framework and machinery, and be closer to "descriptive linguistics" in order to facilitate porting hierarchies from one framework to another framework. We also discuss work in progress such as generating grammars for different languages and frameworks from a single MetaGrammar hierarchy.

1 Introduction

Regardless of the syntactic framework considered, the two most common ways to obtain wide-coverage grammars are either to extract grammars from a treebank, or to hand-craft grammars as is done for LFG in the ParGram project [Butt et al., 2002], for TAG in the Xtag project [XTAG Research Group, 2001], for HPSG in the Matrix [Bender et al., 2002] or even for some application-specific frameworks (e.g. Microsoft WinNLP [Suzuki, 2002]). As discussed in [Kinyon and Prolo, 2002], both approaches have well-known advantages and drawbacks. Here, we propose a "middle-way" between hand-crafted and automatically extracted grammars: the use of a MetaGrammar (MG). The main idea is that a compact MG hierarchy is hand-crafted, from which grammars are automatically generated offline. More specifically, we explain how this MetaGrammar approach, which was successfully used to develop and maintain Tree-Adjoining Grammars, is used to generate Lexical Functional Grammars. Moreover, we focus on crafting MG hierarchies in a framework-neutral manner, which means that we try to remain as close as possible to descriptive linguistics and abstract from the machinery of any given framework in order to facilitate the cross-framework portability of the the grammars we generate.

2 What is a MetaGrammar ?

2.1 Candito's MetaGrammar organization

The notion of MetaGrammar (MG) was originally presented in [Candito, 1996] to automatically generate wide-coverage TAGs for French and Italian[1], using a higher-level and compact layer of linguistic description which imposes a general organization for syntactic information in a three-dimensional hierarchy:

- Dimension 1: initial subcategorization
- Dimension 2: valency alternations and redistribution of functions
- Dimension 3: surface realization of arguments.

[1]A Similar MetaGrammar type of organization for TAGs was independently presented in [Xia, 2001] for English.

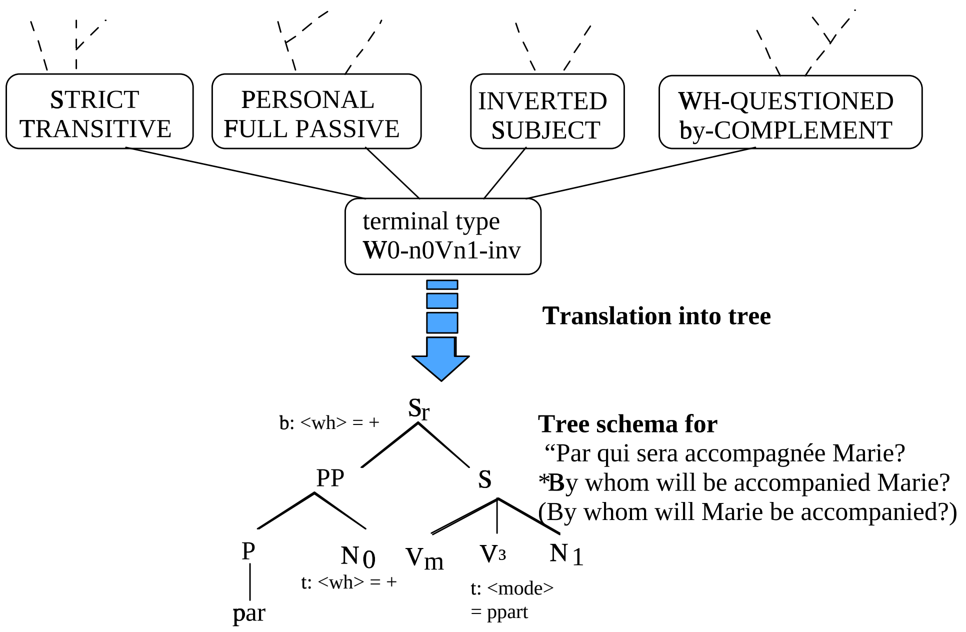
STRICT TRANSITIVE
PERSONAL FULL PASSIVE
INVERTED SUBJECT
WH-QUESTIONED by-COMPLEMENT
terminal type W0-n0Vn1-inv
Translation into tree
Tree schema for
"Par qui sera accompagnée Marie?
*By whom will be accompanied Marie?
(By whom will Marie be accompanied?)
b: <wh> = +
Sr
PP
S
P
N0
Vm
V3
N1
t: <wh> = +
t: <mode> = ppart
par

Figure 1: Generation of an elementary tree for "Par qui sera accompagnée Marie"

A MetaGrammar hierarchy is hand-crafted. From this hierarchy, a grammar is generated automatically offline. Each terminal class in dimension 1 encodes an initial subcategorization (i.e. transitive, ditransitive etc...); Each terminal class in dimension 2 encodes a list of valency alternations i.e. a possible change in the initial functions from dimension 1, including the possibility to increase or decrease the number of syntactic functions to be realized (e.g. to add an argument for causatives, to erase one for passive with no agents ...); Each terminal class in dimension 3 encodes the surface realization of a syntactic function i.e. its phrase-structure realization (e.g. declares if a direct-object is pronominalized, wh-extracted, etc.). Each class in the hierarchy is associated to the partial description of a tree, using a first-order logic language described in [Rogers and Vijay-Shanker, 1994]. These partial descriptions of trees, called **Quasi-trees**, encode **father**, **dominance**, **equality** and **precedence** relations between tree nodes. A well-formed tree is generated by inheriting from exactly one terminal class from dimension 1, one terminal class from dimension 2[2], and n terminal classes from dimension 3 (where n is the number of arguments of the TAG elementary tree being generated). For instance, as illustrated in figure 1, the elementary tree - i.e. TAG rule - for *"Par qui sera accompagnée Marie" ("By whom will be accompanied Mary")* is generated by inheriting from STRICT-TRANSITIVE in dimension 1, from PERSONAL-FULL-PASSIVE in dimension 2 and INVERTED-SUBJECT for its subject and WH-QUESTIONED-BY-COMPLEMENT for its object in dimension 3.[3] This particular tool was used to develop, from a compact hand-crafted hierarchy of a few dozen classes, a wide-coverage TAG for French of 5000

[2]This terminal class may be the result of the crossing of several super-classes, to handle complex phenomena such as *Passive+Causative*.

[3]We thank M.H. Candito for kindly allowing us to reproduce Figure 1 to illustrate her work.

elementary trees [Abeillé et al., 1999], as well as a medium-size TAG for Italian [Candito, 1999]. The compactness of the hierarchy is due to the fact that classes are defined only for **simple** syntactic phenomena: classes for complex syntactic phenomena (e.g. passive-no-agent + dative-shift) are generated by automatic crossings of classes for simple phenomena.

Although we use a different and newer MetaGrammar compiler, we essentially retain the linguistics insight of [Candito, 1996] and retain her three-dimension hierarchy, with a slight rephrasing of dimension 3: in our approach, dimension 3 encodes not only the surface realization of syntactic arguments, but also the surface realization of syntactic heads (including verbs) and of modifiers. It is also worth mentioning that dimension 3 contains in fact several distinct sub-dimensions, namely one sub-dimension for each possible syntactic function (e.g. SubjectRealization, DirObjectRealization, SecondObjRealization, SententialComplementRealization, VerbRealization, ModifiersRealization).

2.2 HyperTags

The grammar rules we generate are sorted by syntactic phenomena, thanks to the notion of *HyperTag*, introduced in [Kinyon, 2000]. HyperTags were inspired by the concept of Supertags, presented in [Joshi and Srinivas, 1994] and [Srinivas, 1997], as well as by Candito's MetaGrammar. Supertagging consists in assigning one or more TAG elementary tree to lexical items, thus providing richer syntactic and morpho-syntactic information than traditional part-of-speech tagging[4]. For example, the TAG elementary from fig. 1 would be a supertag for the verb *"accompagnée"* in *"Par qui sera accompagnée Marie" (By whom will be accompanied M.)*. This supertag informs us not only that the POS is a verb, but also gives us information about the syntactic properties of that verb (It is transitive; Passive with a Wh-extracted by-phrase etc.).

One main problem of supertags though, is that they are framework-dependent (i.e. specific to Tree Adjoining Grammars). Moreover, supertags are dependent on a specific TAG grammar: in figure 1, the supertag has a topology specific to a given grammar. One could associate to the verb *"accompagnée"* a supertag from another grammar, which could be topologically very different (e.g. with branching VP nodes) and yet capture/encode the very same syntactic properties.

Finally, supertags present some readability issues. For instance, the supertag for *"accompagnée"* could be represented using any standard tree notation, such as (S (PP (P par) N0) (S V V N)). Such notation already presents some readability issues. In practice however, esp. when supertagging with a very large grammar, one is more like to use a notation such as *"accompagnée:tree167"*, which is then even less informative than a simple POS tag.

In order to address some of these issues, we proposed in [Kinyon, 2000] the concept of HyperTags. The main idea behind HyperTags is to keep track, when trees (i.e. grammar rules) are generated from a MetaGrammar hierarchy, of which preterminal 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. So for instance, the verb "*accompagnée*" from figure 1 in "*Par qui sera accompagnée Marie*" - instead of being assigned a supertag - would be assigned the following HyperTag, which contains a direct projection of the names of the classes in fig. 1:

[4]One main goal of super-tagging is to perform a first-pass disambiguation in linear time w.r.t. a potential input string in order to improve the performance of TAG parsers, but this is orthogonal to our discussion.

$$\begin{bmatrix} \text{Subcat} & \text{Strict-Transitive} \\ \text{Valency alternations} & \text{Personal-Full-Passive} \\ \text{Argument Realization} & \begin{bmatrix} \text{Subject:} & \text{Inverted} \\ \text{Object:} & \text{Wh-Questioned-by-Complement} \end{bmatrix} \end{bmatrix}$$

So, instead of using Candito's MetaGrammar implementation, we retain her linguistic insights (i.e. her three dimensions to model syntax), but use a more recent MetaGrammar tool, which was developed at LORIA, and which explicitly supports the notion of HyperTags.

Although the LORIA tool was also originally designed to generate TAGs, it turned out to be less framework-dependent. Additionally, this tool is monotonic and is more flexible because it does not impose a fixed-number of dimensions or sub-dimensions.[5]

2.3 The LORIA MetaGrammar tool

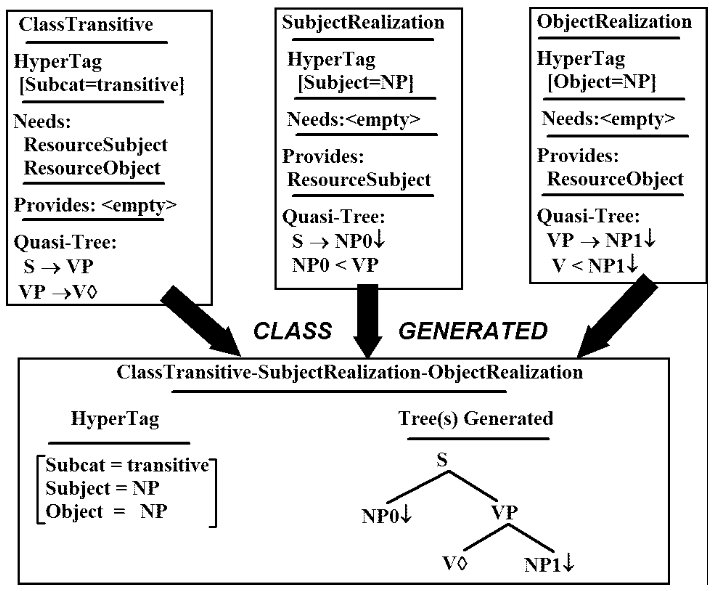

Figure 2: Generation of a simple TAG rule $\rightarrow$ stands for father, $<$ for precedes, $\diamond$ for a lexical anchor and $\downarrow$ for substitution node in the TAG terminology.

To generate TAGs and LFGs, we use a MG compiler developed at LORIA and presented in [Gaiffe et al., 2002] and [Gaiffe et al., 2003]. This compiler is freely available.[6] In the LORIA tool, each class in the MG hierarchy encodes:

- Its SuperClasse(s)[7]
- A HyperTag to capture the main syntactic properties of that class

[5]For example, Candito's compiler did not easily allow one to encode sub-dimensions for Verb-Realizations, or for Modifier-Realizations.

[6]http://www.loria.fr/equipes/led/outils/mgc/mgc.html

[7]Both the LORIA tool and Candito's tool allow multiple inheritance.

- Which resource(s) the class needs and provides
- A Topological content (TC) consisting of:
  - A set of tree nodes
  - Topological relations between those nodes (*father, dominates, precedes, equals*)
  - A function for each node to decorate the tree (e.g. for traditional agreement features and/or LFG functional equations)

The MG tool automatically crosses the classes in the hierarchy, looking to create "balanced" classes, that is classes that do not need nor provide any resource. Then for each balanced terminal class, the HyperTags inherited from its super-classes are unified, and the topological constraints (i.e. quasi-trees) are conjoined; If the HyperTag unification succeeds, one or more minimal satisfying description(s) are computed from the quasi-tree, and one or more <HyperTag, tree> pairs are generated. Figure 2 illustrates the generation of an elementary tree for a canonical transitive ("*J. sees M.*"): CLASSTRANSITIVE needs a ResourceSubject and a ResourceObject, class SUBJECTREALIZATION provides a ResourceSubject, class OBJECTREALIZATION provides a ResourceObject. When these three classes are automatically crossed by the compiler, one obtains a CLASSTRANSITIVE-SUBJECTREALIZATION-OBJECTREALIZATION which is terminal and balanced, i.e. it does not need nor provide any resource. Hence, the HyperTags of the three non terminal classes are unified and a minimal satisfying description is successfully computed for the quasi-tree:
$(S \rightarrow VP) \wedge (VP \rightarrow V) \wedge (S \rightarrow NP0) \wedge (NP0 < VP) \wedge (VP \rightarrow NP1) \wedge (V < NP1)$.

3 Generating LFGs with a MetaGrammar

3.1 Interpretation of the MetaGrammar output

The LORIA MG compiler outputs <HyperTag, tree> pairs. When generating a TAG, $HyperTag$ encodes the main syntactic properties of a TAG elementary tree (i.e. grammar rule), and $tree$ is interpreted as a TAG elementary tree.

When generating an LFG, $HyperTag$ encodes the main syntactic properties of a terminal class, and $tree$ corresponds to one or more LFG rewriting rules.

In order to generate LFGs, we retain (with a few exceptions) the same hierarchy as for TAGs. However, we enrich the topological content of each class in the hierarchy by decorating tree nodes with LFG functional equations.[8] Thus, in a first step the MetaGrammar compiler generates trees which are no longer TAG elementary trees, but rather "hybrid" trees, which are decorated with standard TAG notations (substitution nodes, foot nodes, anchor nodes etc.), but also with standard LFG notations, (in a way which is similar to constituent trees decorated with functional annotation e.g. by [Frank, 2000]). In a second step, these hybrid trees are split on the one hand into one TAG elementary tree and, on the other hand, into a decorated constituent tree which is further broken down into one or more LFG rewriting rule(s).[9] Figure 3 illustrates how a simple decorated tree is generated with the MG compiler, and how the decorated tree is decomposed into one TAG elementary tree and into two LFG rewriting rules for a canonical transitive construction. Note that

[8]These functional equations are attached to quasi-tree nodes using the same mechanism as for traditional feature-structures for agreement.

[9]In a third step, rewriting rules which differ only by the name of their non-terminals are merged, in a manner similar to that of [Hepple and van Genabith, 2000].

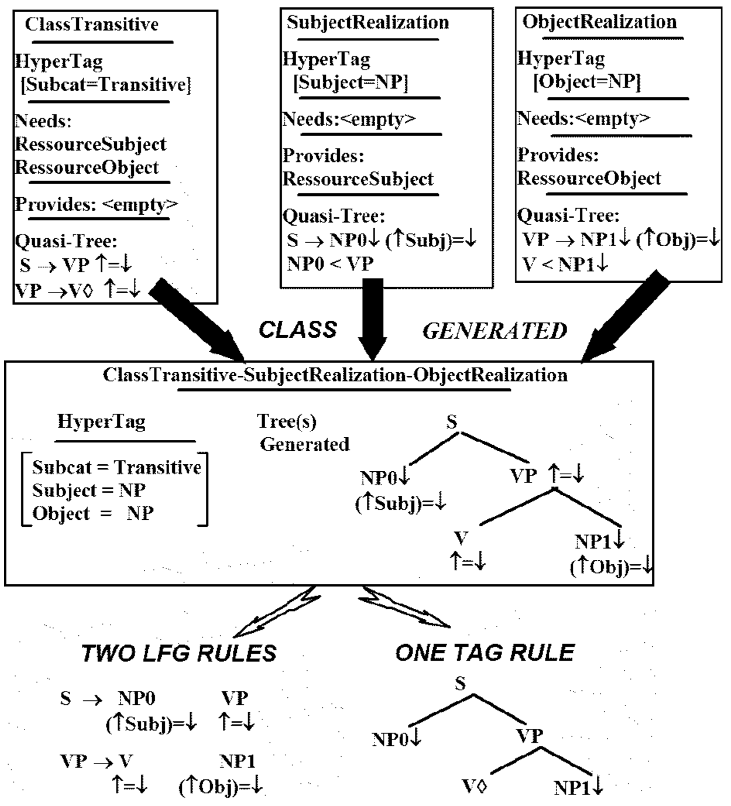
ClassTransitive
HyperTag [Subcat=Transitive]
Needs: RessourceSubject RessourceObject
Provides: <empty>
Quasi-Tree: S → VP ↑=↓ VP → V◊ ↑=↓
SubjectRealization
HyperTag [Subject=NP]
Needs:<empty>
Provides: RessourceSubject
Quasi-Tree: S → NP0↓ (↑Subj)=↓ NP0 < VP
ObjectRealization
HyperTag [Object=NP]
Needs:<empty>
Provides: RessourceObject
Quasi-Tree: VP → NP1↓ (↑Obj)=↓ V < NP1↓
CLASS GENERATED
ClassTransitive-SubjectRealization-ObjectRealization
HyperTag
Subcat = Transitive
Subject = NP
Object = NP
Tree(s) Generated
TWO LFG RULES
ONE TAG RULE

Figure 3: A simple hierarchy which yields one decorated tree, corresponding to one TAG rule and two LFG rules

the only difference between figures 2 and 3 is that in the latter, the tree nodes in the topological content of the classes are decorated with LFG functional equations.

In addition to generating LFG rewriting rules, and in order to ease the grammar-lexicon interface, each decorated tree also yields a LFG lexical template which is generated using the information encoded in the HyperTag: in figure 3, a template for a transitive subcategorization frame SubjObj:V(↑Pred=‘x<(↑Subj)(↑Obj)>’).

[Clément and Kinyon, 2003b] discuss the motivations for generating trees of depth possibly greater than one, and decomposing these trees into one or more LFG rewriting rules (instead of directly generating trees of depth one i.e. rewriting rules), as well as the motivations for using a MetaGrammar to generate LFGs. Note that a notion of resource-sensitivity similar to the one which is present in the MetaGrammar is found in LFG work on semantics [Dalrymple et al., 1995].

Word order variations are handled by leaving some precedence and/or dominance relations between quasi-tree nodes underspecified. For instance figure 4 illustrates then generation of two decorated trees for handling canonical ditranstitives in French. Contrary to English, both word-orders NP$_{Obj}$-PP$_{SecObj}$ and PP$_{SecObj}$-NP$_{Obj}$ are allowed (eg: *“Jean donne à Marie une pomme/ une pomme à Mary” - J. gives to M. an apple/ an apple to M.*). In the topological content of the classes, the precedence relation between the direct object NP and the second object PP is left underspecified. Hence, two decorated trees are obtained from the quasi-tree description: $(S \rightarrow VP) \wedge (VP \rightarrow V) \wedge (S \rightarrow NP0) \wedge (NP0 < VP) \wedge (VP \rightarrow NP1) \wedge (V < NP1) \wedge (VP \rightarrow PP) \wedge (V < PP)$

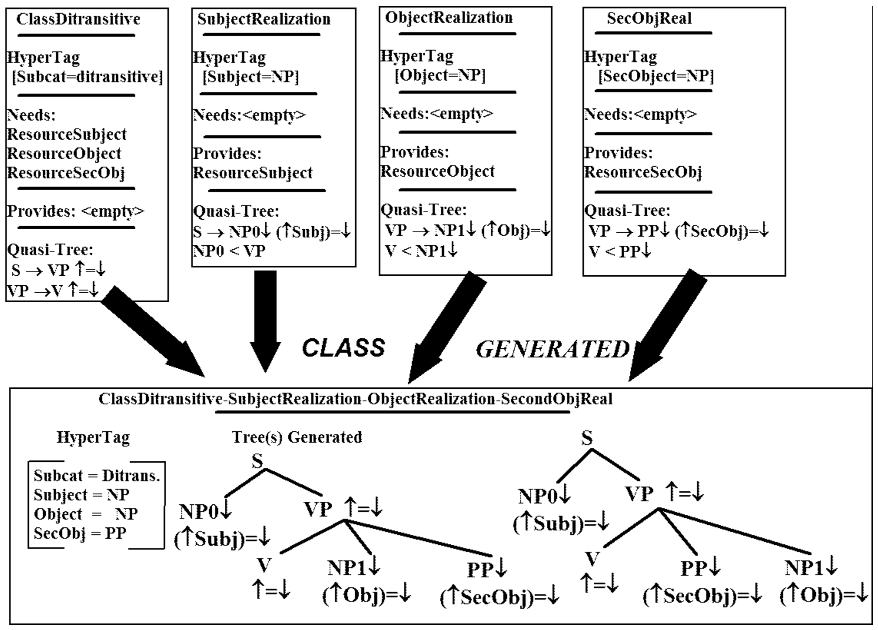
ClassDitransitive
HyperTag [Subcat=ditransitive]
Needs: ResourceSubject ResourceObject ResourceSecObj
Provides: <empty>
Quasi-Tree: S → VP ↑=↓ VP →V ↑=↓
SubjectRealization
HyperTag [Subject=NP]
Needs:<empty>
Provides: ResourceSubject
Quasi-Tree: S → NP0↓ (↑Subj)=↓ NP0 < VP
ObjectRealization
HyperTag [Object=NP]
Needs:<empty>
Provides: ResourceObject
Quasi-Tree: VP → NP1↓ (↑Obj)=↓ V < NP1↓
SecObjReal
HyperTag [SecObject=NP]
Needs:<empty>
Provides: ResourceSecObj
Quasi-Tree: VP → PP↓ (↑SecObj)=↓ V < PP↓
CLASS GENERATED
ClassDitransitive-SubjectRealization-ObjectRealization-SecondObjReal
HyperTag
Subcat = Ditrans. Subject = NP Object = NP SecObj = PP
Tree(s) Generated
S NP0↓ (↑Subj)=↓ VP ↑=↓ V ↑=↓ NP1↓ (↑Obj)=↓ PP↓ (↑SecObj)=↓
S NP0↓ (↑Subj)=↓ VP ↑=↓ V ↑=↓ PP↓ (↑SecObj)=↓ NP1↓ (↑Obj)=↓

Figure 4: French ditransitives: an example of word-order variations

The LFG rewriting rules we generate from a MG hierarchy contain "standard" LFG notations (i.e. ↑, ↓, parenthesis for optionality, Kleene star, constraint equations etc.). However, we do not generate rules containing the additional LFG operators defined in [Kaplan and Maxwell, 1996] (e.g. "shuffle", "ignore or insert"), essentially for two reasons: first, those operators are helpful for developing/maintaining hand-crafted LFGs, but, since our grammars are automatically generated and never edited in a post-generation phase[10], generating rules containing these additional operators would not make much sense. Second, those operators make it possible to express the same rewriting rules in many different ways, which seems to be a not so desirable property. This point is discussed in [Clément and Kinyon, 2003b]. In any event, our decision to generate LFG rules without the [Kaplan and Maxwell, 1996] operators, as well as without lexical rules (see section 4.1 below) has no incidence in interfacing our grammars with existing parsers.[11]

[10]i.e. only the MG hierarchies are edited, never the rules which are automatically generated

[11]So far, we have used the freely available XLFG parser described in [Clément and Kinyon, 2001] and have also experimented with the Medley parser described in [Kaplan and Maxwell, 1996].

3.2 Handling completeness, coherence and uniqueness at the MetaGrammatical level

Linguistic well formedness conditions, such as a “Predicate-Argument Cooccurence Principle” must be defined for TAG elementary trees in order to enforce completeness, coherence and uniqueness constraints, whereas LFG rewriting rules typically do not need to follow such principles. So, still for ditransitives in French, one can have an LFG rewriting rule such as:

VP	→ V	(NP)	(PP)	(NP)
	↑=↓	(↑Obj)=↓	(↑SecondObj)=↓	(↑Obj)=↓

In addition to handling the two word-order variations for canonical ditranstitives, this rule also handles the VP expansion for canonical intransitives, as well as for canonical transitives.

If completeness, coherence or uniqueness constraints are violated (e.g. *“*Mary eats an apple to Peter an apple”*), a C-structure will be built, but no corresponding well-formed F-structure. So, as expected, such an ungrammatical sentence will be rejected.

However, because of the resource-sensitivity of the MetaGrammar, coherence, consistency and uniqueness conditions are intrinsically enforced at the metagrammatical level. For example, we see in fig. 3 that a transitive needs a resourceSubject and a resourceObject and in 4 that a ditransitive needs a resourceSubject, a resourceObject and a resourceSecondObject. Therefore, instead of generating a rewriting rule such as the one above, we quite naturally generate rewriting rules which encode completeness, coherence and uniqueness. Namely, we generate four distinct rewriting rules, rule (1) for canonical intransitives, rule (2) for canonical transitives, rules (3) and (4) for the two word-order variations for canonical ditransitives:

1- VP	→ V		
	↑=↓		
2- VP	→ V	NP	
	↑=↓	(↑Obj)=↓	
3- VP	→ V	NP	PP
	↑=↓	(↑Obj)=↓	(↑SecondObj)=↓
4- VP	→ V	PP	NP
	↑=↓	(↑SecondObj)=↓	(↑Obj)=↓

As a consequence, for a similar coverage, the grammars we generate are typically larger than hand-crafted LFGs. However, this is not a major problem in practice because those rules are not manually developed nor maintained, and also because the increase in the number of rules is not such that it affects parsing performance.

Moreover, with such an approach, fewer spurious C-structures are built (i.e. C-structures with no corresponding well-formed F-structure), potentially reducing the search space when computing F-structures. Also one could envision a pruning strategy (similar to what is done with supertagging), where a first pass selects in linear time the rules which are compatible with the selectional restrictions of lexical items in a sentence (e.g. For a sentence containing only a strict transtitive verb, rewriting rules for intransitives and ditransitives will be abandonned), and a second pass combines these rules to create a C-structure. Of course it does not reduce the worst-case parsing complexity but can greatly speed up parsing in practice.[12]

[12] Another option would be to resort to guided-parsing, as described in [Barthélémy et al., 2001].

4 Main differences between TAG and LFG

We have seen in section 2 that MetaGrammars were originally used to generate Tree Adjoining grammars. When using a MG to generate LFGs, one must then be able to handle LFG machinery which does not necessarily exists in TAG. In this section, we discuss each of the three main differences between the two frameworks, which are:

- The LFG dichotomy "rewriting rules" versus "lexical rules" does not exist for TAG.
- Long-distance dependencies are handled differently in the two frameworks.
- Modifiers can be handled in a more flexible manner with LFG than with TAG

4.1 Lexical rules

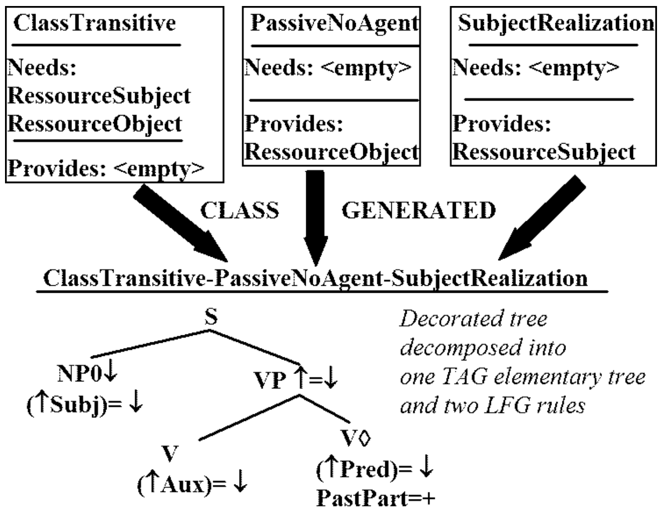
ClassTransitive
Needs: RessourceSubject RessourceObject
Provides: <empty>
PassiveNoAgent
Needs: <empty>
Provides: RessourceObject
SubjectRealization
Needs: <empty>
Provides: RessourceSubject
CLASS
GENERATED
ClassTransitive-PassiveNoAgent-SubjectRealization
S
NP0↓
(↑Subj)= ↓
VP ↑=↓
V
(↑Aux)= ↓
V◊
(↑Pred)= ↓
PastPart=+
Decorated tree decomposed into one TAG elementary tree and two LFG rules

Figure 5: Generating rules for Passive with no agent, without lexical rules

From a linguistic perspective, Candito's dichotomy between "dimension 2" (valency alternations) and "dimension 3" (realization of syntactic functions) amounts to separate what was originally thought of as "syntactic transformations" into two distinct sub-categories: "transformations" where a syntactic function is modified (e.g. passive) versus "transformations" where a syntactic function is not modified (e.g. Wh-movement, cliticization). We were not able to find where this dichotomy originated from. For instance, [Levin, 1993] - who proposes a very extensive and detailed classification of English verbs and their alternations - does not resort to this dichotomy. Our guess is that this distinction between "valency alternations" and "surface realization of syntactic functions" may have been inspired by LFG, partly because it corresponds quite straight-forwardly to the distinction between "lexical rules" and "rewriting rules" in the LFG machinery. Since the distinction is made at the matagrammatical level, there is no need for us to keep - in the grammars we generate - the distinction between lexical rules and rewriting rules. Therefore, we generate LFGs without lexical rules[13] and simply resort to rewriting rules to handle syntactic phenomena

[13]This choice has been made out of linguistic considerations: in practice, the MG compiler could, if we wanted to, generate lexical rules as well.

which are traditionally handled with LFG lexical rules. Figure 5 illustrates the generation of a passive with no agent (ex: *"The apple was eaten"*), which yields two LFG rewriting rules.[14]

Furthermore, LFG lexical rules were to some extent designed to express syntactic generalizations which hold across many subcategorization frames: for instance, the fact that one can passivize transitive verbs, ditransitive verbs, transitive verbs with particles (ex: *''The left-overs were given away to the cats"*) etc. This kind of generalization, which is desirable from a theoretical perspective, but also from a practical perspective to ease grammar development, is already captured by the MetaGrammar. Therefore, if we did generate lexical rules from a MetaGrammar hierarchy, this would amount to express the same kind of linguistic generalization using two distinct mechanisms, which would be redundant.[15]

4.2 Long-distance dependencies

As discussed in [Joshi and Vijay-Shanker, 1989], one major difference between TAG and LFG is the way each of those two frameworks handles long-distance dependencies. To analyze a sentence such as *"Which cat does Mary think that John saw"*, TAG resorts to the extended domain of locality of its elementary trees and to the "adjunction" operation to plug in the elementary tree anchored by *"think"*. This elementary tree anchored by "think" must have a distinguished node, called foot node.[16] The site of the extraction ("*Which cat*") bears no special notation.

To handle the same phenomenon, LFG, resorts to the notion of "functional uncertainty" [Kaplan and Zaenen, 1989]: the node at the site of the extraction must be decorated with a functional uncertainty equation. Figure 6 illustrates the generation of a decorated tree for a Wh-extracted object. The "extracted" tree-node bears a functional uncertainty equation.

4.3 Handling modifiers

With TAG, modifiers are adjoined and therefore modification always yields branching constituent trees. Figure 7 illustrates a simple TAG elementary tree for prenominal adjectives. By adjoining this elementary tree into NP trees, one obtains a constituent structure for e.g. "*Cute little grey kittens*". With LFGs, one can of course obtain such branching structures for modification, but one can also chose - for linguistic reasons - to handle modifiers in a "flatter" manner. For instance, the insertion of modifiers into a VP in French is, contrary to English, very flexible and there is no clear linguistic motivation for handling those VP modifiers in a non flat manner. Figure 8 illustrates how we generate a flat rule for the VP expansion of a ditransitive, which would allow to analyze "*Jean offre chaleureusement un bouquet de fleur chaque matin à Marie avant le départ du train" (lit: J. offers kindly a bouquet of flowers every morning to Mary before the departure of the train).*[17]

[14]The topological content of the classes are omitted for space reasons.

[15]This argument is clearly visible in the TAG community, where some TAGs are developed using MetaRules - a mechanism similar to LFG lexical rules - whereas some other TAGs are developed using MetaGrammars (c.f. [Kinyon and Prolo, 2002] for a discussion), the main advantages of MG being declarativity and monotonicity.

[16]TAG elementary trees with a foot node are called auxiliary trees

[17]Fig. 8 is simplified for readability. Esp. the topological constraints preventing sequences such as "VP $\rightarrow$ V Modif1 NP PP Modif2 Modif3" are not shown. The fig simply aims at giving a general idea of how such phenomena are handled using "inheritance chains" in the MG hierarchy.

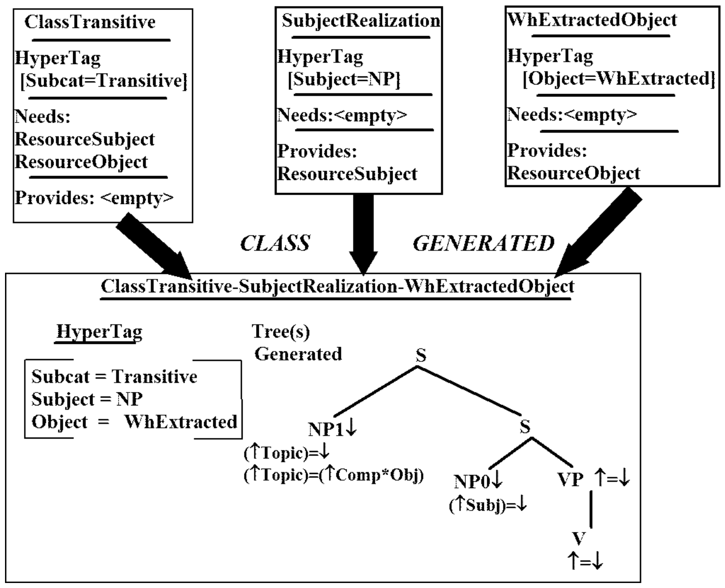

Figure 6: Generating rules with functional uncertainty equations for long distance dependencies

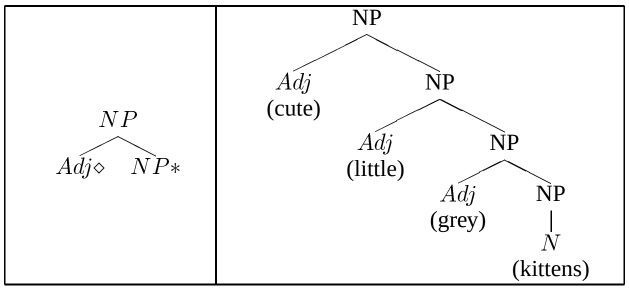

Figure 7: Modifier adjunction yields branching trees.

5 Advantages of the MetaGrammar

A first advantage of using a MetaGrammar, as is detailed in [Kinyon and Prolo, 2002], 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 rules will be generated for handling 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 MG hierarchy (e.g. precedence relation between first and second object for ditransitives in French as shown in fig. 4). This property proves very useful for encoding in a compact manner MG hierarchies for free word-order languages (e.g. Rus-

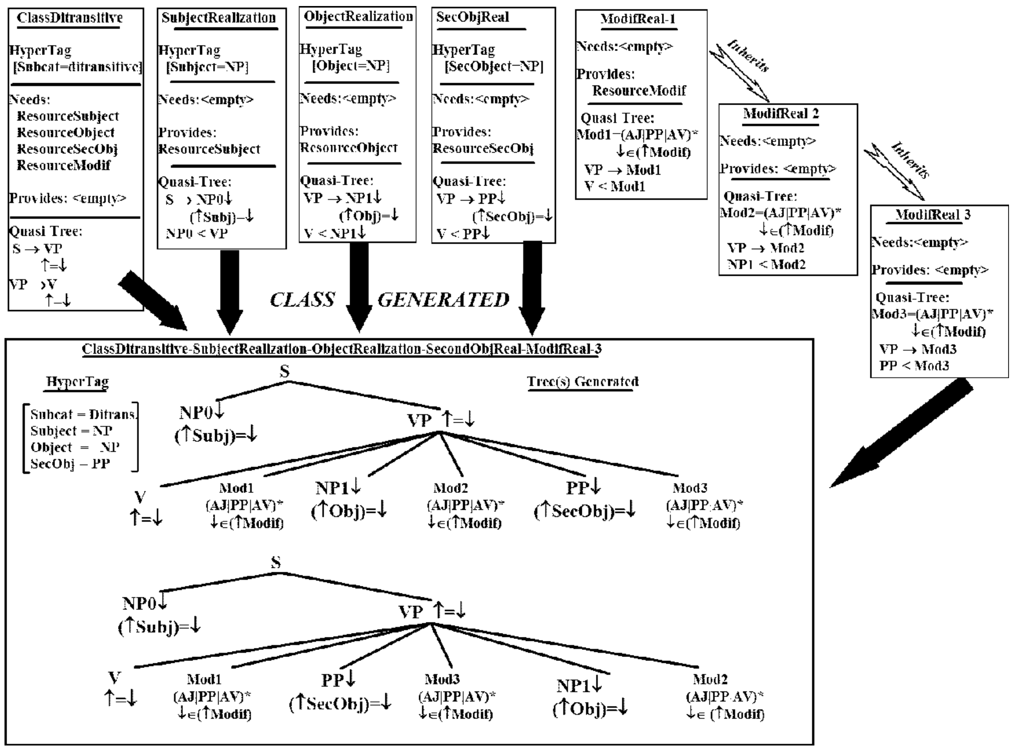

Figure 8: Generating "flat" rules for VP modification

sian, and to some degree German)[18].

A second advantage of the MG is to minimize the need for human intervention in the grammar development process. Humans encode the linguistic knowledge in a compact manner i.e. the MG hierarchy, and then verify 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 (esp. "exceptions" are never handled in a post-generation phase) This ensures a homogeneity not necessarily present with traditional hand-crafted grammars.

A third and essential advantage is that it is straightforward to obtain from a single hierarchy parallel multi-lingual grammars similar to the parallel LFG grammars presented in [Butt et al., 1999] and [Butt et al., 2002], but with an explicit sharing of classes in the MetaGrammar hierarchy plus a cross-framework application.[19]

5.1 Handling exceptionality

The MG offers a very convenient way to express generalization. So one question which arises is that of exceptionality, which is of course a major issue in any grammar development effort.

[18]See [Kinyon and Rambow, 2003a] and [Kinyon and Rambow, 2003b] for a discussion.

[19]This sharing of classes in the hierarchy across languages is similar to what is done in the HPSG Matrix project [Bender et al., 2002].

Many syntactic exceptions are lexically determined: a well known example is that of English ditransitives. Some English ditransitives accept both an NP-PP construction and a double NP construction (ex: *"J. gave an apple to M. / J. gave M. an apple"*)[20]. Some ditransitives accept only the double NP construction (ex: *J. tipped the waitress 10 dollars / *10 dollars to the waitress*). Some ditransitives (essentially from Latin origin) accept only the NP-PP construction (ex: *"J. donated a large amount of money to a NGO / *a NGO a large amount of money"*). In the MG, both word-order variations are of course encoded. The appropriate restrictions for a given verb are encoded in its lexical entry. Some other syntactic exceptions are not lexically-driven. Such is the case of the alternation "qui/que" in French when a subject is extracted from a subordinate clause[21]: "que" may be a complementizer, as in (1)[22] or a relative pronoun for an object (2), "qui" is usually never a complementizer, but a relative pronoun for a subject (3) or for an oblique (4). So, when an object is extracted from a subordinate, "que" appears twice: as a complementizer and also as a relative pronoun (5). When an oblique is extracted from a subordinate, "que" appears as a complementizer, and "qui" as a relative pronoun (6). However, when a subject is extracted from a subordinate, then "que" as a complementizer and "qui" as a relative pronoun is ungrammatical (7). Instead, the two are inverted, with "qui" appearing at the site of the complementizer and "que" at the site of relative pronoun (8).

1. *Jean pense que$_c$ Marie viendra / J. thinks that M. will come*
2. *La pomme que$_r$ Marie mange .../The apple which Mary eats*
3. *La femme qui$_r$/*que$_r$ mange des pommes ... / The woman who eats apples ...*
4. *La femme à qui$_r$ J croit que$_c$ Marie donne des fleurs / The woman to whom J. thinks that M. gives flowers*
5. *La pomme que$_r$ Pierre croit que$_c$ Marie mange / The apple that P. believes that M eats*
6. *La femme à qui$_r$ Marie croit que$_c$ Pierre offre des fleurs / The woman to whom M. believes that P. offers flowers*
7. **La femme qui$_r$ Marie pense que$_c$ viendra / The woman who M. thinks that will come*
8. *La femme que$_r$ Marie pense qui$_c$ viendra / The woman that M. thinks who will come*

In practice, the case of a "simple" subject relative (3) will be much more frequent than that of a subject relative extracted from a subordinate (8). Similarly, when a sentential complement is realized, the site of the complementizer "que" will overwhelmingly be filled with "que" (1), rather than "qui". So (8) can be considered to be a syntactic exception.

Nonetheless, when a subject is relativized, we can not know prior to parsing whether it will be extracted from a subordinate (8) or not (3). We never modify rules that have been generated from the MG, because it would potentially jeopardize most of the advantages of a MG approach. So exceptions are encoded at the metagrammatical level. For the "qui/que" alternation, this means that we encode in the MG not only a class for the standard relative pronoun "qui" for subject relatives, but also a class allowing the site of the subject relative pronoun to be "que". Similarly, we also encode a class in which the complementizer site is filled with "qui". Feature clashes will ensure at parse-time that ungrammatical sentences such as (7) and (3-*que) are rejected.

So basically, exceptions are handled in the MetaGrammar in the same manner as they would be handled in any hand-crafted grammar. The MG does not specifically ease the handling of

[20]With of course some animacy restrictions: *J. sent Mary/*London a package* (see [Levin, 1993] p. 46)

[21]This phenomenon is often compared in the linguistic literature, to the "that-trace" phenomenon in English.

[22]Contrary to English, complementizers cannot be omitted in French.

exceptions. However, most importantly, it does not impose any additional burden to handle exceptionality compared to standard hand-crafted grammars.

"Incompatible classes" can be prevented from generating rules in the MG hierarchy with three mechanisms. First, through the resource allocation model: a class A and a class B will not yield a class A-B if A does not need a resource provided by B, or B a resource provided by A. This is not a straight-forward way to encode exceptionality though, because it could be the case that a crossing involving A and B will be created nonetheless[23]. The second mechanism which prevents the generation of invalid rules is if the quasi-tree resulting from a crossing does not yield any tree. For example, if the description associated to a balanced class states that N1 is father of N2, and N2 is father of N1, assuming that the relation "father" is not reflexive, then no tree will be generated. This is also a non practical way to encode exceptionality.[24] The third solution for preventing the generation of rules from incompatible classes is to prevent those classes from crossing by encoding incompatible features in their HyperTags. This is the solution which is used in practice when crafting a MG hierarchy, for instance to prevent a class such as OBJECTRELATIVE-COMPLEMENTIZERISQUI from being created, or more generally to prevent "double extractions" (e.g. WHFRONTEDSUBJECT-WHFRONTEDOBJECT).

5.2 Compactness

Concerning the compactness of MetaGrammar hierarchies w.r.t. the number of grammar rules which will be generated, Candito's implementation - which we have discussed in section 2.1 - assumes by default that classes do cross, and a specific mechanism has to prevent incompatible classes from crossing. The explicit three-dimensional organization makes classes combine in a cross-product manner. If dimension 1, 2 and 3 have respectively i, j and k terminal classes, the number of combinations to generate grammar rules will be $i * j * k$. However, within each dimension, a given number of classes are hand-crafted, and then classes within each dimension are automatically crossed. Assuming dimensions 1, 2 and 3 have respectively m, n, and l hand-crafted classes, the automatic crossing amounts to creating powersets for each of the three dimensions. Hence the grammar generation process is exponential in the number of grammar rules generated from the hand-crafted classes in the MetaGrammar hierarchy, since we have $i = 2^m$, $j = 2^n$ and $k = 2^l$. Similarly, with the LORIA implementation - which is the one we have used - a hierarchy of n terminal classes may yield in the worst case 2^n decorated trees (i.e. a Powerset). This ensures that large grammars can be generated from a small limited number of hand-crafted classes in the hierarchy and also is a promising property for grammar extraction.

6 Some results and work in progress

Researchers from several groups, in particular from INRIA/Rocquencourt, LORIA, University of Paris 7, University of Pennsylvania and Columbia university are working on the MetaGrammar. These groups try to organize informal meetings to discuss potential improvements to the MG tools and resources. A mailing list is also available.[25] Moreover, a project has just started at Columbia

[23]e.g. if A needs R1 and provides R2, C needs R2 and provides R3, B needs R3 and provides R4 and D needs R4 and provides R1, then a class A-B-C-D will eventually be generated

[24]Moreover, different MG compilers (or even different versions of the same compiler) may make slightly different assumptions: e.g. some versions may implement the relation "father" to be reflexive, in which case the description N1 is father of N2 and N2 is father of N1 will yield a node N1=N2.

[25]At the time of this writing: http://www.ps.uni-sb.de/mailman/listinfo/metagrammar

University,to generate, from a single MG hierarchy, grammars for the different dialects of spoken Arabic.

6.1 Linguistic perspectives: multilingual and cross-framework grammars

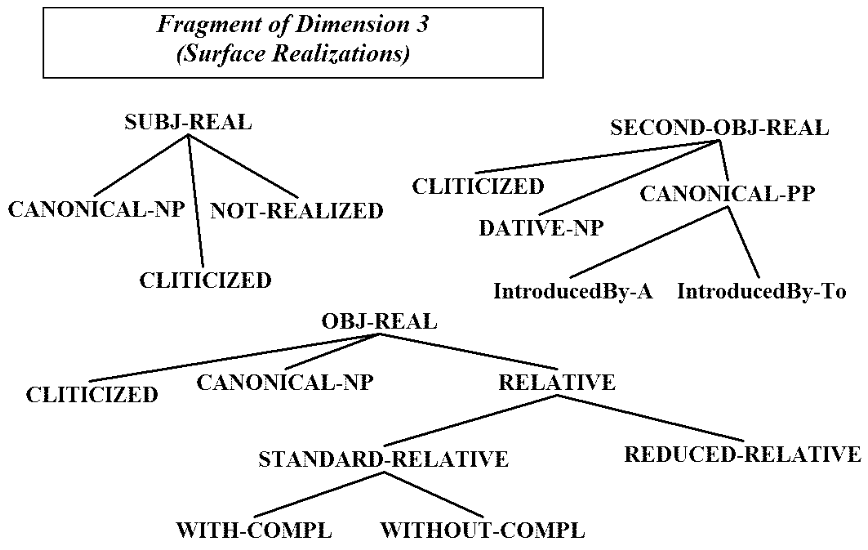

Figure 9: Sharing hierarchy classes for different languages

From a linguistic perspective, work is being conducted on developing MG hierarchies for several languages and frameworks. The hypothesis is that given a framework-neutral MG hierarchy it should be reasonably simple to use that hierarchy to generate new frameworks, and also to re-use as many classes as possible to develop a grammar for a new language. When porting a hierarchy from one framework (e.g. TAG) to another framework (e.g. LFG), one "only" needs to change the topological content of the classes in the hierarchy.[26] When porting a given hierarchy from one language to another language, the assumption is that many classes can be reused for the new language. Of course, the more related the languages are, the more classes will be reusable.

In addition to the TAGs generated for French and Italian [Candito, 1999], Metagrammars - and more specifically the LORIA MG tool - was used at the university of Paris 7 to generate a TAG for German [Gerdes, 2002]. Also at Paris 7, A MG hierarchy is being developed for Korean by Sinwon Yoon. In terms of grammar development, the largest LFG generated with the MG is the grammar for French, which is developed and maintained by Lionel Clément at INRIA. This MG hierarchy for French yields approximately 600 decorated trees, and is also used to generate Range-Concatenation Grammars [Boullier, 1998]. The syntactic phenomena covered by this LFG for French are discussed in [Clément and Kinyon, 2003a].

[26]Of course "only" should be taken with a grain of salt, because if the frameworks are very different and the hierarchy very large, this still involves a substantial amount of work. However, it is still much faster than developing a MG hierarchy or a grammar from scratch.

Concerning parallel multilingual and grammar generation, in addition to proposing a compact representation of syntactic knowledge, [Candito, 1999] explored whether some components of the MG hierarchy could be re-used across similar languages (French and Italian). However, she developed two distinct hierarchies to generate grammars for these two languages and generated only TAG grammars.

We have extended the use of the MetaGrammar to generate LFGs and also have pushed further its cross-language and cross-framework potential by generating parallel TAGs and LFGs for several languages from one single hierarchy: [Clément and Kinyon, 2003b] describe how a single hierarchy yields parallel TAG and LFG grammars for French and English;

[Kinyon and Rambow, 2003b] have used a similar strategy 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. Additionally, this line of work was extended in [Kinyon and Rambow, 2003a] to generate in parallel annotated test-suites and LFGs for English, French and German and to discuss the prtability of an existing hierarchy to Russian. When generating grammars for different languages from a single MG hierarchy, a first general tendency is that classes which are higher in the hierarchy are more likely to be shared for several languages, whereas terminal classes (i.e. leaves) are more likely to be language specific. Moreover, as a rule of thumb, more classes in dimension 1 (initial subcategorization) and in dimension 2 (valency alternations) will be shared than classes in dimension 3 (surface realization of syntactic functions). For dimension 1, most languages have subcategorization frames such as intransitive, transitive, ditransitive etc. Of course, even in dimension 1, some classes may be language-specific. For example, the class TRANSITIVEWITHPARTICLE in dimension 1 is valid for English and German (e.g. "*He gave something up*"), but not for French. Similarly, in dimension 2, the class IMPERSONALPASSIVE is valid for German (e.g. *"Es wurde getanzt"*), but not for English nor French (*"Il a été dansé"* with "il" impersonal is at best extremely marginal).

Figure 9 illustrates a (simplified) fragment of dimension 3 of a MG hierarchy which was crafted for English, French and German. The classes CLITICIZED for subject, object and second object realizations are valid only for French. The class STANDARDRELATIVEWITOUTCOMPLEMENTIZER is valid for German and English, but not for French. The class CANONICAL-PP for a second object realization is valid only for English and French, but not for German. Conversely, the class DATIVE-NP is valid for English and German, but not for French which does not allow a double NP construction. Other classes such as STANDARDRELATIVEWITHCOMPLEMENTIZER are shared by all three languages.

Other uses of the MG include a replication of the Xtag grammar for English [Kinyon and Prolo, 2002], [Kinyon, 2003], in order to detect errors in the hand-crafted Xtag and help the maintenance of the grammar, and also in order to systematically compare within the same project a MG development strategy with a MetaRule development strategy [Prolo, 2002]. As a side-effect, using the technique of generating "hybrid trees" described in section 3, in addition of regenerating the English Xtag grammar, we plan to generate, as a side-effect, a parallel LFG grammar for English.

Although the MG automates grammar development, one bottleneck remains: the lexicon. Although we generate lexical templates from our MG hierarchies, therefore addressing to some extent the grammar-lexicon interface, satisfactory lexical coverage still remains a problem. To illustrate this point, FTAG - the TAG grammar for French generated from a hand-crafted MG hierarchy - contains more than 5000 TAG elementary trees but cannot parse newspaper text essentially because of a lack of lexical coverage and maintenance and of a poor interaction between the grammar and the lexicon. By contrast, Xtag for English, which was entirely handcrafted over more than 15 years, contains only 1200 elementary trees but can parse newspaper text [Prasad and Sarkar, 2000].[27] To remedy this problem, many grammars are extracted from treebanks (esp. from the Penn Treebank) for a variety of frameworks: for LFG [Cahill et al., 2003], for CCG [Hockenmaier et al., 2002], for TAGs [Chiang, 2000], [Chen, 2001], [Xia, 2001]. However, such treebank-extracted grammars yield problems of their own, esp. the lack of portability across frameworks, a potential lack of annotated data, errors and lack of annotation consistency, the risk of extracting rules that are not linguistically sound etc. To remedy some of the problems of treebank extracted grammars while retaining the advantages of an empirical approach, we propose to use the MG as a syntactic interlingua for grammar extraction. From a MG hierarchy automatically extracted from the Penn treebank, the idea is to generate grammars for several frameworks. This work, which is described in detail in [Kinyon, 2003] is a four-step process:

1. Extract HyperTags from the Penn treebank
2. Turn each HyperTag feature into a class in the MG hierarchy
3. Associate a topological content to each class in the hierarchy
4. Generate grammars for several frameworks using and existing MG compiler

6.2 Improving MetaGrammar tools

While extending the use of the MG to generate new frameworks, we have improved the LORIA MG implementation. More specifically, Lionel Clément has implemented a new graphical user interface[28] to ease the development and maintenance of MG hierarchies and a new version of the compiler, which supports the notion of free variables for quasi-tree nodes[29], optional resources, as well as additional relations between tree nodes in the topological content of classes. These additional relations include “sister”, “c-command”, “unequality” and a distinction between reflexive and non reflexive “dominance” and “parenthood”. His implementation also makes it possible to generate sets of rewriting rules (instead of a decorated tree which needs to be broken down in a post-processing phase). We do not further detail these technical points for sake of brevity.

Additionally, yet another version of the MG compiler was implemented by E. de la Clergerie, using the DyAlog programming language (which has similarities with Prolog) [Villemonte de la Clergerie, 2002]. This recent compiler significantly speeds up the grammar generation process.

As the MG is being used by more researchers and to generate frameworks other than TAGs, grammars for more languages etc. there is an increasing need for a standardization effort such as the definition of XML dtds to ease the sharing of MG-related tools. One of the main goals of the MG is to achieve a higher level of consistency in the grammars generated. Of course, it is still a small effort compared e.g. to ParGram. In the long-run, it would be desirable to have better “checking mechanisms” to ensure consistency and help error detection in MG hierarchies - such as typos, misspelled or inconsistent feature names (ex: Gend. versus Gender) etc. - for instance by having a feature declaration mechanism similar to the “feature space” in the ParGram project [Butt et al., 2003].

[27] More than lexical or grammatical coverage, the main problem for the Xtag system is its parser’s performance, which cannot handle long sentences from the WSJ. However, contrary to LFG, the worst-case parsing complexity of TAGs is polynomial and much faster TAG parsers have been developed, such as [Barthélémy et al., 2001]. Unfortunately, integrating parser-grammar-lexicon from those different sources has not been done at this time.

[28] http://atoll.inria.fr/∼ lclement/

[29] This feature makes rules for clitic ordering easier to encode in a compact manner

7 Conclusion and future work

We have presented the notion of MetaGrammar - which was originally designed to develop and maintain Tree-Adjoining Grammars, and have explained how we have used it to generate LFGs. The main idea is that a compact MG hierarchy is handcrafted. From this hierarchy, grammars are generated automatically offline. We focus on crafting hierarchies that are as framework-neutral and close to descriptive linguistics as possible, in order to ease grammar portability. We also try to maximize cross-language sharing by generating, from a single MetaGrammar hierarchy, grammars for more than one language.

In order to further support our claim that the MetaGrammar is framework-independent, we hope to generate in future work grammars for additional frameworks, such as CCG or HPSG.[30]

We keep expanding our MG hierarchy in order to broaden the coverage of our grammars (in particular for French) and are also trying - in collaboration with other research groups - to improve existing MG tools (GUIs, editors, compilers etc.) Moreover, we are focussing on using the MetaGrammar as a syntactic Interlingua for Grammar extraction [Kinyon, 2003], the idea being to extract a MG from the Penn-Treebank, and then generating grammars for several frameworks, including LFG. In particular, we hope to compare LFGs extracted via a MG approach with the LFGs directly extracted from the Penn Treebank e.g. by [Cahill et al., 2003]. Finally, based on the work of [Kameyama, 1986] and on the "hybrid" TAG-LFG trees we generate with the MG, we are investigating whether a framework such as Lexical-Functional-Tree-Adjoining-Grammar (LFTAG) could have interesting properties, both from a theoretical and practical perspective.

Acknowledgements

We would like to thank the ILFGA, for providing partial funding for students to attend the LFG conference. We also would like to thank Aravind Joshi, Owen Rambow, Eric de la Clergerie, Anette Frank, Josef van Genabith, Ruth O'Donovan, the LFG'03 conference participants and the LORIA-INRIA-Paris 7 participants to the MetaGrammar reunion for very helpful discussions and feedback on several aspects of this work.

References

[Abeillé et al., 1999] Abeillé, A., Candito, M., and Kinyon, A. (1999). FTAG: current status and parsing scheme. In *Proc. Vextal-99*, Venice.

[Barthélémy et al., 2001] Barthélémy, F., Boullier, P., Deschamp, P., and de la Clergerie, E. (2001). Guided parsing of range concatenation lanuages. In *Proc. ACL-01*, Toulouse.

[Bender et al., 2002] Bender, E., Flickinger, D., and Oepen, S. (2002). The Grammar Matrix. In *Proc. COLING 2002 Workshop on Grammar Engineering and Evaluation*, Taipei.

[Boullier, 1998] Boullier, P. (1998). Proposal for a natural language processing syntactic backbone. Technical report, Inria. France.

[Butt et al., 1999] Butt, M., Dipper, S., Frank, A., and Holloway-King, T. (1999). Writing large-scale parallel grammars for English, French, and German. In *Proc. LFG-99*.

[Butt et al., 2002] Butt, M., Dyvik, H., Holloway-King, T., Masuichi, H., and Rohrer, C. (2002). The parallel grammar project. In *Proc. COLING 2002 Workshop on Grammar Engineering and Evaluation*, TAIPEI.

[30]the latter being more challenging, since HPSG is head-driven and already relies on a "built-in" hierarchical organization [Flickinger, 1987].

[Butt et al., 2003] Butt, M., Forst, M., Holloway-King, T., and Kuhn, J. (2003). The feature space in parallel grammar writing. In *ESSLLI 2003 Workshop on Ideas and Strategies for Multilingual Grammar Development*, Vienna.

[Cahill et al., 2003] Cahill, A., McCarthy, M., O'Donovan, R., van Genabith, J., and Way, A. (2003). Extracting large-scale lexical resources for LFG from the Penn-II Treebank. In *Proc. LFG-03*, Saratoga-Springs.

[Candito, 1996] Candito, M. (1996). A principle-based hierarchical representation of LTAGs. In *COLING-96*, Copenhagen.

[Candito, 1999] Candito, M. (1999). *Représentation modulaire et paramétrable de grammaires électroniques lexicalisées*. PhD thesis, Univ. Paris 7.

[Chen, 2001] Chen, J. (2001). *Towards Efficient Statistical Parsing using Lexicalized Grammatical Information*. PhD thesis, Univ. of Delaware.

[Chiang, 2000] Chiang, D. (2000). Statistical parsing with an automatically-extracted TAG. In *ACL-00*, Hong-Kong.

[Clément and Kinyon, 2001] Clément, L. and Kinyon, A. (2001). XLFG: an LFG parsing scheme for French. In *Proc. LFG-01*, Hong-Kong.

[Clément and Kinyon, 2003a] Clément, L. and Kinyon, A. (2003a). Automating the generation of a wide-coverage LFG for French using a MetaGrammar. In *Proc. Formal Grammars-03*, Vienna.

[Clément and Kinyon, 2003b] Clément, L. and Kinyon, A. (2003b). Generating parallel multilingual LFG-TAG grammars with a MetaGrammar. In *Proc. ACL-03*, Sapporo.

[Dalrymple et al., 1995] Dalrymple, M., Lamping, J., Pereira, F., and Saraswat, V. (1995). Linear logic for meaning assembly. In *Proc. CLNLP*, Edinburgh.

[Flickinger, 1987] Flickinger, D. (1987). *Lexical rules in the hierarchical lexicon*. PhD thesis, Stanford.

[Frank, 2000] Frank, A. (2000). Automatic F-Structure annotation of treebank trees. In *Proc. LFG-00*, Berkeley.

[Gaiffe et al., 2002] Gaiffe, B., Crabbe, B., and Roussanaly, A. (2002). A new metagrammar compiler. In *Proc. TAG+6*, Venice.

[Gaiffe et al., 2003] Gaiffe, B., Crabbe, B., and Roussanaly, A. (2003). Une plate-forme de conception et d'exploitation d'une grammaire d'arbres adjoints lexicalisés. In *Proc. TALN-03*, Batz-sur-Mer.

[Gerdes, 2002] Gerdes, K. (2002). DTAG. attempt to generate a useful TAG for german using a metagrammar. In *Proc. TAG+6*, Venice.

[Hepple and van Genabith, 2000] Hepple, M. and van Genabith, J. (2000). Experiments in structure preserving grammar compaction. In *Proc. 1st meeting on Speech Technology Transfer*, Sevilla.

[Hockenmaier et al., 2002] Hockenmaier, J., Bierner, G., and Baldridge, J. (2002). Extending the coverage of a CCG system. In *Journal of Language and Computation*.

[Joshi and Srinivas, 1994] Joshi, A. and Srinivas, B. (1994). Disambiguation of Super Parts of Speech (or Supertagging): almost parsing. In *Proc. COLING-94*, Kyoto.

[Joshi and Vijay-Shanker, 1989] Joshi, A. K. and Vijay-Shanker, K. (1989). Treatment of long distance dependencies in LFG and TAG: Functional uncertainty in LFG is a corollary in TAG. In *Proc. ACL-89*, Vancouver.

[Kameyama, 1986] Kameyama, M. (1986). Characterising LFG in terms of TAG. In *Unpublished Technical report*, Univ. of Pennsylvania.

[Kaplan and Maxwell, 1996] Kaplan, R. and Maxwell, J. (1996). LFG grammar writer's workbench. Technical Report version 3.1, Xerox corporation.

[Kaplan and Zaenen, 1989] Kaplan, R. and Zaenen, A. (1989). Long distance dependencies, constituent structure and functional uncertainty. In *Alternatives conceptions of phrase-structure*, Univ. of Chicago press.

[Kinyon, 2000] Kinyon, A. (2000). Hypertags. In *COLING-00*, Sarrebrucken.

[Kinyon, 2003] Kinyon, A. (2003). *MetaGrammars for efficient development, extraction and generation of parallel grammars*. PhD thesis, Proposal. Univ. of Pennsylvania.

[Kinyon and Prolo, 2002] Kinyon, A. and Prolo, C. (2002). A classification of grammar development strategies. In *Proc. COLING 2002 Workshop on Grammar Engineering and Evaluation*, Taipei.

[Kinyon and Rambow, 2003a] Kinyon, A. and Rambow, O. (2003a). Using a MetaGrammar for parallel multilingual grammar development and documentation. In *ESSLLI workshop on multilingual grammar development*, Vienna.

[Kinyon and Rambow, 2003b] Kinyon, A. and Rambow, O. (2003b). Using the MetaGrammar to generate cross-language and cross-framework annotated test-suites. In *LINC-EACL*, Budapest.

[Levin, 1993] Levin, B. (1993). *English verb classes and alternations*. University of Chicago Press, Chicago.

[Prasad and Sarkar, 2000] Prasad, R. and Sarkar, A. (2000). Comparing test-suite based evaluation and corpus-based evaluation of a wide-coverage grammar for English. In *LREC-00*, Athens.

[Prolo, 2002] Prolo, C. (2002). Generating the Xtag english grammar using metarules. In *Proc. COLING-02*, Taipei.

[Rogers and Vijay-Shanker, 1994] Rogers, J. and Vijay-Shanker, K. (1994). Obtaining trees from their description: an application to TAGs. In *Computational Intelligence 10:4*.

[Srinivas, 1997] Srinivas, B. (1997). *Complexity of lexical descriptions and its relevance for partial parsing*. PhD thesis, Univ. of Pennsylvania.

[Suzuki, 2002] Suzuki, H. (2002). A development environment for large-scale multi-lingual parsing systems. In *Proc. COLING 2002 Workshop on Grammar Engineering and Evaluation*, Taipei.

[Villemonte de la Clergerie, 2002] Villemonte de la Clergerie, E. (2002). Construire des analyseurs avec DyALog. In *Proc. of TALN'02*.

[Xia, 2001] Xia, F. (2001). *Automatic grammar generation from two perspectives*. PhD thesis, Univ. of Pennsylvania.

[XTAG Research Group, 2001] XTAG Research Group (2001). A lexicalized tree adjoining grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.