Integrating a unification-based semantics in a large scale Lexicalised Tree Adjoining Grammar for French

Claire Gardent
CNRS / LORIA
Equipe Talaris, Bat. B
615, rue du jardin botanique
54600 Villers les Nancy
France
claire.gardent@loria.fr

Abstract

In contrast to LFG and HPSG, there is to date no large scale Tree Adjoining Grammar (TAG) equiped with a compositional semantics. In this paper, we report on the integration of a unification-based semantics into a Feature-Based Lexicalised TAG for French consisting of around 6 000 trees. We focus on verb semantics and show how factorisation can be used to support a compact and principled encoding of the semantic information that needs to be associated with each of the verbal elementary trees. The factorisation is made possible by the use of XMG, a high-level linguistic formalism designed to specify and compile computational grammars and in particular, grammars based on non-local trees or tree descriptions.

1 Introduction

Whilst there exists large scale LFGs (Lexical Functional Grammar) and HPSGs (Head-Driven Phrase Structure Grammar) equipped with a compositional semantics (Copestake et al., 2001; Frank and van Genabith, 2001), available Tree Adjoining Grammars remain largely syntactic.

One reason for this is that there has been, up to recently, much debate about how best to combine TAG with a compositional semantics. Should it be based on the derived or the derivation tree ? Should Feature-Based LTAG be used or should synchronous TAG? Many proposals have been put forward but only recently did sufficient consensus emerge to support the specification of a TAG based compositional semantics. In a nutshell, it can be achieved either by using a synchronous TAG (Nesson and Shieber, 2006) (in this case, the grammar explicitely describes and synchronises syntax and semantics structures) or by using Feature-Based LTAG (in which case, the synchronisation between syntax and semantics is mediated by the unification of semantic indices associated with the FTAG elementary trees).

Another more practical reason for the absence of large scale TAGs integrating a compositional semantics is the lack of available computational frameworks. Up to recently, there has been no available grammar writing environment and parser that would support the integration of compositional semantics into a TAG. One step in that direction is provided by the development of XMG(Duchier et al., 2004), a formalism which supports the specification of Feature-Based LTAGs equipped with a compositional semantics à la (Gardent and Kallmeyer, 2003).

In this paper, we report on the integration of a unification-based semantics into a Feature-Based LTAG for French which consists of around 6 000 trees. This integration is specified using XMG and we show how this formalism can be used to support a compact and principled encoding of the semantic information that needs to be associated with each of the 6 000 elementary trees.

The article is structured as follows. We start (section 2) by presenting XMG and showing how it supports the specification of Feature-Based LTAGs equipped with a compositional semantics. We then present SEMFRAG, the FTAG grammar for French that we developed (section 3). In section 4, we show how XMG can be used to minimise the development cost involved in enriching such

a grammar with a compositional semantics. Section 5 compares the approach with related work and concludes with pointers for further research.

2 The XMG formalism

The XMG formalism was designed to support the development and factorisation of computational grammars for Natural Language. Like PATR II it is theory neutral in that its use is not restricted to a single grammatical theory. Unlike PATR II however, the language provided by XMG allows the linguist to talk about the basic building blocks not only of rule based computational linguistic theories such as as HPSG (Head Driven Phrase Structure Grammar) and LFG (Lexical Functional Grammar) but also of tree based theories such as TAG (Tree Adjoining Grammar). As we shall see, this involves allowing for sophisticated node naming and identification mechanisms. Other differences between PATR II and XMG include a more general use of disjunction, the use of colours to control tree construction and a more natural encoding of trees and of semantic representation than is permitted by PATR II. A detailed definition of the XMG formalism is given in (Duchier et al., 2004). In what follows, we give an intuitive presentation of XMG emphasising the points that support a strongly factorised specification of grammars and in particular, of SEMFRAG. We start by presenting the basic building blocks XMG allows the linguist to talk about (2.1). We then go on to discuss the factorising mechanisms it supports (2.2). Finally, we introduce the several node naming and identification mechanisms it provides (2.3).

2.1 The basic building blocks

In XMG, the basic building blocks are CLASSES which may be specified along three dimensions : a syntactic dimension (SYN) which consists of a tree description whose node variables can be decorated with feature structures; a semantic dimension (SEM) consisting of a flat semantic formula; and a syntax/semantic interface (INTERFACE) for synchronising semantic formulae and tree descriptions .

SYN. The syntactic dimension in XMG allows the linguist to specify tree descriptions i.e., trees that can be underspecified with respect to both dominance and precedence. The trees described may be either local or extended and their nodes may be decorated with either one or two feature structures (two for TAG TOP and BOTTOM feature structures).

SEM. Using the semantic dimension, the linguist can specify unification based flat semantic formulae in the sense of (Copestake et al., 2001) i.e., non recursive formulae describing first order formulae with lambda binders replaced by unification variables and where scope may be underspecified. Semantic schemas can also be specified in which predicates are replaced by unification variables that will be instantiated during lexical lookup. For instance, the SEM dimension may include the following semantic formula and schema[1]:

(1) a. Every: $l_0 : \forall(X, h_1, h_2), h_1 \geq L_1, h_2 \geq L_2$

b. Binary Relation Schema: $l_1 : P(E), l_1 : Theta_1(E, X), l_1 : Theta_2(E, Y)$

In (1a), the flat semantic formula associated with *every* underspecifies scope by stating that the scope handle h_2 scopes, directly or indirectly ($\geq$), over (the label L_2 associated with) the scopal argument.In (1b) on the other hand, underspecification bears on the predicate P and the theta roles $Theta_1, Theta_2$ which are unification variables whose value will be provided by the lexicon. In this way, this binary relation schema can be used to represent the semantics of all verbs denoting a binary relation. The lexicon will then specify for each verb the relevant relation and theta roles.

INTERFACE. The third XMG dimension permits synchronising syntax and semantics. In essence, features that are used in SYN or in SEM can be assigned global names in the INTERFACE dimension and synchronised using variable sharing. For instance, given a feature-value pair F = X occuring in the SYN dimension and a semantic parameter Y occuring in the SEM dimension, the following interface constraint permits both unifying ("synchronising") X and Y and assigning them the global names IDX and ARG respectively :

(2) IDX = $\boxed{1}X$, ARG = $\boxed{1}Y$

As we shall see in section 4.2.2, the interface allows for a natural and highly factorised means of stating syntax/semantics linking constraints (e.g., the subject constituent provides the semantic index for the first semantic argument).

[1] Here and in what follows, we adopt the convention that identifiers starting with an upper case letter are unification variables.

2.2 Factorising mechanisms

An important feature of a linguistic formalism is that it supports a high level of factorisation thus facilitating grammar development, debugging and maintenance. In XMG, factorising can be achieved using disjunctions, conjunctions and inheritance of classes. As argued in (Crabbé, 2005), classes disjunction supports the description of alternatives, for instance, to describe the alternative possible realisations of a subject (see below). As usual, conjunction and inheritance of classes permits combining the content of two classes[2].

2.3 Node naming and identification mechanisms

In combining tree descriptions, the linguist often wants to identify nodes across descriptions. One distinguishing feature of XMG it that it supports a sophisticated treatment of node naming and node identification (Gardent and Parmentier, 2006).

Node naming. In XMG, node names are by default local to a class. However explicit IMPORT and EXPORT declarations can be used to make names "visible" to children classes. An EXPORT declaration makes the exported name(s) visible to all children classes. Conversely restrictive IMPORT declarations can be used either to block or to rename exported variables that are visible through inheritance.

Node identification. As we have just seen, IMPORT and EXPORT declarations can be used to make names "visible" to children classes and thereby idendify nodes from different classes. For instance, if class C_1 inherits from class C_2, C_1 refers to a node variable X and C_2 exports X, then X denotes the same node in both C_1 and C_2.

However, this mechanism only works within a single branch of the inheritance hierarchy. Indeed in case of multiple inheritance (two classes C_1 and C_2 export the same variable X to a third class inheriting from both C_1 and C_2), identification will fail (X will not be interpreted as denoting the same node in both C_1 and C_2). To remedy this shortcoming, XMG allows for explicit node identifications. Thus in the above case, X can be identified using the constraint $C_1.X = C_2.X$.

[2]The distinction between conjunction and inheritance has to do with some intricate issues concerning node identifications which we will not address here. See (Gardent and Parmentier, 2006) for a detailed discussion on this.

This concludes our informal presentation of XMG. For a more precise definition of its syntax, semantic and compilation process, we refer the reader to (Duchier et al., 2004).

3 SemFraG

To illustrate the expressive power of XMG, we now show how it can be used to specify SEMFRAG, a TAG for French which integrates a unification based compositional semantics. We start by presenting the grammar produced by the XMG specification.

SEMFRAG is a unification based version of LTAG namely, Feature-based TAG. A Feature-based TAG (FTAG, (Vijay-Shanker and Joshi, 1988)) consists of a set of (auxiliary or initial) elementary trees and of two tree composition operations: substitution and adjunction. Initial trees are trees whose leaves are labelled with substitution nodes (marked with a downarrow) or terminal categories. Auxiliary trees are distinguished by a foot node (marked with a star) whose category must be the same as that of the root node. Substitution inserts a tree onto a substitution node of some other tree while adjunction inserts an auxiliary tree into a tree. In an FTAG, the tree nodes are furthermore decorated with two feature structures (called TOP and BOTTOM) which are unified during derivation as follows. On substitution, the top of the substitution node is unified with the top of the root node of the tree being substituted in. On adjunction, the top of the root of the auxiliary tree is unified with the top of the node where adjunction takes place; and the bottom features of the foot node are unified with the bottom features of this node. At the end of a derivation, the top and bottom of all nodes in the derived tree are unified.

To associate semantic representations with natural language expressions, the FTAG is modified as proposed in (Gardent and Kallmeyer, 2003). Each

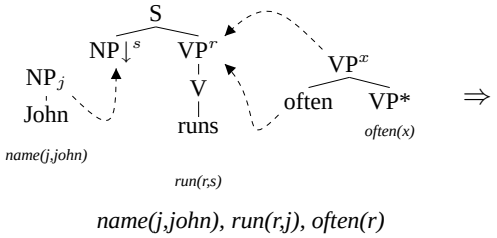

Figure 1: Flat Semantics for "John often runs"

elementary tree is associated with a flat semantic

representation. For instance, in Figure 1,[3] the trees for *John, runs* and *often* are associated with the semantics *name(j,john), run(r,s)* and *often(x)* respectively.

The arguments of a semantic functor are represented by unification variables which occur both in the semantic representation of this functor and on some nodes of the associated syntactic tree. For instance in Figure 1, the semantic index s occurring in the semantic representation of *runs* also occurs on the subject substitution node of the associated elementary tree.

The value of semantic arguments is then determined by the unifications taking place during derivation. For instance, the semantic index s in the tree for *runs* is unified during substitution with the semantic indices labelling the root nodes of the tree for *John*. As a result, the semantics of *John often runs* is

(3) {name(j,john),run(r,j),often(r)}

SEMFRAG describes a core fragment of French and contains around 6 000 elementary trees. It covers some 35 basic verbal subcategorisation frames and for each of these frames, the set of argument redistributions (active, passive, middle, neuter, reflexivisation, impersonal, passive impersonal) and of argument realisations (cliticisation, extraction, omission, permutations, etc.) possible for this frame. Predicative (adjectival, nominal and prepositional) and light verb constructions are also covered as well as subcategorising nouns and adjectives. Basic descriptions are provided for the remaining constructions i.e., adverbs, determiners and prepositions.

4 Implementing SEMFRAG using XMG

We now illustrate the power of XMG by showing how it can be used to produce a highly factorised specification of SEMFRAG, an FTAG of 6 000 trees enriched with a unification based compositional semantics. Given the space constraints, we concentrate on the verbal trees (trees anchored by verbs). We start (4.1) by summarising the specification of SEMFRAG verbal syntactic trees given in (Crabbé, 2005). We then (4.2) show how this specification of the syntax of verbal trees can be enriched with a unification based compositional semantics. We show in particular that this enrichment can be performed using only a limited set of general principles.

4.1 Syntax

The syntactic dimension of SEMFRAG was specified in (Crabbé, 2005). For the verbal trees, it can be summarised as follows.

First, tree families are specified as disjunctions of diatheses. For instance, the N0VN1 family[4] is defined as :

$$\begin{aligned} \textit{n0Vn1} \rightarrow\ & (\ \textit{dian0Vn1Active} \\ & \vee\ \textit{dian0Vn1Passive} \\ & \vee\ \textit{dian0Vn1dePassive} \\ & \vee\ \textit{dian0Vn1ShortPassive} \\ & \vee\ \textit{dian0Vn1ImpersonalPassive} \\ & \vee\ \textit{dian0Vn1middle} \\ & \vee\ \textit{dian0Vn1Reflexive}\ ) \end{aligned} \quad (1)$$

Second, diatheses are defined as conjunctions of classes. For instance, *dian0Vn1Active* is defined as:

$$\begin{aligned} \textit{dian0Vn1Active} \rightarrow\ & (\ Subject \\ & \wedge\ ActiveVerbForm \\ & \wedge\ Object\ ) \end{aligned} \quad (2)$$

Third, each grammatical function appearing in the definition of a diathesis is defined as a disjunction of classes, each class representing a possible realisation of that function. For instance, the *Subject* class is:

$$\begin{aligned} Subject \rightarrow\ & (\ CanonicalSubject \\ & \vee\ RelativisedSubject \\ & \vee\ WhSubject \\ & \vee\ CleftSubject \\ & \vee\ CliticSubject\ ) \end{aligned} \quad (3)$$

Fourth, each class describing a possible grammatical function realisation specifies the adequate tree description. For instance, the fragments for *CanonicalSubject, ActiveVerbForm* and *CanonicalObject* are sketched in Figure 2[5].

In sum, the XMG specification relies on a fairly intuitive use of classes disjunctions and conjunctions. Moreover, the basic leaf classes (i.e., the most deeply embedded disjuncts and conjuncts in the grammar specification) are defined by inheritance, the inheritance hierarchy encoding the sharing of tree description fragments and/or feature

[3] C^x/C_x abbreviate a node with category C and a top/bottom feature structure including the feature-value pair { **index :** x}.

[4] In TAG, a tree family gathers all the elementary trees associated with verbs of a given syntactic type. Thus, the N0VN1 family contains all the trees describing the syntactic contexts in which a verb taking two nominal arguments (i.e., a transitive verb) can occur.

[5] Due to space constraints, these fragments are simplified in that features are omitted.

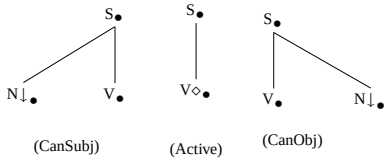

Figure 2: Tree fragments

structures between leaf classes. As a result, several thousand trees are specified using only a few hundred classes.

4.2 Semantics

Just like grammar engineering is a complex issue, enriching a computational grammar with a compositional semantics is potentially time consuming and error prone. We now show that XMG permits this enrichment by means of a few general semantic principles thus minimising both work and the risk of errors. To enrich a purely syntactic FTAG with the type of unification based semantics described in section 3, three main changes need to be carried out.

First, trees must be labelled with appropriate semantic indices and labels. For instance, the subject node of a verbal tree must be labelled with a semantic index.

Second, trees must be associated with appropriate semantic schemas. For instance, the trees of the *n0Vn1* family must be associated with a semantic schema representing a binary relation.

Third, variable sharing between semantic schemas and syntactic trees must be enforced. For instance, the semantic index of the subject node of an active verb should be identified with the first semantic argument of the associated semantic schema.

We now provide an XMG encoding of this information. As for the syntax, we proceed top-down from the verb families down to argument realisation and node labelling.

4.2.1 Associating trees with semantic formulae.

As indicated in the previous section, trees in TAG are grouped into tree families. We use this feature to associate in one fell swoop all the trees of a given family with the appropriate semantic schema. For instance, to associate transitive verbs with a binary relation schema, the syntactic specification given in (1) is modified to:

$$\begin{array}{lcll} \textit{n0Vn1} & \rightarrow & \textit{binaryRel} \ \wedge & (4) \\ & & (\ \textit{dian0Vn1Active} \\ & & \vee\ \textit{dian0Vn1Passive} \\ & & \vee\ \textit{dian0Vn1dePassive} \\ & & \vee\ \textit{dian0Vn1ShortPassive} \\ & & \vee\ \textit{dian0Vn1ImpersonalPassive} \\ & & \vee\ \textit{dian0Vn1middle} \\ & & \vee\ \textit{dian0Vn1Reflexive}\ ) \end{array}$$

4.2.2 Linking constraints

Next the correct syntax/semantic interface constraints must be specified for each diathesis. That is, the correct mapping between syntactic and semantic arguments must be enforced. This is done in two steps.

First, we define a set of INTERFACE constraints of the form

$$index_F = V, arg_i = V$$

which are meant to enforce the identification of the semantic index ($index_F$) labelling a given tree node with grammatical function F (e.g., F = subject) with the index (arg_i) representing the i-th argument in a semantic schema. For instance, when combined with a class C containing a variable X named[6] arg_1 and a variable Y named $index_{subject}$, the *SubjArg1* linking constraint

$$index_{subject} = V, arg_1 = V$$

ensures that X and Y are identified. Assuming further that arg_1 is used to name the first semantic argument and $index_{subject}$ to name the value of the index feature labelling a subject node[7], this constraint ensures a subject/arg$_1$ mapping.

Given such interface constraints, we then refine the diathesis definitions so as to ensure the correct bindings. For instance, the specification in (2) is modified to :

$$\begin{array}{lcll} \textit{dian0Vn1Active} & \rightarrow & (\ SubjArg1 & (5) \\ & & \wedge\ ObjArg2 \\ & & \wedge\ Subject \\ & & \wedge\ ActiveVerbForm \\ & & \wedge\ Object\ ) \end{array}$$

whilst the passive diathesis is specified as:

$$\begin{array}{lcll} \textit{dian0Vn1Passive} & \rightarrow & (\ SubjArg2 & (6) \\ & & \wedge\ CagentArg1 \\ & & \wedge\ Subject \\ & & \wedge\ PassiveVerbForm \\ & & \wedge\ Cagent\ ) \end{array}$$

[6] As explained in section 2, interface constraints can be used to assign global names to values inside a class.

[7] We will see in the next section how to ensure the appropriate naming of syntactic indices and semantic arguments.

4.2.3 Labelling trees with semantic indices.

The above scheme relies on the assumption that tree nodes are appropriately labelled with semantic indices (e.g., the subject node must be labelled with a semantic index) and that these indices are appropriately named (arg_1 must denote the parameter representing the first argument of a binary relation and $index_{subject}$ the value of the index feature on a subject node). As suggested in (Gardent, 2007), a complete semantic labelling of a TAG with the semantic features necessary to enrich this TAG with the unification based compositional semantics sketched in section 3 can be obtained by applying the following *labelling principles*[8]:

Argument labelling: In trees associated with semantic functors, each argument node is labelled with a semantic index[9] named after the grammatical function of the argument node (e.g., $index_{subject}$ for a subject node).

Anchor projection: The anchor node projects its label up to its maximal projection.

Foot projection: A foot node projects its label up to the root[10]

Controller/Controllee: In trees associated with control verbs, the semantic index of the controller is identified with the value of the controlled index occuring on the sentential argument node.

As we shall now see, XMG permits a fairly direct encoding of these principles.

Argument labelling. In the tree associated with a syntactic functor (e.g., a verb), each tree node representing a syntactic argument (e.g., the subject node) should be labelled with a semantic index named after the grammatical function of that node (e.g., $index_{subject}$).

To label argument nodes with an appropriately named semantic index, we first define a set of classes encapsulating a node with an index and a name. We then identify this node with the appropriate tree nodes.

More specifically, we define for each grammatical function $Function \in \{subject, object, cagent, iobject, \dots\}$, a semantic class $FunctionSem$ which associates with an (exported) node called $xFunction$ the feature value pair *index* = I and an interface constraint of the form $index_{Function}$ = I. For instance, the class $SubjectSem$ associates the node $xSubject$ with the feature value pair *index* = I and the interface constraint $index_{subject}$ = I.

$$\begin{array}{lll} subjectSem & \rightarrow & [syn] : xSubject[index = I] \\ & & [interface] : [index_{subject} = I] \end{array} \quad (7)$$

When specifying the tree fragments describing the possible realisations of the grammatical functions, the (exported) argument node is systematically named $xArg$.

Finally, we modify the specification of the grammatical functions realisations to import the appropriate semantic class and identify $xArg$ and $xFunction$ nodes. For instance, 3 above is changed to:

$$\begin{array}{lll} Subject & \rightarrow & SubjectSem \ \wedge \\ & & xArg = xSubject \ \wedge \\ & & (\ CanonicalSubject \\ & & \vee \ RelativisedSubject \\ & & \vee \ WhSubject \\ & & \vee \ CleftSubject \\ & & \vee \ CliticSubject \ ) \end{array} \quad (8)$$

As a result, all $xArg$ nodes in the tree descriptions associated with a subject realisation are labelled with an index I feature whose global name is $index_{subject}$.

Controller/Controllee. Value sharing between the semantic index of the controller (e.g., the subject of the control verb) and that of the controllee (e.g., the empty subject of the infinitival complement) is enforced using linking constraints between the semantic index labelling the controller node and that labelling the sentential argument node of the control verb. Control verb definitions then import the appropriate (object or subject control) linking constraint.

Anchor and foot projection. The anchor (foot) projection principle stipulate the projection of semantic indices from the anchor (foot) node up to the maximal projection (root). To enforce these principles, we define a set of anchor projection classes as illustrated in Figure 3. We then "glue" these projection skeletons onto the relevant syntactic trees by importing them in their definition and explicitely identifying the anchor node of the semantic projection classes with the anchor or foot node of these trees. Since the solutions must be trees, the nodes dominating the anchor node of the

[8] Because of space constraints, the principles required to handle quantification are omitted.

[9] For simplicity, we only talk about indices here. However, to be complete, labels are also need to be taken into account.

[10] The foot projection principle only applies to foot nodes that are not argument nodes i.e., to modifiee nodes.

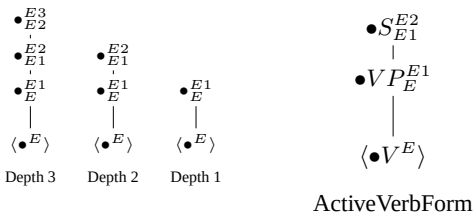

Figure 3: Anchor/Foot projection

projection class will deterministically be unified with those dominating the anchor or foot node of the trees being combined with. For instance, for verbs, the class specifying the verbal spine (e.g., *ActiveVerbForm*, cf. 2) will import a projection class and equate the anchor node of the verbal spine with that of the projection skeleton. As a result, the verb projects its index up to the root (modulo the renaming made necessary by the possibility of an adjunction) as illustrated on the right inside of Figure 3.

4.3 Discussion

Arguably, the XMG encoding we provided to enrich an FTAG with a unification based compositional semantics, is compact and principled.

It is principled in that it provides a direct and transparent formulation of the main principles underlying the integration in a TAG of the unification based semantics sketched in Section 3.

It is compact in that the number of modifications needed to enrich syntax with semantics is relatively small: 76 class definitions and 498 class calls are sufficient to associate the required semantic information (semantic schemas, semantic indices and index projections) with roughly 6000 trees. Crucially, this means that the time involved in integrating the semantics in the grammar is small (roughly a week linguist work) and further that the maintenance, extension and debugging of the semantic component is greatly facilitated.

Both these points rely on the expressivity of XMG. More in particular, the encoding heavily relies on two specific features of XMG namely, *generalised classes disjunctions* and the possibility to use *global names* not only for tree nodes but also for feature values and semantic parameters.

Generalised classes disjunctions are used to associate large sets of trees with semantic schema (section 4.2.1) and to label sets of tree fragments with the appropriately named index (section 4.2.3). Intuitively, generalised classes disjunction and conjunction permit factoring out the common operand of an enumeration (e.g., instead of enumerating $(a \wedge b) \vee (a \wedge c) \vee (a \wedge d) \vee \ldots$, we can specify $a \wedge (b \vee c \vee d)$). In practice, this means that the number of statements necessary to specify the grammar can be greatly reduced. For instance, the association of several thousands of verb trees with a semantic schema is enforced by a total of 176 statements. In contrast, standard linguistic formalisms such as PATR IIor the LKB only allow disjunctions over atomic feature values.

Global names in turn, were used to support a direct encoding of linking constraints (section 4.2.2). 37 linking constraints definitions and 255 linking constraints calls are sufficient to ensure the appropriate mapping between syntactic and semantic arguments in verb trees as well as adjectival and nominal predicative trees. More generally, the possibility to introduce global names not only for tree nodes as in e.g., (Vijay-Shanker and Schabes, 1992) but also for feature values and semantic parameters allows for a simple and direct encoding of constraints applying to identifiers occuring "far apart" in a given tree (for instance, between the index of the subject node in a controlverb tree and that of a PRO index of its infinitival argument).

5 Conclusion

Whilst the development of standard unification-based grammars is well supported by the design of formalisms such as PATR II, the XLE and the LKB, formalisms for developing Tree-Based Grammars have received less attention. XMG aims to remedy this shortcoming by providing a formalism that supports talking about trees, tree sharing and tree labelling.

Trees of arbitrary (finite) depth can be described using underspecified tree descriptions. Additionally, trees can be combined with further linguistic dimensions such as semantic representations and a syntax/semantic interface to form more complex linguistic units.

Tree sharing is supported by the inheritance, the conjunction and the disjunction of tree descriptions together with a sophisticated identifier handling mechanism : identifiers are by default local but can be made global or semi-global, on demand. Furthermore, identifiers can be identified either explicitely (using either the interface or explicit identification constraints) or implicitely (through in-

heritance or through the use of colours, a mechanism not discussed here).

Finally, tree labelling can be expressed by associating tree nodes with one or two (for TAG) feature structures. Importantly, feature values can be assigned global names thereby allowing for the specification of constraints on features that are "far apart from each other" within a tree.

In this paper, we have argued that these features of XMG are effective in supporting an encoding of an FTAG with a unification based compositional semantics which is principled, transparent and compact. These features also markedly distinguish XMG from existing formalisms used to encode tree based grammars such as the non-monotonic encoding of TAG proposed in (Evans et al., 1995) (in contrast, XMG is fully monotonic) and the tree descriptions based approaches proposed in (Candito, 1996; Xia et al., 1998) where in particular, tree descriptions can only be conjoined (not disjoined) and where identification across tree fragments is restricted to nodes.

More in general, we believe that expressive formalisms are necessary to allow not only for the quick development of symbolic tree based grammars but also for their comparison and for the factoring of several grammars be they different wrt to the language they handle (as for instance in the HPSG Delphin or in the LFG Pargram project) or in the semantics they integrate e.g., a glue semantics as proposed in (Frank and van Genabith, 2001), a lambda-based semantics as proposed in (Gardent, 2007) or as shown here, a unification based flat semantics.

References

M.H. Candito. 1996. A principle-based hierarchical representation of LTAGs. In *Proc. of COLING'96, Kopenhagen*.

A. Copestake, A. Lascarides, and D. Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proc. of ACL*, Toulouse, France.

B. Crabbé. 2005. *Représentation informatique de grammaires fortement lexicalisées*. Ph.D. thesis, Université Henri Poincaré, Nancy.

D. Duchier, J. Le Roux, and Y. Parmentier. 2004. The metagrammar compiler. In *2nde confrence internationale Oz/Mozart (MOZ'2004)*, Charleroi.

R. Evans, G. Gazdar, and D. Weir. 1995. Encoding lexicalized tree adjoining grammars with a nonmonotonic inheritance hierarchy. In *Proc. of ACL*.

A. Frank and J. van Genabith. 2001. Ll-based semantics for ltag - and what it teaches us about lfg and ltag. In *Proc. of the LFG'01 Conference*, Hong Kong. CSLI Online Publications.

C. Gardent and L. Kallmeyer. 2003. Semantic construction in ftag. In *Proc. of EACL*, Budapest, Hungary.

C. Gardent and Y. Parmentier. 2006. Coreference handling in xmg. In *Proc. of COLING (Poster)*, Sydney, Australia.

C. Gardent. 2007. Tree adjoining grammar, semantic calculi and labelling invariant. In *Proc. of IWCS*.

Rebecca Nesson and Stuart M. Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*, Malaga, Spain, 29–30 July.

K. Vijay-Shanker and A. K. Joshi. 1988. Feature structures based tree adjoining grammar. In *Proc. of COLING*, pages 714–719, Budapest.

K. Vijay-Shanker and Y. Schabes. 1992. Structure sharing in lexicalised tree adjoining grammar. In *Proc. of COLING 92*, pages 205–211.

F. Xia, M. Palmer, K. Vijay-Shanker, and J. Rosenzweig. 1998. Consistent grammar development using partial-tree descriptions for lexicalized tree adjoining grammar. *Proc. of TAG+4*.