A New Metagrammar Compiler

Bertrand Gaiffe, Benoît Crabbé, Azim Roussanaly

A New Metagrammar Compiler

B. Gaiffe, B. Crabbé, and A. Roussanaly
Loria

1. Why a new metagrammar compiler ?

Writing a TAG grammar manually is known to be a non trivial task (Abeillé 00; Doran et al. 94; Doran et al. 00; VS et al. 92). For that purpose, (Candito 96) has suggested a grammatical framework that allows linguists to describe the syntactic properties of a language at a higher level of abstraction . Given a set of classes, each of these containing a partial tree description, the compiler outputs a set of tree schemata.

According to us, however, the resulting tool suffers two main drawbacks
(Candito 99): (1) the algorithm is closely linked to the related linguistic description. As her analysis put the focus on the study of verbal trees, the structuration is badly adapted for the description of non verbal units ; (2) the current implementation of the compiler is not flexible enough to be easily adapted to other input/output formats.

We therefore propose an enhanced version of such a compiler taking into account our preliminary remarks.

2. Design of our tool

In this section, we present the characteristics of the tool we implemented. Our compiler however ressembles the one described in [Candito 96] and [Candito 99]. We thus present this latter first in order to emphasize only the differences between her proposition and ours.

2.1. Topological factorisation

In [VS and Shabbes 92], the authors propose to use a logic [ref. à rechercher] to describe elementary trees of a TAG grammar so that topological informations shared by trees are factorized in an inheritance hierarchy. Figure 1 illustrate a toy hierarchy that corresponds to the four following trees :

(walks, runs, eats, loves).

In pratice however, informations concerning trees may be factorized according to different point of view quite independant from each other. For instance, subcategorization informations lead to a nice and rather natural herarchy and almost independantly, realisations of syntactic functions lead to another hierarchy. Therefore, attempts to describing a unique hierarchy (almost) leads to repeating a whole hierarchy a the leaves of another (in case of an inheritance tree) or to manually explicit a lot of crossings at the root of hierarchies in case of multiple-inheritance graphs.

2.2. Marie-Hélène Candito's Compiler

[Candito 96] and [Candito 99] precisely explains the preceding point and advocates for three independant hierarchies linguistically motivated which she calls dimensions:

1. subcategorisation (which she represents in terms of initial syntactic functions)
2. syntactic function redistributions (which leads to final syntactic functions)
3. final functions realisations

The underlining idea is that a linguist describes only each these three hierarchies and an automatic tool completes the inheritance graph by crossing final classes of dimension 1 with final classes of dimension 2 and crosses the resulting crossed classes with final classes of dimension 3.

Not all final classes of dimension 1 are to be crossed with all final classes of dimension 2 however. For instance, untransitive verbs do not admit passive constructions. In the resulting crossed classes of dimensions 1

 Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pp. 1–1. Universitá di Venezia.

and 2 only actual syntactic functions mentionned in the crossed class have to be realized. The linguist has thus to give crossing conditions together with his hierarchies in order to constrain teh crossing process.

The algorithm implemented by Candito is thus the following:

```
dim12 = empty set
for each final class c1 of dimension 1
   for each final class c2 of dimension 2 (compatible with c1)
create c12 that inherits c1 and c2 and add it to dim12
   end for
end for
res = empty set
for each c12 in dim12
   resOfC12 = {c12}
   for each final function ff appearing in c12
      for each class c3 of dimension 3 that realizes ff
         build new classes with each element of resOfC12 and c3
 resOfC12 = these new classes
      end for
   end for
   res = res U resOfC12
end for
compute minimal referents for each element of res.
```

As this pseudo-algorithm makes clear, some constants that denote tree nodes are labelled by a final syntactic function. They are actually also labelled by an initial syntactic function[1].In dimension 2, most of the job done by classes consists in modifing the final functions[2]. Moreover, final functions are maintained unique in any description, that is constants bearing the same final function are considered equal.

2.3. Our proposition

Our initial motivation for developping a new meta-grammar compiler had to do with the lexicon: the grammar compiled with Candito's tool is organised in tree families (as is XTAG [ref]) and lemmas are associated to families. The anchoring of a tree thus consists in computing the lemma and the morpho syntactic features associated to a word form, get the families associated to the lemma and finaly attempt to substitute the lemma with the associated morpho syntactic features in the tree.

Such a process may of course fail, either because the morpho-syntactic features do not match (suppose for instance a tree dedicated to an imperative form, together with an infinitive word form) , or because the features associated to the lemma do not match (some transitive verbs for instance do not accept a passive form).

Our initial motivation was then to generate trees together with a feature structure that globaly describes the tree, and Candito's tool did not seem to permit that.

As we had to implement a new tool, we gave ourselves some additional constraints:

- avoiding non monotonous mechanisms such as the modification of the final functions ;
- not limiting *a priori* the number of dimensions. We did not doubt of the quality of the linguistic study, but:
  - the third dimension is *de facto* a collection of dimensions dedicated to realizing each of the possible functions;
  - three dimensions is perhaps not a good choice for other categories than verbs, or for other languages than french, english or italian.

Provided we intend to produce trees together with a feature structure, classes of the meta-grammar contain such a feature structure that describes its content. It then seems natural that these features structure get combined throug unification along the inhéritance lattice (remember that crossing of classes amounts to building new classes by inheritance). This feature structure is then a good mecanism to avoid unwanted class crossings. The exemple

1. Probably in order to keep track of the process.
2. fi nal functions are initialized to the initial function

we mention of untransitive verbs taht do not accepts passive forms is simply taken into account by means of an attribute **transitive** with values **minus** for an intransitive verb and **plus** for all passive classes.

Feature structures in the classes allow us to avoid unwanted crossing, the remaining problem is to find out a mecanism so that classes cross. In candito's compiler, this mecanism relies on a fixed number of dimensions, but we want to avoid dimensions.

We thus decided to make explicit the reason why classes are to be crossed, typically a class of ex-dimension 1 has to be crossed with a class of dimension 2 because it needs redistribution of syntactic functions. Classes of ex-dimension 2 have to be crossed with classes of ex-dimension 3 because they need that a subject, an object or whatever other final function be realized. Conversely, a class of ex-dimension 3 may **provide** the realization of a subject, or an object, or whatever other function.

A class in our system is then described by:

- a name
- a set of super-classes
- a description (which is a feature structure)
- a set of needs
- a set of providings
- a formula describing trees

When a class c12 inherits two classes c1 and c2, the descriptions are unified (in case of failure, c12 is not created), the set of needs is the union of the two set of needs minus the union of the providings, the set of providings is also the union of the providings minus the union of the needs and the formula is the conjunction of the two formulas.

The crossing mechanism then consists in computing all balanced final classes, that is classes whose set of needs and providings are empty[3].

Finally, the formulas corresponding to balanced final classes are used to compute minimal referents [VS Rodgers 94] together with teh description associated with the corresponding class.

3. An overview of the linguistic application

3.1. Generating the trees

We made some experiments on french verbs, according as much as possible to the analysis of (Abeillé 91; Candito 99). Therefore, we give an overview of the way we describe verbs using three 'dimensions' using a functional approach.[4].

Following (Abeillé 91; Candito 99) each tree which belongs to a family represents a predicative structure. The first dimension is dedicated to map initial functions on the predicate's argument structure. Here we define a set of classes which represents the arguments, another set which defines the functions that are to be mapped on them. The final classes of this dimension are a list of all the valid mappings in french. For instance the final class *Subj0VObj1* is the class where the first argument is mapped with a subject and the second is mapped with a direct object[5].

This mapping is expressed with the declaration of a pseudo-node for each argument. Each of these pseudo-nodes is said to be equal (or coreferent) with the one representing its associated function.

To be linguistically well formed, we impose that in a neutralized class each initial function has to be mapped into a final function. Therefore a class where an initial function node is defined emits the need to have a final function.

3. In order to keep with an associative and commutative mechanism, an annulated need as well as an annulated providing is not allowed to appear again
4. Formally speaking, it should be clear that there are no dimensions anymore.
5. For the sake of simplicity, we do not expand here to the whole Abeille's analysis. We do not consider here sentential arguments and verbal auxiliaries which are important for the defi nitions of the families.

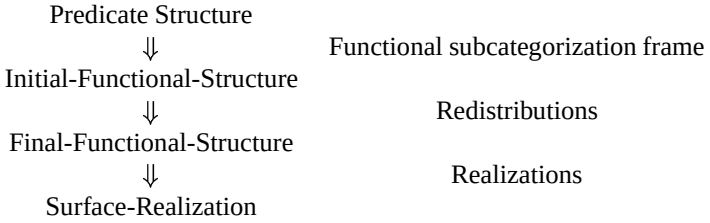
The second step in the generating process defines a final subcategorization frame. Here we map the initial functions given above with final functions. As a side effect, the verb is given a morphological specification. For instance the passive will map the initial-subject to a by-object and the initial object to a final-subject. Additionnally, the passive class requests the verbal morphology to be passive.

We express, in this second dimension, the mapping in the same way as we did before. That is, we define a set of classes where pseudo-nodes with the final-functions are declared. The final classes express equalities between the nodes carrying the initial functions and the ones carying the final functions. The interface with the first dimension is done through classes that provide the related needs. For instance the *Full-personal-passive* class will inherit from classes that provide the initial-subject and the initial-object. The content of the class reflects the mapping : the initial-subject node equals the final-by-object node and the initial-object node equals the final subject node.

Each dimension 2 class that contains a final-function nodes emit a need to be realized. The classes that satisfy these needs are located in the third dimension. The full personnal passive class inherits from the following needs : realization by-object, realization subject and realization passive morphology.

3.2. Feature Structures

4. The quest for monotonicity



Given the initial functions, the second dimension classes are designed to bind these nodes with those that represent the final syntactic functions. As a side effect, the dimension 2 classes also contribute to build the verbal spine of the trees. For instance the dimension 2 class *initialSubject becomes parObj* contains the equation: $FinalparObjectNode = InitialSubjectNode$. A typical final class of dimension 2 is *Full-Pers-Passive*.

At this step, each final function emits the need to be realized. Thus, a class resulting from the crossing process inherits from as many dimension 3 classes as it contains final functions.

Then comes the third dimension where the trees are drawn by both inheriting from a final function realization and a syntactic construction pattern. Sample classes are *Cleft-Subject*, *Questioned-Object*...

The final version of the paper should include a comprehensive example of the generation process of a significant tree sketch (Fig. 1) : *Par qui sera accompagnée Marie ?* (*By whom will be accompanied Mary ?*).

References

Abeillé, A., *Une grammaire lexicalisée d'arbres adjoints pour le français. Application à l'analyse automatique*, Doctoral dissertation, Université de Paris 7, 1991.

Abeillé, A, Candito, M.-H., "FTAG : A Lexicalized Tree Adjoining Grammar for French", *in* Abeillé, A. and Rambow, O. éd. *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing*, Stanford, CSLI, 2000.

Candito, M.-H., Candito, "A Principle Based Hierarchical Representation of LTAGs", *COLING*, 1996.

Candito, M.-H.,*Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien*, Doctoral dissertation, Université de Paris 7, 1999.

Doran, C, Egedi, D, Hockey, A., Srinivas, B., Zaidel, M., "XTAG System - A Wide Coverage Grammar for English", *COLING*, 1994.

Doran, C, Sarkar, A., Srinivas, B., Xia, F., "Evolution of the XTAG System", *in* Abeillé, A. and Rambow, O. éd. *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing*, Stanford, CSLI, 2000.

Joshi, A. K., Levy, L. S., Takahashi, M., "Tree Adjunct Grammars", *Journal of Computer Science*, 1975.

Kinyon, A., "Hypertags", *COLING*, 2000.

Lopez, P., Bonhomme, P., "Resources for Lexicalized Tree Adjoining Grammars and XML encoding : TagML", *LREC*, 2000.

Rogers, J., Vijay-Shanker, K., "Obtaining Trees from Their Descriptions : An Application to Tree-Adjoining Grammars", *Computational Intelligence*, 10, 4, 1994.

Vijay-Shanker, K., Schabes, Y., "Structure Sharing in Lexicalized Tree-Adjoining Grammars", *COLING*, 1992.

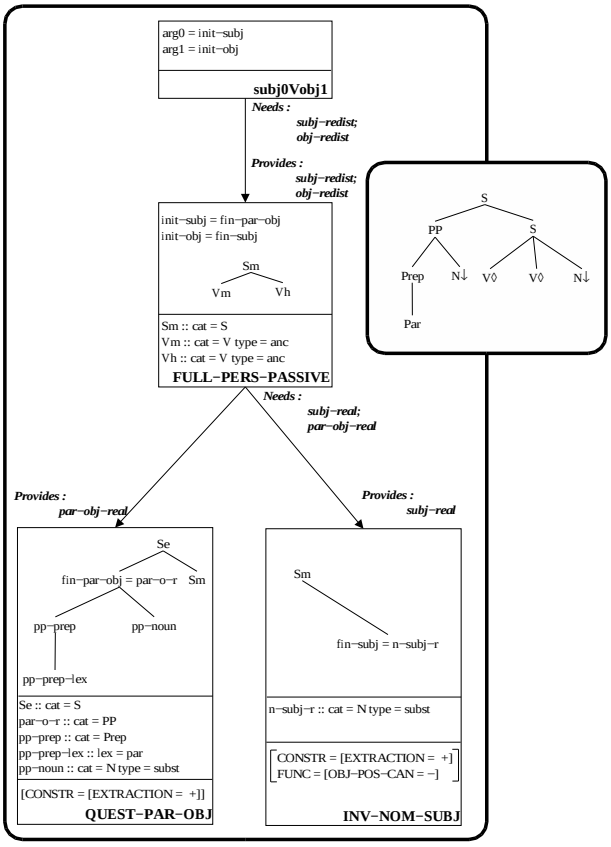


Figure 1: Example of generation