A principle-based hierarchical representation of LTAGs

Marie-Hélène Candito
TALANA
Université Paris 7 Tour centrale 8ème étage pièce 801
75251 Paris cedex 05 FRANCE
e-mail : marie-helene.candito@linguist.jussieu.fr

Abstract

Lexicalized Tree Adjoining Grammars have proved useful for NLP. However, numerous redundancy problems face LTAGs developers, as highlighted by Vijay-Shanker and Schabes (92).

We present a compact hierarchical organization of syntactic descriptions, that is linguistically motivated and a tool that automatically generates the tree families of an LTAG. The tool starts from the syntactic hierarchy and principles of well-formedness and carries out all the relevant combinations of linguistic phenomena.

1 Lexicalized TAGs

Lexicalized Tree Adjoining Grammar (LTAG) is a formalism integrating lexicon and grammar (Joshi, 87; Schabes et al., 88). It has both linguistic advantages (e.g elegant handling of unbounded dependencies and idioms) and computational advantages, particularly due to lexicalization (Schabes et al., 88). Linguists have developed over the years sizeable LTAG grammars, especially for English (XTAG group, 95; Abeillé et al., 90) and French (Abeillé, 91).

In this formalism, the lexical items are associated with the syntactic structures in which they can appear. The structures are lexicalized elementary trees, namely containing at least one lexical node at the frontier (called the anchor of the tree). The elementary tree describes the maximal projection of the anchor. So a verb-anchored tree has a sentential root. Features structures are associated with the trees, that are combined with substitution and adjunction. Adjunction allows the extended domain of locality of the formalism : all trees anchored by a predicate contains nodes for all its arguments.

Such a lexicalized formalism needs a practical organization. LTAGs consist of a morphological lexicon, a syntactic lexicon of lemmas and a set of tree schemata, i.e. trees in which the lexical anchor is missing. In the syntactic lexicon, lemmas select the tree schemata they can anchor. When the grammar is used for parsing for instance, the words of the sentence to be parsed are associated with the relevant tree schemata to form complete lexicalized trees.

The set of tree schemata forms the syntactic part of the grammar. The tree schemata selected by predicative items are grouped into families, and collectively selected. A tree family contains the different possible trees for a given canonical subcategorization (or predicate-argument structure). The arguments are numbered, starting at 0 for the canonical subject. Along with the "canonical" trees, a family contains the ones that would be transformationally related in a movement-base approach. These are first the trees where a "redistribution" of the syntactic function of the arguments has occurred, for instance the passive trees, or middle (for French) or dative shift (for English), leading to an "actual subcategorization" different from the canonical one. When such a redistribution occurs, the syntactic function of the arguments change (or the argument may not be realized anymore, as in the agentless passive). For instance, the subject of a passive tree is number 1, and not 0 (figure 1). This is useful from a semantic point of view, in the case of selectional restrictions attached to the lexical items, or of a syntactic/semantic interface.

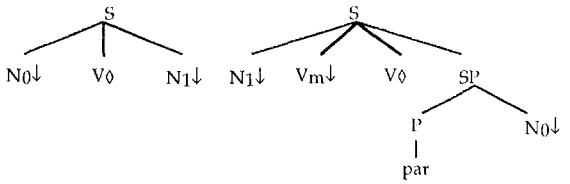

Figure 1. Declarative transitive tree and corresponding full passive for French[1]

And secondly, a family may contain the trees with extracted argument (or cliticized in French). There are different types of trees for extraction. In the English grammar for instance, there are trees for wh-questions and trees for relative clauses (that are adjoined to NPs). In the French grammar there are also separate trees for cleft sentences with gaps in the clause, while the corresponding it-clefts are handled as relative clauses in the English grammar.

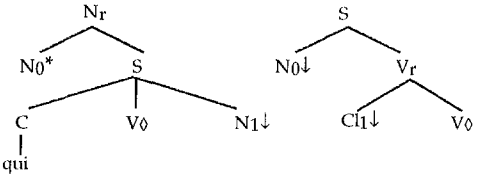

Figure 2. Two trees of the strict transitive family for French : the relativized subject and the cliticized object.

[1]The French LTAG comprises trees with flat structure (no VP node); in the passive tree, the auxiliary is substituted; the same symbol N is used for nominal phrases and nouns, the difference being expressed with a feature <det> (Abeillé, 91). we do not show the feature equations for the sake of clarity. For the French grammar, the average number of equations per tree is 12.

So a family contains all the schemata for a given canonical subcategorization. Yet, in the syntactic lexicon, a particular lemma may select a family only partially. For instance a lemma might select the transitive family, ruling out the passive trees.

On the other hand, the features appearing in the tree schemata are common to every lemma selecting these trees. The idiosyncratic features (attached to the anchor or upper in the tree) are introduced in the syntactic lexicon.

2 Development and maintenance problems with LTAGs

This extreme lexicalization entails that a sizeable LTAG comprises hundreds of elementary trees (over 600 for the cited large grammars). And as highlighted by Vijay-Shanker and Schabes (92), information on syntactic structures and associated features equations is repeated in dozens of tree schemata (hundreds for subject-verb agreement for instance).

This redundancy problem is present at all levels of grammar development. The writing of an LTAG is a rather fastidious task; its extension and/or maintenance is very difficult, since maintaining the grammar means for instance adding an equation to hundreds of trees. Extending it means adding new trees along with their equations, and it can also entail the addition of new features in existing trees. Furthermore, the amount of work may grow exponentially with the size of the grammar, since all combinations of phenomena must be handled.

And finally, in addition to the practical problems of grammar writing, updating and storage, redundancy makes it hard to get a clear vision of the theoretical and practical choices on which the grammar is based.

3 Existing solutions

A few solutions have been proposed for the problems described above. Solutions to the redundancy problem make use of two tools for lexicon representation : inheritance networks and lexical rules. Vijay-Shanker and Schabes (92) have first proposed a scheme for the efficient representation of LTAGs, more precisely of the tree schemata of an LTAG. They have thought of a monotonous inheritance network to represent the elementary trees, using partial descriptions of trees (Rogers and Vijay-Shanker, 92 and 94) (see section 4.1 for further detail). They also propose to use "lexical and syntactic rules" to derive new entries. The core hierarchy should represent the "canonical trees", and the rules derive the ones with redistribution of the functions of arguments (passive, dative shift...) and the ones with extracted argument.

Becker (93; 95) also proposes a hybrid system with the same dichotomy : inheritance network for the dimension of canonical subcategorization frame and "meta-rules" for redistribution or extraction (or both). The language for expressing the meta-rules is very close to the elementary tree language, except that meta-rules use meta-variables standing for subtrees. He proposes to integrate the meta-rules to the XTAG system which would lead to an efficient maintenance and extension tool.

(Evans et al., 95) have proposed to use DATR to represent in a compact and efficient way an LTAG for English, using (default) inheritance (and thus full trees instead of partial descriptions) and lexical rules to link tree structures. They argue the advantage of using already existing software. But some information is not taken into account : the lexical rules do not update argument index. For instance the dative shift rule for English changes the second complement - the PP - into a NP, which is not semantically satisfying. The passive rules simply discards the first complement (representing the canonical direct objet), the other complements moving up. But then the relation between the active object and the passive subject is lost.

The three cited solutions give an efficient representation (without redundancy) of an LTAG, but have in our opinion two major deficiencies.

First these solutions use inheritance networks and lexical rules in a purely technical way. They give no principle about the form of the hierarchy or the lexical rules[2], whereas we believe that addressing the practical problem of redundancy should give the opportunity of formalizing the well-formedness of elementary trees and of tree families.

And second, the *generative* aspect of these solutions is not developed. Certainly the lexical rules are proposed as a tool for generation of new schemata or new classes in a inheritance network. But the automatic triggering, ordering and bounding of the lexical rules is not discussed.

4 Proposed solution : efficient representation and semi-automatic generation

We propose a system for the writing and/or the updating of an LTAG. It comprises a principled and hierarchical representation of lexico-syntactic structures. Using this hierarchy and principles of well-formedness, the tool carries out all the relevant crossings of linguistic phenomena to generate the tree families.

This solution not only addresses the problem of redundancy but also gives a more principle-based representation of an LTAG. The implementation of the principles gives a real generative power to the tool. So in a sense, our work can relate to (Kasper et al., 95) that describes an algorithm to translate a Head-driven Phrase Structure Grammar (HPSG) into an LTAG. The inheritance hierarchy of HPSG and its principles are "flattened" into a lexicalized formalism such as LTAG. The idea is to benefit from a principle-based formalism such as HPSG and from computational properties of an LTAG.

[2]Becker gives a linguistic principle for the bounding of his meta-rules, but he has no solution for the application of this principle.

4.1 Hierarchical representation of an LTAG

4.1.1 Formal choices : a monotonic inheritance network, without meta-rules

Like the solutions described in section 3, our system uses a multiple inheritance network. Yet, it does not use meta-rules. Though they could be a further step of factorization, it seemed interesting to "get the whole picture" of the grammar within the hierarchy, and not only the base trees.

Further, we have chosen monotonic inheritance, especially as far as syntactic descriptions are concerned. Default inheritance does not seem to be justified to represent tree schemata, from the linguistic point of view. Default inheritance is often necessary to deal with exceptions. One may want to express generalizations despite a few more specific exceptions. Now the set of tree schemata we intend to describe hierarchically is empty of lexical idiosyncrasies, which are in the syntactic lexicon (cf. section 1). The set of tree schemata represents syntactic phenomena that are all productive enough to allow monotonicity. This resulting hierarchy will then be more transparent and will benefit from more declarativity.

Technically, monotonicity in syntactic descriptions is allowed by the use of partial descriptions of trees (Rogers and Vijay-Shanker, 92; 94), as was proposed in (Vijay-Shanker and Schabes, 92) (see section 4.1.3).

4.1.2 General organization of the hierarchy

Section 1 briefly described the organization of an LTAG in families of trees. The rules for the organization of a family, its coherence and completeness, are flattened into the different trees. With the approach of an automatic generation of TAG trees, we have found necessary to explicit these rules, which are defined using the notions of argument and syntactic function.

Following a functional approach to subcategorization (see for instance Lexical Functional Grammar, (Bresnan, 82)), we clearly separate the "redistributions" of syntactic functions of the arguments from the different realizations of a given syntactic function (in canonical, extracted, cliticized... position). We intend the term *redistribution* in a broad sense for manipulation of the number and functions of arguments. It includes cases of reduction of arguments (e.g. agentless passive), restructuration (dative-shift for English) or even augmentation of arguments (some causative constructions[3], introducing an agent whose function is subject). Redistribution is represented in our system by pairing arguments and functions, and not in terms of movement.

So the proposed hierarchy of syntactic descriptions (for the family anchored by a verb) comprises the three following dimensions :

dimension 1 : the canonical subcategorization frame
This dimension defines the types of canonical subcategorization. Its classes contain information on the arguments of a predicate, their index, their possible categories and their canonical syntactic function.

dimension 2 : the redistribution of syntactic functions
This dimension defines the types of redistribution of functions (including the case of no redistribution at all). The association of a canonical subcategorization frame and a compatible redistribution gives an actual subcategorization, namely a list of argument-function pairs, that have to be locally realized.

dimension 3 : the syntactic realizations of the functions
It expresses the way the different syntactic functions are positioned at the phrase-structure level (in canonical position or in cliticized or extracted position). This last dimension is itself partitioned according to two parameters : the syntactic function and the syntactic construction.

4.1.3 Monotonic inheritance and partial descriptions of trees

The hierarchy is a strict multiple inheritance network whose terminal classes represent the elementary trees of the LTAG. These terminal classes are not written by hand but automatically generated following principles of well-formedness, either technical or linguistic.

A partial description is a set of constraints that characterizes a set of trees. Adding information to the description reduces monotonically the set of satisfying trees. The partial descriptions of Rogers and Vijay-Shanker (94)[4] use three relations : left-of, parent and dominance (represented with a dashed line). A dominance link can be further specified as a path of length superior or equal to zero. These links are obviously useful to underspecify a relation between two nodes at a general level, that will be specified at an either lower or lateral level. Figure 3 shows a partial description representing a sentence with a nominal subject in canonical position, giving no other information about possible other complements. The link between the S and V nodes is underspecified, allowing either presence or absence of a cliticized complement on the verb. In the case of a clitic, the path between the S and V nodes can be specified with the description of figure 4. Then, if we have the information that the nodes labelled respectively S and V of figures 3 and 4 are the same, the conjunction of the two descriptions is equivalent to the description of figure 5.

[3] We talk about some causative constructions analysed as complex predicates with co-anchors in French as in :
Jean a fait s'assoir les enfants. *Jean made sit the children. (Jean made the children sit)

[4] Vijay-Shanker & Schabes (92) have used the partial descriptions introduced in (Rogers & Vijay-Shanker, 92), but we have used the more recent version of (Rogers & Vijay-Shanker, 94). The difference between the two versions lies principally in the definition of quasi-trees, first seen as partial models of trees and later as distinguished sets of constraints.

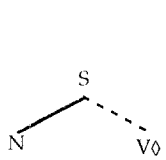

Figure 3

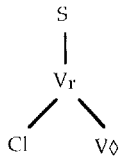

Figure 4

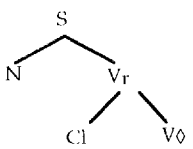

Figure 5

In the hierarchy of syntactic descriptions we propose, the partial description associated with a class is the unification of the own description of the class with all inherited partial descriptions. As shown in the above example, the conjunction of two descriptions may require statements of identity of nodes. Rogers and Vijay-Shanker (94) foresee, in the case of an application to TAG, the systematic identity of lexical anchors. Further, Vijay-Shanker and Schabes (92) make also use of a particular function to state identity of argumental nodes. But this is not enough as one might need to state equality of any type of nodes (like the S nodes in the above example). To achieve this in our system, one simply needs to "name" both nodes in the same way. Remember we talk about descriptions of trees. In these objects, nodes are referred to by constants. Two nodes, in two conjunct descriptions, referred to by the same constant are the same node, and two nodes referred to by different constants can either be equal or different. Equality of nodes can also be inferred, mainly using the fact that a tree node has only one direct parent node.

We have added atomic features associated with each constant, such as category, index, quality (i.e. foot, anchor or substitution node), canonical syntactic function and actual syntactic function. These features belong to the meta-formalism of LTAG hierarchical organization. We will call them meta-features (as opposed to the features attached to the nodes of the TAG trees). In the conjunction of two descriptions, the identification of two nodes known to be the same (either by inference or because they have the same constant) requires the unification of such meta-features. In case of failure, the whole conjunction fails, or rather, leads to an unsatisfiable description.

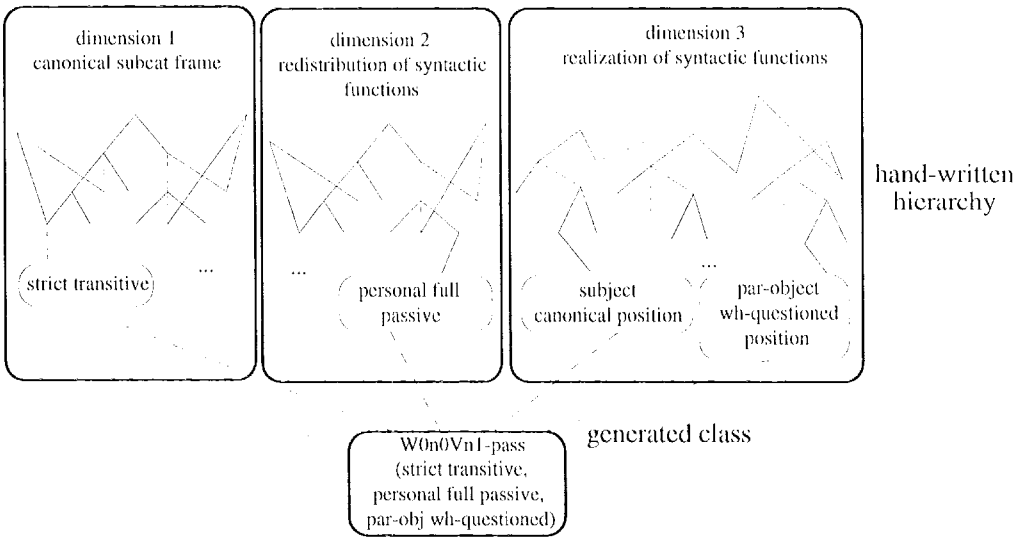

Figure 6. Creation of a terminal class totally defined by its super-classes.

4.2 Automatic generation of elementary trees

The three dimensions introduced in section 4.1.2 constitute the core hierarchy. Out of this syntactic database and following principles of well-formedness the generator creates elementary trees. This is a two-steps process : it first creates some *terminal classes* with inherited properties only - they are totally defined by their list of super-classes. Then it translates these terminal classes into the relevant elementary tree schemata, in the XTAG[5] format, so that they can be used for parsing.

The tree schemata are generated grouped in families. This is simply achieved by fixing a canonical subcat frame (dimension 1), associating with it all relevant redistributions (dimension 2) and relevant realizations of functions (dimension 3). At the development stage, generation can also be done following other criterions. For instance, one can generate all the passive trees, or all trees with extracted complements...

4.2.1 Principles of well-formedness

The generation of elementary trees from more abstract data needs the characterization of what is a well-formed elementary tree in the framework of LTAG. The common factor to various expressions of linguistic principles made for LTAGs is the argument-predicate co-occurrence principle (Kroch and Joshi, 85; Abeillé, 91) : the trees for a predicative item contain positions for all its arguments.

But for a given predicate, we expect the canonical arguments to remain constant through redistribution of functions. The canonical subject

[5]XTAG (Paroubek et al., 92) is a tool for writing and using LTAGs, including among other things a tree editor and a syntactic parser.

(argument 0) in a passive construction, even when unexpressed, is still an argument of the predicate. So the principle should be a principle of predicate-*functions* co-occurrence : the trees for a predicative item contain positions for all the functions of its actual subcategorization. In the solution we propose, this principle is translated as :

1- **subcat principle** : a terminal class must inherit of a canonical subcategorization (dimension 1) and a *compatible* redistribution, including the case of no redistribution at all (dimension 2). This pair of super-classes defines an actual subcategorization.

2- **completeness/coherence/unicity principle** : the terminal class must inherit *exactly one* type of realization for each function of the actual subcategorization[6].

Well-formedness of elementary trees is also expressed through the form of the hierarchy itself (the content of the classes, the inheritance links, the inheritance modes for the different slots...). This information spread into the hierarchy is used for tree generation following *technical* principles of well-formedness. Due to a lack of space we detail only the following principle, useful to understand next section.

3- **unification principle** : the unifications of partial descriptions and meta-equations required by inheritance must succeed; the unification of nodes with same constant is mandatory; moreover two nodes with the same value for the meta-feature "function" must unify.

Figure 6 shows an example of generation of a terminal class, corresponding to the tree, for French, for the full passive of a strict transitive verb, in a wh-question on the agent (see figure 7). It can be illustrated by the sentence :

(Je me demande) par qui Jean sera accompagné.
By whom will Jean be accompanied?

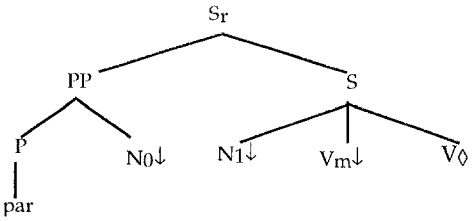

Figure 7. Tree for French, for the full passive of a strict transitive verb, in a wh-question on the agent.

The corresponding terminal class W0n0Vn1-pass inherits the canonical subcat STRICT TRANSITIVE and the redistribution PERSONAL FULL PASSIVE. This defines the following actual subcategorization : arg0/par-object; arg1/subject. Then the terminal class inherits the relevant realization for each of the cited functions (SUBJECT IN CANONICAL POSITION and PAR-OBJ-QUESTIONED).

[6]Following from the functional representation of subcategorization, this principle relates to the principles of well-formedness of functional structures in LFG.

4.2.2 From terminal classes to elementary trees

The terminal classes representing elementary trees inherit a (constructed) partial description of tree, with meta-equations and equations. To get elementary trees from these classes, we need to translate the partial descriptions into trees. This is done by taking the least tree(s) satisfying the description. We do not go into the details for brevity reasons, but intuitively the minimal tree is computed by taking the underspecified links to be path of length zero when their ends are compatible, of length one otherwise (figure 8). A description can leave underspecified the order of some daughters, leading to several minimal trees. Rogers and Vijay-Shanker (94) give a formal mechanism to obtain trees from descriptions.

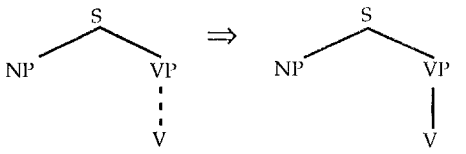

Figure 8. Translating a dashed line into a path of length one.

After obtaining tree(s) from the partial description, the generator translates the node constants into the concatenation of syntactic category and index (if it exists).

4.2.3 A detailed example

Let us go back to the tree of figure 7. The next figure shows in detail the super-classes[7] (introduced at figure 6) for the class W0n0Vn1-pass representing this tree :

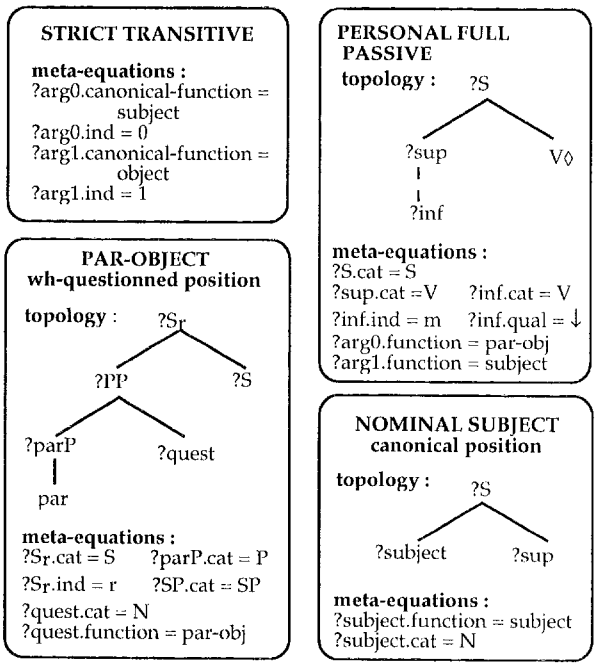

Figure 9. Super-classes of W0n0Vn1-pass.

[7]We only show the direct super-classes. They are given with their specific properties and with their inherited properties as well. The "equations" slot is not shown. In the partial descriptions shown, the constants naming the nodes start with ?.

The conjunction of the inherited partial descriptions leads to the following description :

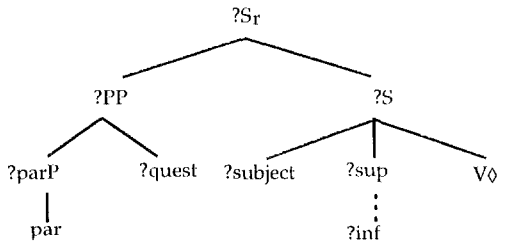

Figure 10. Inherited partial description.

The nodes with same constants have unified (?S/?S) and the constants with same "function" meta-feature have also unified : ?subject/?arg1 and ?quest/?arg0 (cf. principle 3). Then the node constants are translated and the least satisfying tree is computed, leading to the target tree of figure 7.

5 Applications

The tool has been used to update and augment the French LTAG developed at Paris 7. A hierarchy has been written that gives a compact and transparent representation of the verbal families already existing in the grammar. The writing of the hierarchy has been the occasion of updating structures and equations, insuring uniform and coherent handling of phenomena. Furthermore the automatic generation from the hierarchy guarantees the well-formedness of the families, with all possible conjunctions of phenomena. Extra phenomena such as nominal subject inversion, impersonal middle constructions, some causative constructions or free order of complements have been added.

The generative power of the tool is effective : out of about 90 hand-written classes, the tool generates 730 trees for the 17 families for verbs without sentential complements[8], 400 of which were present in the pre-existing grammar. The tool is currently used to add trees for some elliptical coordinations.

We see several possible applications of the tool. We could try to generate a grammar with weaker constraints, useful for corpora with recurrent ill-formed sentences. Secondly, we could obviously use the tool to build a grammar for another language, either from scratch or using the hierarchy designed for French. Using this already existing hierarchy and the implemented principles of well-formedness will lead to a grammar for another language "compatible" with the French grammar. This could be an advantage in the perspective of machine translation for instance.

Because the principles of well-formedness implemented are general and capture mainly the extended domain of locality of LTAG, the generator we have presented can very well be used to generate a grammar with different underlying linguistic choices (for instance the GB perspective used in the English grammar cited).

[8] By the time of conference, we will be able to give figures for the families with sentential complements also.

6 Conclusion

We have presented a hierarchical and principle-based representation of syntactic information. It insures transparency and coherence in syntactic descriptions and allows the generation of the elementary trees of an LTAG, with systematic crossing of linguistic phenomena.

7 References

A. Abeillé, K. Bishop, Sharon Cote and Y. Schabes. 1990. A lexicalized Tree Adjoining Grammar for English. Technical Report, University of Pennsylvania.

A. Abeillé. 1991. Une grammaire lexicalisée d'Arbres Adjoints pour le français, PhD thesis, University Paris 7.

T. Becker. 1993. HyTAG : a new type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Order Languages, PhD thesis, University of Saarbrücken.

T. Becker. 1994. Patterns in Metarules. Proceedings of the third International Workshop on Tree Adjoining Grammars (TAG+3), Paris.

C. Doran, D. Egedi, B.A. Hockey, B. Srinivas and M. Zaidel. 1994. XTAG System - A wide Coverage Grammar for English. Proceedings of COLING'94, Kyoto.

R. Evans, G. Gazdar and D. Weir. 1995. Encoding Lexicalized Tree Adjoining Grammar with a Nonmonotonic Inheritance Hierarchy. Proceedings of ACL'95, Boston.

A. Joshi. 1987. Introduction to Tree Adjoining Grammar, in A. Manaster Ramer (ed), *The Mathematics of Language*, J. Benjamins, pp. 87-114.

R. Kasper, B. Kiefer, K. Netter and K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. Proceedings of ACL'95, Boston.

A. Kroch and A. Joshi. 1985. The linguistic relevance of Tree Adjoining Grammars. Technical report, University of Pennsylvania.

P. Paroubek, Y. Schabes and A. Joshi. 1992. XTAG - A graphical Workbench for developing Tree Adjoining Grammars. Proceedings of 3-ANLP, Trento.

J. Rogers and K. Vijay-Shanker. 1992. Reasoning with descriptions of trees. Proceedings ACL'92, pp. 72-80.

J. Rogers and K. Vijay-Shanker. 1994. Obtaining trees from their descriptions : an application to Tree-Adjoining Grammars. Computational Intelligence, vol. 10, N° 4, pp. 401-421.

Y. Schabes, A. Abeillé and A. Joshi. 1988. Parsing strategies with lexicalized grammars : Tree Adjoining Grammars. Proceedings of COLING'88, Budapest, vol. 2, pp. 578-583.

K. Vijay-Shanker and Y. Schabes. 1992. Structure Sharing in Lexicalized Tree Adjoining Grammar. Proceedings of COLING'92, Nantes, pp. 205-211.

XTAG research group. 1995. A lexicalized Tree Adjoining Grammar for English, Technical Report IRCS 95-03, University of Pennsylvania.