Generating the XTAG English grammar using metarules

Carlos A. Prolo
Computer and Information Science Department
University of Pennsylvania
Suite 400A, 3401 Walnut Street
Philadelphia, PA, USA, 19104-6228
prolo@linc.cis.upenn.edu

Abstract

We discuss a grammar development process used to generate the trees of the wide-coverage Lexicalized Tree Adjoining Grammar (LTAG) for English of the XTAG Project. Result of the coupling of Becker's metarules and a simple yet principled hierarchy of rule application, the approach has been successful to generate the large set of verb trees in the grammar, from a very small initial set of trees.

1 Introduction

The XTAG Project (Joshi, 2001) is an ongoing project at the University of Pennsylvania since about 1988, aiming at the development of natural language resources based on Tree Adjoining Grammars (TAGs) (Joshi and Schabes, 1997). Perhaps the most successful experience in it has been the construction of a wide-coverage Lexicalized TAG for English (Doran et al., 2000; XTAG Research Group, 2001), based on ideas initially developed in (Krock and Joshi, 1985).

As the grammar grew larger, the process of consistent grammar development and maintenance became harder (Vijay-Shanker and Schabes, 1992). An LTAG is a set of lexicalized elementary trees that can be combined, through the operations of tree adjunction and tree substitution, to derive syntactic structures for sentences. Driven by locality principles, each elementary tree for a given lexical head is expected to contain its projection, and slots for its arguments (e.g., (Frank, 2001)). Keeping up with these principles, one can easily see that the number of required elementary trees is huge for a grammar with reasonable coverage of syntactic phenomena. Under the XTAG project, for engineering reasons, the grammar has been split up in (roughly) two main components[1]: a set tree templates lexicalized by a syntactic category, and a lexicon with each word selecting its appropriate tree templates. Figure 1 shows typical grammar template trees that can be selected by lexical items and combined to generate the structure in Figure 2. The *derivation tree*, to the right, contains the history of the tree grafting process that generated the *derived tree*, to the left.[2]

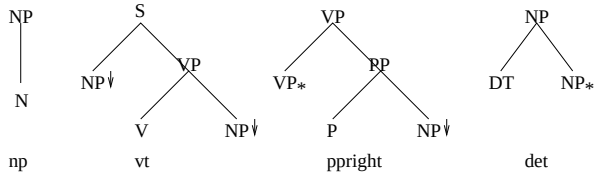


Figure 1: An example of Tree Adjoining Grammar

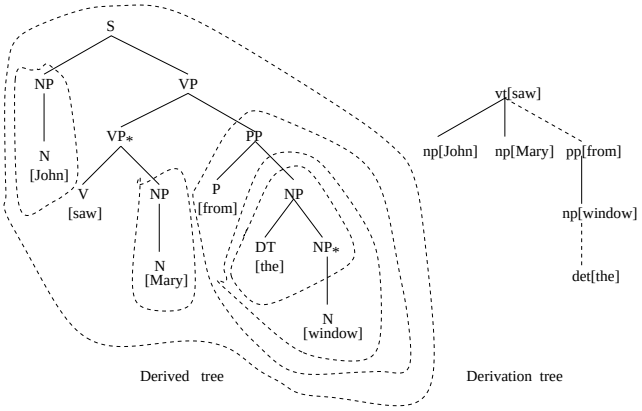


Figure 2: Derivation of *John saw Mary from the window*

Although various syntactic categories have multiple syntactic frames available (e.g., prepositions may have different kinds of arguments, nouns and adjectives may have arguments or not, etc.), it is the verbs that exhibit the most wild variety of domains of locality: from the 1004 template trees in

[1]For a more accurate description of the XTAG system architecture, see (XTAG Research Group, 2001) or (Doran et al., 2000).

[2]For a more comprehensive introduction to TAGs and Lexicalized TAGs we refer the reader to (Joshi and Schabes, 1997).

the XTAG grammar, 783 are for verbs, almost 80%. That happens because the grammar tries to capture in elementary trees the locality for each of the diverse syntactic structures related transformationally to each other (the effect of long distance movement is captured by adjunction of the intervening material). Examples of required tree templates are: declarative transitive (the example above); ditransitive passive with wh-subject moved; and intransitive with PP object with the PP-object relativized.

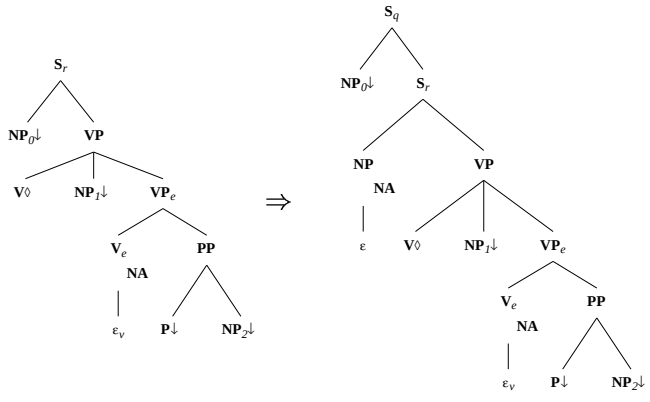
As early noticed by (Vijay-Shanker and Schabes, 1992) the information regarding syntactic structure and feature equations in (feature-based) LTAGs is repeated across templates trees in a quite regular way, that perhaps could be more concisely captured than by just having a plain set of elementary trees. Besides the obvious linguistic relevance, as a pure engineering issue, the success of such enterprise would result in enormous benefits for grammar development and maintenance.

Several approaches have been proposed in the literature describing compact representations methods for LTAGs, perhaps the best known being (Vijay-Shanker and Schabes, 1992), (Candito, 1996; Candito, 1998), (Evans et al., 1995; Evans et al., 2000), (Xia et al., 1998; Xia, 2001), and (Becker, 1993; Becker, 1994; Becker, 2000). We describe in this paper how we combined Becker's metarules with a hierarchy of rule application to generate the verb tree templates in the XTAG English grammar, from a very small initial set of trees.[3]

2 Metarules

We present in this section an introductory example of metarules.[4] Consider the two trees in Figure 3 anchored by verbs that take as arguments an NP and a PP (e.g., *put*).

The one to the left corresponds to its declarative structure; the other to the wh-subject extracted form. Despite their complexity, they share most of their structure: the only differences being the wh-site in the right tree (higher NP) and the trace at subject position. That observation would not be very useful if the differential description we have made was idiosyncratic to this pair, which is not the case. Clearly, many other pairs all over the grammar will share the same differential description.



(a) declarative (b) subject extracted

Figure 3: Some related trees for the verb put

Figure 4 shows a metarule for wh-subject extraction that captures the similarities mentioned above. It describes how to automatically generate the tree in Figure 3.b, given as input the tree in Figure 3.a. Here is how it works. First the input tree has to match the *left-hand side* of the metarule, *lhs* in Figure 4, starting from their roots. In the example, the *lhs* tree requires the candidate tree to have its root labeled $S_?$. Then, its leftmost child has to be an *NP*, as indicated by the node $?2NP_?$ in *lhs*: $?2$ indicates it is the variable $\#2$; $NP_?$ indicates we need an *NP*, regardless of the subscript. Next, the *lhs* tree requires the rest of the tree to match variable $?1$. That is trivial, because such variables with just an identification number are "wild cards" that match any range of subtrees. The matches of each variable in *lhs*, for the application to the input tree in Figure 3.a, are shown in Figure 5.

Had the matching process failed no new tree would have been generated. Since in the example above the matching succeeded, the processor move

[3]This work started years ago, already mentioned in (Doran et al., 2000, p. 388). There has been some confusion on the issue, perhaps driven by a somewhat ambiguous statement in (Becker, 2000, p. 331): "In this paper, we present the various patterns which are used in the implementation of metarules which we added to the XTAG system (Doran et al. 2000)". The work of Becker conceived and developed the idea of metarules for TAGs (Becker, 1993; Becker, 1994). He also created the original implementation of the metarule interpreter as part of the XTAG software, from 1993 to 1995, thereafter improved to reach a first stable form as documented in (XTAG Research Group, 1998). However, with respect to grammar development, he only created the necessary example patterns to support the concepts of metarules, while the work described here is the first to actually evaluate metarules in-the-large as part of the XTAG project (a preliminary version of this paper was in the TAG+6 workshop).

[4]For a more comprehensive introduction of its linguistic motivations and the basic patterns it allows, see (Becker, 2000).

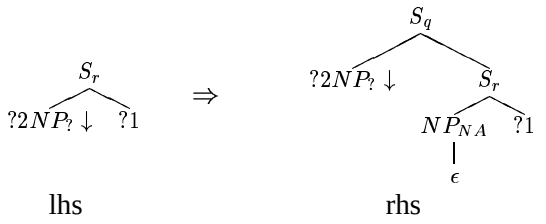


Figure 4: Metarule for wh-movement of subject

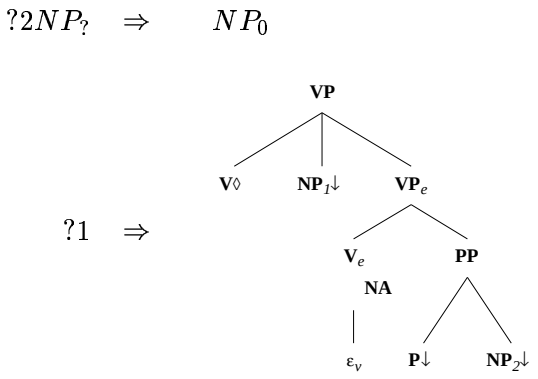


Figure 5: Variable Matching for the tree in Fig. 3.a

to the final step, which is to generate the new tree. We look at the *right-hand side* of the metarule *rhs* and just replace the instances of the variables there with their matched values, obtaining the tree in Figure 3.b. The same process can be applied for the many other pairs related by the same metarule.

In a feature-based grammar as the one we are focusing on, to create tree structures without the proper feature equations is of little use. On the other hand, experience has shown that feature equations are much harder to maintain correct and consistent in the grammar than the tree structures. The XTAG metarules use features in two ways: as matching requirements, and for transformation purposes.

3 An ordered set of metarules

The set of verbal trees can be seen as a subset of the Cartesian product of three dimensions: subcategorization (e.g., transitive, intransitive), redistribution (e.g., passive), and realization (e.g., wh-subject movement) – discounted, of course, combinations blocked by linguistic constraints (e.g., there can not be object movement in intransitives). The verb trees in the XTAG English grammar are organized in families that roughly reflect a subcategorization frame. Hence, each family contains trees

SUBCATEGORIZATION GROUP	No. of Fams.	No. of Trees
Intransitive	1	12
Transitive	1	39
Adjectival complement	1	11
Ditransitive	1	46
Prepositional complement	4	182
Verb particle constructions	3	100
Light verb constructions	2	53
Sentential Complement (full verb)	3	75
Sentential Subject (full verb)	4	14
Idioms (full verb)	8	156
Small Clauses/Predicative	20	187
Equational "be"	1	2
Ergative	1	12
Resultatives	4	101
It Clefts	3	18
Total	57	1008

Table 1: Current XTAG Grammar Coverage

for each combination of redistribution and realization alternatives compatible with the subcategorization frame. The *base* tree of a family is the one corresponding to its declarative usage (no redistribution, arguments in canonical position). Table 1 summarizes the current coverage of the XTAG English grammar. The grouping of the families is just for presentational convenience.

Becker (1993; 1994; 2000) proposes that a grammar is the closure of the set of base trees under metarule application, raising a heated discussion on the unboundedness of the process of recursive application. We understand the issue is artificial and we show in this section that a simple ordering mechanism among the metarules suffices.[5]

Our strategy for generation of the verbal trees is the following. There is a unique ordered set of 21 metarules (Table 2). For each family, we start with the base, declarative tree, apply the sequence of metarules, and the result is the whole family of trees. The sequence of metarules are applied in a way we call cumulative mode of application represented in Figure 6. The generated set start with the declarative tree. The first metarule is applied to the set, generating new trees, which are themselves included in the generated set. Then the second rule is applied, and so on, until the sequence is finished.

Redistribution rules are applied before realization

[5]Notice that in the context of TAGs, metarules are used "off-line" to generate a finite grammar, a bounded process, which is radically different from their use in the Transformational Grammar tradition or in any other "on-the-fly" environment.

Metarule	Description
passive	Generate the passive form
passive-fromPP	Passive form for PP complements: "Results were accounted for by ..."
dropby	Passive without by-clause
gerund	Trees for NPs like in "John eating cake (is unbelievable)"
imperative	Imperative
wh-subj	Wh-subject movement
wh-sentsubj	Wh-subj. mov. for sentential subjs.
wh-npobj	NP extraction from inside objects
wh-smallnpobj	NP obj. extr. for small clauses
wh-apobj	AP complement extraction
wh-advobj	ADVP complement extraction
wh-ppobj	PP complement extraction
rel-adj-W	Adjunct rel. clause with wh-NP
rel-adj-noW	Adj. rel. clause with compl.
rel-subj-W	Subject rel. clause with wh-NP
rel-subj-noW	Subj. rel. clause with compl.
rel-subj-noW-forpassive	Subj. rel. clause with compl. for passives
rel-obj-W	NP Object rel. clause with wh-NP
rel-obj-noW	NP Obj. rel. clause with compl.
rel-ppobj	PP Object rel. clause
PRO	PRO Subject

Table 2: Metarules used to generate the verb families of the XTAG English Grammar

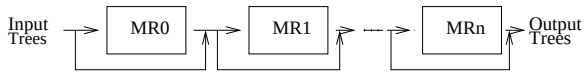


Figure 6: Cumulative application of metarules

rules. It is usual for a metarule to fail to apply to many of the already generated trees. Partly, this is due to the obvious fact that not all rules are compatible with any given subcategorization frame or after another metarule has been applied to it. But also, because the linear order is clearly a simplification of what in fact should be a partial order, e.g. subject relativization should not apply to a wh-subject extracted tree. Constraints expressed in the metarules are responsible for blocking such applications.

We chose one of the largest families, with 52 trees, for verbs like *put* that take both an NP and a PP as complements, to detail the process of generation. For the sake of simplicity we omit the 26 relative clause trees. The remaining 25 trees [6] are described in Table 3, and the generation graph is shown in Figure 7. Numbers assigned to the trees in the Table are used to refer to them in the Figure.

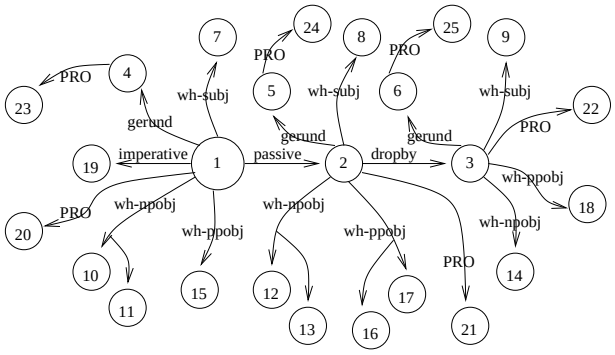


Figure 7: Partial generation of the *put* family

4 Evaluation

An important methodological issue is that the grammar was generated towards a pre-existent English grammar. So we can claim that the evaluation was quite accurate. Differences between the generated and pre-existent trees had to be explained and discussed with the group of grammar developers. Often this led to the discovery of errors and better ways of modeling the grammar. Perhaps the best expression of the success of this enterprise was to be able to generate the 53 verb families (783 trees) from only the corresponding 53 declarative trees (or so) plus 21 metarules, a quite compact initial set. More importantly this compact set can be effectively used for grammar development. We turn now to the problems found as well as some interesting observations.

4.1 We undergenerate:[7]

There are about 20 idiosyncratic trees not generated, involving trees for "-ed" adjectives, restricted to transitive and ergative families, and Determiner Gerund trees, which lack a clear pattern across the families.[8] These trees should be separately added to the families. Similarly, there are 10 trees involving punctuation in the sentential complement families which are not worth generating automatically.

We do not handle yet: the passivization of the second object (from inside a PP) in families for idiomatic expressions ("The warning was taken heed

[6] There is one tree, for nominalization with determiner, we have found not worth generating. We comment on that ahead.

[7] We overlooked it-cleft families, with unusual tree structures, and the equational *be* family with two trees.

[8] For instance, the nominalization of the transitive verb *find* selects a prepositional complement introduced by the preposition *of*: "The finding of the treasure (by the pirates) was news for weeks." But the "of" insertion is not uniform across families: cf. "the accounting for the book."

No.	DESCRIPTION	EXAMPLE
1	Declarative	He put the book on the table
2	Passive w. by	The book was put on the table by him
3	Passive w.o. by	The book was put on the table
4	Gerundive nominals	He putting the book on the table was unexpected
5	Gerundive for passive w. by	The book being put on the table by him ...
6	Gerundive for passive w.o. by	The book being put on the table ...
7	Subject extraction	Who put the book on the table ?
8	Subj. extr. from passive w. by	What was put on the table by him ?
9	Subj. extr. from passive w.o. by	What was put on the table ?
10	1st obj. extraction	What did he put on the table ?
11	2nd obj. NP extraction	Where did he put the book on ?
12	2nd obj. NP extr. from pass. w. by	Where was the book put on by him ?
13	Agent NP extr. from pass. w. by	Who (the hell) was this stupid book put on the table by ?
14	2nd obj. NP extr. from pass. w.o. by	Where was the book put on ?
15	PP obj. extr.	On which table did he put the book ?
16	PP obj. extr. from pass. w. by	On which table was the book put by him ?
17	By-clause extr. from pass. w. by	By whom was the book put on the table ?
18	PP obj. extr. from pass. w.o. by	On which table was the book put ?
19	Imperative	Put the book on the table !
20	Declarative with PRO subject	I want to [PRO put the book on the table]
21	Passive w. by w. PRO subject	The cat wanted [PRO to be put on the tree by J.]
22	Passive w.o. by w. PRO subject	The cat wanted [PRO to be put on the tree]
23	Ger. noms. with PRO subject	John approved of [PRO putting the cat on the tree]
24	Ger. noms. for passive w. by w. PRO subj.	The cat approved of [PRO being put on the tree by J.]
25	Ger. noms. for passive w.o. by w. PRO subj.	The cat approved of [PRO being put on the tree]

Table 3: Partial view of the trees from the *put* family

of"); the occurrence of the "*by phrase*" before sentential complements ("I was told by Mary that ..."); and wh-extraction of sentential complements and of exhaustive PPs. Except for the first case all can be easily accounted for.

4.2 We overgenerate:

We generate 1200 trees (instead of 1008).[9] However things are not as bad as they look: 206 of them are for passives related to multi-anchor trees, as we explain next. It is acknowledged the existence of a certain amount of overgeneration in the tree families due to the separation between the lexicon and the tree templates. For instance, it is widely known that not all transitive verbs can undergo passivization. But the transitive family contains passive trees. The reconciliation can be made through features assigned to verbs that allow blocking the selection of the particular tree. However in the family for verb particle with two objects (e.g., for "John opened up Mary a bank account"), the four lexical entries were judged not to undergo passivization and the corresponding trees (64) were omitted from the family. It is not surprising then that the metarules overgenerate them. Still, 100 out of the 206 are for passives in the unfinished idiom families and are definitely lexically dependent. The other 42 overgenerated passives are in the light verb families. There are a few other cases of overgeneration due to lexically dependent judgments, not worth detailing. Finally, a curious case involved empty elements that could be generated at slightly different positions which are not distinguished at surface (e.g., before or after a particle). The choice for having only one alternative in the grammar is of practical nature (related to parsing efficiency) as opposed to linguistic.

4.3 Limitations to further compaction:

All the metarules for wh-object extraction do essentially the same, but currently they cannot be unified. Further improvements in the metarule system implementation could solve the problem at least partially, by allowing to treat symbols and indices as separate variables. A more difficult problem are some subtle differences in the feature equations across the grammar (e.g., causing the need of a separate tree for relativization of the subject in passive trees). By far, feature equations constitute the hardest issue to handle with the metarules.

4.4 A metarule shortcoming:

Currently they do not allow for the specification of negative structural constraints to matching. There is one feature equation related to punctuation that needed 5 separate metarules (not described above)

[9] Which means more than an excess of 192 trees since there is also some undergeneration, already mentioned.

to handle (by exhaustion) the following constraint: the equation should be added if and only if the tree has some non-empty material after the verb which is not a "by-phrase".

4.5 Other cases:

A separate metarule was needed to convert foot nodes into substitution nodes in sentential complement trees. This families departs from the rest of the grammar in that their base tree is an auxiliary tree to allow extraction from the sentential complement. But the corresponding relative clauses have to have the S complement as a substitution node.

5 Discussion

A question might arise about the rationale behind the ordering of the rules. There has been some debate about how lexical or syntactic rules should apply to generate an LTAG. Becker's metarules have been targeted due to the unboundedness in the process of their recursive application. He has been defending himself (Becker, 2000) suggesting principles under which boundedness would arise as a natural consequence. What we have been proposing here is a clear separation between the metarules as a formal system for deriving trees from trees and the control mechanism that says which rule is applied when. Given the experiment we have reported in this paper, it seems undeniable that such approach should be considered at least valid.

As for the particular order we adopted, as mentioned before, it comes partly from reasonable assumptions about precedence of lexical redistribution rules over extraction rules (which can also be empirically observed), and partly as a mere simplification of a partial order relation.

In a related issue, it is important to notice also that the ordering is not among rules, but among instances of rule applications as observed in (Evans et al., 2000). It was just by "accident" that rules were applied only once. For instance, one could imagine that in languages where double wh-movement is possible, a wh-rule have to be effectively applied twice. That does not entitle one to reject an *a priory* ordering between the instances. In this case the wh-rule would appear twice in the graph.

Still another issue that can be raised is related to the monotonicity of the approach, especially in face of the problems we had with passives. As in (Candito, 1996), we overgenerate: ultimately, trees are incorrectly being assigned to some lexical items. In our particular case, however this can be charged to the architecture of the XTAG English grammar. The obvious way to handle this kind of problem in the XTAG grammar is by way of features in the lexical items that block their effective selection of a template. On the other hand if one wants to adopt a stronger lexicalist approach, it is easy to see how one could allow the lexical item to influence the base trees so as to control what rules in the chain are effectively applied, e.g., as in (Evans et al., 2000). Or, in other words: a metarule by itself is just a mechanism for tree-transformation.[10]

6 Conclusions

The ideas of compact representation of the lexicon are certainly not new, with well known concrete proposals for diverse frameworks (Bresnan, 1982; Gazdar et al., 1985; Pollard and Sag, 1997). For LTAGs, in particular, there has been quite a few proposals, as we have already mentioned (Vijay-Shanker and Schabes, 1992; Becker, 1993; Candito, 1996; Evans et al., 1995; Xia, 2001), and even large-scale grammars built with them, e.g., the French grammar in (Abeille and Candito, 2000) and an English one in (Xia, 2001).

The work we described in this paper evaluates a particular approach to grammar generation from compact representation. On the one hand, it tests the hypothesis that Becker's tree-transformation rules, the *'metarules'*, fit well the LTAG formalism and can be effectively and efficiently used to build large-scale such grammars. On the other hand, the facility with which a natural partial ordering of such rules is obtained (here simplified as a total order for practical reasons), dismisses the debate concerning free-generation, unboundedness, and also weakens the arguments concerning the non-directionality of the metarules, suggesting that they might be more of an *academic* nature.

A major strength of the approach is to have set a target grammar with which to compare. A detailed qualitative evaluation of the mismatches between the existing and generated grammars was obtained that allows us to access not only the weaknesses of the generation process but also the problems of the original grammar development: e.g., the inconsistency in the treatment of the interface between the lexicon and the tree templates.

Future work in the XTAG group includes the construction of a graph based interface for metarules that allows the application of metarules according

[10] Of course, this may not reflect Becker's view.

to a partial order, as well as distinct treatment for different families.[11] We are also interested in aspects of the use of metarules to enhance extracted grammars (Kinyon and Prolo, 2002).

References

Anne Abeille and Marie-Helene Candito. 2000. Ftag: A lexicalized Tree Adjoining Grammar for French. In Abeille and Rambow (Abeille and Rambow, 2000), pages 305–329.

Anne Abeille and Owen Rambow, editors. 2000. *Tree Adjoining Grammars: formalisms, linguistic analysis and processing.* CSLI, Stanford, CA.

Tilman Becker. 1993. *HyTAG: A new Type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Word Order Lang uages*. Ph.D. thesis, Universität des Saarlandes.

Tilman Becker. 1994. Patterns in metarules. In *Proceedings of the 3rd TAG+ Conference*, Paris, France.

Tilman Becker. 2000. Paterns in metarules for TAG. In Abeille and Rambow (Abeille and Rambow, 2000), pages 331–342.

Joan Bresnan, editor. 1982. *The Mental Representation of Grammatical Relations.* MIT Press, Cambridge, MA.

Marie-Helene Candito. 1996. A principle-based hierarchical representation of LTAGs. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 194–199, Copenhagen, Denmark.

Marie-Helene Candito. 1998. Building parallel LTAG for french and italian. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 16th International Conference on Computational Linguistics*, pages 211–217, Montreal, Canada.

Christine Doran, Beth Ann Hockey, Anoop Sarkar, B. Srinivas, and Fei Xia. 2000. Evolution of the XTAG system. In Abeille and Rambow (Abeille and Rambow, 2000), pages 371–404.

Roger Evans, Gerald Gazdar, and David Weir. 1995. Encoding lexicalized Tree Adjoining Grammars with a nonmonotonic inheritance hierarchy. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 77–84, Cambridge, MA, USA.

Roger Evans, Gerald Gazdar, and David Weir. 2000. 'Lexical Rules' are just lexical rules. In Abeille and Rambow (Abeille and Rambow, 2000), pages 71–100.

Robert Frank. 2001. *Phrase Structure Composition and Syntactic Dependencies.* to be published.

Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard Un. Press, Cambridge, MA.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In *Handbook of Formal Languages*, volume 3, pages 69–123. Springer-Verlag, Berlin.

Aravind K Joshi. 2001. The XTAG project at Penn. In *Proceedings of the 7th International Workshop on Parsing Technologies (IWPT-2001)*, Beijing, China. Invited speaker.

Alexandra Kinyon and Carlos A. Prolo. 2002. A classification of grammar development strategies. In *Proceedings of the Workshop on Grammar Engineering and Evaluation*, Taipei, Taiwan.

Anthony S. Krock and Aravind K. Joshi. 1985. The linguistic relevance Tree Adjoining Grammar. Technical Report MS-CIS-85-16, University of Pennsylvania.

Carl Pollard and Ivan Sag. 1997. *Information-based Syntax and Semantics. Vol 1: Fundamentals*, volume 13 of *CSLI Lecture Notes*. CSLI, Menlo Park, CA.

K. Vijay-Shanker and Yves Schabes. 1992. Structure sharing in lexicalized Tree-Adjoining Grammars. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, pages 205–211, Nantes, France.

Fei Xia, Martha Palmer, K. Vijay-Shanker, and Joseph Rosenzweig. 1998. Consistent grammar development using partial-tree descriptions for lexicalized Tree-Adjoining Grammars. In *Proceedings of the 4th Int. Workshop on Tree Adjoining Grammars (TAG+4)*, Philadelphia, USA.

Fei Xia. 2001. *Investigating the Relationship between Grammars and Treebanks for Natural Languages*. Ph.D. thesis, Department of Computer and Information Science, Un. of Pennsylvania.

The XTAG Research Group. 1998. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 98-18, University of Pennsylvania.

The XTAG Research Group. 2001. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 01-03, University of Pennsylvania.

[11] The new interface to the XTAG development system is thanks to Eric Kow and Nikhil Dinesh.

Testing for different languages is done using various tools for example JUNIT for java, FunSuite for scala and TestRig (grun on command line) for grammar(.g4). In this blog our focus will be on understanding what ANTLR is and mainly testing the correctness of an input(string generally) with respect to the developed grammar using Grun. ANTLR (ANother Tool For Language Recognition) is a tool that converts grammars into programs (java programs for now) that recognize sentences in the language described by the grammar. For example, given a grammar for JSON, the ANTLR tool generates java files that