A Classification of Grammar Development Strategies

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

Abstract

In this paper, we propose a classification of grammar development strategies according to two criteria : **hand-written** *versus* **automatically acquired** grammars, and grammars based on a **low** *versus* **high level of syntactic abstraction**. Our classification yields four types of grammars. For each type, we discuss implementation and evaluation issues.

1 Introduction: Four grammar development strategies

There are several potential strategies to build wide-coverage grammars, therefore there is a need for classifying these various strategies. In this paper, we propose a classification of grammar development strategies according to two criteria :

- **Hand-crafted** *versus* **Automatically acquired** grammars
- Grammars based on a **low** *versus* **high level of syntactic abstraction**.

As summarized in table 1, our classification yields four types of grammars, which we call respectively type A, B, C and D.

Of these four types, three have already been implemented to develop wide-coverage grammars for English within the Xtag project, and an implementation of the fourth type is underway [1]. Most of our examples are based on the development of wide coverage Tree Adjoining Grammars (TAG), but it is important to note that the classification is relevant within other linguistic frameworks as well (HPSG, GPSG, LFG etc.) and is helpful to discuss portability among several syntactic frameworks.

We devote a section for each type of grammar in our classification. We discuss the advantages and drawbacks of each approach, and especially focus on how each type performs w.r.t. grammar coverage, linguistic adequacy, maintenance, over- and under- generation as well as to portability to other syntactic frameworks. We discuss grammar replication as a mean to compare these approaches. Finally, we argue that the fourth type, which is currently being implemented, exhibits better development properties.

2 TYPE A Grammars: hand-crafted

The limitations of Type A grammars (hand-crafted) are well known : although linguistically motivated, developing and maintaining a totally hand-crafted grammar is a challenging (perhaps unrealistic ?) task. Such a large hand-crafted grammar for TAGs is described for English in (XTAG Research Group, 2001). Smaller hand-crafted grammars for TAGs have been developed for other languages (e.g. French (Abeille, 1991)), with similar problems. Of course, the limitations of hand-crafted grammars are not specific to the TAG framework (see e.g. (Clement and Kinyon, 2001) for LFG).

2.1 Coverage issues

The Xtag grammar for English, which is freely downloadable from the project homepage [2] (along with tools such as a parser and an extensive documentation), has been under constant development for approximately 15 years. It consists of more than 1200 elementary trees (1000 for verbs) and has been tested on real text and test suites. For instance, (Doran et al., 1994) report that 61% of 1367 grammatical sentences from the TSNLP test-suite (Lehman and al, 1996) were parsed with an early version of the grammar. More recently, (Prasad and Sarkar, 2000) evaluated the coverage of the grammar on "the weather corpus", which contained rather complex sentences with an average length of 20 words per sentence, as well as on the "CSLI LKB test suite" (Copestake, 1999). In addition, in order to

[1] We do not discuss here shallow-parsing approaches, but only *f*ull grammar development. Due to space limitations, we do not introduce the TAG formalism and refer to (Joshi, 1987) for an introduction.

[2] http://www.cis.upenn.edu/ xtag/

	High level of syntactic abstraction	*Low level of syntactic abstraction*
Hand-crafted	**Type A:** Traditional hand-crafted grammars	**Type C:** Hand-crafted level of syntactic abstraction Automatically generated grammars
Automatically acquired	**Type B:** Traditional treebank extracted grammars	**Type D:** Automatically acquired level of syntactic abstraction Automatically generated grammar

Table 1: A classification of grammars

evaluate the range of syntactic phenomena covered by the Xtag grammar, an internal test-suite which contains all the example sentences (grammatical and ungrammatical) from the continually updated documentation of the grammar is distributed with the grammar. (Prasad and Sarkar, 2000) argue that constant evaluation is useful not only to get an idea of the coverage of a grammar, but also as a way to continuously improve and enrich the grammar [3].

Parsing failures were due, among other things, to POS errors, missing lexical items, missing trees (i.e. grammar rules), feature clashes, bad lexicon grammar interaction (e.g. lexical item anchoring the wrong tree(s)) etc.

2.2 Maintenance issues

As a hand-crafted grammar grows , consistency issues arise and one then needs to develop maintenance tools. (Sarkar and Wintner, 1999) describe such a maintenance tool for the Xtag grammar for English, which aims at identifying problems such as typographical errors (e.g. a typo in a feature can prevent unification at parse time and hurt performance), undocumented features (features from older versions of the grammar, that no longer exist), type-errors (e.g. English verb nodes should not be assigned a *gender* feature), etc. But even with such maintenance tools, coverage, consistency and maintenance issues still remain.

[3]For instance, at first, Xtag parsed only 20% of the sentences in the weather corpus because this corpus contained frequent free relative constructions not handled by the grammar. After augmenting the grammar, 89.6% of the sentences did get a parse.

2.3 Are hand-crafted grammars useful ?

Some degree of automation in grammar development is unavoidable for any real world application : small and even medium-size hand-crafted grammar are not useful for practical applications because of their limited coverage, but larger grammars give way to maintenance issues. However, despite the problems of coverage and maintenance encountered with hand-crafted grammars, such experiments are invaluable from a linguistic point of view. In particular, the Xtag grammar for English comes with a very detailed documentation, which has proved extremely helpful to devise increasingly automated approaches to grammar development (see sections below) [4].

3 TYPE B Grammars: Automatically extracted

To remedy some of these problems, Type B grammars (i.e. automatically acquired, mostly from annotated corpora) have been developed. For instance (Chiang, 2000), (Xia, 2001) (Chen, 2001) all automatically acquire large TAGs for English from the Penn Treebank (Marcus et al., 1993). However, despite an improvement in coverage, new problems arise with this type of grammars : availability of annotated data which is large enough to avoid sparse data problems, possible lack of linguistic adequacy, extraction of potentially unreasonably large grammars (slows down parsing and increases ambiguity),

[4]Perhaps fully hand-crafted grammars can be used in practice on limited domains, e.g. the weather corpus. However, a degree of automation is useful even in those cases, if only to insure consistency and avoid some maintenance problems.

lack of domain and framework independence (e.g. a grammar extracted from the Penn Treebank will reflect the linguistic choices and the annotation errors made when annotating the treebank).

We give two examples of problems encountered when automatically extracting TAG grammars: The extraction of a wrong domain of locality; And The problem of sparse-data regarding the integration of the lexicon with the grammar.

3.1 Wrong domain of locality

Long distance dependencies are difficult to detect accurately in annotated corpora, even when such dependencies can be adequately modeled by the grammar framework used for extraction (which is the case for TAGs, but not for instance for Context Free Grammars). For example, (Xia, 2001) extracts two elementary trees from a sentence such as *Which dog does Hillary Clinton think that Chelsea prefers*. These trees are shown on figure 1. Unfortunately, because of the potentially unbounded dependency, the two trees exhibit an incorrect domain of locality: the Wh-extracted element ends up in the wrong elementary tree, as an argument of "think", instead of as an argument of "prefer" [5]

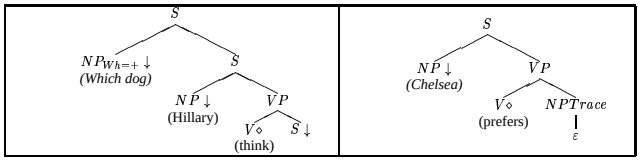

Figure 1: Extraction of the wrong domain of locality

This problem is not specific to TAGs, and would translate in other frameworks into the extraction of the "wrong" dependency structure[6].

3.2 Sparse data for lexicon-grammar integration

Existing extraction algorithms for TAGs acquire a fully lexicalized grammar. A TAG grammar may be viewed as consisting of two components: on the one hand "tree templates" and on the other hand a lexicon which indicates which tree template(s) should be associated to each lexical item [7].

Suppose the following three sentences are encountered in the training data :

1. *Peter watches the stars*
2. *Mary eats the apple*
3. *What does Peter watch ?*

From these three sentences, two tree templates will be correctly acquired, as shown on figure 2 : The first one covers the canonical order of the realization of arguments for sentences 1 and 2, the second covers the case of a Wh-extracted object for sentence 3. Concerning the interaction between the lexicon and the grammar rules, the fact that "watch" should select both trees will be accurately detected. However, the fact that "eat" should also select both trees will be missed since "eat" has not been encountered in a Wh-extractedObject construction.

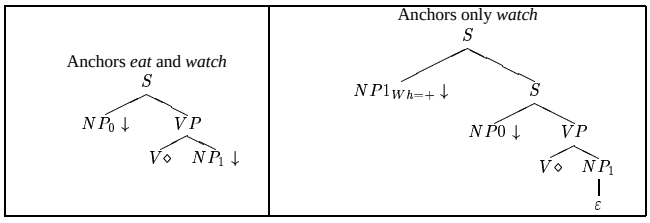

Figure 2: Correct templates, but incomplete lexicon-grammar interface

A level of syntactic abstraction is missing : in this case, the notion of subcategory frame. This is especially noticeable within the TAG framework from the fact that in a TAG hand-crafted grammar the grammar rules are grouped into "tree families", with one family for each subcategorization frame (transitive, intransitive, ditransitive, etc.), whereas automatically extracted TAGs do not currently group trees into families.

4 TYPE C Grammars

To remedy the lack of coverage and maintenance problems linked to hand-crafted grammars, as well as the lack of generalization and linguistic adequacy of automatically extracted grammars, new syntactic levels of abstraction are defined. In the context of TAGs, one can cite the notion of MetaRules (Becker, 2000), (Prolo, 2002)[8], and the notion of MetaGrammar (Candito, 1996), (Xia, 2001).

[5] Some extraction algorithms such as those of (Chen, 2001) or (Chiang, 2000) do retrieve the right the right domain of locality for this specific example, but do extract a domain of locality which is incorrect in some other cases.

[6] One can argue that the problem does not appear when using simple CFGs, and/or that this problem is only of interest to linguists. A counter-argument is that linguistic adequacy of a grammar, whether extracted or not, DOES matter. An extreme caricature to illustrate this point : the context free grammar *S* $\rightarrow$ *S word* | *word* allows one to robustly and unambiguously parse any text, but is not very useful for any further NLP.

[7] This subdivision avoids an combinatoric explosion in the number of rules if the grammar was fully lexicalized

[8] For other MetaRule based approaches based on the DATR formalism, see (Carroll et al., 2000) or (Evans et al., 2000)

4.1 MetaRules

A MetaRule works as a pattern-matching tool on trees. It takes as input an elementary tree and outputs a new, generally more complex, elementary tree. Therefore, in order to create a TAG, one can start from one *canonical* elementary tree for each subcategorization frame and a finite number of MetaRules which model syntactic transformations (e.g. passive, wh-questions etc) and automatically generate a full-size grammar. (Prolo, 2002) started from 57 elementary trees and 21 hand-crafted MetaRules, and re-generated the verb trees of the hand-crafted Xtag grammar for English described in the previous section.

The replication of the hand-crafted grammar for English, using a MetaRule tool, presents interesting aspects : it allows to directly compare the two approaches. Some trees generated by (Prolo, 2002) were not in the hand-crafted grammar (e.g. various orderings of "by phrase passives") while some others that were in the hand-crafted grammar were not generated by the MetaRules[9]. This replication process makes it possible, with detailed scrutiny of the results, to :

- Identify what should be consider as under- or over- generation of the MetaRule tool.
- Identify what should be considered to be under- or over- generation of the hand-crafted grammar.

Thus, grammar replication tasks make it possible to improve both the hand-crafted and the MetaRule generated grammars.

4.2 MetaGrammars

Another possible approach for compact and abstract grammar encoding is the MetaGrammar (MG), initially developed by (Candito, 1996). The idea is to compact linguistic information thanks to an additional layer of linguistic description, which imposes a general organization for syntactic information in a three-dimensional hierarchy :

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

Each terminal class in dimension 1 describes a possible initial subcategorization (i.e. a TAG tree family). Each terminal class in dimension 2 describes a list of ordered redistributions of functions (e.g. it allows to add an argument for causatives, to erase one for passive with no agents ...). Each terminal class in dimension 3 represents the surface realization of a surface function (ex: declares if a direct-object is pronominalized, wh-extracted, etc.). Each class in the hierarchy corresponds to the partial description of a tree (Rogers and Vijay-Shanker, 1994). A TAG elementary tree is generated by inheriting from exactly one terminal class from dimension 1, one terminal class from dimension 2, and *n* terminal classes from dimension 3 (where *n* is the number of arguments of the elementary tree being generated). For instance the elementary tree for "Par qui sera accompagnee Marie" (By whom will Mary be accompanied) is generated by inheriting from *transitive* in dimension 1, from *impersonal-passive* in dimension 2 and *subject-nominal-inverted* for its subject and *questioned-object* for its object in dimension 3. This compact representation allows one to generate a 5000 tree grammar from a hand-crafted hierarchy of a few dozens of nodes, esp. since nodes are explicitly defined only for **simple** syntactic phenomena [10]. The MG was used to develop a wide-coverage grammar for French (Abeille et al., 1999). It was also used to develop a medium-size grammar for Italian, as well as a generation grammar for German (Gerdes, 2002) using the newly available implementation described in (Gaiffe et al., 2002). A similar MetaGrammar approach has been described in (Xia, 2001) for English [11].

4.3 MetaGrammars versus MetaRules: which is best ?

It would be desirable to have a way of comparing the results of the MetaGrammar approach with that of the MetaRule approach. Unfortunately, this is not possible because so far none of the two approaches have been used within the same project(s). Therefore, in order to have a better comparison between these two approaches, we have started a second replication of the Xtag grammar for English, this time using a MG. This replication should allow us to make a direct comparison between the hand-crafted grammar, the grammar generated with MetaRules and the grammar generated with a MG.

For this replication task, we use the more recent implementation presented in (Gaiffe et al., 2002) because their tool :

[9] Due to space limitations, we refer to (Prolo, 2002) for a detailed discussion.

[10] Nodes for complex syntactic phenomena are generated by automatic crossings of nodes for simple phenomena

[11] but that particular work did not attempt to replicate the Xtag grammar, and thus the generated grammar is not directly comparable to the hand-crafted version of the grammar.

- Is freely available [12], portable (java), well maintained and includes a Graphical User Interface.
- Outputs a standardized XML format [13]
- Is flexible (one can have more than 3 dimensions in the hierarchy) and strictly monotonic w.r.t. the trees built
- Supports "Hypertags", i.e. each elementary tree in the grammar is associated with a feature structure which describes its salient linguistic properties [14].

In the (Gaiffe et al., 2002) implementation, each class in the MG hierarchy can specify :

- Its SuperClasse(s)
- A Feature structure (i.e. Hypertag) which captures the salient linguistic characteristics of that class.
- What the class needs and provides
- A set a quasi-nodes
- Constraints between quasi-nodes (father, dominates, precedes, equals)
- traditional feature equations for agreement.

The MG tool automatically crosses the nodes in the hierarchy, looking to create "balanced" classes, that is classes that do not need nor provide anything. From these balanced terminal classes, elementary trees are generated. Figure 3 shows how a canonical transitive tree is automatically generated from 3 hand-written classes and the quasi-trees associated to these classes [15].

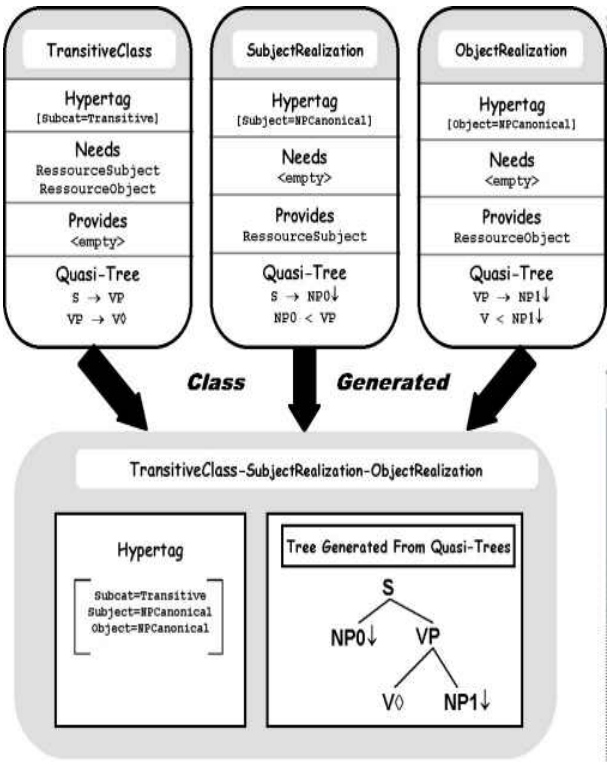

Figure 3: Generating a canonical transitive tree with a MetaGrammar of 3 classes : $\rightarrow$ stands for "father of", $<$ for "precedes", $\diamond$ for anchor nodes and $\downarrow$ for substitution nodes.

4.4 Advantages and drawbacks of TYPE C grammars

It is often assumed that Metarule and MetaGrammar approaches exhibit some of the advantages of hand-crafted grammars (linguistic relevance) as well as some of the advantages of automatically extracted grammars (wide-coverage), as well as easier maintenance and better coherence. However, as is pointed out in (Barrier et al., 2000), grammar development based on hand-crafted levels of abstraction give rise to new problems while not necessarily solving all the *old* problems: Although the automatic generation of the grammar insures some level of consistency, problems arise if mistakes are made while hand-crafting the abstract level (hierarchy or MetaRules) from which the grammar is automatically generated. This problem is actually more serious than with simple hand-crafted grammars, since an error in one node will affect ALL trees that inherit from this node. Furthermore, a large portion of the generated grammar covers rare syntactic phenomena that are not encountered in practice, which unnecessarily augments the size of the resulting grammars, increases ambiguity while not significantly improving coverage [16]. One crucial problem is that despite the automatic generation of the grammar (which eases maintenance), the interface between lexicon and grammar is still mainly man-

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

[13] See http://atoll.inria.fr/ clerger/tag20.dtd,xml for more details on format standardization efforts for TAG related tools.

[14] The idea of "featurization" is very useful for applications such as text generation, supertagging (Kinyon, 2002), and is especially relevant for the automatic acquisition of a MG (see section 5)

[15] This example is of course a simplification: for sake of clarity it does not reflect the complex structure of our real "hierarchy".

[16] For instance, the 5000 tree grammar for French parses 80% of (simple) TSNLP sentences, and does not parse newspaper text, whereas the 1200 tree hand-crafted Xtag grammar for English does. Basically, instead of solving both under-generation and over-generation problems, a hand-crafted abstract level of syntactic encoding runs the risk of increasing both

ually maintained (and of course one of the major sources of parsing failures is due to missing or erroneous lexical entries).

5 TYPE D Grammars

However, the main potential advantage of such an abstract level of syntactic representation is framework independence. We argue that the main drawbacks of an abstract level of syntactic representation (over-generation, propagation of manual errors to generated trees, interface with the lexicon) may be solved if this abstract level is acquired automatically instead of being hand-crafted. Other problems such as sparse data problems are also handled by such a level of abstraction [17]. This corresponds to type D in our classification. A preliminary description of this work, which consist in automatically extracting the hierarchy nodes of a MetaGrammar from the Penn Treebank (i.e. a high level of syntactic abstraction) may be found in (Kinyon and Prolo, 2002). The underlying idea is that a lot of abstract framework independent syntactic information is implicitly present in the treebank, and has to be retrieved. This includes : subcategorization information, potential valency alternations (e.g. passives are detected by a morphological marker on the POS of the verb, by the presence of an NP-Object "trace", and possibly by the presence of a Prepositional phrase introduced by "by", and marked as "logical-subject"), and realization of arguments (e.g. Wh-extractions are noticed by the presence of a Wh constituent, co-indexed with a trace). In order to retrieve this information, we have examined all the possible tag combinations of the Penn Treebank 2 annotation style, and have determined for each combination, depending on its location in the annotated tree whether it was an argument (optional or compulsory) or a modifier. We mapped each argument to a syntactic function [18]. This allowed us to extract fine-grained subcategorization frames for each verb in the treebank. Each subcategorization frame is stored as a finite number of final classes using the (Gaiffe et al., 2002) MG tool : one class for each subcategorization frame (dimension 1 in Candito's terminology), and one class for each function realization (dimension 3 in Candito's terminology). The same technique is used to acquire the valency alternation for each verb, and non-canonical syntactic realizations of verb arguments (Wh extractions etc...). This amounts to extracting "hypertags" (Kinyon, 2000) from the treebank, transforming these Hypertags into a MetaGrammar, and automatically generating a TAG from the MG. An example of extraction may be seen on figure 4 : *expose* appears here in a reduced-relative construction. However, from the trace occupying the canonical position of a direct object, the program retrives the correct subcategorization frame (i.e. tree family) for this verb. Hence, just this occurence of *expose* correctly extracts the MG nodes from which both the "canonical tree" and the "Reduced relative tree" will be generated. If one was extracting a simple type B grammar, the canonical tree would not be retrieved in this example.

```
Input Sentence :
-------------
(NP (NP (DT a)
        (NN group) )
    (PP (IN of)
        (NP (NP (NNS workers) )
            (RRC (VP (VBN exposed)
                 (NP (-NONE- *) )
                 (PP-CLR (TO to)
                     (NP (PRP it) ))
           (ADVP-TMP (NP (QP (RBR more)
                             (IN than)
                         (CD 30) )
                         (NNS years) )
                         (IN ago) ))))))

Extracted Output :
##########
#VB: exposed
#Subj: NP-SBJ
#Arguments: NP#DirObj//PP-CLR#PrepObj(to)
##########
```

Figure 4: An example of extraction from the Penn Treebank

This work is still underway [19]. From the abstract level of syntactic generalization, a TAG will be automatically generated. It is interesting to note that the resulting grammar does not have to closely reflect the linguistic choices of the annotated data from which it was extracted (contrary to type B grammars). Moreover, from the same abstract syntactic data, one could also generate a grammar in another framework (ex. LFG). Hence, this abstract

[17] As discussed in section 3, if one sees *eat* in the data, and one sees some other transitive verb with a Wh extracted object, the elementary tree for "What does J. eat" is correctly generated, even if *eat* has never been encountered in such a construction in the data, which is not the case with the automatic extraction of *traditional* lexicalized grammars

[18] We use the following functions : *subject, predicative, direct object, second object, indirect object, LocDir object.*

[19] For now, this project has already yielded, as a byproduct, a freely available program for extracting verb subcategorization frames (with syntactic functions) from the Penn Treebank

level may be viewed as a syntactic interlingua which can solve some portability issues [20].

6 Conclusion

We have proposed a classification of grammar development strategies and have examined the advantages and drawbacks of each of the four approaches. We have explained how "grammar replication" may prove an interesting task to compare different development strategies, and have described how grammar replication is currently being used in the Xtag project at the University of Pennsylvania in order to compare hand-crafted grammars, grammars generated with MetaRules, and grammars generated with a MetaGrammar. We have reached the conclusion that of the four grammar development strategies proposed, the most promising one consists in automatically acquiring an abstract level of syntactic representation (such as the MetaGrammar). Future work will consist in pursuing this automatic acquisition effort on the Penn Treebank. In parallel, we are investigating how the abstract level we acquire can be used to generate formalisms other than TAGs (e.g. LFG).

Aknowledgements:
We thank the Xtag group, and more particularly W. Schuler and R. Prasad for helpful comments on earlier versions of this work. We also thank B. Crabbé and B. Gaiffé for their help with the LORIA MetaGrammar compiler.

References

A. Abeille, M. Candito, and A. Kinyon. 1999. FTAG: current status and parsing scheme. In *Proc. Vextal-99*, Venice.

A. Abeille. 1991. *Une grammaire lexicalisee d'arbres adjoints pour le francais*. Ph.D. thesis, Univ. of Paris 7.

N. Barrier, S. Barrier, and A. Kinyon. 2000. Lexik : a maintenance tool for FTAG. In *Proc. TAG+5*, Paris.

T. Becker. 2000. Patterns in metarules for TAG. In Abeille Rambow, editor, *Tree Adjoining Grammars*, CSLI.

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

J. Carroll, N. Nicolov, O. Shaumyan, M. Smets, and D. Weir. 2000. Engineering a wide-coverage lexicalized grammar. In *Proc. TAG+5*, Paris.

[20]The notion of "syntactic interlingua" was used in other papers as an analogy to the terminology used for Machine translation : "simple" grammar extraction algorithms could be seen as "transfer approaches" (i.e. low level of abstraction) whereas MetaGrammar extraction could be seen as "interlingua" approaches, in the sense that a higher level of abstraction is needed (the "lingua" being a syntactic framework such as TAGs, LFG etc.)

J. Chen. 2001. *Towards Efficient Statistical Parsing using Lexicalized Grammatical Information*. Ph.D. thesis, Univ. of Delaware.

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

L. Clement and A. Kinyon. 2001. XLFG: an LFG parsing scheme for french. In *LFG-01*, Hong-Kong.

A. Copestake. 1999. The (new) LKB system. In *CSLI*, Stanford University.

C. Doran, D. Egedi, B. Hockey, B. Srinivas, and M. Zaidel. 1994. XTAG system- a wide coverage grammar for English. In *COLING-94*, Kyoto.

R. Evans, G. Gazdar, and D. Weir. 2000. Lexical rules are just lexical rules. In Abeille Rambow, editor, *Tree Adjoining Grammars*, CSLI.

B. Gaiffe, B. Crabbe, and A. Roussanaly. 2002. A new metagrammar compiler. In *Proc. TAG+6*, Venice.

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

A.K. Joshi. 1987. An introduction to tree adjoining grammars. In *Mathematics of language*, John Benjamins Publishing Company.

A. Kinyon and C. Prolo. 2002. Identifying verb arguments and their syntactic function in the Penn Treebank. In *LREC-02*, Las Palmas.

A. Kinyon. 2000. Hypertags. In *COLING-00*, Sarrebrucken.

A. Kinyon. 2002. Featurizing a tree adjoining grammar. In *Proc. TAG+6*, Venice.

S. Lehman and al. 1996. Tsnlp — test suites for natural language processing. In *Proc. COLING-96*, Copenhagen.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English : the penn treeban. In *Computational Linguistics*, Vol 19.

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

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

J. Rogers and K. Vijay-Shanker. 1994. Obtaining trees from their description: an application to TAGS. In *Computational Intelligence 10:4*.

A. Sarkar and S. Wintner. 1999. Typing as a means for validating feature structures. In *CLIN-99*, Utrecht.

F. Xia. 2001. *Automatic grammar generation from two perspectives*. Ph.D. thesis, Univ. of Pennsylvania.

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