Towards Semi-automating Grammar Development

Fei Xia, Martha Palmer
Dept of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
fxia/mpalmer@linc.cis.upenn.edu

K. Vijay-Shanker
Dept of Computer and Information Science
University of Delaware
Newark, DE 19716, USA
vijay@cis.udel.edu

Abstract

In order to extend Information Processing technology to new languages, we are developing a system that automatically generates an LTAG grammar from an abstract specification of a language. Our approach uses language-independent specifications that can be tailored to specific languages by eliciting linguistic information from native informants, thus partially automating the grammar development process.

1 Introduction

Lexicalized Tree Adjoining Grammar (LTAG) is a tree-rewriting formalism. It is more expressive than a context-free grammar (CFG), and therefore a better formalism for representing various phenomena in natural languages. In the last decade, it has been applied to various NLP tasks such as parsing (Srinivas, 1997), machine translation (Palmer et al., 1998), information retrieval (Chandrasekar and Srinivas, 1997), generation (Stone and Doran, 1997; McCoy et al., 1992), and summarization applications (Baldwin et al., 1997), and substantial grammars now exist for French, German, Japanese and English. However, there is a substantial development time required for LTAGs, lessening their appeal for languages of low diffusion, such as Portuguese.

In this paper we present a system developed to generate an LTAG automatically from an abstract specification of a language. In addition to providing obvious benefits with respect to performing maintenance and ensuring consistency, we believe this approach has exciting potential for partially automating the LTAG development process. We have found that the abstract specification that lends itself most readily to automatic tree generation also corresponds closely to a division into language independent and language dependent properties.

2 LTAG Development System

LTAGs are based on the Tree Adjoining Grammar (TAG) formalism developed by Joshi, Levy, and Takahashi (Joshi et al., 1975). An important characteristic of an LTAG is that it is lexicalized, i.e., each lexical item *anchors* one or more tree structures that encode its subcategorization information. Trees with the same canonical subcategorizations are grouped into tree families. Figure 1 shows a few trees in the tree family for a verb such as *break* in English. Each individual tree includes two types of grammatical information: one is the subcategorization frame: *break* takes one argument in tree 1(a)-(c), and two in tree 1(d)-(e); the other type of information is the transformational information: tree 1(a) and 1(d) share the structure for declarative, tree 1(b) and 1(e) for wh-movement, and 1(c) and 1(f) for relativization.

As the size of the grammar grows, developing and maintaining those trees by hand faces two major problems: first, the reuse of tree substructures in many elementary trees creates redundancy. To make certain change of the grammar, all the related trees have to be manually checked. The process is inefficient and can not guarantee the consistency(Vijay-Shanker and Schabes, 1992);[1] second, the underlying linguistic information is not expressed explicitly. As a result, from the grammar itself (i.e. a set of thousands of trees), it is hard to grasp the characteristics of a particular language, to compare languages, and to build a grammar for a new language given existing grammars for other languages.

Our system aims at solving those problems. It requires the grammar developers to state the linguistic information explicitly and assumes the syntactic information of a language can be represented in three types of specifications: subcategorization frames, lexical redistribution rules (LRRs), and tree descriptions which are called *blocks* in our system. Blocks are further divided into subcategorization blocks and transformation blocks according to their functions. To produce the grammar, our system takes those specifications as the input and combines them to automatically generate the elementary trees.

Figure 2 shows the framework of the system. The input to the system are marked by * and in bold

[1]For a discussion of other approaches that address this issue, (Becker, 1994; Evans et al., 1995; Candito, 1996), see (Xia et al., 1998).

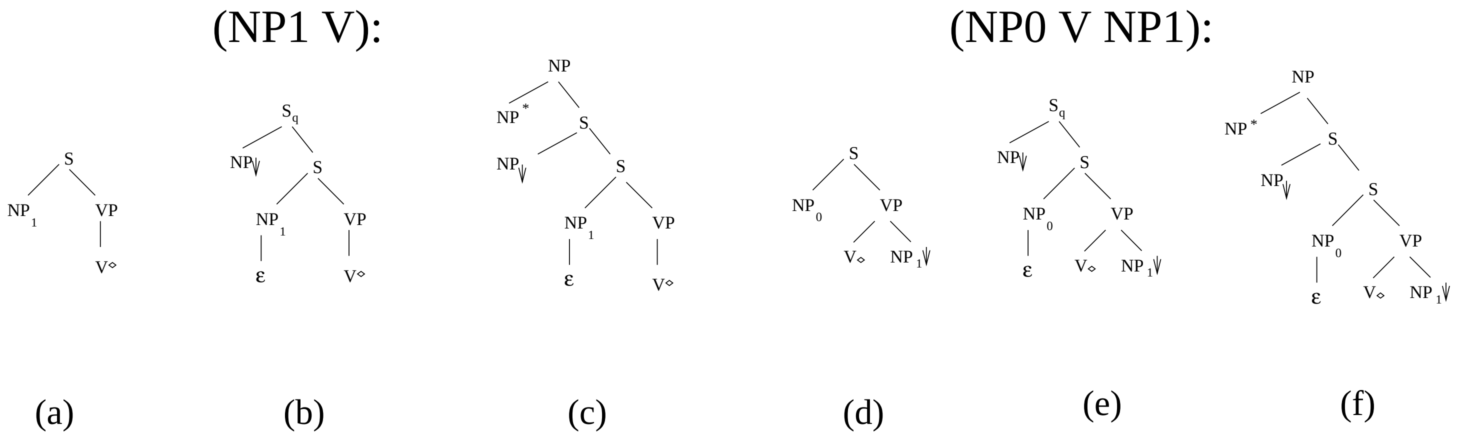


Figure 1: Some elementary trees that *break* anchors

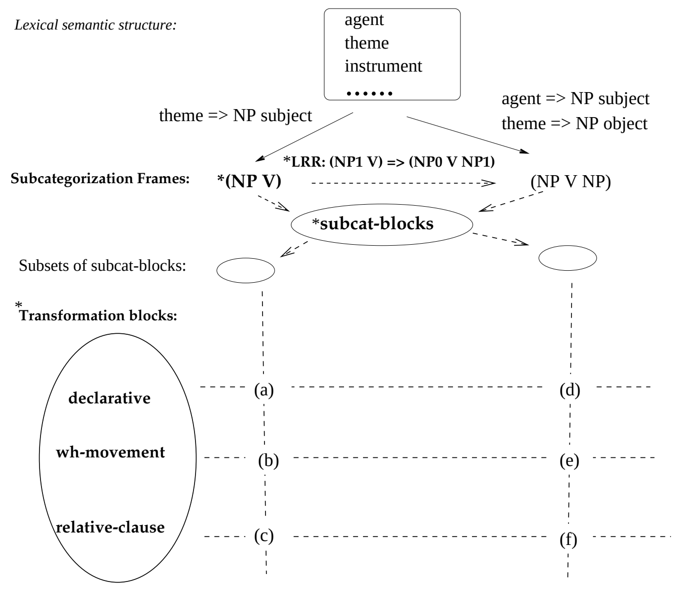


Figure 2: The framework of the system

font. The output are elementary trees, marked as (a)-(f), corresponding to the ones in Figure 1. Conceptually, each verb has a lexical semantic representation, which lists the thematic roles that the verb has. Those roles can be realized in various ways in syntax. For example, the theme can be realized as the subject (e.g. *The window broke*) as in the first subcategorization frame, or the object (e.g. *Mike broke the window*) as in the second subcategorization frame. Because it is not clear what kind of the lexical semantic representation is appropriate for each verb, our current system does not include that part. Instead, we assume there is a canonical subcategorization frame, and other frames can be derived from it by applying Lexical Redistribution Rules(LRRs).

(a) (NP1 V) (b) (NP0 V NP1)

Figure 3: Two subcategorization frames that *break* anchors

$$(NP1\ \ V) \Rightarrow (NP0\ \ V\ \ NP1)$$

Figure 4: The ergative LRR

2.1 Input to the System: Three Types of Specifications

Three types of specifications are defined more precisely below.

2.1.1 Subcategorization Frames:

Subcategorization frames specify the category of its anchor, the number of its arguments, each argument's category and other information such as feature equations.

2.1.2 Lexical Redistribution Rules (LRRs):

Lexical Redistribution Rules (LRRs) specify the relations between subcategorization frames. An LRR is a pair of subcategorization frames. It can be seen as a function that takes a subcategorization frame as the input and generates a new frame as the output. For example, The LRR shown in Figure 4 creates the subcategorization frame *(NP V NP)* when it is applied to the frame *(NP V)*.

2.1.3 Blocks

Blocks are tree descriptions specified in a logical language patterned after (Rogers and Vijay-Shankar, 1994). A block specifies categorical labels of nodes, feature value assignments, and structural relationships between nodes. There are four types of structural relations: dominance, immediate dominance (i.e. parent), strictly dominance, and precedence. Figure 5 and 6 are some blocks used in English

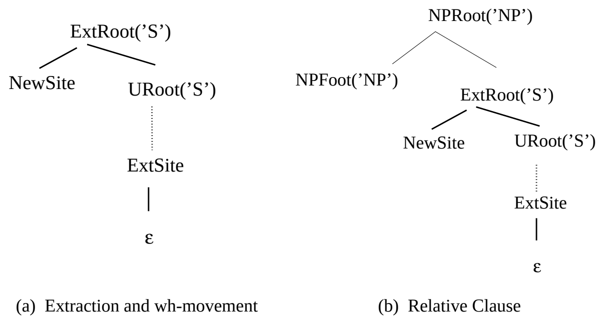


Figure 6: Transformation block for extraction

XTAG grammar.[2]

Blocks are very similar to elementary trees except that the former can leave some information unspecified. For example, when x dominates y, the number of intermediate nodes between x and y is unspecified. Elementary trees can be seen as a combination of blocks where all the structural relations between each pair of nodes are totally specified.

Blocks are divided into two types according to their functions: subcategorization blocks and transformation blocks. The former describes the structural configuration incorporating the various arguments in a subcategorization frame in their canonical positions.[3] Some of the subcategorization blocks used in the development of the English grammar are shown in Figure 5. For example, the *is_main_frame* block describes the spine of a clause. While in most cases the verb will be the anchor (specified in *main_anchor_is_verb*) , *is_main_frame* does not equate the anchor with the verb. This allows for the analysis used in the Penn English LTAG where a noun or an adjective can serve as an anchor in the tree for small clauses.

The transformation blocks are used for various transformations such as wh-movement.[4] Figure 6(a) depicts our representation of phrasal extraction. This can be specialized to give the blocks for wh-movement, relative clause formation, etc. For example, relative clause, as in Figure 6(b), is defined by further specifying that the ExtRoot modifies an NP node. wh-movement is the same as the phrasal extraction except the node NewSite has a +wh feature, which is not shown in the Figure 6(a).

[2]In order to focus on the use of tree descriptions and to make the figures less cumbersome, we show only the structural aspects and do not show the feature specification. Dotted lines, solid lines and dash-dotted lines denote dominance, immediate dominance and strictly dominance relation respectively. The arc between nodes shows the precedence order of the nodes are unspecified. The nodes' categories are enclosed in parentheses.

[3]Here *canonical* positions roughly corresponds to the positions in deep structure in GB-theory.

[4]These transformation blocks do not encode rules for modifying trees, but rather describe the properties of a particular syntactic construction.

2.2 Tree Generation from the Specification

Once we introduce the notion of LRRs, a *tree family* is defined as the set of elementary trees with the same "canonical" subcategorization frame.[5] Given a subcategorization frame f and sets of LRRs and blocks, to generate a tree family for that frame, the system takes several steps:

1. **Derive subcategorization frames** :
   Apply sequences of LRRs to f and generate a set $Fset$ of the related subcategorization frames.
2. **Select subcategorization blocks** :
   For each frame f_i in Fset, select a set of subcategorization blocks $SBset_i$.[6]
3. **Combine with transformation blocks** :
   For each subset $TBset_j$ of transformation blocks,[7] combine it with $SBset_i$ to form a new set of blocks $Bset_{i,j}$.
4. **Generate trees** :
   For each $Bset_{i,j}$, generate a set of elementary trees that are consistent with the tree description in $Bset_{i,j}$. If there are more than one tree, choose the ones with the minimal number of nodes.

Given the frame *(NP V)* in Figure 3(a), the LRR in Figure 4, and blocks in Figure 5 and 6, the system will generate the same trees as the ones in Figure 1 plus the trees in Figure 7. For instance, the trees in Figure 1(f) and 7(b) are automatically generated by applying the LRR to the frame *(NP V)* to get the frame in 3(b), then choosing the subcategorization blocks in Figure 5, and next combining them with the relative-clause block in 6(b), as shown Figure 2.

[5]As mentioned before, if we introduce the notion of lexical semantic representation, we don't have to assume the existence of a canonical subcategorization frame. Rather, we can define *tree family* as the set of elementary trees with the same underlying lexical semantics representation.

[6]The system uses a default mapping from the information in the frame to the name of subcategorization blocks. For example, if the anchor in the frame takes an NP subject, the system will select the blocks *pred_has_subject* and *subject_is_NP*. The default mapping can be easily modified.

[7]Theoretically, the system can try all the the subsets of transformational blocks. However, some transformation blocks are *incompatible* in that the combinations of them will fail to produce any elementary tree. For example, an elementary tree may use the block for wh-movement or relative-clause, but it will never use both at the same time. The system can rule out those combinations in tree generation stage, but for the purpose of efficiency, instead of letting the system try those combinations and fail, the grammar developer can partition the transformation blocks into several parts such as the blocks in the same part are incompatible. The system will take the partition and only try the subsets where each block in the subset comes from different parts.

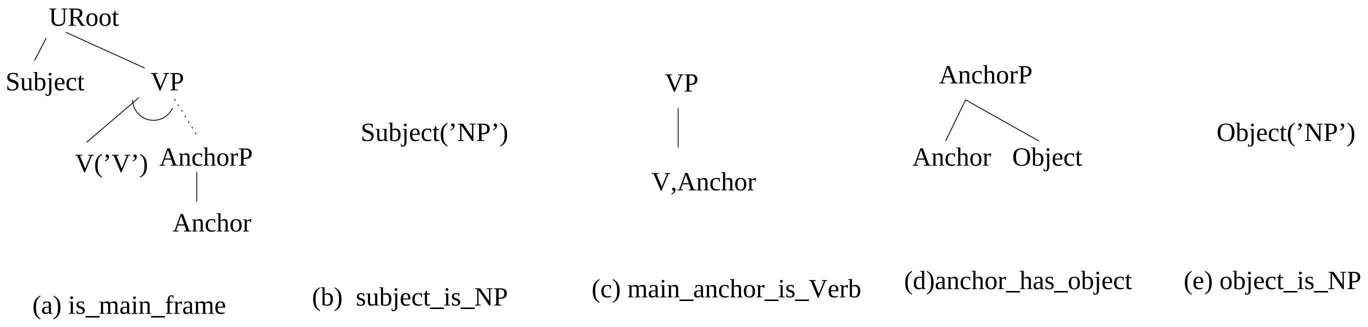


Figure 5: Some subcategorization blocks selected by the canonical frame of transitive verbs

	English	Chinese
LRRs	passive dative-shift ergative etc.	existential-const causative ergative
trans blocks	wh-question relativization gerund etc	topicalization relativization arg-drop etc.
# LRRs	6	8
# subcat blocks	34	24
# trans blocks	8	15
# tree families	43	35
# trees	638	482

Table 1: Major features of English and Chinese grammars

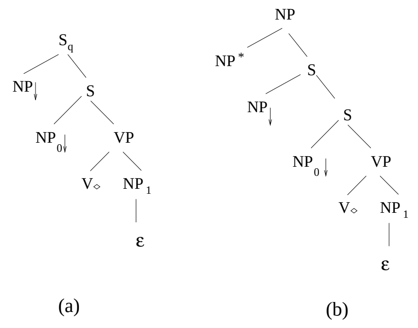


Figure 7: Trees generated by the system besides the ones in Figure 1

2.3 Building Grammar

We have used our system to develop grammars for English and for Chinese. The major features of these two grammars are summarized in Table 1.

The specification of LRRs and blocks in our system highlight the similarities and differences between languages. For example, both languages have relative-clauses, imperatives, etc. As a result, both grammars have similar LRRs or blocks for these phenomena. For phenomena that occur in one language, only that language will have the corresponding LRRs or blocks, such as the argument-drop block in Chinese, and the dative-shift LRR and the gerund block in English.

3 Eliciting language specific information

Section 2 described the procedure used to automatically build a grammar by combining subcategorization frames, LRRs, and blocks.[8] It presumes that the user provides this information to the system. Defining such information from scratch for a new language is easier than building all the elementary trees by hand, but it is still a difficult and time-consuming task. Bracketed corpora such as Penn Treebank(Marcus et al., 1993) can be used for extracting those information if such corpora exist, as shown in (Xia, 1999), but quite often such corpora are not available for low diffusion languages.

A central assumption in the field of formal syntax is there exists a universal grammar, and the difference among languages can be captured by different setting of a parameter list. Based on this assumption, We have extended our system to include language-independent structures. Our goal is to couple these language-independent structures with an interface which elicits language-dependent details from a native speaker. These language-dependent details instantiate certain parameter settings, and

[8]An LTAG may also include trees for modification and conjunction. Our system can easily produce those trees from other forms of abstract specification. This is omitted from the paper due to space limitations.

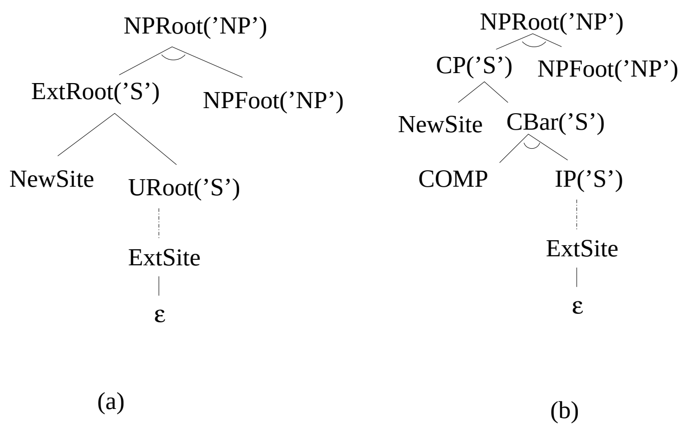


Figure 8: The possible meta-blocks for relative clause

thus generate blocks tailored to the specific language. The grammar developer may still need to add additional details to these blocks, but the development time should be shortened significantly. In this section, we will illuminate the way how transformation blocks are built by this process. Other kinds of specifications can be elicited similarly.

To build a transformation block, we start with the definition of the corresponding phenomenon, which is language-independent. For example, *relative clause* can be roughly defined as *an NP is modified by a clause in which one constituent is extracted (or co-indexed with an operator).* We build a tree description (for clarity, we will call it *meta-block*) according to the definition. Notice the exact shape of the meta-block often depends on the theory. For example, both meta-blocks in Figure 8 are consistent with the definition of relative clause, the former follows the way that Penn XTAG group treats the complementizer(COMP) as adjunct, the latter follows more closely to the GB theory where COMP is the functional head of CP. The meta-block must be general enough to be language-independent. Next, the system will recognize parts in the meta-block that are not fully specified and prompt the user for answers. Then, add those information to be meta-block so it is tailored to our target language. Meta-blocks plus language-specific information form our transformation blocks for that language.

For example, Figure 8 shows the possible meta-blocks for relative clauses. Table 2 lists the questions about those meta-blocks and the answers in four languages. In relative clause, relative pronoun(RelPron) occupies the position marked by NewSite. If we start with the meta-block in 8(b), the second part of the questions under double lines should also be used. The corresponding blocks are shown in Figure 9 if we start with the meta-block in Figure 8(a).[9]

Several points are worth noting. First, the setting of some parameters follows from higher-level generalizations and some pairs of parameters are related. For example, the position of NPFoot follows from the head position in that language. Korean is a SOV language, so we can infer the position of the NPFoot without asking native speakers. Second, the setting of the parameters provides a way of measuring the similarities between the languages. According to the settings, Chinese is more similar to Korean than to English.

A word of caution is also in order. Both the construction of the meta-block and the correct answers to the questions require some degree of linguistic expertise. Also, certain language specific details can not be easily expressed as yes-no questions. For example, the answers marked with *** mean they are true only under certain conditions which need more specification, e.g. in English, COMP and RelPron can be both dropped only when the relativized NP is not the subject.

4 Conclusion

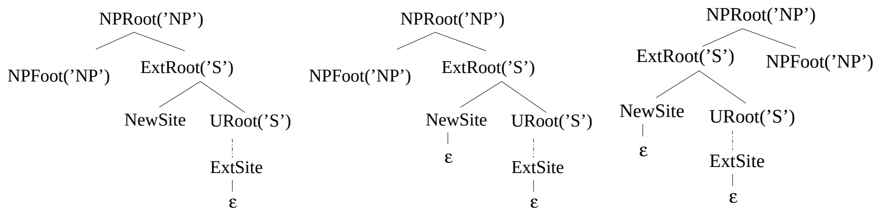
In summary, we have presented an LTAG development system that shows interesting promise with respect to semi-automating the grammar development process. The LTAG generation is driven by an abstract specification of different types of linguistic information: subcategorization frames, blocks and lexical redistribution rules. An appropriate elicitation process can glean this linguistic information from the user, thus allowing the system to semi-automatically begin the definition of the abstract specification, and actively supporting the user during the development process. The abstract level of representation for the grammar both necessitates and facilitates an examination of the linguistic assumptions. This can be very useful for gaining an overview of the theory that is being implemented and exposing gaps that remain unmotivated and need to be investigated. The grammar development then becomes an interactive process between the system and the language expert, with the system assisting in the precise definition of the linguistic categories, and then highlighting areas that need further definition. Once an LTAG is built by the system, all the IP technologies developed for LTAG become readily available.

The system has benefits building translation tools as well as grammar development, since the language dependent properties of a language will be clearly

[9]The blocks for relative clause in English and Portuguese are the same as shown in Figure 9(a) and 9(b) but English and Portuguese differ in one aspect: when the ExtSite is not the subject, in English, both COMP and NewSite are optional, but in Portuguese, one of them must be present. The difference is captured by features which are not shown in the figure.

	English	Portuguese	Chinese	Korean
position of NPFoot?	left	left	right	right
overt wh-movement?	yes	yes	no	no
has overt RelPron?	yes	yes	no	no
RelPron can be dropped?	yes*	yes*	-	-
position of COMP?	left	left	right	suffix
COMP can be dropped?	yes*	yes*	yes*	no
COMP and RelPron co-occurs?	no	no	-	-
COMP and RelPron both be dropped?	yes*	no	-	-

Table 2: Settings for relative clauses in four languages



(a) English and Portuguese with Relative Pronoun (b) English and Portuguese without Relative Pronoun (c) Chinese and Korean

Figure 9: The blocks for relative clauses in four languages

specified, and can be contrasted with those of other languages. By focusing on syntactic properties at a higher level, our approach allows new opportunities for the investigation of how languages relate to themselves and to each other.

References

Breckenridge Baldwin, Christine Doran, Jeffrey Reynar, Michael Niv, B. Srinivas, and Mark Wasson. 1997. EAGLE: An Extensible Architecture for General Linguistic Engineering. In *Proceedings of RIAO97*, Montreal.

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

Marie-Helene Candito. 1996. A principle-based hierarchical representation of ltags. In *Proceedings of COLING-96*, Copenhagen, Denmark, August.

R. Chandrasekar and B. Srinivas. 1997. Gleaning information from the web: Using syntax to filter out irrelevant information. In *Proceedings of AAAI 1997 Spring Symposium on NLP on the World Wide Web*.

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(ACL '95)*, Cambridge, MA.

Aravind K. Joshi, L. Levy, and M. Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and System Sciences*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2).

K. F. McCoy, K. Vijay-Shanker, and G. Yang. 1992. A functional approach to generation with tag. In *Proceedings of the 30th ACL*.

Martha Palmer, Owen Rambow, and Alexis Nasr. 1998. Rapid prototyping of domain-specific machine translation system. In *Proceedings of ATMA-98*, Langhorne, PA, October.

James Rogers and K. Vijay-Shankar. 1994. Obtaining Trees from their Descriptions: An Application to Tree Adjoining Grammars. *Computational Intelligence*, 10(4).

Bangalore Srinivas. 1997. *Complexity of Lexical Descriptions and Its relevance to Partial Parsing*. Ph.D. thesis, University of Pennsylvania.

Matthew Stone and Christine Doran. 1997. Sentence planning as description using tree adjoining grammar. In *Proceedings of the 35th ACL*.

K. Vijay-Shanker and Yves Schabes. 1992. Structure sharing in lexicalized tree adjoining grammar. In *Proceedings of the 15^{th} International Conference on Computational Linguistics (COLING '92)*, Nantes, France, August.

F. Xia, M. Palmer, K. Vijay-shanker, and J. Rosenzweig. 1998. Consistent grammer development using partial-tree descriptions for lexicalized tree-adjoining grammar. In *4th International Workshop on TAG and Related Frameworks*.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of NLRPS-99*.