Evolution of the XTAG System

Christine Doran Beth Ann Hockey Anoop Sarkar
B. Srinivas and Fei Xia*
Institute for Research in Cognitive Science
University of Pennsylvania
Philadelphia, PA 19104-6228, USA
{cdoran, beth, anoop, srini, fxia}@unagi.cis.upenn.edu

1 Introduction

The XTAG Project has been ongoing at Penn in some form or another since 1987. It began with a toy grammar run on LISP machines, and has since developed into a full-scale system with a large English grammar, small grammars for several other languages, a sophisticated X-windows based grammar development environment and numerous satellite tools. Approximately 35 people have worked extensively on the system, and at least that many have worked more peripherally. Thus, while it is not a geographically distributed project, it has been temporally distributed. At any given time, there is no single person who is completely familiar with all aspects of either the grammar or the tool kit. As a result, careful documentation has proven to be invaluable. Historically, this has taken the form of distinct papers on individual components; this is still the case for the tools. For the grammar, however, there is now a single document, available as a (frozen) technical report (XTAG-Group, 1995) or a constantly updated HTML document.[1] The technical report has been useful not only for the people working on the project at Penn, but also for those outside of Penn who are either interested in Tree Adjoining Grammar specifically, or simply interested in seeing how we handled some particular aspect of the grammar.

The system has language independent components such as the LTAG parser, the X-windows development environment, and the maintenance tools. It also has language dependent components such as lexical and tree structures for several languages, and the morphological database and part of speech tagger for English. The rest of this section will discuss some of the major system components in greater detail. The remainder of the paper will focus on the English grammar. We also have a large French grammar (started at Penn and expanded at Paris 7, by Anne Abeillé) and others,

*Thanks to Aravind Joshi for his support of the XTAG project and his assistance on this paper, and to the other members of the project, past and present, whose work has contributed to the XTAG System. This work was partially supported by ARO grant DAAH04-94-G-0426, ARPA grant N66001-94-C-6043, and NSF STC grant DIR-8920230.

[1] Both are freely available from the project's web page, at http://www.cis.upenn.edu/~xtag.

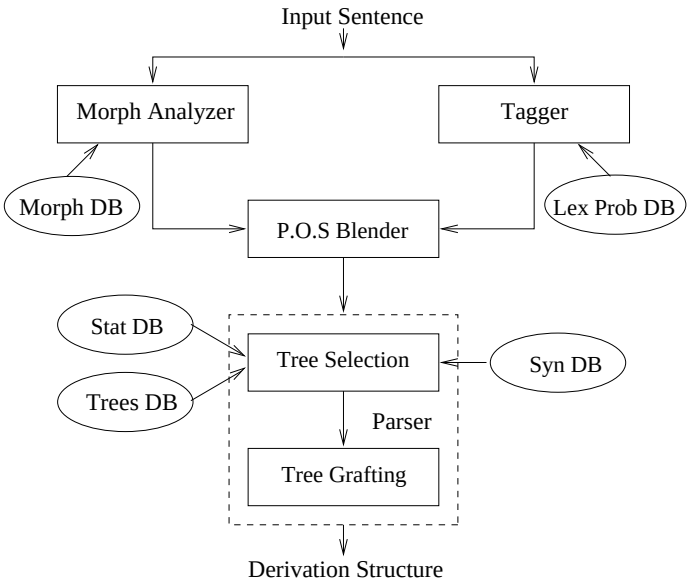


FIGURE 1 Overview of XTAG system

and small grammars for Korean, Chinese and Hindi. The X-windows interface is completely modular and can be (and has been) used with any of these grammars.

1.1 Formalism

The XTAG system uses a feature-based, lexicalized TAG (and everywhere we say LTAG, read "FB-LTAG"). For more detailed discussion about the formalism the reader is directed to the following references: (Joshi, 1987) gives an introduction to the TAG formalism, (Vijay-Shanker and Joshi, 1991) describes the use of unification features in a TAG, the advantages of lexicalization are discussed in (Schabes and Joshi, 1991). We do not use multi-component trees (Weir, 1988), although we simulate tree-local multi-component adjunction using features for auxiliary inversion (Hockey and Srinivas, 1993).

1.2 System Description

Figure 1 shows the overall flow of the system when parsing a sentence; a summary of each component is presented in Table 1. At the heart of the system is a parser for lexicalized TAGs (Schabes and Joshi, 1988, Schabes, 1990) which produces all legitimate parses for the sentence. The parser has two phases: **Tree Selection** and **Tree Grafting**.

Component	Details
Morphological Analyzer and Morph Database	Consists of approximately 317,000 inflected items derived from over 90000 stems. Entries are indexed on the inflected form and return the root form, POS, and inflectional information.
POS Tagger and Lex Prob Database	Wall Street Journal-trained trigram tagger (Church, 1988) extended to output N-best POS sequences (Soong and Huang, 1990). Decreases the time to parse a sentence by an average of 93% .
Syntactic Database	More than 105,000 entries. Each entry consists of: the uninflected form of the word, its POS, the list of trees or tree-families associated with the word, and a list of feature equations that capture lexical idiosyncrasies.
Tree Database	886 trees, divided into 48 tree families and 187 individual trees. Tree families represent subcategorization frames; the trees in a tree family would be related to each other transformationally in a movement-based approach.
X-Interface	Menu-based facility for creating and modifying tree files. User controlled parser parameters: parser's start category, enable/disable/retry on failure for POS tagger. Storage/retrieval facilities for elementary and parsed trees. Graphical displays of tree and feature data structures. Hand combination of trees by adjunction or substitution for grammar development. Ability to manually assign POS tag and/or Supertag before parsing

TABLE 1 System Summary

1.3 Tree Selection

Since we are working with lexicalized TAGs, each word in the sentence selects at least one tree. The advantage of a lexicalized formalism like LTAGs is that rather than parsing with all the trees in the grammar, we can parse with only the trees selected by the words in the input sentence.

In the XTAG system, the selection of trees by the words is done in several steps. Each step attempts to reduce ambiguity, i.e. reduce the number of trees selected by the words in the sentence.

Morphological Analysis and POS Tagging The input sentence is first submitted to the **Morphological Analyzer** and the **Tagger**. The morphological analyzer (Karp *et al.*, 1992) consists of a disk-based database (a compiled version of the derivational rules) which is used to map an inflected word into its stem, part of speech and feature equations corresponding to inflectional information. These features are inserted at the anchor node of the tree eventually selected by the

stem. The POS Tagger can be disabled in which case only information from the morphological analyzer is used.

POS Blender The output from the morphological analyzer and the POS tagger go into the **POS Blender** which uses the output of the POS tagger as a filter on the output of the morphological analyzer. Any words that are not found in the morphological database are assigned the POS given by the tagger.

Syntactic Database The syntactic database contains the mapping between particular stem(s) and the tree templates or tree-families stored in the **Tree Database** (see Table 1). The syntactic database also contains a list of feature equations that capture lexical idiosyncrasies. The output of the POS Blender is used to search the **Syntactic Database** to produce a set of lexicalized trees with the feature equations associated with the word(s) in the syntactic database unified with the feature equations associated with the trees. Note that the features in the syntactic database can be assigned to any node in the tree and not just to the anchor node.

Default Assignment For words that are not found in the syntactic database, default trees and tree-families are assigned based on their POS tag.

Filters Some of the lexicalized trees chosen in previous stages can be eliminated in order to reduce ambiguity. Two methods are currently used: structural filters which eliminates trees which have impossible spans over the input sentence and a statistical filter based on unigram probabilities of non-lexicalized trees (from a hand corrected set of approximately 6000 parsed sentences).

The **Tree Database** contains the tree templates that are lexicalized by following the various steps given above. The lexical items are inserted into distinguished nodes in the tree template called the *anchor nodes*. The part of speech of each word in the sentence corresponds to the label of the anchor node of the trees. Hence the tagset used by the POS Tagger corresponds exactly to the labels of the anchor nodes in the trees. The tagset used in the XTAG system is given in Table 2. The tree templates are subdivided into tree families (for verbs and other predicates), and tree files which are simply collections of trees for lexical items like prepositions, determiners, etc[2].

1.4 Tree Grafting

Once a particular set of lexicalized trees for the sentence have been selected, XTAG uses an Earley-style predictive left-to-right parsing algorithm for LTAGs (Schabes and Joshi, 1988, Schabes, 1990) to find all derivations for

[2]The nonterminals in the tree database are `A, AP, Ad, AdvP, Comp, Conj, D, N, NP, P, PP, Punct, S, V, VP`.

the sentence. The derivation trees and the associated derived trees can be viewed using the X-interface (see Table 1). The X-interface can also be used to save particular derivations to disk.

Part of Speech	Description
A	Adjective
Ad	Adverb
Comp	Complementizer
D	Determiner
G	Genitive Noun
I	Interjection
N	Noun
P	Preposition
PL	Particle
Punct	Punctuation
V	Verb

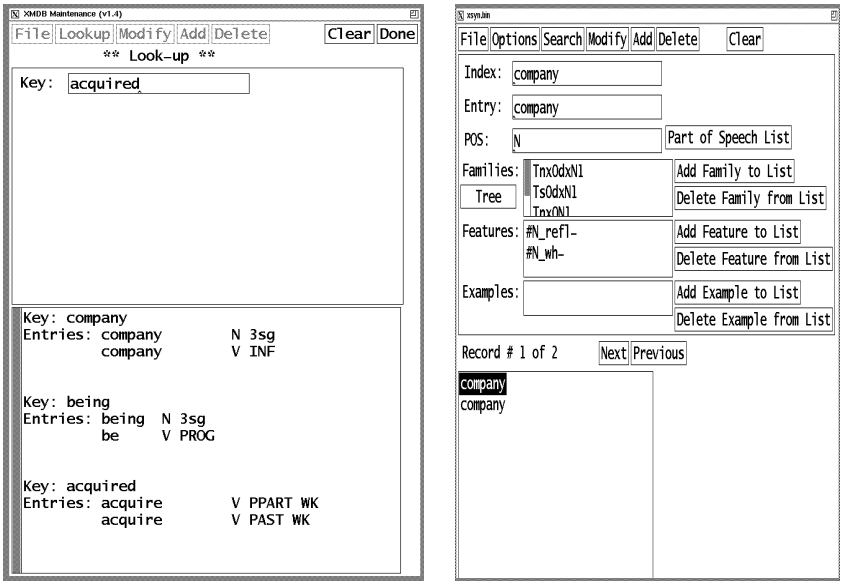
TABLE 2 XTAG tagset

1.5 The Grammar Development Environment

Working with and developing a large grammar is a challenging process, and the importance of having good visualization tools cannot be overemphasized. Currently the XTAG system has X-windows based tools for viewing and updating the morphological and syntactic databases (Karp *et al.*, 1992, Egedi and Martin, 1994). These are available in both ASCII and binary-encoded database format. The ASCII format is well-suited for various UNIX utilities (awk, sed, grep) while the database format is used for fast access during program execution. However even the ASCII formatted representation is not well-suited for human readability. An X-windows interface[3] for the databases allows users to easily examine them. Searching for specific information on certain fields of the syntactic database is also available. Also, the interface allows a user to insert, delete and update any information in the databases. Figure 2(a) shows the interface for the morphology database and Figure 2(b) shows the interface for the syntactic database.

XTAG also has a sophisticated parsing and grammar development interface (Paroubek *et al.*, 1992). This interface includes a tree editor, the ability to vary parameters in the parser, work with multiple grammars and/or parsers, and use metarules for more efficient tree editing and construction (Becker, 1994). The interface is shown in Figure 3.

[3]The interface uses the MIT Athena Toolkit, which is distributed with the standard MIT X release.



(a) Morphology database (b) Syntactic database

FIGURE 2 Interfaces to the database maintenance tools

2 Development Methodologies

Extending the English Grammar requires expansion of two different types of data: associations of lexical items with syntactic frames and the trees encoding those syntactic frames. Section 2.1 describes extension of the former and Section 2.2 the later.

2.1 Lexicon Development

There have been two major expansions of the lexicon from the original toy grammar. The first expansion used automatic extraction/translation from online dictionaries to produce syntactic entries for the lexicon. This effort had mixed success due to two problems with using dictionaries for lexical information. One problem is that dictionaries often do not supply enough of the right type of information, and the resulting lexical entries can be grossly under-specified. A more serious problem is that dictionaries can misclassify lexical items as to subcategorization. We had a very trying instance of this second type of problem with the particle verbs. Many of the transitive particle verb entries generated by the automatic procedure were for items that did not undergo particle movement (e.g. *put off the paper*, *put the paper off*) or show any other signs of being particle verbs. Consequently, the particle verb portion of the lexicon had to be completely redone by hand.

The second large scale expansion of the lexicon was executed by com-

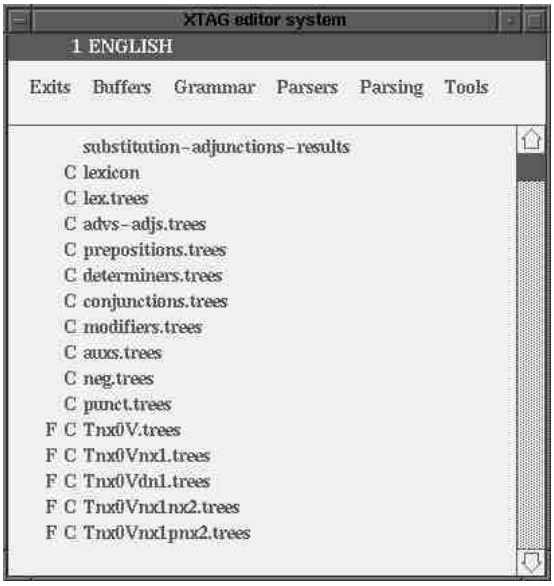


FIGURE 3 Interface to the XTAG system

paring the XTAG lexicon to COMLEX (COMLEX is an English lexicon developed at New York University under Linguistic Data Consortium sponsorship (Grishman *et al.*, 1994)). A separate lexicon was created containing the subset of COMLEX items that were in not in the XTAG lexicon. The XTAG lexicon and the COMLEX subset lexicon can be used together to provide the lexical coverage of both.[4]

In addition to these two large expansions of the lexicon, additional entries are added as part of the implementation of new syntactic analyses. We use a variety of sources to compile lists of items to be added. For example, the subordinating conjunctions were taken from a descriptive grammar (Quirk *et al.*, 1985) and the lists of nouns that anchor partitive constructions was compiled from the same descriptive grammar, a thesaurus, and from online corpora.

2.2 Tree Building

In the early days of the project, when the system was running on LISP machines, the graphical interface was quite unstable necessitating hand editing of trees in LISP as text files, particularly for large scale changes. Since the development of the X-windows interface, trees have been more easily constructed by hand using its tree editing tools. Even with the improved interface, we have found that we still needed a more "batch process"

[4]It was necessary to maintain separation between the COMLEX-based lexicon and the XTAG lexicon because we are not able to distribute COMLEX.

method to perform major changes across tree families, enforce consistency across the grammar and to create substantial numbers of new trees. We have, in the past, resorted to `emacs` macros or PERL scripts but recently we have incorporated two automatic methods for producing trees which are described in Section 4.

New syntactic analyses can require new tree structure in addition to new lexical items. Adding a new subcategorization frame to the grammar requires adding both new lexical entries and the corresponding trees. The question of whether something is an argument (and hence part of the subcategorization frame) or an adjunct is usually resolved by checking whether it is optional. If omission leads to ungrammaticality, it is treated as an argument[5]. For example, in digesting the COMLEX data, if there was a subcategorization in COMLEX for which there was no corresponding tree family in XTAG, the appropriate tree family was added. In other cases we add new trees to cover a syntactic phenomenon not handled by the grammar. In the early days of the grammar, phenomena were often added based on linguistic interest. Now the grammar is extensive enough that it handles most interesting and well known constructions. Table 2.2 lists the linguistic phenomena we currently cover. Phenomena which we are aware the grammar does not cover, some of which we are currently working on, are: predicate and non-constituent coordination; idioms; binding; semantic features (other than those on determiners); extraposition; and comparatives with gaps.

While we still add some analyses based on linguistic interest, for example "tough-movement" (e.g. *[The artichoke]$_i$ was easy to cook t_i*), we primarily use corpus analysis to direct our grammar extension efforts.

2.3 Using Corpora to Expand the Grammar

In order to continue to improve the coverage of XTAG, we periodically parse a batch of sentences from some corpus, and perform an error analysis on those which are rejected. Based on the results of this analysis, we prioritize upcoming grammar development efforts. The results of a recent error analysis are shown in Table 4. The table does not show errors in parsing due to mistakes made by the POS tagger which contributed the largest number of errors: 32. At this point, we have added a treatment of punctuation to handle #1, an analysis of time NPs (#2), a large number of multi-word prepositions (part of #3), gapless relative clauses (#7), bare infinitives (#14) and have added the missing subcategorization (#3) and missing lexical entry (#12). We are in the process of extending the parser to handle VP coordination (#9) (Sarkar and Joshi, 1996). We find that this method of error analysis is very useful in focusing our research efforts in a productive direction.

[5]Cases where something considered to be an adjunct is subject to movement are handled by an approximation to multi-component adjunction.

adjuncts	infinitives
appositives	inversion
auxiliaries	it-clefts
auxiliary contractions	multi-word prepositions and adverbs
bare infinitives	negation
clausal adjuncts	noun-noun modification
control constructions	noun-verb contraction
copular constructions	particle movement
determiner sequencing	passives
ECM	punctuation
ergatives	quoted speech
genitives	raising
gerunds	relative clauses
imperatives	small clauses
infinitives	time NPs
inversion	topicalization
it-clefts	wh- questions

TABLE 3 Phenomena Covered by the XTAG English Grammar

2.3.1 TSNLP

We also ran the English Grammar on the Test Suites for Natural Language Processing (TSNLP) English corpus (Lehmann *et al.*, 1996). The corpus is intended to be a systematic collection of English grammatical phenomena, including complementation, agreement, modification, diathesis, modality, tense and aspect, sentence and clause types, coordination, and negation. It contains 1409 grammatical sentences and phrases and 3036 ungrammatical ones.

Before parsing the grammatical subset of the TSNLP data, we made a few tokenization changes: we changed contractions from two tokens to one, downcased the first words of sentences, changed a pair of square brackets to parentheses and changed quotes to pairs of opens and closes. There were 42 examples which we judged ungrammatical, and removed from the test corpus. These were sentences with conjoined subject pronouns, where one or both were accusative, e.g. *Her and him succeed.* Overall, we parsed 61.4% of the 1367 remaining sentences and phrases. The errors were of various types, broken down in Table 5.

The missing lexical items are obviously the easiest of these to remedy, and we have added them. This class also highlighted the fact that our grammar is heavily slanted toward American English—our grammar did not handle *dare* or *need* as auxiliary verbs, and there were a number of very British particle constructions, e.g. *She misses him out.* The missing trees are slightly harder to address, but the data obtained here is very useful in helping us fill gaps in our grammar. Based on these results an analysis of the

Rank	No of errors	Category of error
#1	11	Parentheticals and appositives
#2	8	Time NP
#3	8	Missing subcat
#4	7	Multi-word construction
#5	6	Ellipsis
#6	6	Not sentences
#7	3	Relative clause with no gap
#8	2	Funny coordination
#9	2	VP coordination
#10	2	Inverted predication
#11	2	Who knows
#12	1	Missing entry
#13	1	Comparative?
#14	1	Bare infinitive

TABLE 4 Results of Corpus Based Error Analysis

Error Class	%	Example
POS Tag	19.7%	She adds to/V it , He noises/N him abroad
Missing lex item	43.3%	*used* as an auxiliary V, *calm NP down*
Missing tree	21.2%	*should've*, *bet NP NP S*, *regard NP as Adj*
Feature clashes	3%	*My every firm*, *All money*
Rest	12.8%	*approx*, *e.g.*

TABLE 5 Breakdown of TSNLP Errors

modal+*'ve* contractions was added to the grammar, along with a treatment of 'semi-modals' like *used*. The feature clashes are mostly in sequences of determiners, and would need to be looked at more closely to see whether the changes needed to correct them would do more harm than good. One general problem with the corpus is that, because it uses a very restricted lexicon, if there is one problematic lexical item it is likely to appear a large number of times and cause a disproportionate amount of grief. *Used to* appears 33 times and we got all 33 wrong. However, it must be noted that the XTAG grammar has analyses for syntactic phenomena that were not represented in the TSNLP test suite such as sentential subjects and subordinating clauses among others.

2.4 Where Do Analyses Come From

When we encounter a phenomenon that the English grammar does not handle, there are three basic approaches to building an LTAG analysis. If there is an existing analysis in some other formalism like GB or HPSG that is compatible with LTAG, we can **borrow** it directly. If there is no pre-existing analysis, we can **invent** our own. But the more common scenario

is that there are analyses, possibly conflicting, which are not completely compatible with LTAG, and we **merge** the existing analysis with our LTAG specific constraints. Each of these three scenarios is illustrated below.

2.4.1 Borrowed: Relative Clauses

Relative clauses have some uncontroversial features—an element is extracted from a clause, and a nominal modifier results; the extracted element is co-indexed with its trace—and some controversial ones—are they NP or N modifiers? Where in CP is the moved element?

Our analyses of relative clauses has changed twice over the course of the project. (Based on examples like *The teachers and the students who attended the party* we have chosen to make our relative clauses NP modifiers.) The basic elements have remained unchanged: the trees are adjunction structures anchored by a verb, with NP foot and root nodes, and there is a separate tree for each possible extraction site inside the clause.

(1) The artichoke which$_i$ Max cooked t$_i$.

(2) A person [whose mother]$_i$ t$_i$ cooks artichokes.

(3) A person [with whom]$_i$ Max cooked artichokes t$_i$.

(4) The artichoke that$_i$/$\emptyset_i$ Max cooked t$_i$.

The first analysis had two sets of relative clause trees. One had a substitution site for a relative pronoun (e.g. *who,which*, ex. (1)), extracted NP (e.g. *whose mother*, ex. (2)) or pied-piped PP (e.g. *at whose house*, ex. 3)), and the other allowed a complementizer to adjoin, handling the *that/nil* alternation (ex. 4). These trees are shown in Figure 4(a) and Figure 4(b), respectively.

However, the distinction between complementizers and relative pronouns seemed to us in retrospect to be an artificial one, so we collapsed the trees into a single set, with all relativizing elements adjoining (as in Figure 4(b)). This had the unfortunate consequence of disallowing extracted full NPs and PPs, and we also realized that both the past and current analyses had no consistent mechanism for co-indexing the extracted NP and its trace, since the extracted item was often adjoined. Neither analysis handled adjunct relative clauses.

The current analysis, developed and implement by Rajesh Bhatt, again has two sets of trees and is virtually identical to the GB analysis. Unlike the first analysis, nothing adjoins. There are two substitution positions,one for relative pronouns, full NPs or PPs (labelled NP or PP) and one for complementizers (overt or null, labelled Comp). In each tree, one of these positions is already filled by an empty element. The two are illustrated in Figure 4(c) and (d). Trees like 4(c) handle examples such as (1)-(3), while 4(d) handles the alternation exemplified in (4). This allows us to co-index the extraction site with its trace, since the extracted NP is always represented in the initial tree, while capturing the doubly-filled comp restriction. We

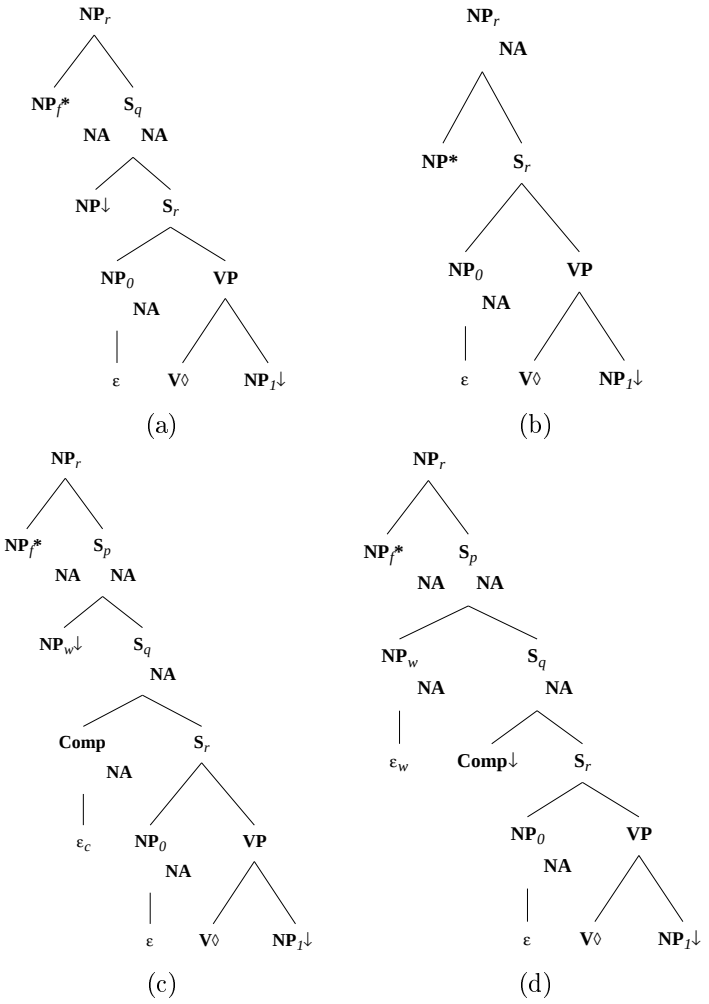


FIGURE 4 Three versions of transitive subject relative clauses

also now have gap-less adjunct relative clauses. A more detailed description of the analysis is given in the on-line version of the XTAG technical report, (XTAG-Group, 1995):Chapter 14.

2.4.2 Invented: Determiner Sequencing

While there is a vast quantity of literature on determiners, it has focused on aspects other than determiner sequencing, which was the issue of concern to us. Examples of possible and impossible orderings of determiners are shown below. Consequently, while we derived some inspiration from the literature, we had to develop an entirely new analysis for determiner sequencing.

The semantic properties of determiners as quantifiers has been a very active area of research in linguisitics. The starting point of our analysis of determiner sequencing was the realization that the semantic properties discussed in such research (e.g. (Keenan and Stavi, 1986, Barwise and Cooper, 1981)) when taken as features in our grammar would provide many of the necessary constraints for determiner sequencing. Our first implementation, the trees from which are shown in Figure 5, used several features from (Keenan and Stavi, 1986) plus agreement, genitive, and the wh- value of a word. The Wall Street Journal and Brown Corpus, online dictionaries, a descriptive grammar of English (Quirk *et al.*, 1985), and native speaker judgements were used as sources for developing a list of individual determiners to be accounted for, and for determining what sequences of determiners were possible. In this first implementation singular count nouns were differentiated from mass nouns in the grammar and the count nouns anchored an NP tree structure that contained a DetP node. One determiner anchoring a non-branching DetP substituted into the DetP position in the noun phrase. Additional determiners anchored an auxilliary tree that adjoined onto DetP.

While this first implementation handled the actual determiner sequencing well, there were other aspects of the analysis which were not satisfactory. There were problems both of linguistic and implementational elegance. The required division of nouns into count and mass seemed fairly unmotivated and in practice it was often difficult to make the assignment to one or the other category. The acceptability of a mass or count interpretation of a noun phrase seems to depend much more on context than on the particular noun involved, which argues for treating this as a problem of semantic interpretation rather than a syntactic problem. On the implementational front, the mass/count distinction forced us into the unpleasant situation of needing a count and non-count version of every tree in the grammar with a noun anchor.

Another problem with the first analysis was that it created an NP structure in which a functional category, the determiner, was selected by the noun. In the rest of the LTAG grammar, anchors select for their complements and functional categories adjoin. In addition, there has been debate

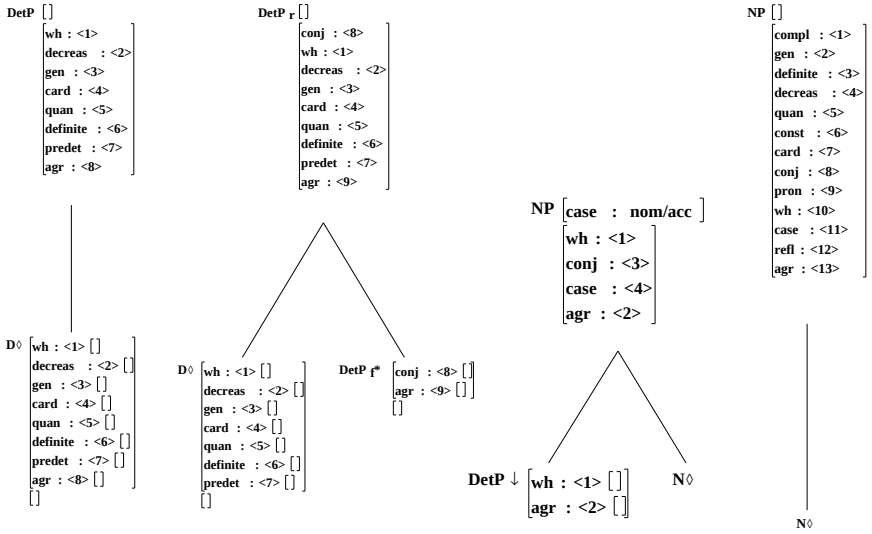


FIGURE 5 Trees for Constructing Simple NPs Under Substitution
Analysis of Determiners (old analysis)

about the structure of NPs in the linguistics literature fueled by the observations that both nouns and determiners exhibit selectional restrictions on each other and that different types of NPs (e.g. with and without determiners, pronouns, proper nouns) have somwhat different distributions. One influential approach to accounting for these observations was the DP hypothesis (Abney, 1987), which takes the head of what has traditionally been called an NP to be the determiner. While we wished to retain nouns as heads of NPs, it was clear that our first determiner sequencing analysis did not allow us to capture any of the insights of the DP-hypothesis.

Changing to an analysis in which determiners adjoined to NP, as illustrated in Figure 6, made the treatment of nouns uniform, eliminated many trees, allowed us to capture the insights of the DP hypothesis with respect to selection of determiners toward nouns, and gave the determiners a treatment that was consistent with other functional categories in our grammar. The features that had done well on the determiner sequencing in the first implementation transferred to the new adjunction analysis without a problem. In the course of changing the analysis we had an opportunity to fine tune and improve the features, which we took advantage of by reevaluating some of the features, adding further lexical items, and adding an additional feature to improve coverage. The current analysis is presented in the paper by (Hockey and Mateyek, in this volume).

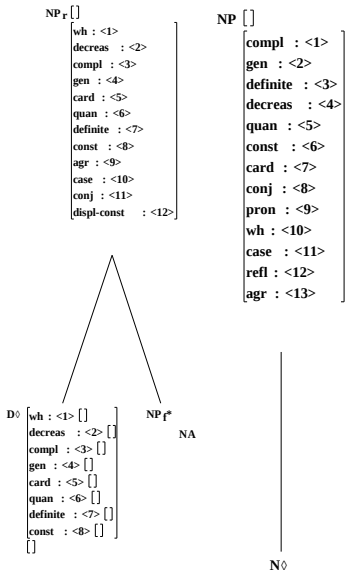


FIGURE 6 Trees for Constructing Simple NPs
Under Adjunction Analysis of Determiners (current analysis)

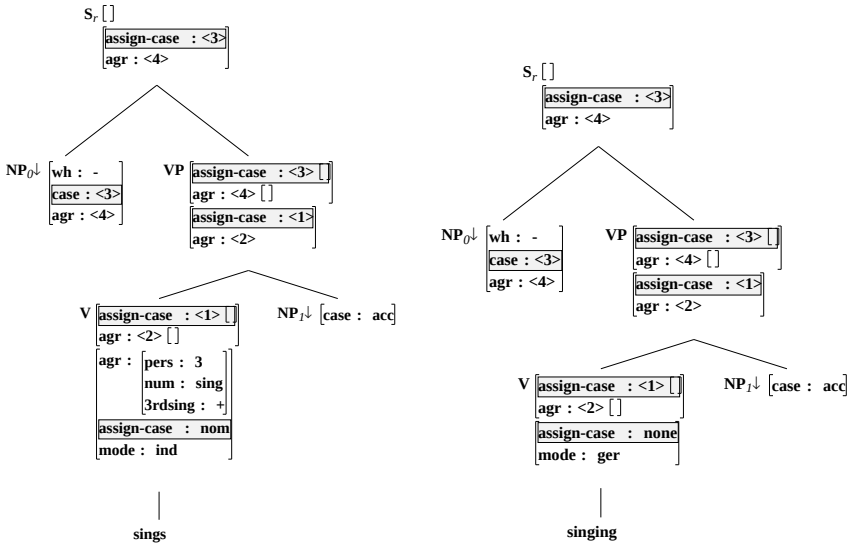
2.4.3 Merged: PRO Distribution

In the English XTAG grammar, we require that every clausal tree project all of the arguments of its head. This means that we do not allow VP trees in clausal complementation, as CCG and HPSG do. It also makes it natural to adopt the PRO mechanism along with other notions of case and the case filtering from GB theory.

Object case is treated as structural and is built into all transtive trees. The case assigned to the subject position varies with verb form. Since the XTAG grammar treats the inflected verb as a single unit rather than dividing it into INFL and V nodes, case, along with tense and agreement, is expressed in the features of verbs, and must be passed in the appropriate manner. The morphological form of the verb determines the value of the <**assign-case**> feature. Figures 7(a) and 7(b) show the same tree[6] anchored by different morphological forms of the verb *sing*, which give different values for the <**assign-case**> feature.

Inflected (indicative or imperative) verbs assign nominative case, and the remaining forms—infinitival, past participial, etc.—assign case **none**, as shown for the progressive form of the verb *sing* in Figure 7(b). The distinction between case **none** and no case is indicative of a divergence from the standard GB theory. In GB theory, the absence of case on an

[6] Again, the feature structures shown have been restricted to those that pertain to the V/NP interaction.



(a) (b)

FIGURE 7 Assigning case according to verb form

NP position means that only PRO can fill that position. However, with feature unification as it is used in the XTAG grammar, the absence of case on an NP means that *any* NP can appear there, regardless of its case. This is due to the mechanism of unification, in which if something is unspecified, it can unify with anything. Thus we must have a specific case **none** to handle verb forms that in standard GB would not assign case. PRO is the only NP with case **none**. Note that although we developed this treatment to allow our grammar to use PRO in a feature unification environment, our treatment is very similar to the assignment of null case to PRO in (Chomsky and Lasnik, 1993). (Watanabe, 1993) also proposes a very similar approach within Chomsky's Minimalist framework.

3 Translating the XTAG Grammar into Other Frameworks

The utility of the XTAG English Grammar extends beyond the XTAG project. It has been used as a resource for other "constituent-based" frameworks. There have been a number of efforts to use either the XTAG lexicon or just the trees to construct grammars in other formalisms. (Doran and Srinivas, in this volume), discuss building a large Combinatory Categorial Grammar by translating the trees into CCG categories and then translating the lexical associations with the categories as well. Yuka Tateisi of the University of Tokyo is working on converting a smaller version of the

English grammar into HPSG, and has gotten very promising parsing speed-ups over other HPSG parsers. (Evans *et al.*, 1995) encode the XTAG trees in DATR as lexical rules in a non-monotonic inheritance hierarchy.

4 Grammar Development Tools

As noted above, when a grammar reaches the size the XTAG English Grammar has, it is a challenge to maintain consistency across the grammar, and to make wide-ranging changes efficiently. Consistency suffers when changes are made to some trees of a certain type, but not to others. Making changes by hand to large numbers of trees is time-consuming, tedious, and risks introducing errors into the grammar. We have explored two automatic tools for constructing and maintaining an LTAG, which are described in the next sections.

4.1 Metarule implementation

In the XTAG English grammar, the trees for a class of verbs (which have the same subcategorization frame) are grouped into a **tree family**. Tree families include variations such as wh-questions, relative clauses, topicalized, and passive sentences. **Metarules** can be used to generate the trees in the tree family from the basic declarative tree. Becker's article in this volume describes metarules in more detail. By using metarules, the number of trees that have to be stated in an LTAG can be reduced considerably. Ideally, for every tree family only one representative tree, which could be called the **base** tree, has to be given; all the other trees can be derived by the application of metarules. Although one could do this at run time, our assumption is that it would be more practical to do it at compile time.

Becker's implementation of metarules has been incorporated into the XTAG system and is accessed via the X-interface, and has recently been used to perform the major change in our analysis of relative clauses described in Section 2.4.1. It is also currently being used by Carlos Prolo to generate the entire XTAG English grammar from base trees for each tree family. First, Prolo is checking for consistency within the existing grammar. The second phase will be to compare the metarule-produced grammar with that produced by the lexical organization tool described below, and to evaluate the relative strengths of the two approaches.

4.2 Lexical Organization

The XTAG English grammar currently consists of 886 tree templates, so grammar maintenance is no small task. One source of redundancy is the reuse of tree substructures in many different tree templates. This redundancy poses a problem for grammar maintenance and revision. For example, in every wh-question tree, there is a node under S which dominates a trace and is coindexed with wh-constituent at the sentence-initial position. If some changes are made in this substructure, all the trees that

use this substructure must be inspected and edited. Furthermore, one can only manually verify that such an update does not conflict with any other principle already instantiated. As the grammar grows, this difficulty of this task grows with it.

The main idea of the lexical organization is that instead of building elementary trees manually, we define *blocks*.[7] Each block abstractly describes all trees incorporating the partial structure it represents. Elementary trees are generated automatically from different combinations of the blocks. In maintaining the grammar, only the blocks need ever be manipulated; the larger sets of actual trees which they subsume are computed automatically from these high-level descriptions.

Similar approaches have been pursued for a large French LTAG by (Candito, 1996) and for the XTAG English grammar by (Becker, 1994). Following the ideas set forth in (Vijay-Shanker and Schabes, 1992), Candito constructs a description hierarchy in much the same way as the present work. Becker's meta-rules can also been seen as partial descriptions, wherein the inputs and outputs of the meta-rules are sisters in a description hierarchy and the parent is the common structure shared by both. However, there is still redundancy across meta-rules whose inputs apply to the same partial descriptions. For instance, the subject wh- extraction and subject relative metarules would be specified independently and both refer to an NP in subject position of a clause.

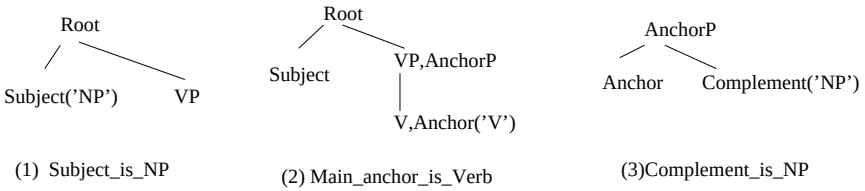
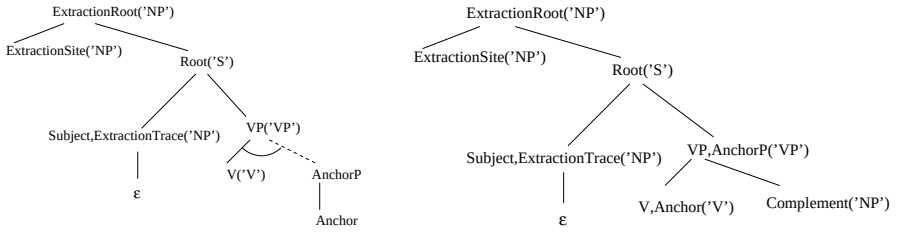


FIGURE 8 Subcategorization quasi-trees

4.2.1 Hierarchical Organization of the Current English Grammar

Lexical organization is used to build the tree templates for the English Grammar. We decompose an elementary tree into a conjunction of blocks which will be reused in the description of many other tree templates elsewhere in the grammar. The blocks are of two types: the subcategorization blocks and the ones for "transformations" such as wh-question formation. The former is further divided into four fairly orthogonal subtypes: (1) the

[7] Blocks are partial descriptions of trees specified in a logical language patterned after Rogers and Vijay-Shanker (1994). Since we are using a feature-based LTAG, our language has also been equipped with descriptive predicates allowing us to specify a tree's feature-structure equations, in addition to its structural characteristics.



(1) quasi-tree for relative clause (2) tree generated from the four quasi-trees

FIGURE 9 Quasi-tree for subject extraction in relative clause, and tree generated by combining it with the 3 quasi-trees in Figure 8

set of blocks describing the syntactic subject, (2) those for the main anchor(s), (3) those describing objects and (4) those for structure below a subject or object.

Consider, for example, the description of the relative clause tree for transitive verbs which contains four blocks: one specifying that its subject is extracted, one that the subject is an NP, one that the main anchor is a verb, and one that the complement is an NP. These blocks correspond to the quasi-trees (partially specified trees) shown in Figure 8, and 9(1), and when combined will generate the quasi-tree in Figure 9(2). For the sake of simplicity, feature equations are not shown. In these figures, solid lines and dashed lines denote the parent and dominance relations respectively; each node has a label, enclosed in parentheses, and at least one name. Multiple names for the same node are separated by commas such as *VP, AnchorP* in Figure 8(2). The arc in Figure 9(1) indicates that the precedence order of V and AnchorP is unspecified. (In small clauses, the main anchor is a preposition, adjective or noun, not a verb, so AnchorP and VP are not always the same node.)

Our system uses a logical tree description language implemented in Prolog to represent blocks, tree well-formedness constraints[8] and a node-minimality constraint (to reject trees which are consistent with set of descriptions, but has non-minimal number of nodes). There is also an interface to the Prolog module, written in Perl, and a visualization tool for displaying pieces of the description lattice, implemented in C and using the X-window graphical utility package. The system has been used to generate about 90% of tree families in the English grammar.

4.2.2 A tool for grammar examination

Being able to specify the grammar in a high-level description language has obvious advantages for maintenance and updating of the grammar, in that changes need only be made in one place and are automatically percolated

[8] e.g. the children of every node must be totally ordered with respect to precedence, a tree must have a unique node which has no parent, while all other nodes have exactly one parent, etc.

appropriately throughout the grammar. We expect to reap additional benefits from this approach when developing a grammar for another language. Beyond these issues of efficiency and consistency, this approach also gives us a unique perspective on the existing grammar as a whole. Defining blocks for the grammar both necessitates and facilitates an examination of the linguistic assumptions that have been made with regard to feature specification and tree-family definition. This can be very useful for gaining a overview of the theory that is being implemented and exposing gaps that have not yet been explained. Because of the organic way in which the grammar was built over the years, we have always suspected that there might exist a fair amount of inconsistency either within the feature structures, or within the tree families. The effort in organizing the lexicon has so far turned up very few non-linguistically motivated inconsistencies, which is a gratifying validation of the constraints imposed by the LTAG formalism.

Our work in tree organization has allowed us to characterize three principal types of exceptions in the XTAG English grammar: (1) a class of trees is missing from the grammar, though this class would be expected from allowing the descriptive blocks to combine freely (for example, a sentential subject with a verb anchor and a PP complement); (2) within a class of trees, some member is missing, though an analogous member is present in another class (extraction of the clausal complement of a noun-anchored predicative); (3) one tree in a class can be generated by combining quite general descriptions, but there is an exceptional piece of structure or feature equation (the ergative alternation of certain transitive verbs). While these may sometimes reflect known syntactic generalizations (e.g. extraction islands, as with the example in (2)), they may also reflect inconsistencies which have arisen over the lengthy time-course of grammar development and need to be corrected. As previously noted, the latter have so far been quite limited in number and significance.

Our approach makes it incumbent on us to seek principled explanations for these irregularities, since they must be explicitly encoded in the description hierarchy. Without the description hierarchy, there would be no need to reconcile these differences, since they would be entirely independent pieces of a flat grammar.

5 Grammar Evaluation

Performance evaluation of a grammar system can be distinguished into three kinds depending on the purpose it serves. [9] First, *intrinsic evaluation*, measures the performance of a system in the context of the framework it is developed in. This kind of evaluation helps system developers and maintainers to measure the performance of successive generations of the

[9] These evaluation methodologies are applicable to general purpose speech and natural language processing systems (Cole *et al.*, 1996, Jones and Galliers, 1995).

system and identify the shortcomings and weaknesses in the grammar, thus providing a direction for productive development of the grammar. We have discussed examples of such an evaluation for the XTAG system in Section 2.3.

A second method of evaluation is *comparative evaluation*. The objective here is to directly compare the performance of different grammar systems that use different grammar formalisms (and possibly different statistical models). Comparative evaluation helps in identifying the strengths and weaknesses of different systems and suggests possibilities for combining different approaches. However, this evaluation scheme requires a metric that is insensitive to the representational differences in the output produced by different parsers. For this purpose, the metric may have to be sufficiently abstracted away from individual representations so as to reach a level of agreement among the different representations produced by parsers. However, as a result of the abstraction process, the strengths of representations of certain parsers might be lost completely.

We have compared the performance of the XTAG system against other systems in the past. Using the crossing bracket accuracy as a metric, we have compared the performance of XTAG against the IBM statistical parsing system on the IBM manual sentences, against the Alvey Natural Language Tools Parser on the LDOCE Noun Phrases and against the CLARE parser on similar corpus. The results of these evaluations are reported in (Srinivas *et al.*, 1996). Although XTAG performed comparably with each of these systems, we feel that a metric that measures the accuracy of the derivation structures (dependency structures) would be a more suitable metric to measure the performance of the XTAG system.

A third method of evaluation of a parsing system is *extrinsic evaluation*. Extrinsic evaluation is meaningful when a parsing system is embedded in an application and it refers to the evaluation of the parsing system's contribution to the overall performance of the application. Extrinsic evaluation could be used as an indirect method of comparing parsing systems even if they produce different representations for their outputs as long as the output can be converted into a form usable by the application that the parser is embedded in.

6 Improving Parsing Efficiency

There are a number of stages where syntactic ambiguity—both lexical and structural—can be reduced in parsing. We will discuss:

1. part-of-speech tagging prior to parsing and,
2. tree/subcat filtering and weighting techniques

The first is a general technique which is applicable to all kinds of grammars; tree filtering and tree weighting take advantage of the particular properties of lexicalized grammars. The combination of these three tech-

niques has proven extremely effective in attacking the problem of ambiguity while simultaneously improving the efficiency of the parser in the XTAG system.

6.1 Part of Speech disambiguation

It is well known that lexical ambiguity with regard to the part-of-speech (POS) of a word is one of the greatest sources of overall ambiguity. This is particularly important in a lexicalized grammar, where each word is associated with multiple structures for each POS it may have. In our grammar, for example, the word *try* selects 59 verb trees and 17 noun trees; simply by identifying its POS we substantially reduce the number of trees it contributes to the parsing process. When this is done for each word in a sentence, the reduction in number of trees finally considered by the parser is enormous. Consider two examples: the NP *the act of allowing fresh air into a room* receives 26 parses untagged, and POS tagging reduces that number to 4; the sentence *the second part is the name of your personal computer* receives 32 parses without POS tagging, and only 8 parses with tagging.[10]

6.2 Supertag disambiguation

The elementary trees of LTAG localize dependencies, including long distance dependencies, by requiring that all and only the dependent elements be present within the same tree. As a result of this localization, a lexical item may be (and almost always is) associated with more than one elementary tree. Figure 10 illustrates the set of elementary trees assigned to each word of the sentence *the purchase price includes two ancillary companies.* We call these elementary trees *supertags*, since they contain more information (such as subcategorization and agreement information) than standard part-of-speech tags. Supertags for recursive and non-recursive constructs are labeled with βs and αs respectively.

Although each word is initially associated with many supertags, in a complete parse each word is associated with just one supertag (assuming there is no global ambiguity). The task of a lexicalized grammar parser can be viewed as a two step process. The first step is to select the appropriate supertags for each word of the input and the second step is to combine the selected supertags with substitution and adjunction operations. We call the first step as *Supertagging*. Figure 11 illustrates the initial set of supertags assigned to each word of the sentence *the purchase price includes two ancillary companies.* It also shows the final supertag sequence associated with the words in a complete parse of the sentence.

Note that, as in standard part-of-speech disambiguation, supertagging could have been done by a parser. However, just as carrying out part-of-

[10]The first example is from the Alvey test sentences and the second from the IBM Manual Corpus.

speech disambiguation prior to parsing makes the job of the parser much easier and therefore faster, supertagging reduces the work of the parser even further. The result of supertagging is an *almost parse* in the sense that the parser need 'only' link the individual structures to arrive at a complete parse. The *almost parsing* method can also be used to parse sentence fragments where it may not be possible to link the disambiguated supertag sequence into a single structure.

6.2.1 Reducing supertag ambiguity using structural information

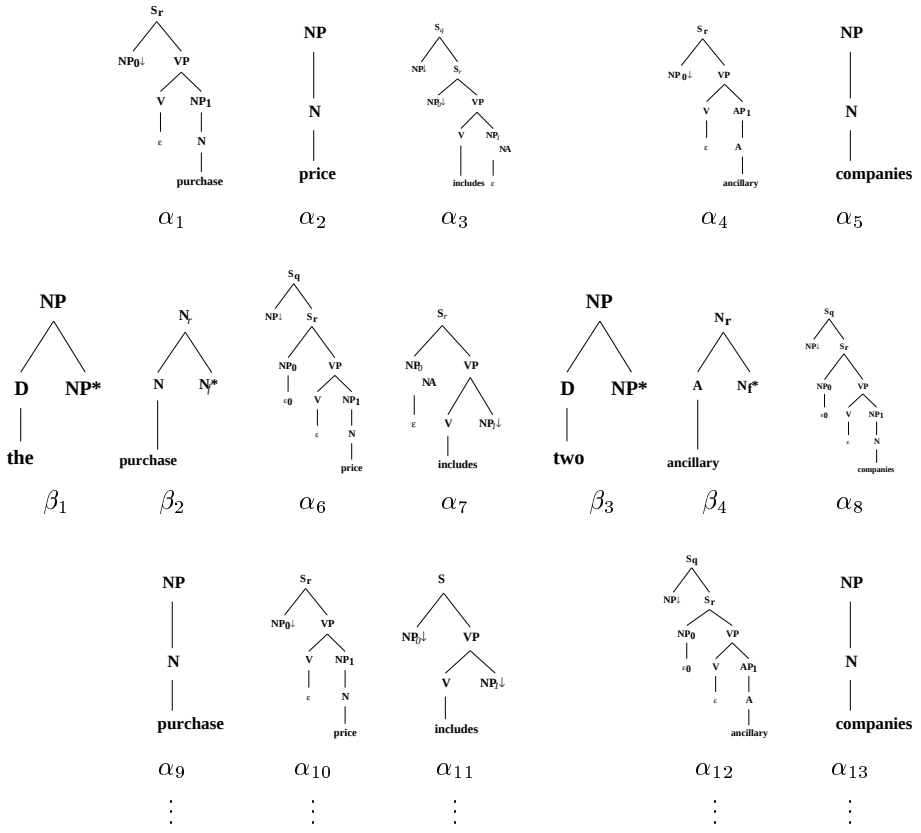
The structure of the supertag can be best seen as providing admissibility constraints on syntactic environments in which it may be used. Some of these constraints can be checked locally. The following are a few constraints that can be used to determine the admissibility of a syntactic environment for a supertag.[11]

- Span of the supertag : Span of a supertag is the minimum number of lexical items that the supertag can cover. Each substitution site of a supertag will cover at least one lexical item in the input. A simple rule can be used to eliminate supertags based on the *span constraint*: if the span of a supertag is larger than the input string, then the supertag cannot be used in any parse of the input string.
- Left (Right) span constraint: If the span of the supertag to the left (right) of the anchor is larger than the length of the string to the left (right) of the word that anchors the supertag, then the supertag cannot be used in any parse of the input string.
- Lexical items in the supertag: A supertag can be eliminated if the terminals appearing on the frontier of the supertag do not appear in the input string. Supertags with the built-in lexical item *by*, that represent passive constructions are typically eliminated from being considered during the parse of an active sentence.
  More generally, these constraints can be used to eliminate supertags that cannot have their features satisfied in the context of the input string. An example of this is the elimination of supertag that requires a *wh+* NP when the input string does not contain wh-words.

Table 6 indicates the decrease in supertag ambiguity for 2012 WSJ sentences (48,763 words)[12] by using the structural constraints relative to the supertag ambiguity without the structural constraints.

These filters prove to be very effective in reducing supertag ambiguity. The graph in Figure 12 plots the number of supertags at the sentence level for sentences of length 2 to 50 words with and without the filters. As can

[11] Prof. Mitch Marcus pointed out that these tests are similar to the *generalized shaper tests* used in the Harvard Predictive Analyzer (Kuno, 1966).
[12] wsj_20 of the Penn Treebank



the purchase price includes two ancillary companies.

FIGURE 10 A selection of the supertags associated with each word of the sentence *the purchase price includes two ancillary companies*

Sent:	the	purchase	price	includes	two	ancillary	companies.
Initial		α_1	α_2	α_3		α_4	α_5
Assig.	β_1	β_2	α_6	α_7	β_3	β_4	α_8
		α_9	α_{10}	α_{11}		α_{12}	α_{13}
		$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
Final Assig.	β_1	β_2	α_2	α_{11}	β_3	β_4	α_{13}

FIGURE 11 Supertag disambiguation for the sentence
the purchase price includes two ancillary companies

System	Total # of words	Av. # of S'tags/word
Without struct. constraints	48,783	47.0
With struct. constraints	48,783	25.0

TABLE 6 Supertag ambiguity with and without the use of structural constraints

be seen from the graph, the supertag ambiguity is significantly lower when the filters are used. The average reduction in supertag ambiguity is about 50%. This means that given a sentence, close to 50% of the supertags can be eliminated even before parsing begins by just using structural constraints of the supertags. This reduction in supertag ambiguity speeds up the parser significantly. In fact, the supertag ambiguity in XTAG system is so large that the parser is prohibitively slow without the use of these filters.

Even though structural constraints are effective in reducing supertag ambiguity, the search space for the parser is still sufficiently large. We use a trigram model in order to reduce the ambiguity further.

6.2.2 Trigram Model

The task of supertagging is to select the appropriate supertag for each word from the initial set of supertags it is assigned, given the context of the sentence. Thus the task of a supertagger is very similar to a part-of-speech tagger. A trigram disambiguation model has proved very successful in part-of-speech tagging. Owing to the similarity of supertagging to part-of-speech tagging, we use a trigram model to disambiguate supertags. A detailed discussion of the model can be found in (Srinivas, 1997a). Table 7 shows the performance of the trigram supertagger. Trained on 1,000,000 word/supertag pairs [13] and tested on 47,000 words[14] the supertagger assigned the correct supertag to 92% of the words. A total of 300 different supertags were used in these experiments.

[13] Sentences in wsj_00 through wsj_24, except wsj_20 of Penn Treebank.
[14] Sentences in wsj_20 of Penn Treebank.

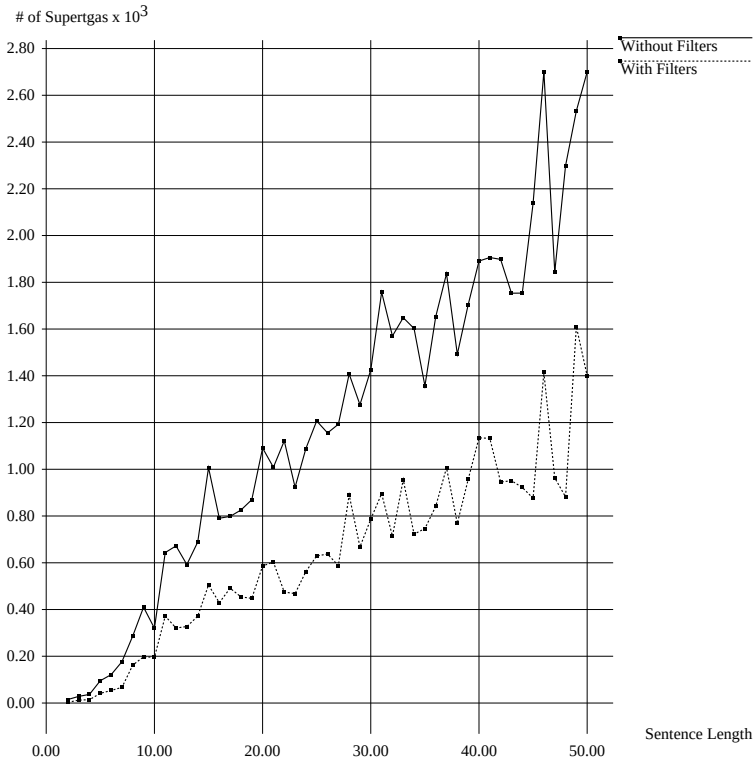


FIGURE 12 Comparison of number of supertags with and without filtering for sentences of length 2 to 50 words.

Once the supertagger selects the appropriate supertag for each word, the second stage of the parser is needed 'only' to combine the individual supertags to arrive at the parse of the input. Tested on about 1300 WSJ sentences with each word in the sentence correctly supertagged, the XTAG parser took approximately 4 seconds per sentence to yield a parse (combine the supertags and perform feature unification). In contrast, the same 1300 WSJ sentences without the supertag annotation took nearly 120 seconds per sentence to yield a parse. Thus the parsing speed-up gained by this integration is a factor of about 30.

6.3 Trade-offs in ambiguity resolution

The techniques used to reduce ambiguity and improve parser efficiency such as POS tagging (see Section 6.1) and Supertagging (see Section 6.2) although highly accurate are not infallible. Relying on ambiguity resolution before parsing the sentence means that when either the POS Tagger or the Supertagger assigns an incorrect POS tag or supertag, parsing fails.

Size of training set (words)	Model	Size of test set (words)	% Correct
1,000,000	(Baseline)	47,000	77.2%
	Trigram	47,000	92.2%

TABLE 7 Performance of the supertagger on the WSJ corpus

The sentence is then parsed without the POS tagger or the Supertagger, increasing the overall time needed to parse that particular sentence. Despite this tradeoff, techniques such as POS tagging and Supertagging still reduce the overall time needed to parse a set of sentences. Performance can be increased further by exploiting n-best techniques in the ambiguity resolution stages.

7 Applications

The trees developed as part of the XTAG system have been put to use with the help of the supertagger in many application areas including information filtering, information extraction and language modeling. A detailed list of these applications is given in (Srinivas, 1997b). The XTAG system also has an extension to Synchronous TAGs (Shieber and Schabes, 1990) which is being used for applications in Machine Translation (Egedi *et al.*, 1994, Han *et al.*, 1996, Palmer and Rosenzweig, 1996).

8 Other efforts

The following are few of the efforts currently underway:

- The large grammar (database) version of XTAG has recently been ported to CLISP, contemporary public-domain software, with the specific goal of permitting XTAG to run under the (public-domain) Linux operating system.
- Grammars for other languages such as Chinese, Hindi and Korean are currently being developed.
- There is also work underway to improve the lexical organization's tree-generation algorithm and the user interface to enable easier specification of grammars.
- We are checking for consistency of the grammar using metarules and the lexical organization tool. We also intend to compare the grammar produced using metarules with that produced by the lexical organization tool, and to evaluate the relative strengths of the two approaches.

References

Steven Abney. *The English Noun Phrase in its Sentential Aspects.* PhD thesis, MIT, 1987.

John Barwise and Robin Cooper. Generalized Quantifiers and Natural Language. *Linguistics and Philosophy*, 4, 1981.

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

Marie-Helene Candito. A Principle-Based Hierarchical Representation of LTAGs. In *Proceedings of COLING-96*, Copenhagen, Denmark, 1996.

Noam Chomsky and Howard Lasnik. The minimalist program. ms., 1993.

Kenneth Ward Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *2nd Applied Natural Language Processing Conference*, Austin, Texas, 1988.

Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue. Survey of the state of the art in human language technology, 1996. http://www.cse.ogi.edu/CSLU/HLTsurvey/.

Dania Egedi and Patrick Martin. A Freely Available Syntactic Lexicon for English. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, Nara, Japan, August 1994.

D. Egedi, M. Palmer, H.S. Park, and A. Joshi. Korean to english translation using synchronous tags. In *First Conference of the Association for Machine Translation in the Americas*, Columbus, MD, October 1994.

Roger Evans, Gerald Gazdar, and David Weir. Encoding Lexicalized Tree Adjoining Grammars with a Nonmonotonic Inheritance Hierarchy. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, 1995.

Ralph Grishman, Catherine Macleod, and Adam Meyers. Comlex Syntax: Building a Computational Lexicon. In *Proc. 15th Int'l Conf. Computational Linguistics (COLING 94)*, Kyoto, Japan, August 1994.

Chunghye Han, Fei Xia, Martha Palmer, and Joseph Rosenzweig. Capturing language specific constraints on lexical selection with feature-based lexicalized tree-adjoining grammar. In *Proceedings of International Conference on Chinese Computing(ICCC'96)*, 1996.

Beth Ann Hockey and B. Srinivas. Feature-Based TAG in Place of Multicomponent Adjunction: Computational Implications. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS)*, Fukuoka, Japan, December 1993.

Karen Sparck Jones and Julia R. Galliers. *Evaluating natural language processing systems : an analysis and review.* Number 1083 in Lecture notes in computer science. Lecture notes in artificial intelligence. Springer, Berlin ; New York, 1995.

A. K. Joshi. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam, 1987.

Daniel Karp, Yves Schabes, Martin Zaidel, and Dania Egedi. A Freely Available Wide Coverage Morphological Analyzer for English. In *Proceedings of the* 15^{th} *International Conference on Computational Linguistics (COLING '92)*, Nantes, France, August 1992.

E. L. Keenan and J. Stavi. A Semantic Characterization of Natural Language Determiners. *Linguistics and Philosophy*, 9, August 1986.

S. Kuno. Harvard predictive analyzer. In David G.Hays, editor, *Readings in automatic language processing*. American Elsevier Pub. Co., New York, 1966.

Sabine Lehmann, Stephan Oepen, Sylvie Regnier-Prost, Klaus Netter, Veronika Lux, Judith Klein, Kirsten Falkedal, Frederik Fouvry, Dominique Estival, Eva Dauphin, Hervé Compagnion, Judith Baur, Lorna Balkan, and Doug Arnold. TSNLP — Test Suites for Natural Language Processing. In *Proceedings of COLING 1996*, Kopenhagen, 1996.

Martha Palmer and Joseph Rosenzweig. Capturing motion verb generalizations with synchronous tags. In *Proceedings of AMTA-96*, Montreal, Quebec, October 1996.

Patrick Paroubek, Yves Schabes, and Aravind K. Joshi. Xtag – a graphical workbench for developing tree-adjoining grammars. In *Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. *A Comprehensive Grammar of the English Language*. Longman, London, 1985.

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

Anoop Sarkar and Aravind Joshi. Coordination in Tree Adjoining Grammars: Formalization and Implementation. In *Proceedings of the* 18^{th} *International Conference on Computational Linguistics (COLING '94)*, Copenhagen, Denmark, August 1996.

Yves Schabes and Aravind K. Joshi. An Early-Type Parsing Algorithm for Tree Adjoining Grammars. In *Proceedings of the* 26^{th} *Meeting of the Association for Computational Linguistics*, Buffalo, June 1988.

Yves Schabes and Aravind K. Joshi. Parsing with Lexicalized Tree Adjoining Grammar. In M. Tomita, editor, *Current Issues in Parsing Technologies*. Kluwer Academic Publishers, 1991.

Yves Schabes. *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, Computer Science Department, University of Pennsylvania, 1990.

Stuart Shieber and Yves Schabes. Synchronous Tree Adjoining Grammars. In *Proceedings of the* 13^{th} *International Conference on Computational Linguistics (COLING'90)*, Helsinki, Finland, 1990.

Frank K. Soong and Eng-Fong Huang. Fast Tree-Trellis Search for Finding the N-Best Sentence Hypothesis in Continuous Speech Recognition. *Journal of Acoustic Society, AM.*, May 1990.

B. Srinivas, Christine Doran, Beth Ann Hockey, and Aravind Joshi. An approach to robust partial parsing and evaluation metrics. In *Proceedings of the Workshop on Robust Parsing at European Summer School in Logic, Language and Information*, Prague, August 1996.

B. Srinivas. *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA, August 1997.

B. Srinivas. Performance evaluation of supertagging for partial parsing. Submitted for publication., January 1997.

K. Vijay-Shanker and Aravind K. Joshi. Unification Based Tree Adjoining Grammars. In J. Wedekind, editor, *Unification-based Grammars*. MIT Press, Cambridge, Massachusetts, 1991.

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

Akira Watanabe. The Notion of Finite Clauses in AGR-Based Case Theory. *MIT Working Papers in Linguistics*, 18:281–296, 1993.

D. Weir. *Characterizing mildly context-sensitive grammar formalisms*. PhD thesis, University of Pennsylvania, Philadelphia, PA, August 1988.

The XTAG-Group. A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS 95-03, University of Pennsylvania, 1995. Updated version available at http://www.cis.upenn.edu/ xtag/tr/tech-report.html.