A Lexicalized Tree Adjoining Grammar for English

The XTAG Research Group

Institute for Research in Cognitive Science
University of Pennsylvania
3401 Walnut St., Suite 400A
Philadelphia, PA 19104-6228

http://www.cis.upenn.edu/~xtag

February 26, 2001

Abstract

This document describes a sizable grammar of English written in the TAG formalism and implemented for use with the XTAG system. This report and the grammar described herein supersedes the TAG grammar described in [XTAG-Group, 1998]. The English grammar described in this report is based on the TAG formalism developed in [Joshi *et al.*, 1975], which has been extended to include lexicalization ([Schabes *et al.*, 1988]), and unification-based feature structures ([Vijay-Shanker and Joshi, 1991]). The range of syntactic phenomena that can be handled is large and includes auxiliaries (including inversion), copula, raising and small clause constructions, topicalization, relative clauses, infinitives, gerunds, passives, adjuncts, ditransitives (and datives), ergatives, it-clefts, wh-clefts, PRO constructions, noun-noun modifications, extraposition, determiner sequences, genitives, negation, noun-verb contractions, sentential adjuncts, imperatives and resultatives. The XTAG grammar is continuously updated with the addition of new analyses and modification of old ones, and an online version of this report can be found at the XTAG web page: `http://www.cis.upenn.edu/~xtag/`.

Authors

The contributors to this release of the XTAG English Grammar are as follows.

- **Authors of the Technical Report:**
  Tonia Bleam ⟨`tbleam@linc.cis.upenn.edu`⟩
  Chung-hye Han
  Rashmi Prasad ⟨`rjprasad@linc.cis.upenn.edu`⟩
  Carlos Prolo
  Anoop Sarkar

- **Grammar Developers:**
  Tonia Bleam
  Matt Feldman
  Chung-hye Han
  Martin Kappus
  Seth Kulick
  Virginie Nanta
  Rashmi Prasad
  Carlos Prolo

- **System Developers:**
  Moses Kimanzi
  Eric Kowey
  Carlos Prolo
  Anoop Sarkar
  William Schuler
  Fei Xia

This work was done under the supervision of Prof. Aravind Joshi.

Acknowledgements

We are immensely grateful to Aravind Joshi for supporting this project.

The following people have been grammar developers in previous releases of the XTAG English Grammar: Anne Abeille, Jason Baldridge, Rajesh Bhatt, Kathleen Bishop, Tonia Bleam, Raman Chandrasekar, Sharon Cote, Beatrice Daille, Christine Doran, Dania Egedi, Tim Farrington, Matthew Feldman, Jason Frank, Chung-Hye Han, Caroline Heycock, Beth Ann Hockey, Roumyana Izvorski, Martin Kappus, Karin Kipper, Daniel Karp, Seth Kulick, Young-Suk Lee, Heather Matayek, Patrick Martin, Megan Moser, Virginie Nanta, Sabine Petillon, Rashmi Prasad, Laura Siegel, Yves Schabes, Victoria Tredinnick and Raffaella Zanuttini.

The Common Lisp version of the XTAG system and the morphological and syntactic database system were developed and maintained by: Tilman Becker, Richard Billington, Andrew Chalnick, Dania Egedi, Devtosh Khare, Moses Kimanzi, Eric Kowey, Albert Lee, David Magerman, Alex Mallet, Patrick Paroubek, Rich Pito, Gilles Prigent, Carlos Prolo, Anoop Sarkar, Yves Schabes, William Schuler, B. Srinivas, Fei Xia, Yuji Yoshiie and Martin Zaidel.

We would also like to thank Michael Hegarty, Lauri Karttunen, Anthony Kroch, Mitchell Marcus, Martha Palmer, Owen Rambow, Philip Resnik, Maribel Romero, Beatrice Santorini and Mark Steedman.

In addition, Jeff Aaronson, John Batzel, Rahul Dave, Douglas DeCarlo, Mark-Jason Dominus, Mark Foster, Gaylord Holder, David Magerman, Ken Noble, Steven Shapiro, Kristofor Varhus and Ira Winston have provided technical support. Adminstrative support was provided by Laurel Sweeney, Ann Bies, Nicole Bolden, Susan Deysher, Carolyn Elken, Jodi Kerper, Christine Sandy and Trisha Yannuzzi.

This work was partially supported by NSF Grant SBR8920230, ARO Grant DAAH0404-94-G-0426, and DARPA Grant N66001-00-1-8915.

Contents

List of Figures

Part I

General Information

Chapter 1

Getting Around

This technical report presents the English XTAG grammar as implemented by the XTAG Research Group at the University of Pennsylvania. The technical report is organized into four parts, plus a set of appendices. Part 1 contains general information about the XTAG system and some of the underlying mechanisms that help shape the grammar. Chapter 2 contains an introduction to the formalism behind the grammar and parser, while Chapter 3 contains information about the entire XTAG system. Linguists interested solely in the grammar of the XTAG system may safely skip these chapters. Chapter 4 contains a description of the organization of the grammar as tree families and subcategorization frames, and discusses some of the linguistic principles that underlie the XTAG grammar, including the distinction between complements and adjuncts, and how case is handled.

The actual description of the grammar begins with Part 2, and is contained in the following three parts. Parts 2 and 3 contains information on the verb classes and the types of trees allowed within the verb classes, respectively, while Part 4 contains information on trees not included in the verb classes (e.g. NP's, PP's, various modifiers, etc). Chapter 5 of Part 2 contains a table that attempts to provide an overview of the verb classes and tree types by providing a graphical indication of which tree types are allowed in which verb classes. This has been cross-indexed to tree figures shown in the tech report. Chapter 6 contains an overview of all of the verb classes in the XTAG grammar. The rest of Part 2 contains more details on several of the more interesting verb classes, including ergatives, sentential subjects, sentential complements, copula, raising verbs, small clauses, ditransitives, ditransitives, datives, verbs selecting PP complements, resultatives and it-clefts.

Part 3 contains information on some of the tree types that are available within the verb classes. These tree types correspond to what would be transformations in a movement based approach. Not all of these types of trees are contained in all of the verb classes. The table (previously mentioned) in Part 2 contains a list of the tree types and indicates which verb classes each occurs in.

Part 4 focuses on the non-verb class trees in the grammar. NP's and determiners are presented in Chapter 20, while the various modifier trees are presented in Chapter 21. Auxiliary verbs, which are classed separate from the verb classes, are presented in Chapter 22, while certain types of conjunctions are shown in Chapter 23. The XTAG treatment of comparatives is presented in Chapter 24, and our treatment of punctuation is discussed in Chapter 25.

Throughout the technical report, mention is occasionally made of changes or analyses that we hope to incorporate in the future. Appendix 26 details a list of these and other future work. The appendices also contain information on some of the nitty gritty details of the XTAG grammar, including a system of metarules which can be used for grammar development and maintenance in Appendix 27. A system for the organization of the grammar in terms of an inheritance hierarchy is in Appendix 28. The tree naming conventions used in XTAG are explained in detail in Appendix 29, and a comprehensive list of the features used in the grammar is given in Appendix 30. Appendix 31 contains an evaluation of the XTAG grammar, including comparisons with other wide coverage grammars.

Chapter 2

Introduction to Lexicalized Tree Adjoining Grammars

Tree-adjoining grammar (TAG) is a formal tree rewriting system. TAG and Lexicalized Tree-Adjoining Grammar (LTAG) have been extensively studied both with respect to their formal properties and to their linguistic relevance. TAG and LTAG are formally equivalent, however, from the linguistic perspective LTAG is the system we will be concerned with in the XTAG system. We will often use these terms TAG and LTAG interchangeably.

The motivations for the study of LTAG are both linguistic and formal. The elementary objects manipulated by LTAG are structured objects (trees or directed acyclic graphs) and not strings. Using structured objects as the elementary objects of the formal system, it is possible to construct formalisms whose properties relate directly to the study of strong generative capacity (i.e., structural descriptions), which is more relevant to the linguistic descriptions than the weak generative capacity (sets of strings).

Each grammar formalism specifies a domain of locality, i.e., a domain over which various dependencies (syntactic and semantic) can be specified. It turns out that the various properties of a formalism (syntactic, semantic, computational, and even psycholinguistic) follow, to a large extent, from the initial specification of the domain of locality.

2.0.1 Domain of locality of CFGs

In a context-free grammar (CFG) the domain of locality is the one level tree corresponding to a rule in a CFG (Fig. 2.1). It is easily seen that the arguments of a predicate (for example, the two arguments of *likes*) are not in the same local domain. The two arguments are distributed over the two rules (two domains of locality)– $S \rightarrow NP\ VP$ and $VP \rightarrow V\ NP$. They can be brought together by introducing a rule $S \rightarrow NP\ V\ VP$. However, then the structure provided by the VP node is lost. We should also note here that not every rule (domain) in the CFG in (Fig. 2.1) is lexicalized. The three rules on the right are lexicalized, i.e., they have a lexical anchor. The rules on the right are not lexicalized. The second and the third rules on the left are almost lexicalized, in the sense that they each have a preterminal category (V in the second rule and ADV in the third rule), i.e., by replacing V by *likes* and ADV by *passionately* these two rules will become lexicalized. However, the first rule on the left ($S \rightarrow NP\ VP$) cannot be lexicalized. Can a CFG be lexicalized, i.e., given a CFG, G, can we construct another CFG, G', such that

CFG G

S -> NP VP
VP -> V NP
VP -> VP ADV
NP -> Harry
NP -> peanuts
V -> likes
ADV -> passionately

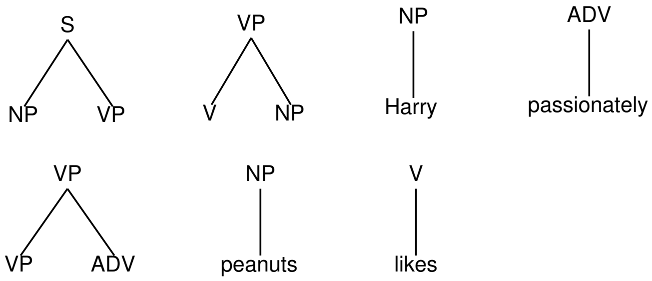


Figure 2.1: Domain of locality of a context-free grammar

every rule in G' is lexicalized and $T(G)$, the set of (sentential) trees (i.e., the tree language of G) is the same as the tree language $T(G')$ of G'? It can be shown that this is not the case [Joshi and Schabes, 1996]. Of course, if we require that only the string languages of G and G' be the same (i.e., they are weakly equivalent) then any CFG can be lexicalized. This follows from the fact that any CFG can be put in the Greibach normal form where each rule is of the form $A \rightarrow w\ B1\ B2\ ...\ Bn$ where w is a lexical item and the $B's$ are nonterminals. The lexicalization we are interested in requires the tree languages (i.e., the set of structural descriptions) be the same, i.e., we are interested in the 'strong' lexicalization. To summarize, a CFG cannot be strongly lexicalized by a CFG. This follows from the fact that the domain of locality of CFG is a one level tree corresponding to a rule in the grammar. Note that there are two issues we are concerned with here– lexicalization of each elementary domain and the encapsulation of the arguments of the lexical anchor in the elementary domain of locality. The second issue is independent of the first issue. From the mathematical point of view the first issue, i.e., the lexicalization of the elementary domains of locality is the crucial one. We can obtain strong lexicalization without satisfying the requirement specified in the second issue (encapsulation of the arguments of the lexical anchor). Of course, from the linguistic point view the second issue is very crucial. What this means is that among all possible strong lexicalizations we should choose only those that meet the requirements of the second issue. In implementing a grammar in the XTAG system we will assume that we always make such a choice.

2.0.2 Lexicalization of CFGs

Now we can ask the following question. Can we strongly lexicalize a CFG by a grammar with a larger domain of locality? Fig. 2.2 and Fig. 2.3 show a tree substitution grammar where the

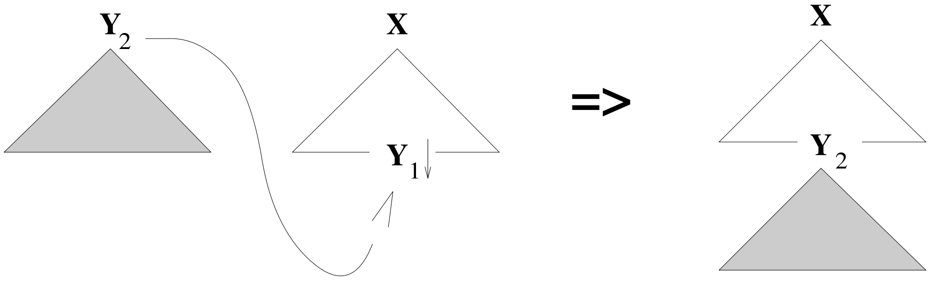


Figure 2.2: Substitution

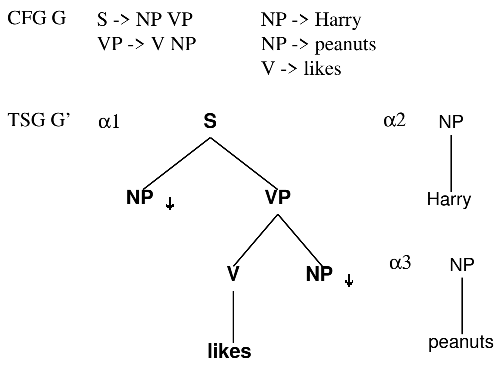


Figure 2.3: Tree substitution grammar

elementary objects (building blocks) are the three trees in Fig. 2.3 and the combining operation is the tree substitution operation shown in Fig. 2.2. Note that each tree in the tree substitution grammar (TSG), G' is lexicalized, i.e., it has a lexical anchor. It is easily seen that G' indeed strongly lexicalizes G. However, TSG's fail to strongly lexicalize CFG's in general. We show this by an example. Consider the CFG, G, in Fig. 2.4 and a proposed TSG, G'. It is easily seen that although G and G' are weakly equivalent they are not strongly equivalent. In G', suppose we start with the tree α_1 then by repeated substitutions of trees in G' (a node marked with a vertical arrow denotes a substitution site) we can grow the right side of α_1 as much as we want but we cannot grow the left side. Similarly for α_2 we can grow the left side as much as we want but not the right side. However, trees in G can grow on both sides. Hence, the TSG, G', cannot strongly lexicalize the CFG, G [Joshi and Schabes, 1996].

We now introduce a new operation called 'adjoining' as shown in Fig. 2.5. Adjoining involves splicing (inserting) one tree into another. More specifically, a tree β as shown in Fig. 2.5 is inserted (adjoined) into the tree α at the node X resulting in the tree γ. The tree β, called an

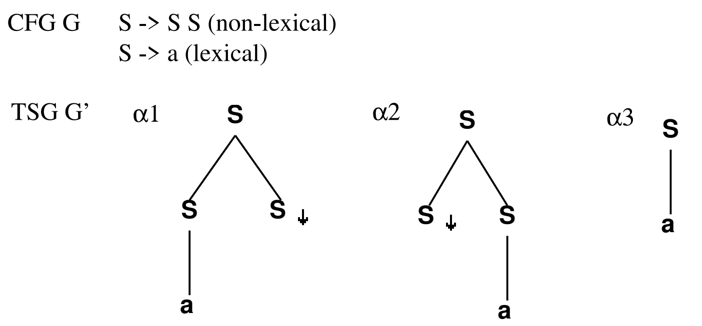


Figure 2.4: A tree substitution grammar

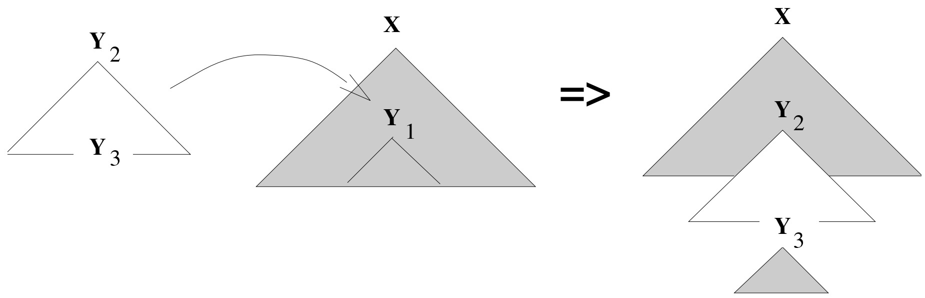


Figure 2.5: Adjoining

auxiliary tree, has a special form. The root node is labeled with a nonterminal, say X and on the frontier there is also a node labeled X called the foot node (marked with *). There could be other nodes (terminal or nonterminal) nodes on the frontier of β, the nonterminal nodes will be marked as substitution sites (with a vertical arrow). Thus if there is another occurrence of X (other than the foot node marked with *) on the frontier of β it will be marked with the vertical arrow and that will be a substitution site. Given this specification, adjoining β to α at the node X in α is uniquely defined. Adjoining can also be seen as a pair of substitutions as follows: The subtree at X in α is detached, β is substituted at X and the detached subtree is then substituted at the foot node of β. A tree substitution grammar when augmented with the adjoining operation is called a tree-adjoining grammar (lexicalized tree-adjoining grammar because each elementary tree is lexically anchored). In short, LTAG consists of a finite set of elementary trees, each lexicalized with at least one lexical anchor. The elementary trees are either initial or auxiliary trees. Auxiliary trees have been defined already. Initial trees are those for which all nonterminal nodes on the frontier are substitution nodes. It can be shown that any CFG can be strongly lexicalized by an LTAG [Joshi and Schabes, 1996].

In Fig. 2.6 we show a TSG, G', augmented by the operation of adjoining, which strongly lexicalizes the CFG, G. Note that the LTAG looks the same as the TSG considered in Fig. 2.4. However, now trees α_1 and α_2 are auxiliary trees (marked with *) that can participate in

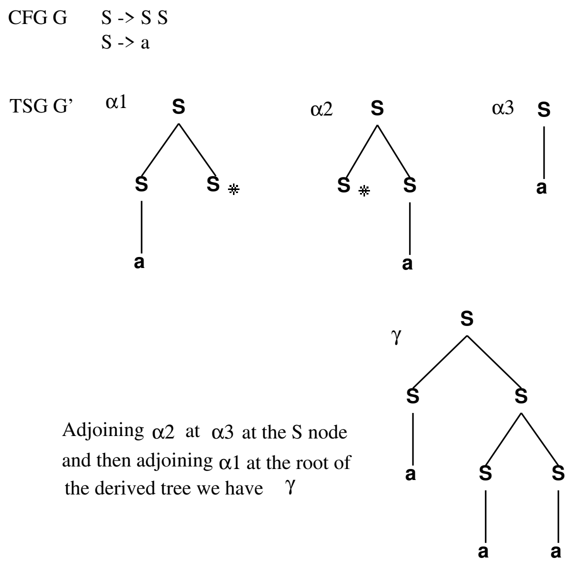


Figure 2.6: Adjoining arises out of lexicalization

adjoining. Since adjoining can insert a tree in the interior of another tree it is possible to grow both sides of the tree α_1 and tree α_2, which was not possible earlier with substitution alone. In summary, we have shown that by increasing the domain of locality we have achieved the following: (1) lexicalized each elementary domain, (2) introduced an operation of adjoining, which would not be possible without the increased domain of locality (note that with one level trees as elementary domains adjoining becomes the same as substitution since there are no interior nodes to be operated upon), and (3) achieved strong lexicalization of CFGs.

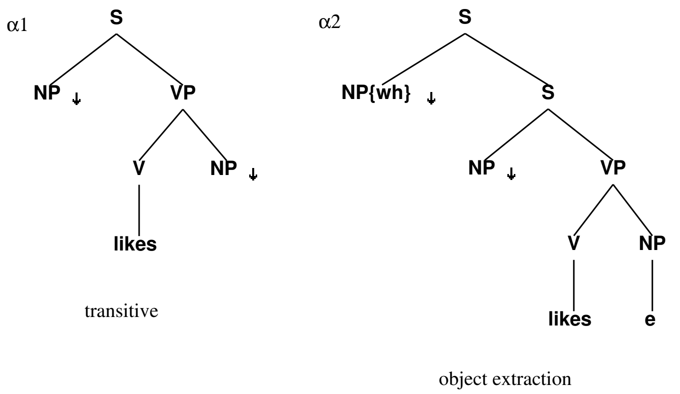


Figure 2.7: LTAG: Elementary trees for *likes*

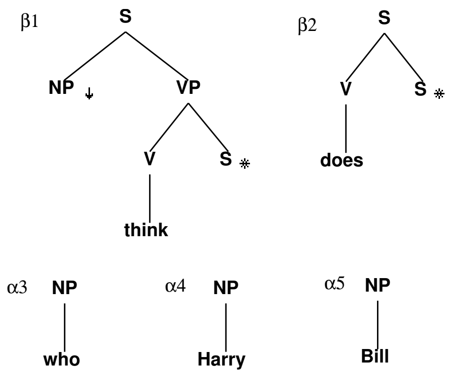


Figure 2.8: LTAG: Sample elementary trees

2.0.3 Lexicalized tree-adjoining grammars

Rather than giving formal definitions for LTAG and derivations in LTAG we will give a simple example to illustrate some key aspects of LTAG. We show some elementary trees of a toy LTAG grammar of English. Fig. 2.7 shows two elementary trees for a verb such as *likes*. The tree α_1 is anchored on *likes* and encapsulates the two arguments of the verb. The tree α_2 corresponds to the object extraction construction. Since we need to encapsulate all the arguments of the verb in each elementary tree for *likes*, for the object extraction construction, for example, we need to make the elementary tree associated with *likes* large enough so that the extracted argument is in the same elementary domain. Thus, in principle, for each 'minimal' construction in which *likes* can appear (for example, subject extraction, topicalization, subject relative, object relative, passive, etc.) there will be an elementary tree associated with that construction. By 'minimal' we mean when all recursion has been factored away. This factoring of recursion away from the domain over which the dependencies have to be specified is a crucial aspect of LTAGs as they are used in linguistic descriptions. This factoring allows all dependencies to be localized in the elementary domains. In this sense, there will, therefore, be no long distance dependencies as such. They will all be local and will become long distance on account of the composition operations, especially adjoining.

Fig. 2.8 shows some additional trees. Trees α_3, α_4, and α_5 are initial trees and trees β_1 and β_2 are auxiliary trees with foot nodes marked with *. A derivation using the trees in Fig. 2.8 is shown in Fig. 2.9. The trees for *who* and *Harry* are substituted in the tree for *likes* at the respective NP nodes, the tree for *Bill* is substituted in the tree for *think* at the NP node, the tree for *does* is adjoined to the root node of the tree for *think* tree (adjoining at the root node is a special case of adjoining), and finally the derived auxiliary tree (after adjoining β_2 to β_1) is adjoined to the indicated interior S node of the tree α_2. This derivation results in the derived tree for *who does Bill think Harry likes* as shown in Fig. 2.10. Note that the dependency between *who* and the complement NP in α_2 (local to that tree) has been stretched in the derived tree in Fig. 2.10. This tree is the conventional tree associated with the sentence.

However, in LTAG there is also a derivation tree, the tree that records the history of composition of the elementary trees associated with the lexical items in the sentence. This

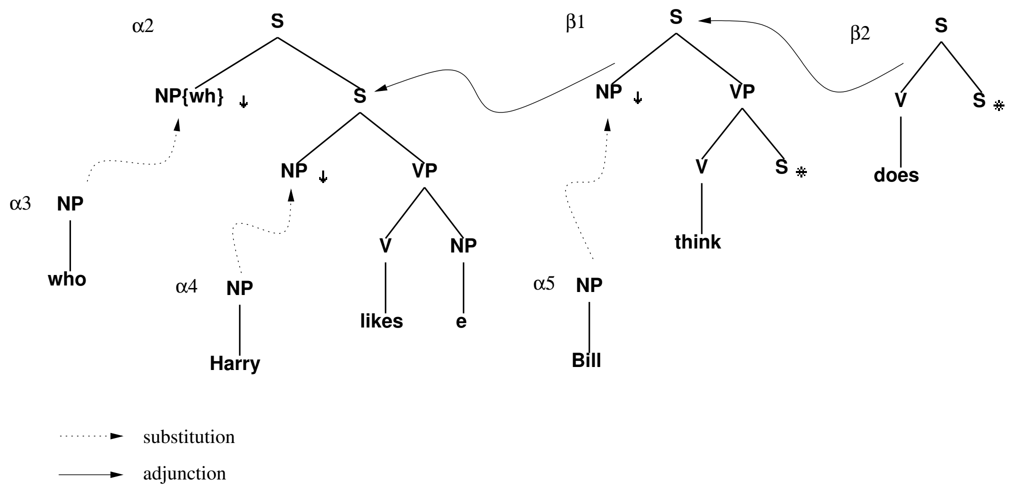


Figure 2.9: LTAG derivation for *who does Bill think Harry likes*

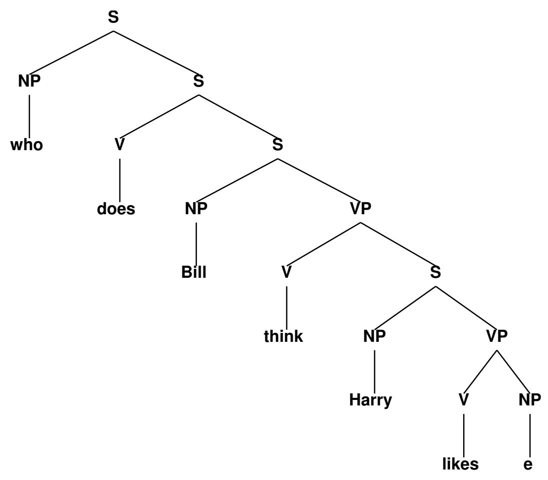


Figure 2.10: LTAG derived tree for *who does Bill think Harry likes*

derivation tree is shown in Fig. 2.11. The nodes of the tree are labeled by the tree labels such as α_2 together with the lexical anchor[1]. The derivation tree is the crucial derivation structure for LTAG. We can obviously build the derived tree from the derivation tree. For semantic computation the derivation tree (and not the derived tree) is the crucial object. Compositional semantics is defined on the derivation tree. The idea is that for each elementary tree there is a semantic representation associated with it and these representations are composed using the

[1]The derivation trees of LTAG have a close relationship to the dependency trees, although there are some crucial differences; however, the semantic dependencies are the same.

derivation tree. Since the semantic representation for each elementary tree is directly associated with the tree there is no need to reproduce necessarily the internal hierarchy in the elementary tree in the semantic representation [Joshi and Vijay-Shanker, 1999]. This allows the so-called 'flat' semantic representation as well as helps in dealing with some non-compositional aspects as in the case of rigid and flexible idioms.

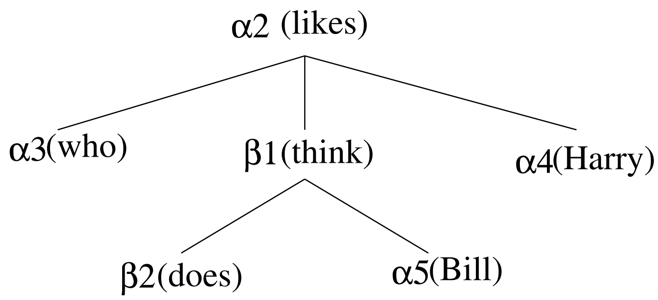


Figure 2.11: LTAG derivation tree

2.1 Some important properties of LTAG

The two key properties of LTAG are (1) extended domain of locality (EDL) (for example, as compared to CFG), which allows (2) factoring recursion from the domain of dependencies (FRD), thus making all dependencies local. All other properties of LTAG (mathematical, linguistic, and even psycholinguistic) follow from EDL and FRD. TAGs (LTAGs) belong to the so-called class of mildly context-sensitive grammars [Joshi, 1985]. CFL's are properly contained in the class of languages of LTAG, which in turn are properly contained in the class of context-sensitive languages. There is a machine characterization of TAG (LTAG), called embedded pushdown automaton (EPDA) [Vijay-Shanker, 1987], i.e., for every TAG language there is an EPDA which corresponds to this (and only this) language and the language accepted by any EPDA is a TAG language. EPDAs have been used to model some psycholinguistic phenomena, for example, processing crossed dependencies and nested dependencies have been in discussed in [Joshi, 1990]. With respect to formal properties, the class of TAG languages enjoys all the important properties of CFLs, including polynomial parsing (with complexity $O(n^6)$).

2.2 Unification-based features

In the XTAG system, each node in each LTAG tree is decorated with two feature structures (top and bottom feature structures), in contrast to the CFG based feature structure grammars, because adjoining can augment a tree internally, while in a CFG based grammar a tree can be augmented only at the frontier. It is possible to define adjoining and substitution (as it is done in the XTAG system) in terms of appropriate unifications of the top and bottom feature structures. Because of FRD (factoring recursion from the domain of dependencies), there is no recursion in the feature structures. Therefore, in principle, feature structures can be eliminated.

However, they are crucial for linguistic descriptions. Constraints on substitution and adjoining are modeled via these feature structures [Vijay-Shanker, 1987]. This method of manipulating feature structures is a direct consequence of the extended domain of locality of LTAG.

In a unification framework, a feature structure is associated with each node in an elementary tree. This feature structure contains information about how the node interacts with other nodes in the tree. It consists of a top part, which generally contains information relating to the supernode, and a bottom part, which generally contains information relating to the subnode. Substitution nodes, however, have only the top features, since the tree substituting in logically carries the bottom features.

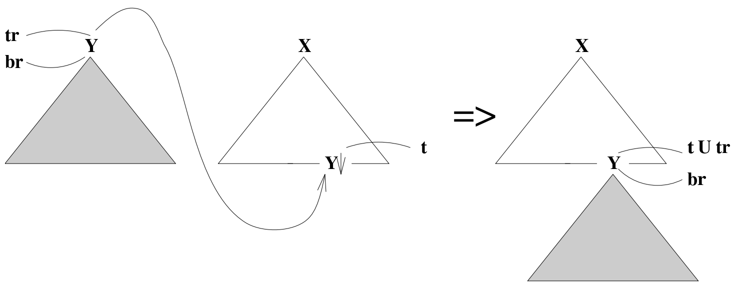


Figure 2.12: Substitution in FB-LTAG

The notions of substitution and adjunction must be augmented to fit within this new framework. The feature structure of a new node created by substitution inherits the union of the features of the original nodes. The top feature of the new node is the union of the top features of the two original nodes, while the bottom feature of the new node is simply the bottom feature of the top node of the substituting tree (since the substitution node has no bottom feature). Figure 2.12[2] shows this more clearly.

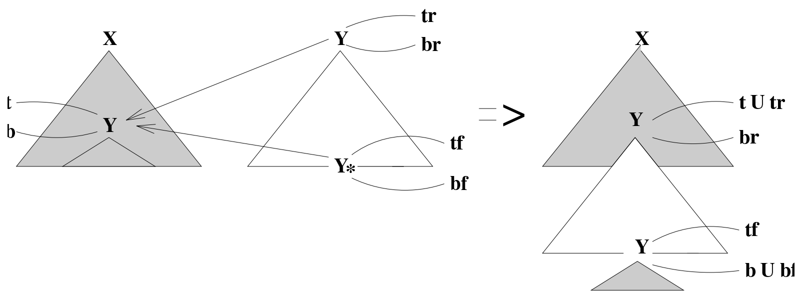
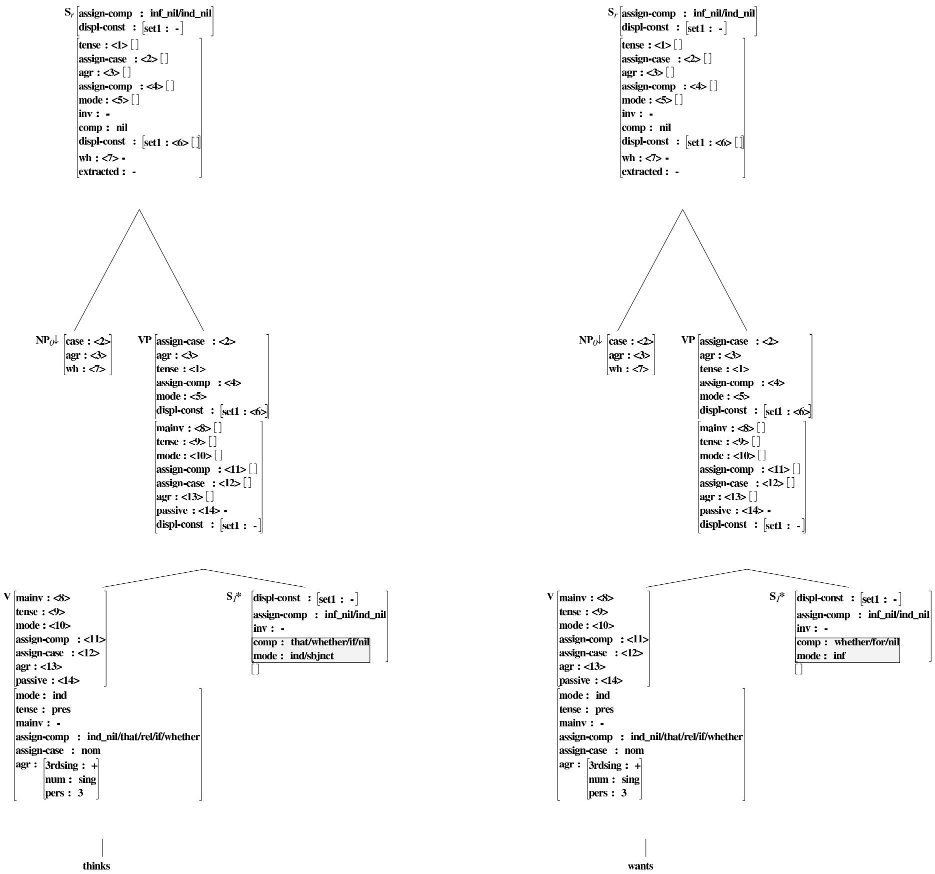


Figure 2.13: Adjunction in FB-LTAG

Adjunction is only slightly more complicated. The node being adjoined into splits, and its top feature unifies with the top feature of the root adjoining node, while its bottom feature unifies with the bottom feature of the foot adjoining node. Again, this is easier shown graphically, as in Figure 2.13[3].

[2]abbreviations in the figure: t=top feature structure, tr=top feature structure of the root, br=bottom feature structure of the root, U=unification

[3]abbreviations in the figure: t=top feature structure, b=bottom feature structure, tr=top feature structure



think tree | *want* tree

Figure 2.14: Lexicalized Elementary Trees with Features

The embedding of the TAG formalism in a unification framework allows us to dynamically specify local constraints that would have otherwise had to have been made statically within the trees. Constraints that verbs make on their complements, for instance, can be implemented through the feature structures. The notions of Obligatory and Selective Adjunction, crucial to the formation of lexicalized grammars, can also be handled through the use of features.[4] Perhaps more important to developing a grammar, though, is that the trees can serve as a schemata to be instantiated with lexical-specific features when an anchor is associated with the tree. To illustrate this, Figure 2.14 shows the same tree lexicalized with two different verbs, each of which instantiates the features of the tree according to its lexical selectional restrictions.

In Figure 2.14, the lexical item *thinks* takes an indicative sentential complement, as in the sentence *John thinks that Mary loves Sally. Want* takes a sentential complement as well, but an infinitive one, as in *John wants to love Mary.* This distinction is easily captured in the features and passed to other nodes to constrain which trees this tree can adjoin into, both cutting down the number of separate trees needed and enforcing conceptual Selective Adjunctions (SA).

2.3 Some related systems

Categorial Grammars: Categories assigned to lexical items in a categorial grammar framework do encapsulate the arguments of the lexical anchor. It is of interest to see how the basic ideas in LTAG could be incorporated in a categorial framework. The idea is not to translate LTAG into a categorial grammar but rather construct a categorial system with properties similar to LTAG. This is achieved by associating partial proof trees with lexical items. These partial proof trees are obtained by starting with the type assignment for a lexical item as specified by a categorial grammar and then 'unfolding' it by using certain categorial inference rules such as function application. This unfolding is done until the slots for the arguments of the lexical anchor are exposed. We thus have a finite collection of partial proof trees (lexicalized, of course) which serve as the building blocks of our system (analogous to the finite set of elementary trees in LTAG). These proof trees are then combined by universal categorial inference rules in terms of cuts. Informally speaking, the proof trees are hooked up by linking the conclusion nodes of one tree to the assumption nodes of another tree. For a further discussion of such systems and their relationship to LTAG can be found in [Joshi and Kulick, 1997; Joshi *et al.*, 1999].

From sentence structure to discourse structure: Using the insights from LTAG a structural and presuppositional a ccount of local discourse structure has been presented in [Webber *et al.*, 1999]. The idea is to start the analysis of discourse in the same way as one starts the analysis of a clause, looking at how its syntax and semantics project from the lexicon. This is complementary to the issue of discourse pragmatics –how these small syntactic units of discourse are used in achieving communicative intentions –and to the other discourse processes that provide additional organizational overlays on these units. A key feature of this approach is that semantic discourse relations are associated with syntactic structures and anaphoric links, and that the properties of the two are (not surprisingly) different. Together they allow more complex semantics to be conveyed through simpler structures.

of the root, br=bottom feature structure of the root, tf=top feature structure of the foot, bf=bottom feature structure of the foot, U=unification

[4] The remaining constraint, Null Adjunction (NA), must still be specified directly on a node.

Phrase structure composition and syntactic dependencies: [Frank, 2000] presents a comprehensive perspective on phrase structure composition and syntactic dependencies in a TAG-based grammatical architecture and compares it to the minimalist framework, showing that a number of stipulative and problematic aspects of that theory can be eliminated.

2.4 Summary

The domain of locality of a grammar formalism, i.e., the domain over which various dependencies can be specified determines to a large extent the syntactic, semantic, computational, and even psycholinguistic properties of the formalism. From this perspective the extended domain of Lexicalized Tree-Adjoining Grammars (LTAG) – extended as compared to the domain of locality of CFGs, for example – was explored. This extended domain is achieved by specifying the elementary objects of the grammar as structured objects instead of strings, together with two universal combining operations (substitution and adjoining). This perspective allows us to study directly many aspects of strong generative capacity which are more useful for linguistic description.

Chapter 3

Components of the XTAG System

This section focuses on the various components that comprise the parser and English grammar in the XTAG system. Persons interested only in the linguistic analyses in the grammar may skip this section without loss of continuity, although a quick glance at the tagset used in XTAG and the set of non-terminal labels used will be useful. We may occasionally refer back to the various components mentioned in this section.

3.1 System Description

Figure 3.1 shows the overall flow of the system when parsing a sentence; a summary of each component is presented in Table 3.1. 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**.

3.1.1 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. The morphology data was originally extracted from the Collins English Dictionary ([Hanks, 1979]) and Oxford

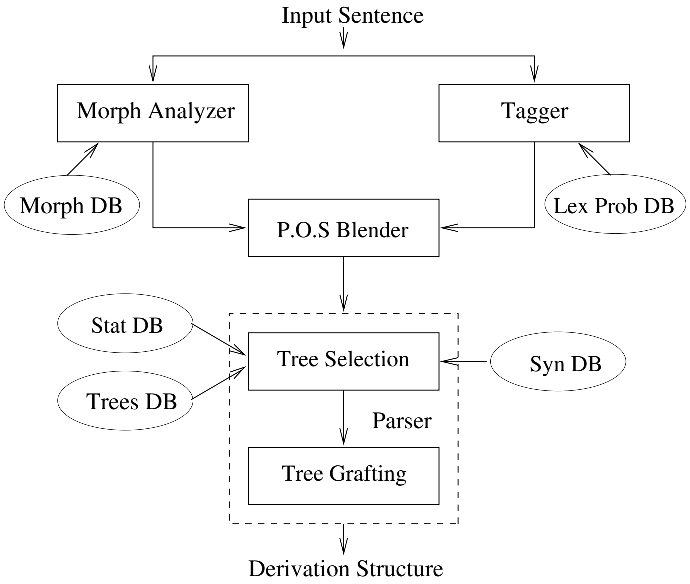


Figure 3.1: Overview of XTAG system

Advanced Learner's Dictionary ([Hornby, 1974]) available through ACL-DCI ([Liberman, 1989]), and then cleaned up and augmented by hand ([Karp *et al.*, 1992]).

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 3.1). 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. The syntactic database entries were originally extracted from the Oxford Advanced Learner's Dictionary ([Hornby, 1974]) and Oxford Dictionary for Contemporary Idiomatic English ([Cowie and Mackin, 1975]) available through ACL-DCI ([Liberman, 1989]), and then modified and augmented by hand ([Egedi and Martin, 1994]). There are more than 31,000 syntactic database entries.[1] Selected entries from this database are shown in Table 3.2.

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

[1]This number does not include trees assigned by default based on the part-of-speech of the word.

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 30,000 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	1004 trees, divided into 53 tree families and 221 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 3.1: System Summary

reduce ambiguity. Two methods are currently used: structural filters which eliminate 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). These methods speed the runtime by approximately 87%.

Supertagging Before parsing, one can avail of an optional step of *supertagging* the sentence. This step uses statistical disambiguation to assign a unique elementary tree (or *supertag*) to each word in the sentence. These assignments can then be hand-corrected. These supertags are used as a filter on the tree assignments made so far. More information on supertagging can be found in ([Srinivas, 1997a; Srinivas, 1997b]).

3.1.2 Tree Database

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

```
<<INDEX>>porousness<<ENTRY>>porousness<<POS>>N
<<TREES>>^BNXN ^BN ^CNn
<<FEATURES>>#N_card- #N_const- #N_decreas- #N_definite- #N_gen-
#N_quan- #N_refl-

<<INDEX>>coo<<ENTRY>>coo<<POS>>V<<FAMILY>>Tnx0V

<<INDEX>>engross<<ENTRY>>engross<<POS>>V<<FAMILY>>Tnx0Vnx1
<<FEATURES>>#TRANS+

<<INDEX>>forbear<<ENTRY>>forbear<<POS>>V<<FAMILY>>Tnx0Vs1
<<FEATURES>>#S1_WH- #S1_inf_for_nil

<<INDEX>>have<<ENTRY>>have<<POS>>V<<ENTRY>>out<<POS>>PL
<<FAMILY>>Tnx0Vplnx1
```

Table 3.2: Example Syntactic Database Entries.

given in Table 3.3. 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].

3.1.3 Tree Grafting

Once a particular set of lexicalized trees for the sentence have been selected, XTAG a parsing algorithm for LTAGs ([Schabes and Joshi, 1988; Schabes, 1990]) to find all derivations for the sentence. The derivation trees and the associated derived trees can be viewed using the X-interface (see Table 3.1). The X-interface can also be used to save particular derivations to disk.

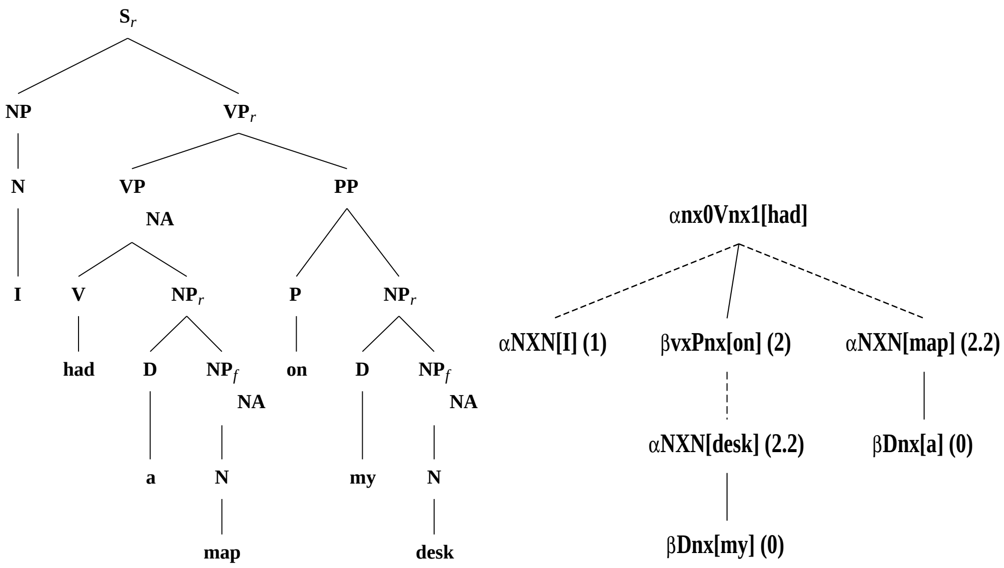
The output of the parser for the sentence *I had a map yesterday* is illustrated in Figure 3.2. The parse tree[3] represents the surface constituent structure, while the derivation tree represents the derivation history of the parse. The nodes of the derivation tree are the tree names anchored by the lexical items[4]. The composition operation is indicated by the nature of the arcs: a dashed line is used for substitution and a bold line for adjunction. The number beside each tree name is the address of the node at which the operation took place. The derivation tree can also be interpreted as a dependency graph with unlabeled arcs between words of the sentence.

The morphological and syntactic databases are documented in ([Karp *et al.*, 1992; Egedi and Martin, 1994]). XTAG also has a parsing and grammar development interface documented in ([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

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

[3]The feature structures associated with each note of the parse tree are not shown here.

[4]Appendix 29 explains the conventions used in naming the trees.



Parse Tree Derivation Tree

Figure 3.2: Output Structures from the Parser

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 3.3: XTAG tagset

tree editing and construction ([Becker, 1994]). These papers refer to an earlier version of the grammar development tools. They have been replaced by newer implementations which are described on the XTAG web page. Also, more information and download instructions about the XTAG grammar, the parser and the grammar development interface can be obtained on the XTAG web page at
`http://www.cis.upenn.edu/~xtag/`.

Chapter 4

Tree Families and Subcategorization Frames

This chapter is meant to be an introductory look at grammar organization in XTAG. It serves as basic information the reader would need to read the more detailed information about the grammar in subsequent parts of the technical report.

4.1 Tree Families and Subcategorization Frames

Elementary trees for predicative words[1] are used to represent the linguistic notion of subcategorization. The anchor of the elementary tree subcategorizes for the other elements that appear in the tree, forming a clausal or sentential structure. Tree families group together trees that belong to the same subcategorization frame. Consider the following uses of the verb *buy*:

(1) Srini bought a book.

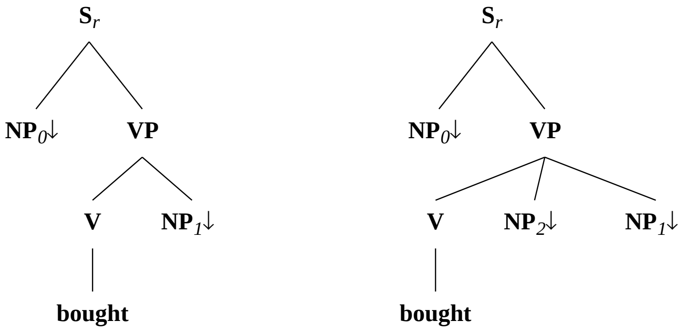
(2) Srini bought Beth a book.

In sentence (1), the verb *buy* subcategorizes for a direct object NP. The elementary tree anchored by *buy* is shown in Figure 4.1(a) and includes nodes for the NP complement of *buy* and for the NP subject. In addition to this declarative tree structure, the tree family also contains the trees that would be related to each other transformationally in a movement based approach, i.e passivization, imperatives, wh-questions, relative clauses, and so forth. Each tree family selected by its anchor represents all the various syntactic environments that it can appear in.

Sentence (2) shows that *buy* also subcategorizes for a double NP object. This means that *buy* also selects the double NP object subcategorization frame, or tree family, with its own set of transformationally related sentence structures. Figure 4.1(b) shows the declarative structure for this set of sentence structures.

Entire classes of anchors select a tree family. All the transitive verbs form a class which selects the transitive tree family. Recall that a tree family is a group of trees related by some

[1]Such as non-auxiliary verbs or predicative nouns.



(a) (b)

Figure 4.1: Different subcategorization frames for the verb *buy*

syntactic transformations. Since an entire class of anchors are selecting this tree family, the assumption is that these syntactic transformations are valid for each member of this class. For instance, wh- extraction is a syntactic transformation that will apply regardless of any idiosyncratic properties of any particular anchor of the tree family. Chapter 6 contains more information about the different tree families in the grammar.

There are some syntactic transformations, however, that are sensitive to the properties of a particular anchor within the same subcategorization frame. The ergative (or transitive-inchoative) alternation for transitive verbs is one such transformation. Only a subset of the transitive verbs can undergo this transformation. A more rigorous definition of tree family that accounts for such lexical idiosyncrasies within an otherwise homogeneous family is discussed in detail in Appendix 4.2.

4.2 Tree Families and Lexical Rules

In the XTAG English grammar, tree families provide us with a compact means to specify the set of trees which a particular lexical anchor takes. This set represents all syntactic environments in which this anchor appears. One way to represent tree families is to have a universal set of trees, with each individual anchor taking some subset of that larger set. However, since syntactic transformations are not usually sensitive to the anchor of the tree family, in practice we assign a tree family to entire classes of words.

Chapter 4 defines the notion of tree family used in the XTAG grammar. In this chapter we discuss in more detail what it means for a particular anchor to select several tree families keeping in mind that entire classes of words which share the same subcategorization are assigned tree families in the syntactic database. Also, there are some syntactic transformations that are sensitive to the properties of a particular word within the same subcategorization frame.

In general, the trees within a particular tree family can be thought of as being those which are syntactically or transformationally related. The tree in Figure 4.2 allows us to parse sentences such as (3). This tree represents the subcategorization of the verb and forms the base for the generation of an entire tree family. Each family has one such *base tree*. One method for generating all the trees in the family that are syntactically related to the base tree is to use metarules (see Appendix 27). One such member of the transitive tree family is the tree in

Figure 4.3 which is the tree where the object of the αnx0Vnx1 has been extracted. This new tree αW1nx0Vnx1 now allows us to parse sentences such as (4).

(3) Susie likes Hobbes.

(4) Who does Susie like?

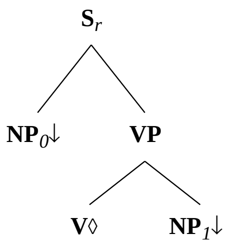


Figure 4.2: Declarative Transitive Tree: αnx0Vnx1

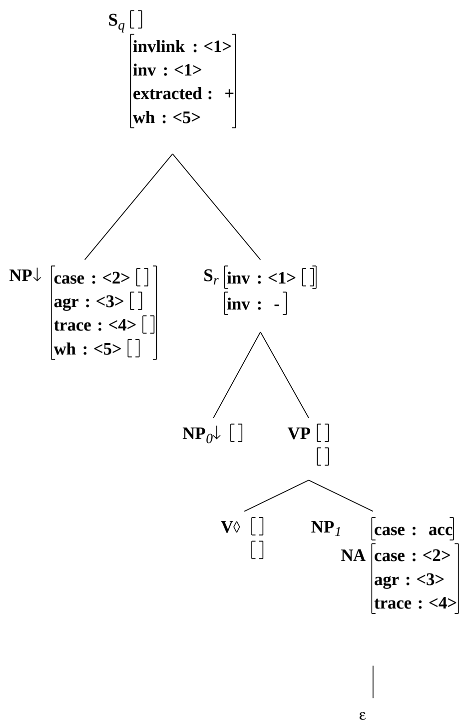


Figure 4.3: Transitive tree with object extraction: αW1nx0Vnx1

The semantic interpretation of arguments across different trees in a family remains constant. That is, if NP_1 is the recipient argument of the predicate in the base indicative tree, then it is the recipient when it is fronted as well.

There are some syntactic transformations, however, that are sensitive to the properties of a particular word within the same subcategorization frame. The ergative (or transitive-inchoative alternation) for transitive verbs is one such transformation. Only a subset of the transitive verbs can undergo this transformation. Let us call such transformations *lexical rules* to distinguish them from more general *syntactic transformations* (like wh-extraction) which are not lexically idiosyncratic.

To explain how these two different kinds of transformations interact let us take the example of the ergative subset of the transitive verbs:

Ergative verbs such as *melt* undergo the ergative transformation (see Chapter 7). Such verbs are a subset of the set of transitive verbs, which do not all take this transformation; for example verbs like *borrow* cannot take this transformation. To handle the transitive/ergative distinction, we could create a tree family which exclusively handles the regular transitives and another one which handles the ergatives. In doing so, however, we would be duplicating structure since the ergative family would contain all of the trees found in the transitive family in addition to the purely ergative trees. An alternative is to consider the ergative distinction as arising from a *lexical rule* which takes the base tree of the transitive family (αnx0Vnx1) and produces the base tree for a strictly ergative family (αEnx1V). Then all of the syntactic transformations which are available in the grammar can be applied to that new base tree to form the entire ergative tree family. In this particular instance, the indexation on the syntactic object NP of αnx0Vnx1 (NP_1) is retained on the syntactic subject of the ergative tree, which has the same semantic role in both constructions.

The same problem appears in the ditransitive tree family, in which a subset of anchors also undergo the dative-shift transformation. We give an identical solution to this case as we do for the ergatives.

Hence, tree families can be related to each other by lexical rules. Construing tree families in this way allows us to take advantage of shared structure when creating a grammar. Introducing this shallow hierarchy in the definition of family allows us to have a more compact organization of the grammar. This departs from earlier conceptions of tree families which maintained that argument positions in different tree families were in no way related; thus, before, if a lexical item anchored two different tree families, each anchoring instance was considered a different predicate. Under both the previous and the present conceptions, tree families are the exhausitive domain of a base tree and the trees generated by syntactic transformations on that base. However, the present conception of tree families provides for a new level of distinction – that of families related by a lexical rule. These related families are the exhaustive domain of a particular predicate that is affected by a lexical rule.

The XTAG grammar encodes the notion of lexical rules implicitly rather than explicitly. It is implicit in the anchorings of certain lexical items. For example, ergative verbs such as *melt* anchor both Tnx0Vnx1 and TEnx1V, and their lexical family is the set of trees in the union of these two tree families. See the chapters on ergatives (see Chapter 7) and ditransitives (see Chapter 10) for more in-depth discussions of particular cases of lexical families.

To briefly summarize, we can now give the following definitions:

tree family: the set of trees generated from a base tree by the syntactic transformations which are available in the grammar.

lexical family: the set of tree families generated for a given predicate from an initial base tree and the base trees which are generated by the lexical rules which are specified for that predicate.

It should be noted, however, that we will continue to refer to tree families and lexical families both by the generic name *tree family*, unless the distinction is crucial to the point we are trying to make.

4.3 Complements and Adjuncts

Complements and adjuncts have very different structures in the XTAG grammar. Complements are included in the elementary tree anchored by the verb that selects them, while adjuncts do not originate in the same elementary tree as the verb anchoring the sentence, but are instead added to a structure by adjunction. The contrasts between complements and adjuncts have been extensively discussed in the linguistics literature and the classification of a given element as one or the other remains a matter of debate (see [Rizzi, 1990], [Larson, 1988], [Jackendoff, 1990], [Larson, 1990], [Cinque, 1990], [Obernauer, 1984], [Lasnik and Saito, 1984], and [Chomsky, 1986]). The guiding rule used in developing the XTAG grammar is whether or not the sentence is ungrammatical without the questioned structure.[2] Consider the following sentences:

(5) Srini bought a book.

(6) Srini bought a book at the bookstore.

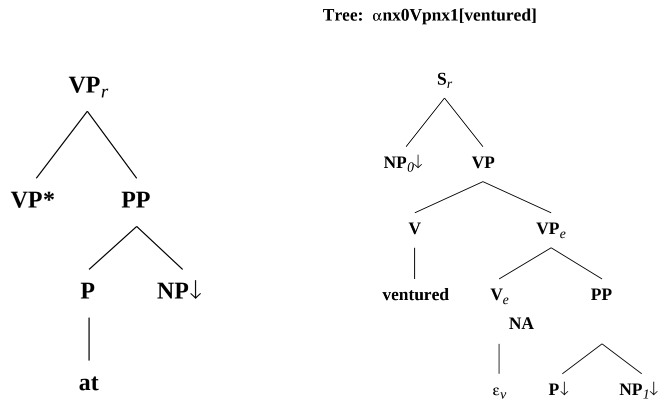
(7) Fei ventured into the cave.

(8) ∗Fei ventured.

Prepositional phrases are common adjuncts, and when they are used as adjuncts they have a tree structure such as that shown in Figure 4.4(a). This adjunction tree would adjoin into the tree shown in Figure 4.1(a) to generate sentence (6). There are verbs, however, such as *venture*, *hunger* and *differentiate*, that take prepositional phrases as complements. Sentences (7) and (8) clearly show that the prepositional phrase are not optional for *venture*. For verbs such as this, the prepositional phrase is articulated within the verb's elementary tree, as shown in Figure 4.4(b). The preposition and its complement noun phrase are substituted into this elementary tree.

Virtually all parts of speech, except for main verbs, function as both complements and adjuncts in the grammar. More information is available in this report on various parts of speech as complements: adjectives (e.g. section 6.14), nouns (e.g. section 6.3), and prepositions (e.g. section 6.11); and as adjuncts: adjectives (section 21.1), adverbs (section 21.6), nouns (section 21.3), and prepositions (section 21.5).

[2] Iteration of a structure can also be used as a diagnostic: *Srini bought a book at the bookstore on Walnut Street for a friend.*



(a) (b)

Figure 4.4: Trees illustrating the difference between Complements and Adjuncts

4.4 Other assumptions made in grammar development

The morphological, syntactic, and tree databases together comprise the English grammar[3]. A lexical item that is not in the databases receives a default tree selection and features for its part of speech and morphology.

In designing the grammar, a decision was made early on to err on the side of acceptance whenever there are conflicting opinions as to whether or not a construction is grammatical. In this sense, the XTAG English grammar is intended to function primarily as an acceptor rather than a generator of English sentences. The range of syntactic phenomena that can be handled is large and includes auxiliaries (including inversion), copula, raising and small clause constructions, topicalization, relative clauses, infinitives, gerunds, passives, adjuncts, it-clefts, wh-clefts, PRO constructions, noun-noun modifications, determiner sequences, genitives, negation, noun-verb contractions, clausal adjuncts and imperatives.

4.4.1 Non-S constituents

Although sentential trees are generally considered to be special cases in any grammar, insofar as they make up a 'starting category', it is the case that any initial tree constitutes a phrasal constituent. These initial trees may have substitution nodes that need to be filled (by other initial trees), and may be modified by adjunct trees, exactly as the trees rooted in S. Although grouping is possible according to the heads or anchors of these trees, we have not found any classification similar to the subcategorization frames for verbs that can be used by a lexical entry to create a tree family of the set of trees selected by such entries. These trees are selected one by one by each lexical item, according to each lexical item's idiosyncrasies. The grammar described by this technical report places them into several files for ease of use, but these files do not constitute tree families in the way that the subcategorization frames do.

[3]See Chapter 3 for details on these levels of representation.

Part II

Verb Classes

Chapter 5

Where to Find What

The two tables that follow give an overview of what types of trees occur in various tree families, with pointers to discussion in this report. The first table gives the expansion of abbreviations in the headers of the second table. The abbeviations and their expansions can be matched in the second table in the order in which they appear - first for the column headers (left-right) and second for the row header (top-down). Notice that the non-abbreviated headers will not be seen in this table, so they should be skipped when following the order of appearance. The second table contains two kinds of information related to the XTAG grammar. Firstly, the column headers give the list all the tree families in the grammar (along with the XTAG names for them). These can be also be referenced in the more detailed decsriptions of each tree family in Chapter 6. Secondly, each column, corresponding to a tree family, contains a list of the constructions available for that tree family so that the row headers gives the constructions available in the grammar across the tree families. An entry in a cell of the table indicates that the tree(s) for the construction named in the row header are included in the tree family named in the column header. Entries are of two types. If the particular tree(s) are displayed and/or discussed in this report the entry gives a page number reference to the relevant discussion or figure.[1] Otherwise, a $\surd$ indicates inclusion in the tree family but no figure or discussion related specifically to that tree in this report. Blank cells indicate that there are no trees for the construction named in the row header in the tree family named in the column header.

[1]Since Chapter 6 has a brief discussion and a declarative tree for every tree family, page references are given only for other sections in which discussion or tree diagrams appear.

Abbreviation	Full Name
COLUMN HEADERS	
V,P Pred.	Predicative Multi-word with Verb, Prep anchors
V,P Ditr. Pred.	Ditransitive with PP with Verb and Prep anchors
Sent. compl. w/ NP	Sentential complement with NP
Intr. Verb Particle	Intransitive verb particle
Trans. Verb Particle	Transitive verb particle
Ditrans. Verb Particle	Ditransitive verb particle
Sent. compl.	Sentential complement
Intransitive w/ Adj.	Intransitive with adjective
Transitive SS.	Transitive sentential subject
Ditr. LV. w/ PP	Ditransitive light verb with PP
Adj.. Sm-Cl.	Adjective small clause
Adj. Sm-Cl. w/ Sent. compl.	Adjective small clause with sentential complement
Adj. Sm-Cl. w/ SS.	Adjective small clause with sentential subject
NP Sm-Cl.	NP small clause
PP Sm-Cl.	PP small clause
Exh. PP Sm-Cl.	Exhaustive PP small clause
PP Sm-Cl. w/ SS.	PP small clause with sentential subject
PP Sm-Cl. w/ Ad,P Pred.	PP small clause with Adv and Prep anchors
PP Sm-Cl. w/ A,P Pred.	PP small clause with Adjective and Prep anchors
PP Sm-Cl. w/ N,P Pred.	PP small clause with Noun and Prep anchors
PP Sm-Cl. w/ P,P Pred.	PP small clause with two Prep anchors
PP Sm-Cl. w/ P,N Pred.	PP small clause with Prep and Noun anchors
PP Sm-Cl. w/ SS., & Ad,P Pred.	PP small clause with sentential subject and Adverb and Prep anchors
PP Sm-Cl. w/ SS., & A,P Pred.	PP small clause with sentential subject and Adjective and Prep anchors
PP Sm-Cl. w/ SS., & N,P Pred.	PP small clause with sentential subject and Noun and Prep anchors
PP Sm-Cl. w/ SS., & P,P Pred.	PP small clause with sentential subject and Prep anchors
PP Sm-Cl. w/ SS., & P,N Pred.	PP small clause with sentential subject and Prep and Noun anchors
Intr. SS.	Intransitive sentential subject
SS. w/ 'to' compl.	Sentential subject with "to" complement
Loc. Sm-Cl. w/ Ad. Pred.	Locative small clause with adverb anchor
Tr/Intr. Res. w/ V,A Pred.	(transitive/intransitive) resultatives with verb and adjective anchors
Tr/Intr. Res. w/ V,P Pred.	(transitive/intransitive) resultatives with verb and prep anchors
Erg. Res. w/ V,A Pred.	Ergative resultatives with verb and adjective anchors
Erg. Res. w/ V,P Pred.	Ergative resultatives with verb and prep anchors
SS. w/ Sm-Cl. compl.	Sentential subject with small clause complement
ROW HEADERS	
Y/N quest.	Yes/No question
Wh-mov. S compl.	Wh-moved Sentential complement
Wh-mov. Adj or Ad compl.	Wh-moved Adjective or adverb complement
Wh-mov. object of a mod.	Wh-moved object of a modifier
Wh-mov. PP	Wh-moved PP
Topic. NP complement	Topicalized NP complement
Det. gerund	Determiner gerund
Rel. cl. on NP compl.	Relative clause on NP complement
Rel. cl. on PP compl.	Relative clause on PP complement
Rel. cl. on NP object of P	Relative clause on NP object of Prep
Pass. with wh-moved subj.	Passive with wh-moved subject (with and without *by* phrase)
Pass. w. wh-mov. ind. obj.	Passive with wh-moved indirect object (with and without *by* phrase)
Pass. w. wh-mov. obj. of the *by phrase*	Passive with wh-moved object of the *by* phrase
Pass. w. wh-mov. *by* phrase	Passive with wh-moved *by* phrase
Trans. Idiom with V, D and N	Transitive Idiom with Verb, Det and Noun anchors

Constructions	Intransitive, Tnx0V	Transitive Ergative, TEnx1V	Transitive, Tnx0Vnx1	Intransitive w/ PP, Tnx0Vpnx1	V,P Pred., Tnx0VPnx1	Ditransitive, Tnx0Vnx2nx1	Ditransitive with PP, Tnx0Vnx1pnx2	V,P Ditr. Pred., Tnx0Vnx1Pnx2	Sent. compl. w/ NP, Tnx0Vnx1s2	Intr. Verb Particle, Tnx0Vpl	Trans. Verb Particle, Tnx0Vplnx1	Ditrans. Verb Particle, Tnx0Vplnx2nx1	Sent. compl. , Tnx0Vs1	Intransitive w/ Adj., Tnx0Vax1	Transitive SS., Ts0Vnx1
	Tree families														
Declarative	✓	✓	291	27	✓	291	✓	115	✓	✓	✓	✓	14, 91	✓	96
Passive w/ & w/o *by* phrase			✓		✓	✓	✓	✓	129		✓	✓			
Wh-moved subject	135	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	92	✓
Wh-mov. NP compl., DO or IO			132			136	✓	✓	✓		✓	✓			
Wh-mov. S compl.									✓				✓		
Wh-mov. Adj. or Ad. compl.														138	
Wh-mov. object of a mod.				✓	✓		136	✓							
Wh-mov. PP				✓	✓		137	✓							
Topic. NP compl.			✓			✓	✓	✓	✓		✓	✓			
Imperative	✓	✓	157	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Det. gerund	✓	✓	166	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
NP gerund	✓	✓	167	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Rel. cl. on subj. w/ NP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Rel. cl. on subj. w/ COMP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Rel. cl. on NP compl., DO, IO w/ NP			✓	✓		✓	✓	✓	✓		✓	✓			
Rel. cl. on NP compl., DO, IO w/ COMP			✓	✓		✓	✓	✓	✓		✓	✓			
Rel. cl. on PP compl. w/ pied-piping			✓	✓		✓	✓	✓	✓		✓				
Rel. cl. on NP object of P w/ NP			✓	✓		✓	✓	✓	✓		✓				
Rel. cl. on NP object of P w/ COMP			✓	✓		✓	✓	✓	✓		✓				
Rel. cl. on adjunct w/ PP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rel. cl. on adjunct w/ COMP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Past part. as arg. Adj.			✓		✓										
Paren. quot. cl.									✓				✓		
Participial NP pre-mod	✓	✓	✓		✓										
Pass. w/ wh-mov. subj.			✓	✓	✓	✓	✓	✓	✓		✓				
Pass. w/ wh-mov. ind. obj.				✓	✓	✓	✓	✓	✓		✓				
Pass. w/ wh-mov. obj. of *by* phrase			✓	✓	✓	✓	✓	✓	✓		✓				
Pass. w/ wh-mov. *by* phrase			✓	✓	✓	✓	✓	✓	✓		✓				
PRO in Decl.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
PRO in NP ger.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
PRO subj. in decl. pass. w/ *by* phrase			✓	✓	✓	✓	✓	✓	✓		✓				
PRO subj. in decl. pass. w/o *by* phrase			✓	✓	✓	✓	✓	✓	✓		✓				
PRO subj. in ger. pass. w/ *by* phrase			✓	✓	✓	✓	✓	✓	✓		✓				
PRO subj. in ger. pass. w/o *by* phrase			✓	✓	✓	✓	✓	✓	✓		✓				

	Tree families													
Constructions	Transitive Light Verb, Tnx0lVN1	Ditr. LV. w/ PP, Tnx0lVN1Pnx2	NP It-cleft, TItVnx1s2	PP It-cleft, TItVpnx1s2	Adverb, It-cleft, TItVads2	Adj.. Sm-Cl., Tnx0Ax1	Adj. Sm-Cl. w/ Sent. compl., Tnx0A1s1	Adj. Sm-Cl. w/ SS., Ts0Ax1	Equative BE, Tnx0BEnx1	NP Sm-Cl., Tnx0N1	NP w/ Sent. compl. Sm-Cl., Tnx0N1s1	NP Sm-Cl. w/ SS., Ts0N1	PP Sm-Cl., Tnx0Pnx1	Exh. PP Sm-Cl., Tnx0Px1
Declarative	√	√	√	123	√	109	√	√	113	106	√	106	√	√
Passive w/ & w/o *by* phrase		√												
Y/N quest.			√	123	√				√					
Wh-moved subject	√	√				92	√	√		√	√	√	√	√
Wh-mov. NP compl., DO or IO		√	√							√			√	
Wh-mov. Adj. or Ad. compl.			√		√	√								
Wh-mov. object of a mod.		√												
Wh-mov. PP		√		√										√
Topic. NP compl.		√								√				
Imperative	√	√				√	√			√	√		√	√
Det. gerund	√	√												
NP gerund	√	√				√	√			√	√		√	√
Rel. cl. on subj. w/ NP	√	√				√	√			√	√		√	√
Rel. cl. on subj. w/ COMP	√	√				√	√			√	√		√	√
Rel. cl. on NP compl., DO, IO w/ NP		√											√	
Rel. cl. on NP compl., DO, IO w/ COMP		√											√	
Rel. cl. on PP compl. w/ pied-piping		√											√	
Rel. cl. on NP object of P w/ NP		√											√	
Rel. cl. on NP object of P w/ COMP		√												
Rel. cl. on adjunct w/ PP	√	√	√	√	√	√	√	√		√	√	√	√	√
Rel. cl. on adjunct w/ COMP	√	√	√	√	√	√	√	√		√	√	√	√	√
Pass. w/ wh-mov. ind. obj.		√												
PRO in Decl.	√	√				√	√			√	√		√	√
PRO in NP ger.	√	√				√	√			√	√		√	√
PRO subj. in decl. pass. w/ *by* phrase		√												
PRO subj. in decl. pass. w/o *by* phrase		√												
PRO subj. in ger. pass. w/ *by* phrase		√												
PRO subj. in ger. pass. w/o *by* phrase		√												

	Tree families													
Constructions	PP Sm-Cl. w/ SS., Ts0Pnx1	PP Sm-Cl. w/ Ad,P Pred., Tnx0ARBPnx1	PP Sm-Cl. w/ A,P Pred., Tnx0APnx1	PP Sm-Cl. w/ N,P Pred., Tnx0NPnx1	PP Sm-Cl. w/ P,P Pred., Tnx0PPnx1	PP Sm-Cl. w/ P,N Pred., Tnx0PNaPnx1	PP Sm-Cl. w/ SS., & Ad,P Pred., Ts0ARBPnx1	PP Sm-Cl. w/ SS., & A,P Pred., Ts0APnx1	PP Sm-Cl. w/ SS., & N,P Pred., Ts0NPnx1	PP Sm-Cl. w/ SS., & P,P Pred., Ts0PPnx1	PP Sm-Cl. w/ SS., & P,N Pred., Ts0PNaPnx1	Intr. SS., Ts0V	SS. w/ 'to' compl., Ts0Vtonx1	ECM, TXnx0Vs1
Declarative	√	√	√	√	√	√	√	√	√	√	√	√	√	94
Wh-moved subject	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Wh-mov. NP compl., DO or IO			√	√										
Wh-mov. object of a mod.					√	√								
Wh-mov. PP		√			√	√								
Imperative		√			√	√								√
NP gerund		√			√	√								√
Rel. cl. on subj. w/ NP		√	√	√	√	√								√
Rel. cl. on subj. w/ COMP		√	√	√	√	√								√
Rel. cl. on NP compl., DO, IO w/ NP	√				√	√								
Rel. cl. on NP compl., DO, IO w/ COMP	√													
Rel. cl. on PP compl. w/ pied-piping	√				√	√								
Rel. cl. on NP object of P w/ NP		√	√	√	√	√	√	√	√	√	√			
Rel. cl. on NP object of P w/ COMP		√	√	√	√	√	√	√	√	√	√			
Rel. cl. on adjunct w/ PP	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Rel. cl. on adjunct w/ COMP	√	√	√	√	√	√	√	√	√	√	√	√	√	√
PRO in Decl.		√	√	√	√	√								√
PRO in NP ger.		√			√	√								

	Tree families													
Constructions	Loc. Sm-Cl. w/ Ad. Pred., Tnx0nx1ARB	Idiom w/ V,D,N anchors, Tnx0VDN1	Idiom w/ V,D,A,N anchors, Tnx0VDAN1	Idiom w/ V,N Y anchors, Tnx0VN1	Idiom w/ V,A,N anchors, Tnx0VAN1	Idiom w/ V,D,A,N,P anchors, Tnx0VDAN1Pnx2	Idiom w/ V,A,N,P anchors, Tnx0VAN1Pnx2	Idiom w/ V,N,P anchors, Tnx0VN1Pnx2	Idiom w/ V,D,N,P anchors, Tnx0VDN1Pnx2	Tr/Intr. Res. w/ V,A Pred., TRnx0Vnx1A2	Tr/Intr. Res. w/ V,P Pred., TRnx0Vnx1Pnx2	Erg. Res. w/ V,A Pred., TREnx1VA2	Erg. Res. w/ V,P Pred., TREnx1VPnx2	SS. w/ Sm-Cl. compl., Ts0Vs1
Declarative	72	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Passive w/ & w/o *by* phrase		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Wh-moved subject	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Wh-mov. NP compl., DO or IO										✓				
Wh-mov. Adj. or Ad. compl.	72													
Wh-mov. object of a mod.										✓		✓		
Wh-mov. PP	✓										✓		✓	
Imperative	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
NP gerund	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Rel. cl. on subj. w/ NP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Rel. cl. on subj. w/ COMP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Rel. cl. on NP compl., DO, IO w/ NP										✓	✓			
Rel. cl. on NP compl., DO, IO w/ COMP										✓	✓		✓	
Rel. cl. on PP compl. w/ pied-piping										✓	✓	✓		
Rel. cl. on NP object of P w/ NP											✓			
Rel. cl. on NP object of P w/ COMP											✓			
Rel. cl. on adjunct w/ PP		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Rel. cl. on adjunct w/ COMP		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Participial NP pre-mod										✓	✓			
Pass. w/ wh-mov. subj.										✓	✓			
Pass. w/ wh-mov. obj. of *by* phrase		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Pass. w/ wh-mov. *by* phrase		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
PRO in Decl.	✓									✓	✓			
PRO in NP ger.														
PRO subj. in decl. pass. w/ *by* phrase	✓									✓	✓			
PRO subj. in decl. pass. w/o *by* phrase										✓	✓			
PRO subj. in ger. pass. w/ *by* phrase										✓	✓			
PRO subj. in ger. pass. w/o *by* phrase										✓	✓			
Outer Pass. w/ and w/o *by* phrase	✓					✓	✓	✓	✓					
Outer Pass. w/ Rel. cl. on subj. w/ COMP	✓					✓	✓	✓	✓					
Outer Pass. w/ Rel. cl. on subj. w/ NP	✓					✓	✓	✓	✓					

Chapter 6

Verb Classes

Each main[1] verb in the syntactic lexicon selects at least one tree family[2] (subcategorization frame). Since the tree database and syntactic lexicon are already separated for space efficiency (see Chapter 3), each verb can efficiently select a large number of trees by specifying a tree family, as opposed to each of the individual trees. This approach allows for a considerable reduction in the number of trees that must be specified for any given verb or form of a verb.

There are currently 57 tree families in the system.[3] This chapter gives a brief description of each tree family and shows the corresponding declarative tree[4], along with any peculiar characteristics or trees. It also indicates which transformations are in each tree family, and gives the number of verbs that select that family.[5] A few sample verbs are given, along with example sentences.

6.1 Intransitive: Tnx0V

Description: This tree family is selected by verbs that do not require an object complement of any type. Adverbs, prepositional phrases and other adjuncts may adjoin on, but are not required for the sentences to be grammatical. 1,941 verbs select this family.

Examples: *eat*, *sleep*, *dance*
Al ate .
Seth slept .
Hyun danced .

Declarative tree: See Figure 6.1.

Other available trees: wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP, adjunct (gap-less) relative clause with covert extracted *wh*-NP, adjunct

[1]Auxiliary verbs are handled under a different mechanism. See Chapter 22 for details.
[2]See section 3.1.2 for explanation of tree families.
[3]An explanation of the naming convention used in naming the trees and tree families is available in Appendix 29.
[4]Before lexicalization, the $\diamond$ indicates the anchor of the tree.
[5]Numbers given are as of December 2000 and are subject to some change with further development of the grammar.

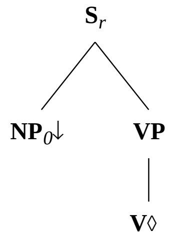


Figure 6.1: Declarative Intransitive Tree: αnx0V

(gap-less) relative clause with PP pied-piping, imperative, determiner gerund, NP gerund, pre-nominal participial modifier, PRO subject, NP gerund with PRO subject.

6.2 Ergative: TEnx1V

Description: This tree family is selected by verbs for which the syntactic subject corresponds to the logical object, which is indicated by the index of '1' on the subject NP. See Chapter 7 for more information about the ergative alternation. 292 verbs select the ergative tree family.

Examples: *sink*, *melt*
The ship sank .
The ice melted .

Declarative tree: See Figure 6.2.

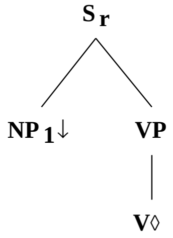


Figure 6.2: Declarative Ergative Tree: αEnx1V

Other available trees: wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP, adjunct (gap-less) relative clause with PP pied-piping, NP gerund, determiner gerund, imperative, PRO subject, NP gerund with PRO subject, prenominal participial modifier.

6.3 Transitive: Tnx0Vnx1

Description: This tree family is selected by verbs that require only an NP object complement.

The NP's may be complex structures, including gerund NP's and NP's that take sentential complements. This does not include light verb constructions (see sections 6.16 and 6.17). 4,506 verbs select the transitive tree family.

Examples: *eat*, *dance*, *take*, *like*

Al ate an apple .

Seth danced the tango .

Hyun is taking an algorithms course .

Anoop likes the fact that the semester is finished .

Declarative tree: See Figure 6.3.

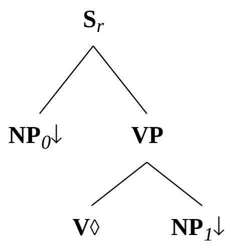


Figure 6.3: Declarative Transitive Tree: αnx0Vnx1

Other available trees: wh-moved subject, wh-moved object, subject relative clause with overt and covert extracted *wh* NP's, adjunct (gap-less) relative clause with covert extracted *wh* NP/with PP pied-piping, object relative clause with overt and covert extracted *wh*-NP's, imperative, determiner gerund, NP gerund, passive with *by* phrase, passive without *by* phrase, passive with wh-moved subject and *by* phrase, passive with wh-moved subject and no *by* phrase, passive with wh-moved object out of the *by* phrase, passive with wh-moved *by* phrase, passive with relative clause on subject and *by* phrase overt and covert extracted *wh* NP's, passive with relative clause on subject and no *by* phrase with overt and covert extracted *wh* NP's, passive with relative clause on object on the *by* phrase with overt and covert extracted *wh* NP's/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase, PRO subject, NP gerund with PRO subject, passive with *by*-phrase with PRO subject, passive without *by* phrase with PRO subject, gerund passive with *by* phrase with PRO subject, gerund passive without *by* phrase with PRO subject, prenominal participial modifiers, derived adjectives.

6.4 Ditransitive: Tnx0Vnx2nx1

Description: This tree family is selected by verbs that take exactly two NP complements. The apparent ditransitive alternates involving verbs in this class and benefactive PP's (e.g. *John baked a cake for Mary*) are analyzed as transitives (see section 6.3) with a PP adjunct. Benefactives are taken to be adjunct PP's because they are optional (e.g. *John baked a cake* vs. *John baked a cake for Mary*). 167 verbs select the ditransitive tree family.

Examples: *ask*, *cook*, *win*

Christy asked Mike a question .
Doug cooked his father dinner .
Dania won her sister a stuffed animal .

Declarative tree: See Figure 6.4.

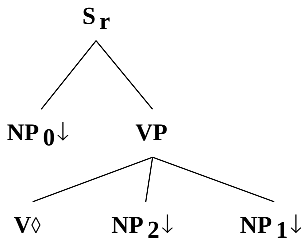


Figure 6.4: Declarative Ditransitive Tree: αnx0Vnx2nx1

Other available trees: wh-moved subject, wh-moved direct object, wh-moved indirect object, subject relative clause with covert and overt extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh* NP/with PP pied-piping, direct object relative clause with covert and overt extracted *wh*-NP's, indirect object relative clause with covert and overt extracted *wh*-NP's, imperative, determiner gerund, NP gerund, passive with *by* phrase, passive without *by* phrase, passive with wh-moved subject and *by* phrase, passive with wh-moved subject and no *by* phrase, passive with wh-moved object out of the *by* phrase, passive with wh-moved *by* phrase, passive with wh-moved indirect object and *by* phrase, passive with wh-moved indirect object and no *by* phrase, passive with relative clause on subject and *by* phrase with covert and overt extracted *wh*-NP's, passive with relative clause on subject and no *by* phrase with covert and overt extracted *wh*-NP's, passive with relative clause on object of the *by* phrase with covert and overt extracted *wh*-NP's/with PP pied-piping, passive with relative clause on the indirect object and *by* phrase with covert and overt extracted *wh*-NP's, passive with relative clause on the indirect object and no *by* phrase with covert and overt extracted *wh*-NP's, passive with/without *by*-phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase, PRO subject, passive with and without *by*-phrase with PRO subject, NP gerund with PRO subject, NP gerund passive with and without the *by*-phrase and PRO subject.

6.5 Ditransitive with PP: Tnx0Vnx1pnx2

Description: This tree family is selected by ditransitive verbs that take a noun phrase followed by a prepositional phrase. The preposition is not constrained in the syntactic lexicon. The preposition must be required and not optional - that is, the sentence must be ungrammatical with just the noun phrase (e.g. $*$*John put the table*). No verbs, therefore, should select both this tree family and the transitive tree family (see section 6.3). There are 62 verbs that select this tree family.

Examples: *ensconce*, *put*, *usher*

Mary ensconced herself on the sofa .

He put the book on the table .

He ushered the patrons into the theater .

Declarative tree: See Figure 6.5.

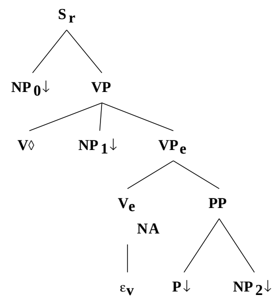


Figure 6.5: Declarative Ditransitive with PP Tree: αnx0Vnx1pnx2

Other available trees: wh-moved subject, wh-moved direct object, wh-moved object of PP, wh-moved PP, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, direct object relative clause with overt and covert extracted *wh*-NP's, object of PP relative clause with overt and covert extracted *wh*-NP's/with PP pied-piping, imperative, determiner gerund, NP gerund, passive with *by* phrase, passive without *by* phrase, passive with wh-moved subject and *by* phrase, passive with wh-moved subject and no *by* phrase, passive with wh-moved object out of the *by* phrase, passive with wh-moved *by* phrase, passive with wh-moved object out of the PP and *by* phrase, passive with wh-moved object out of the PP and no *by* phrase, passive with wh-moved PP and *by* phrase, passive with wh-moved PP and no *by* phrase, passive with relative clause on subject and *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on subject and no *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on object of the *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with relative clause on the object of the PP and *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with relative clause on the object of the PP and no *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with and without *by* phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase, PRO subject, passive with and without *by*-phrase with PRO subject, NP gerund with PRO subject, NP gerund passive with and without the *by*-phrase and PRO subject.

6.6 Multiple anchor ditransitive with PP: Tnx0Vnx1Pnx2

Description: This tree family is selected by ditransitive verbs that take a noun phrase followed by a prepositional phrase headed by a particular preposition. The preposition is constrained by making it one of the anchors. There are 84 verbs that select this tree family.

Examples: *gear for*, *give to*, *remind of*
The attorney geared his client for the trial . He gave the book to his teacher .
The city reminded John of his home town .

Declarative tree: See Figure 6.6.

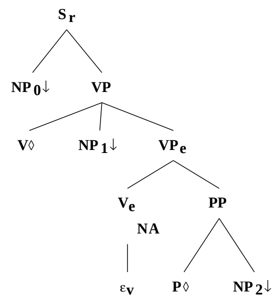


Figure 6.6: Declarative Multiple anchor Ditransitive with PP Tree: αnx0Vnx1Pnx2

Other available trees: wh-moved subject, wh-moved direct object, wh-moved object of PP, wh-moved PP, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, direct object relative clause with overt and covert extracted *wh*-NP's, object of PP relative clause with overt and covert extracted *wh*-NP's/with PP pied-piping, imperative, determiner gerund, NP gerund, passive with *by* phrase, passive without *by* phrase, passive with wh-moved subject and *by* phrase, passive with wh-moved subject and no *by* phrase, passive with wh-moved object out of the *by* phrase, passive with wh-moved *by* phrase, passive with wh-moved object out of the PP and *by* phrase, passive with wh-moved object out of the PP and no *by* phrase, passive with wh-moved PP and *by* phrase, passive with wh-moved PP and no *by* phrase, passive with relative clause on subject and *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on subject and no *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on object of the *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with relative clause on the object of the PP and *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with relative clause on the object of the PP and no *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with and without *by* phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase, PRO subject, passive with and without *by*-phrase with PRO subject, NP gerund with PRO subject, NP gerund passive with and without the *by*-phrase and PRO subject.

6.7 Sentential Complement with NP: Tnx0Vnx1s2

Description: This tree family is selected by verbs that take both an NP and a sentential complement. The sentential complement may be infinitive or indicative. The type of clause is specified by each individual verb in its syntactic lexicon entry. A given verb may select more than one type of sentential complement. The declarative tree, and many other trees in this family, are auxiliary trees, as opposed to the more common initial trees. These auxiliary trees adjoin onto an S node in an existing tree of the type specified by the sentential complement. This is the mechanism by which TAGs are able to maintain long-distance dependencies (see Chapter 15), even over multiple embeddings (e.g. *What did Bill tell Mary that John said?*). 83 verbs select this tree family.

Examples: *beg*, *expect*, *tell*

Srini begged Mark to increase his disk quota .

Beth told Jim that it was his turn .

Declarative tree: See Figure 6.7.

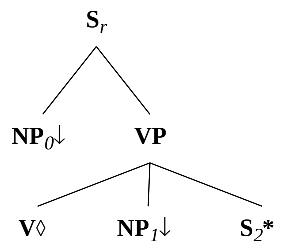


Figure 6.7: Declarative Sentential Complement with NP Tree: βnx0Vnx1s2

Other available trees: wh-moved subject, wh-moved object, wh-moved sentential complement, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, object relative clause with overt and covert extracted *wh*-NP's, imperative, determiner gerund, NP gerund, passive with *by* phrase before sentential complement, passive with *by* phrase after sentential complement, passive without *by* phrase, passive with wh-moved subject and *by* phrase before sentential complement, passive with wh-moved subject and *by* phrase after sentential complement, passive with wh-moved subject and no *by* phrase, passive with wh-moved object out of the *by* phrase, passive with wh-moved *by* phrase, passive with relative clause on subject and *by* phrase before sentential complement with overt and covert extracted *wh*-NP's, passive with relative clause on subject and *by* phrase after sentential complement with overt and covert extracted *wh*-NP's, passive with relative clause on subject and no *by* phrase with overt and covert extracted *wh*-NP's, passive with/without *by*-phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase before sentential complement, gerund passive with *by* phrase after the sentential complement, gerund passive without *by* phrase, parenthetical reporting clause, PRO subject, passive with PRO subject with *by*-phrase before and after sentential complement, passive with PRO subject without *by*-phrase, NP gerund with PRO subject, NP gerund passive with PRO subject with *by*-phrase before and after sentential complement, NP gerund passive with PRO subject without *by*-phrase.

6.8 Intransitive Verb Particle: Tnx0Vpl

Description: The trees in this tree family are anchored by both the verb and the verb particle. Both appear in the syntactic lexicon and together select this tree family. Intransitive verb particles can be difficult to distinguish from intransitive verbs with adverbs adjoined on. The main diagnostics for including verbs in this class are whether the meaning is compositional or not, and whether there is a transitive version of the verb/verb particle combination with the same or similar meaning. The existence of an alternate compositional meaning is a strong indication for a separate verb particle construction. There are 159 verb/verb particle combinations.

Examples: *add up*, *come out*, *sign off*
The numbers never quite added up .
John finally came out (of the closet) .
I think that I will sign off now .

Declarative tree: See Figure 6.8.

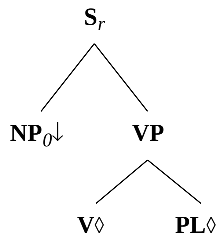


Figure 6.8: Declarative Intransitive Verb Particle Tree: αnx0Vpl

Other available trees: wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, imperative, determiner gerund, NP gerund, PRO subject, NP gerund with PRO subject.

6.9 Transitive Verb Particle: Tnx0Vplnx1

Description: Verb/verb particle combinations that take an NP complement select this tree family. Both the verb and the verb particle are anchors of the trees. Particle movement has been taken as the diagnostic to distinguish verb particle constructions from intransitives with adjoined PP's. If the alleged particle is able to undergo particle movement, in other words appear both before and after the direct object, then it is judged to be a particle. Items that do not undergo particle movement are taken to be prepositions. In many, but not all, of the verb particle cases, there is also an alternate prepositional meaning in which the lexical item did not move. (e.g. *He looked up the number (in the phonebook). He looked the number up. Srini looked up the road (for Purnima's car). *He looked the road up.*) There are 548 verb/verb particle combinations.

Examples: *blow off*, *make up*, *pick out*

He blew off his linguistics class for the third time .
He blew his linguistics class off for the third time .
The dyslexic leprechaun made up the syntactic lexicon .
The dyslexic leprechaun made the syntactic lexicon up .
I would like to pick out a new computer .
I would like to pick a new computer out .

Declarative tree: See Figure 6.9.

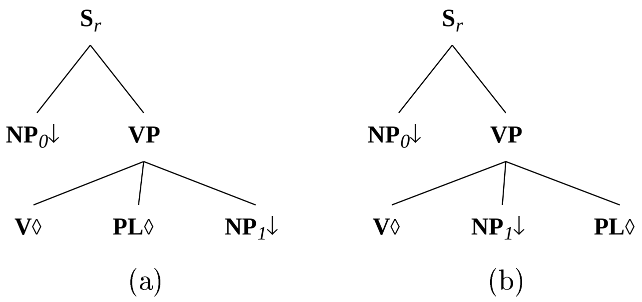


Figure 6.9: Declarative Transitive Verb Particle Tree: αnx0Vplnx1 (a) and αnx0Vnx1pl (b)

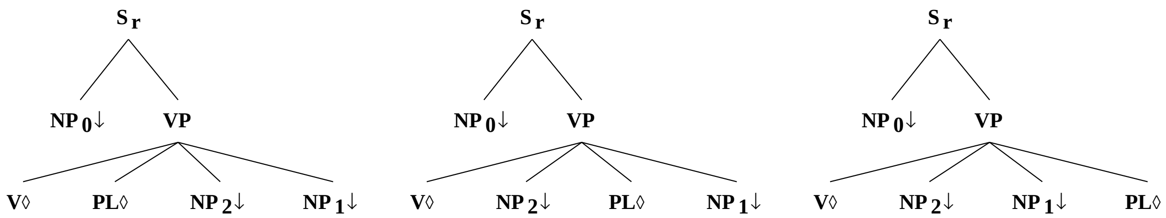
Other available trees: wh-moved subject with particle before the NP, wh-moved subject with particle after the NP, wh-moved object, subject relative clause with particle before the NP with overt and covert extracted *wh*-NP's, subject relative clause with particle after the NP with overt and covert extracted *wh*-NP's, object relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with particle before the NP with covert extracted *wh*-NP/with PP pied-piping, adjunct (gap-less) relative clause with particle after the NP with covert extracted *wh*-NP/with PP pied-piping, imperative with particle before the NP, imperative with particle after the NP, determiner gerund with particle before the NP, NP gerund with particle before the NP, NP gerund with particle after the NP, passive with *by* phrase, passive without *by* phrase, passive with wh-moved subject and *by* phrase, passive with wh-moved subject and no *by* phrase, passive with wh-moved object out of the *by* phrase, passive with wh-moved *by* phrase, passive with relative clause on subject and *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on subject and no *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on object of the *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with/without *by*-phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase, PRO subject with verb particle before and after the NP, passive with PRO subject with and without the *by* phrase, NP gerund with PRO subject with verb particle before and after the NP, NP gerund passive with PRO subject with and without the *by* phrase.

6.10 Ditransitive Verb Particle: Tnx0Vplnx2nx1

Description: Verb/verb particle combinations that select this tree family take 2 NP complements. Both the verb and the verb particle anchor the trees, and the verb particle can occur before, between, or after the noun phrases. Perhaps because of the complexity of the sentence, these verbs do not seem to have passive alternations (*∗A new bank account was opened up Michelle by me*). There are 4 verb/verb particle combinations that select this tree family. The exhaustive list is given in the examples.

Examples: *dish out*, *open up*, *pay off*, *rustle up*
I opened up Michelle a new bank account .
I opened Michelle up a new bank account .
I opened Michelle a new bank account up .

Declarative tree: See Figure 6.10.



(a) (b) (c)

Figure 6.10: Declarative Ditransitive Verb Particle Tree: αnx0Vplnx2nx1 (a), αnx0Vnx2plnx1 (b) and αnx0Vnx2nx1pl (c)

Other available trees: wh-moved subject with particle before the NP's, wh-moved subject with particle between the NP's, wh-moved subject with particle after the NP's, wh-moved indirect object with particle before the NP's, wh-moved indirect object with particle after the NP's, wh-moved direct object with particle before the NP's, wh-moved direct object with particle between the NP's, subject relative clause with particle before the NP's with overt and covert extracted *wh*-NP's, subject relative clause with particle between the NP's with overt and covert extracted *wh*-NP's, subject relative clause with particle after the NP's with overt and covert extracted *wh*-NP's, indirect object relative clause with particle before the NP's with overt and covert extracted *wh*-NP's, indirect object relative clause with particle after the NP's with overt and covert extracted *wh*-NP's, direct object relative clause with particle before the NP's with overt and covert extracted *wh*-NP's, direct object relative clause with particle between the NP's with and without comp, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, imperative with particle before the NP's, imperative with particle between the NP's, imperative with particle after the NP's, determiner gerund with particle before the NP's, NP gerund with particle before the NP's, NP gerund with particle between the NP's, NP gerund with particle after the NP's, PRO subject with particles before NP's, PRO subject with particles between the NP's, PRO subject with particles after the NP's, NP gerund with PRO subject with particles before NP's, NP gerund with PRO subject with particles between the NP's, NP gerund with PRO subject with particles after the NP's,

6.11 Intransitive with PP: Tnx0Vpnx1

Description: The verbs that select this tree family are not strictly intransitive, in that they **must** be followed by a prepositional phrase. Verbs that are intransitive and simply **can** be followed by a prepositional phrase do not select this family, but instead have the PP adjoin onto the intransitive sentence. Accordingly, there should be no verbs in both this class and the intransitive tree family (see section 6.1). The prepositional phrase is not restricted to being headed by any particular lexical item. The PP is expanded to facilitate extraction of the PP-internal NP. Note that these are not transitive verb particles (see section 6.9), since the head of the PP does not move. 22 verbs select this tree family.

Examples: *grab*, *venture slog*
Seth grabbed for the brass ring.
Jones ventured into the cave.
The soldiers slogged grudgingly through the mud.

Declarative tree: See Figure 6.11.

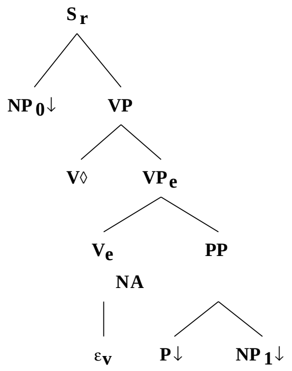


Figure 6.11: Declarative Intransitive with PP Tree: αnx0Vpnx1

Other available trees: wh-moved subject, wh-moved object of the PP, wh-moved PP, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, object of the PP relative clause with overt and covert extracted *wh*-NP's/with PP pied-piping, imperative, determiner gerund, NP gerund, passive with *by* phrase, passive without *by* phrase, passive with wh-moved subject and *by* phrase, passive with wh-moved subject and no *by* phrase, passive with wh-moved *by* phrase, passive with relative clause on subject and *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on subject and no *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on object of the *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with/without *by*-phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase, PRO subject, passive with PRO subject with and without the *by* phrase, NP gerund with PRO subject, NP gerund passive with PRO subject with and without the *by* phrase.

6.12 Multiple anchor PP complement: Tnx0VPnx1

Description: This tree family is selected by multiple anchor verb/preposition pairs which together have a non-compositional interpretation. For example, *think of* has the non-compositional interpretation involving the inception of a notion or mental entity in addition to the interpretation in which the agent is thinking about someone or something. To allow adverbs to appear between the verb and the preposition, the trees contain an extra VP level. 206 verb/preposition pairs select this tree family.

Examples: *think of*, *believe in*, *depend on*
Calvin thought of a new idea .
Hobbes believes in sleeping all day .
Bill depends on drinking coffee for stimulation .

Declarative tree: See Figure 6.12.

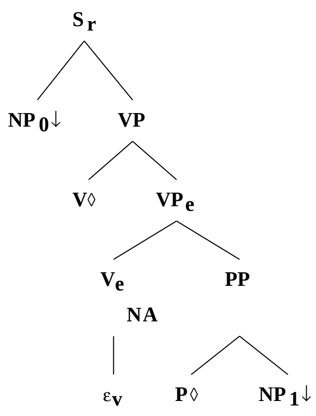


Figure 6.12: Declarative PP Complement Tree: αnx0VPnx1

Other available trees: wh-moved subject, wh-moved object, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, object relative clause with overt and covert extracted *wh*-NP's, imperative, determiner gerund, NP gerund, passive with *by* phrase, passive without *by* phrase, passive with wh-moved subject and *by* phrase, passive with wh-moved subject and no *by* phrase, passive with wh-moved object out of the *by* phrase, passive with wh-moved *by* phrase, passive with relative clause on subject and *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on subject and no *by* phrase with overt and covert extracted *wh*-NP's, passive with relative clause on object on the *by* phrase with overt and covert extracted *wh*-NP's/with PP pied-piping, passive with/without *by*-phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase. In addition, two other trees that allow transitive verbs to function as adjectives (e.g. *the thought of idea*) are also in the family, PRO subject, passive with PRO subject with and without the *by* phrase, NP gerund with PRO subject, NP gerund passive with PRO subject with and without the *by* phrase.

6.13 Sentential Complement: Tnx0Vs1

Description: This tree family is selected by verbs that take just a sentential complement. The sentential complement may be of type infinitive, indicative, or small clause (see Chapter 9). The type of clause is specified by each individual verb in its syntactic lexicon entry, and a given verb may select more than one type of sentential complement. The declarative tree, and many other trees in this family, are auxiliary trees, as opposed to the more common initial trees. These auxiliary trees adjoin onto an S node in an existing tree of the type specified by the sentential complement. This is the mechanism by which TAGs are able to maintain long-distance dependencies (see Chapter 15), even over multiple embeddings (e.g. *What did Bill think that John said?*). 353 verbs select this tree family.

Examples: *consider*, *think*
Dania considered the algorithm unworkable .
Srini thought that the program was working .

Declarative tree: See Figure 6.13.

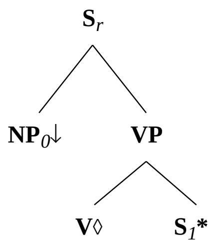


Figure 6.13: Declarative Sentential Complement Tree: βnx0Vs1

Other available trees: wh-moved subject, wh-moved sentential complement, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, imperative, determiner gerund, NP gerund, parenthetical reporting clause trees, PRO subject, NP gerund with PRO subject.

6.14 Intransitive with Adjective: Tnx0Vax1

Description: The verbs that select this tree family take an adjective as a complement. The adjective may be regular, comparative, or superlative. It may also be formed from the special class of adjectives derived from the transitive verbs (e.g. *agitated, broken*). See section 6.3). Unlike the Intransitive with PP verbs (see section 6.11), some of these verbs may also occur as bare intransitives as well. This distinction is drawn because adjectives do not normally adjoin onto sentences, as prepositional phrases do. Other intransitive verbs can only occur with the adjective, and these select only this family. The verb class is also distinguished from the adjective small clauses (see section 6.21) because these verbs are not raising verbs. 34 verbs select this tree family.

Examples: *become*, *grow*, *smell*
The greenhouse became hotter .
The plants grew tall and strong .
The flowers smelled wonderful .

Declarative tree: See Figure 6.14.

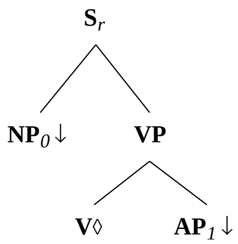


Figure 6.14: Declarative Intransitive with Adjective Tree: αnx0Vax1

Other available trees: wh-moved subject, wh-moved adjective (*how*), subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, imperative, NP gerund, PRO subject, NP gerund with PRO subject.

6.15 Transitive Sentential Subject: Ts0Vnx1

Description: The verbs that select this tree family all take sentential subjects, and are often referred to as 'psych' verbs, since they all refer to some psychological state of mind. The sentential subject can be indicative (complementizer required) or infinitive (complementizer optional). 98 verbs that select this tree family.

Examples: *delight*, *impress*, *surprise*
that the tea had rosehips in it delighted Christy .
to even attempt a marathon impressed Dania .
For Jim to have walked the dogs surprised Beth .

Declarative tree: See Figure 6.15.

Other available trees: wh-moved subject, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

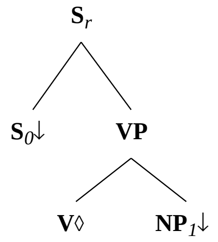


Figure 6.15: Declarative Sentential Subject Tree: αs0Vnx1

6.16 Light Verbs: Tnx0lVN1

Description: The verb/noun pairs that select this tree families are pairs in which the interpretation is non-compositional and the noun contributes argument structure to the predicate (e.g. *The man took a walk.* vs. *The man took a radio*). The verb and the noun occur together in the syntactic database, and both anchor the trees. The verbs in the light verb constructions are *do*, *give*, *have*, *make* and *take*. The noun following the light verb is (usually) in a bare infinitive form (*have a good cry*) and usually occurs with *a(n)*. However, we include deverbal nominals (*take a bath*, *give a demonstration*) as well. Constructions with nouns that do not contribute an argument structure (*have a cigarette*, *give* NP *a black eye*) are excluded. In addition to semantic considerations of light verbs, they differ syntactically from Transitive verbs (section 6.3) as well in that the noun in the light verb construction does not extract. Some of the verb-noun anchors for this family, like *take aim* and *take hold* disallow determiners, while others require particular determiners. For example, *have think* must be indefinite and singular, as attested by the ungrammaticality of **John had the think/some thinks.* Another anchor, *take leave* can occur either bare or with a possesive pronoun (e.g., *John took his leave*, but not **John took the leave*). This is accomplished through feature specification on the lexical entries. There are 242 verb/noun pairs that select the light verb tree.

Examples: *give groan*, *have discussion*, *make comment*
The audience gave a collective groan .
We had a big discussion about closing the libraries .
The professors made comments on the paper .

Declarative tree: See Figure 6.16.

Other available trees: wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, imperative, determiner gerund, NP gerund, PRO subject, NP gerund with PRO subject.

6.17 Ditransitive Light Verbs with PP Shift: Tnx0lVN1Pnx2

Description: The verb/noun pairs that select this tree family are pairs in which the interpretation is non-compositional and the noun contributes argument structure to the predicate

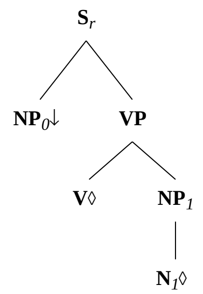


Figure 6.16: Declarative Light Verb Tree: αnx0lVN1

(e.g. *Dania made Srini a cake.* vs. *Dania made Srini a loan*). The verb and the noun occur together in the syntactic database, and both anchor the trees. The verbs in these light verb constructions are *give* and *make.* The noun following the light verb is (usually) a bare infinitive form (e.g. *make a promise to Anoop*). However, we include deverbal nominals (e.g. *make a payment to Anoop*) as well. Constructions with nouns that do not contribute an argument structure are excluded. In addition to semantic considerations of light verbs, they differ syntactically from the Ditransitive with PP Shift verbs (see section 6.6) as well in that the noun in the light verb construction does not extract. Also, passivization is severely restricted. Special determiner requirments and restrictions are handled in the same manner as for the Tnx0lVN1 family. There are 18 verb/noun pairs that select this family.

Examples: *give look*, *give wave*, *make promise*
Dania gave Carl a murderous look .
Amanda gave us a little wave as she left .
Dania made Doug a promise .

Declarative tree: See Figure 6.17.

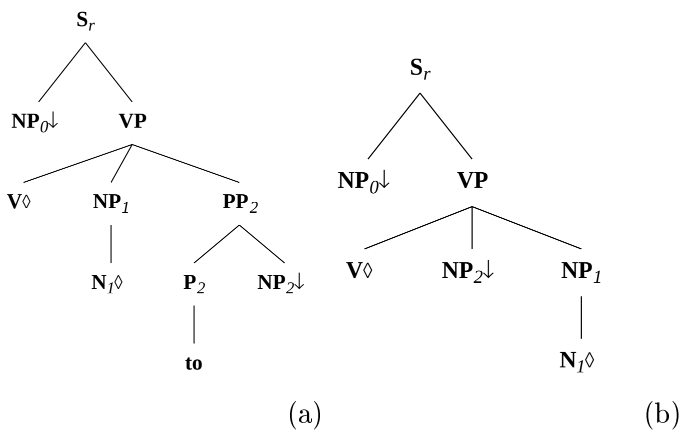


Figure 6.17: Declarative Light Verbs with PP Tree: αnx0lVN1Pnx2 (a), αnx0lVnx2N1 (b)

Other available trees: Non-shifted: wh-moved subject, wh-moved indirect object, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, indirect object relative clause with overt and covert extracted *wh*-NP's/with PP pied-piping, imperative, NP gerund, passive with *by* phrase, passive with *by*-phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase, PRO subject, NP gerund with PRO subject, passive with PRO subject with *by* phrase, NP gerund passive with and without the *by* phrase

Shifted: wh-moved subject, wh-moved object of PP, wh-moved PP, subject relative clause with overt and covert extracted *wh*-NP's, object of PP relative clause with overt and covert extracted *wh*-NP's/with PP pied-piping, imperative, determiner gerund, NP gerund, passive with *by* phrase with adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, gerund passive with *by* phrase, gerund passive without *by* phrase, PRO subject, NP gerund with PRO subject

6.18 NP It-Cleft: TItVnx1s2

Description: This tree family is selected by *be* as the main verb and *it* as the subject. Together these two items serve as a multi-component anchor for the tree family. This tree family is used for it-clefts in which the clefted element is an NP and there are no gaps in the clause which follows the NP. The NP is interpreted as an adjunct of the following clause. See Chapter 12 for additional discussion.

Examples: *it be*

it was yesterday that we had the meeting .

Declarative tree: See Figure 6.18.

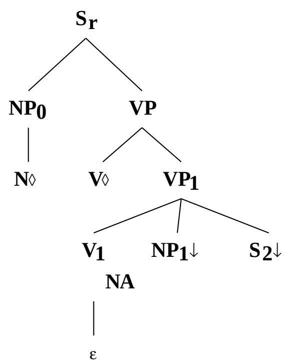


Figure 6.18: Declarative NP It-Cleft Tree: αItVnx1s2

Other available trees: inverted question, wh-moved object with *be* inverted, wh-moved object with *be* not inverted, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.19 PP It-Cleft: TItVpnx1s2

Description: This tree family is selected by *be* as the main verb and *it* as the subject. Together these two items serve as a multi-component anchor for the tree family. This tree family is used for it-clefts in which the clefted element is a PP and there are no gaps in the clause which follows the PP. The PP is interpreted as an adjunct of the following clause. See Chapter 12 for additional discussion.

Examples: *it be*
it was at Kent State that the police shot all those students .

Declarative tree: See Figure 6.19.

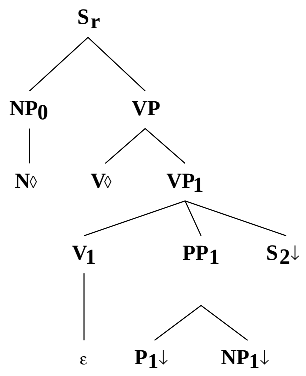


Figure 6.19: Declarative PP It-Cleft Tree: αItVpnx1s2

Other available trees: inverted question, wh-moved prepositional phrase with *be* inverted, wh-moved prepositional phrase with *be* not inverted, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.20 Adverb It-Cleft: TItVad1s2

Description: This tree family is selected by *be* as the main verb and *it* as the subject. Together these two items serve as a multi-component anchor for the tree family. This tree family is used for it-clefts in which the clefted element is an adverb and there are no gaps in the clause which follows the adverb. The adverb is interpreted as an adjunct of the following clause. See Chapter 12 for additional discussion.

Examples: *it be*
it was reluctantly that Dania agreed to do the tech report .

Declarative tree: See Figure 6.20.

Other available trees: inverted question, wh-moved adverb *how* with *be* inverted, wh-moved adverb *how* with *be* not inverted, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

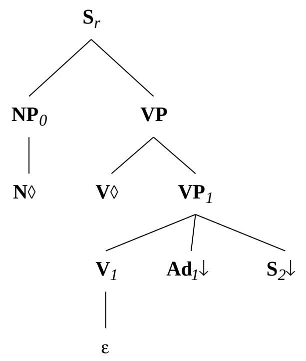


Figure 6.20: Declarative Adverb It-Cleft Tree: αItVad1s2

6.21 Adjective Small Clause Tree: Tnx0Ax1

Description: These trees are not anchored by verbs, but by adjectives. They are explained in much greater detail in the section on small clauses (see section 9.3). This section is presented here for completeness. 4150 adjectives select this tree family.

Examples: *addictive*, *dangerous*, *wary*
cigarettes are addictive .
smoking cigarettes is dangerous .
John seems wary of the Surgeon General's warnings .

Declarative tree: See Figure 6.21.

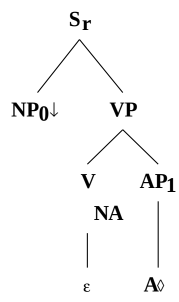


Figure 6.21: Declarative Adjective Small Clause Tree: αnx0Ax1

Other available trees: wh-moved subject, wh-moved adjective *how*, relative clause on subject with overt and covert extracted *wh*-NP's, imperative, NP gerund, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject, NP gerund with PRO subject.

6.22 Adjective Small Clause with Sentential Complement: Tnx0A1s1

Description: This tree family is selected by adjectives that take sentential complements. The sentential complements can be indicative or infinitive. Note that these trees are anchored by adjectives, not verbs. Small clauses are explained in much greater detail in section 9.3. This section is presented here for completeness. 673 adjectives select this tree family.

Examples: *able*, *curious*, *disappointed*
Christy was able to find the problem .
Christy was curious whether the new analysis was working .
Christy was sad that the old analysis failed .

Declarative tree: See Figure 6.22.

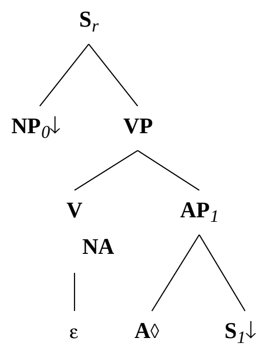


Figure 6.22: Declarative Adjective Small Clause with Sentential Complement Tree: αnx0A1s1

Other available trees: wh-moved subject, wh-moved adjective *how*, relative clause on subject with overt and covert extracted *wh*-NP's, imperative, NP gerund, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject, NP gerund with PRO subject.

6.23 Adjective Small Clause with Sentential Subject: Ts0Ax1

Description: This tree family is selected by adjectives that take sentential subjects. The sentential subjects can be indicative or infinitive. Note that these trees are anchored by adjectives, not verbs. Most adjectives that take the Adjective Small Clause tree family (see section 6.21) take this family as well.[6] Small clauses are explained in much greater detail in section 9.3. This section is presented here for completeness. 4,007 adjectives select this tree family.

Examples: *decadent*, *incredible*, *uncertain*
to eat raspberry chocolate truffle ice cream is decadent .
that Carl could eat a large bowl of it is incredible .
whether he will actually survive the experience is uncertain .

Declarative tree: See Figure 6.23.

[6] No great attempt has been made to go through and decide which adjectives should actually take this family and which should not.

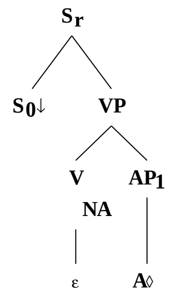


Figure 6.23: Declarative Adjective Small Clause with Sentential Subject Tree: αs0Ax1

Other available trees: wh-moved subject, wh-moved adjective, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.24 Equative *BE*: Tnx0BEnx1

Description: This tree family is selected only by the verb *be*. It is distinguished from the predicative NP's (see section 6.25) in that two NP's are equated, and hence interchangeable (see Chapter 9 for more discussion on the English copula and predicative sentences). The XTAG analysis for equative *be* is explained in greater detail in section 9.4.

Examples: *be*
That man is my uncle.

Declarative tree: See Figure 6.24.

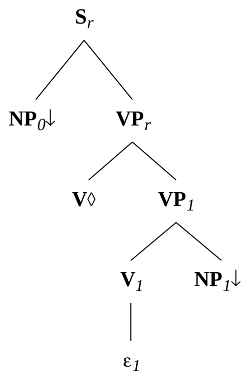


Figure 6.24: Declarative Equative *BE* Tree: αnx0BEnx1

Other available trees: inverted-question.

6.25 NP Small Clause: Tnx0N1

Description: The trees in this tree family are not anchored by verbs, but by nouns. Small clauses are explained in much greater detail in section 9.3. This section is presented here for completeness. 5,476 nouns select this tree family.

Examples: *author*, *chair*, *dish*

Dania is an author .

that blue, warped-looking thing is a chair .

those broken pieces were dishes .

Declarative tree: See Figure 6.25.

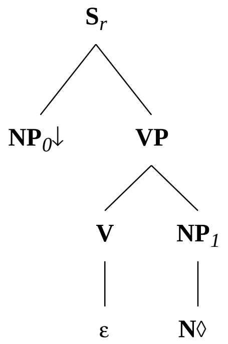


Figure 6.25: Declarative NP Small Clause Trees: αnx0N1

Other available trees: wh-moved subject, wh-moved object, relative clause on subject with overt and covert extracted *wh*-NP's, imperative, NP gerund, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject, NP gerund with PRO subject.

6.26 NP Small Clause with Sentential Complement: Tnx0N1s1

Description: This tree family is selected by the small group of nouns that take sentential complements by themselves (see section 8.8). The sentential complements can be indicative or infinitive, depending on the noun. Small clauses in general are explained in much greater detail in the section 9.3. This section is presented here for completeness. 144 nouns select this family.

Examples: *admission*, *claim*, *vow*

The affidavits are admissions that they killed the sheep .

there is always the claim that they were insane .

this is his vow to fight the charges .

Declarative tree: See Figure 6.26.

Other available trees: wh-moved subject, wh-moved object, relative clause on subject with overt and covert extracted *wh*-NP's, imperative, NP gerund, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject, NP gerund with PRO subject.

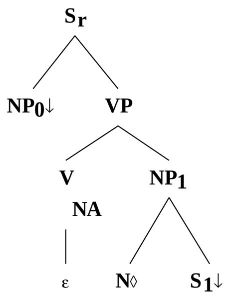


Figure 6.26: Declarative NP with Sentential Complement Small Clause Tree: αnx0N1s1

6.27 NP Small Clause with Sentential Subject: Ts0N1

Description: This tree family is selected by nouns that take sentential subjects. The sentential subjects can be indicative or infinitive. Note that these trees are anchored by nouns, not verbs. Most nouns that take the NP Small Clause tree family (see section 6.25) take this family as well.[7] Small clauses are explained in much greater detail in section 9.3. This section is presented here for completeness. 5,466 nouns select this tree family.

Examples: *dilemma*, *insanity*, *tragedy*
whether to keep the job he hates is a dilemma .
to invest all of your money in worms is insanity .
that the worms died is a tragedy .

Declarative tree: See Figure 6.27.

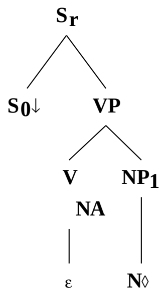


Figure 6.27: Declarative NP Small Clause with Sentential Subject Tree: αs0N1

Other available trees: wh-moved subject, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject, NP gerund with PRO subject.

6.28 PP Small Clause: Tnx0Pnx1

Description: This family is selected by prepositions that can occur in small clause construc-

[7]No great attempt has been made to go through and decide which nouns should actually take this family and which should not.

tions. For more information on small clause constructions, see section 9.3. This section is presented here for completeness. 39 prepositions select this tree family.

Examples: *around*, *in*, *underneath*
Chris is around the corner .
Trisha is in big trouble .
The dog is underneath the table .

Declarative tree: See Figure 6.28.

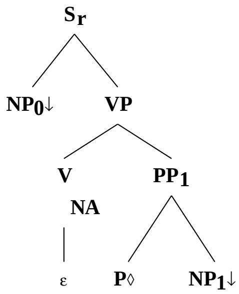


Figure 6.28: Declarative PP Small Clause Tree: αnx0Pnx1

Other available trees: wh-moved subject, wh-moved object of PP, relative clause on subject with overt and covert extracted *wh*-NP's, relative clause on object of PP with overt and covert extracted *wh*-NP's/with PP pied-piping, imperative, NP gerund, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject, NP gerund with PRO subject.

6.29 Exhaustive PP Small Clause: Tnx0Px1

Description: This family is selected by **exhaustive** prepositions that can occur in small clauses. Exhaustive prepositions are prepositions that function as prepositional phrases by themselves. For more information on small clause constructions, please see section 9.3. The section is included here for completeness. 33 exhaustive prepositions select this tree family.

Examples: *abroad*, *below*, *outside*
Dr. Joshi is abroad .
The workers are all below .
Clove is outside .

Declarative tree: See Figure 6.29.

Other available trees: wh-moved subject, wh-moved PP, relative clause on subject with overt and covert extracted *wh*-NP's, imperative, NP gerund, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject, NP gerund with PRO subject.

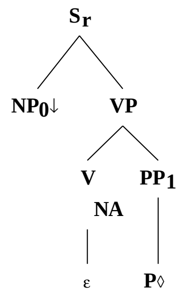


Figure 6.29: Declarative Exhaustive PP Small Clause Tree: αnx0Px1

6.30 PP Small Clause with Sentential Subject: Ts0Pnx1

Description: This tree family is selected by prepositions that take sentential subjects. The sentential subject can be indicative or infinitive. Small clauses are explained in much greater detail in section 9.3. This section is presented here for completeness. 39 prepositions select this tree family.

Examples: *beyond*, *unlike*

that Ken could forget to pay the taxes is beyond belief .

to explain how this happened is outside the scope of this discussion .

for Ken to do something right is unlike him .

Declarative tree: See Figure 6.30.

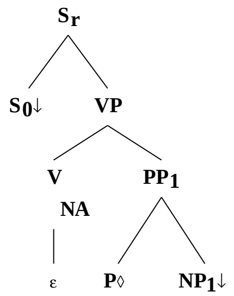


Figure 6.30: Declarative PP Small Clause with Sentential Subject Tree: αs0Pnx1

Other available trees: wh-moved subject, relative clause on object of the PP with overt and covert extracted *wh*-NP's/with PP pied-piping, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.31 Intransitive Sentential Subject: Ts0V

Description: Only the verb *matter* selects this tree family. The sentential subject can be indicative (complementizer required) or infinitive (complementizer optional).

Examples: *matter*

to arrive on time matters considerably .

that Joshi attends the meetings matters to everyone .

Declarative tree: See Figure 6.31.

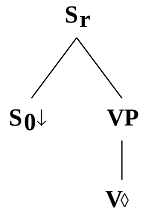


Figure 6.31: Declarative Intransitive Sentential Subject Tree: αs0V

Other available trees: wh-moved subject, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.32 Sentential Subject with 'to' complement: Ts0Vtonx1

Description: The verbs that select this tree family are *fall*, *occur* and *leak*. The sentential subject can be indicative (complementizer required) or infinitive (complementizer optional).

Examples: *fall*, *occur*, *leak*

to wash the car fell to the children .

that he should leave occurred to the party crasher .

whether the princess divorced the prince leaked to the press .

Declarative tree: See Figure 6.32.

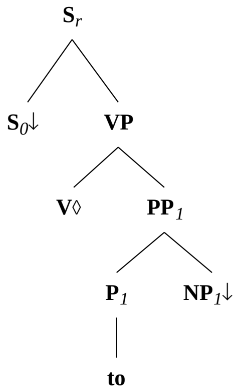


Figure 6.32: Sentential Subject Tree with 'to' complement: αs0Vtonx1

Other available trees: wh-moved subject, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.33 PP Small Clause, with Adv and Prep anchors: Tnx0ARBPnx1

Description: This family is selected by multi-word prepositions that can occur in small clause constructions. In particular, this family is selected by two-word prepositions, where the first word is an adverb, the second word a preposition. Both components of the multi-word preposition are anchors. For more information on small clause constructions, see section 9.3. 8 multi-word prepositions select this tree family.

Examples: *ahead of*, *close to*
The little girl is ahead of everyone else in the race .
The project is close to completion .

Declarative tree: See Figure 6.33.

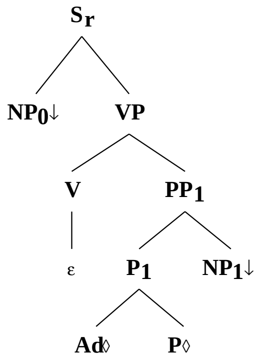


Figure 6.33: Declarative PP Small Clause tree with two-word preposition, where the first word is an adverb, and the second word is a preposition: αnx0ARBPnx1

Other available trees: wh-moved subject, wh-moved object of PP, relative clause on subject with overt and covert extracted *wh*-NP's, relative clause on object of PP with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, imperative, NP Gerund, PRO subject, NP gerund with PRO subject.

6.34 PP Small Clause, with Adj and Prep anchors: Tnx0APnx1

Description: This family is selected by multi-word prepositions that can occur in small clause constructions. In particular, this family is selected by two-word prepositions, where the first word is an adjective, the second word a preposition. Both components of the multi-word preposition are anchors. For more information on small clause constructions, see section 9.3. 7 multi-word prepositions select this tree family.

Examples: *according to*, *void of*
The operation we performed was according to standard procedure .

He is void of all feeling .

Declarative tree: See Figure 6.34.

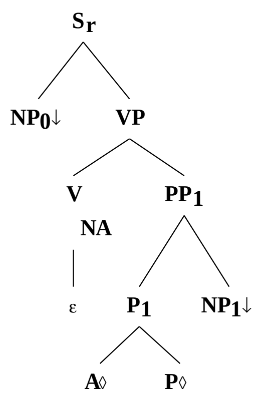


Figure 6.34: Declarative PP Small Clause tree with two-word preposition, where the first word is an adjective, and the second word is a preposition: αnx0APnx1

Other available trees: wh-moved subject, relative clause on subject with overt and covert extracted *wh*-NP's, relative clause on object of PP with overt and covert extracted *wh*-NP's, wh-moved object of PP, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject.

6.35 PP Small Clause, with Noun and Prep anchors: Tnx0NPnx1

Description: This family is selected by multi-word prepositions that can occur in small clause constructions. In particular, this family is selected by two-word prepositions, where the first word is a noun, the second word a preposition. Both components of the multi-word preposition are anchors. For more information on small clause constructions, see section 9.3. 1 multi-word preposition selects this tree family.

Examples: *thanks to*
The fact that we are here tonight is thanks to the valiant efforts of our staff .

Declarative tree: See Figure 6.35.

Other available trees: wh-moved subject, wh-moved object of PP, relative clause on subject with overt and covert extracted *wh*-NP's, relative clause on object with covert extracted *wh*-NP, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject.

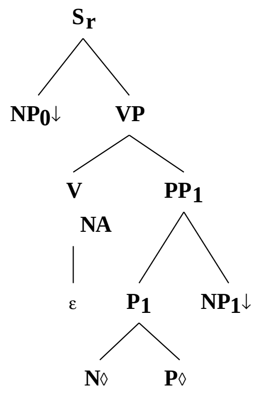


Figure 6.35: Declarative PP Small Clause tree with two-word preposition, where the first word is a noun, and the second word is a preposition: αnx0NPnx1

6.36 PP Small Clause, with Prep anchors: Tnx0PPnx1

Description: This family is selected by multi-word prepositions that can occur in small clause constructions. In particular, this family is selected by two-word prepositions, where both words are prepositions. Both components of the multi-word preposition are anchors. For more information on small clause constructions, see section 9.3. 9 multi-word prepositions select this tree family.

Examples: *on to*, *inside of*
that detective is on to you .
The red box is inside of the blue box .

Declarative tree: See Figure 6.36.

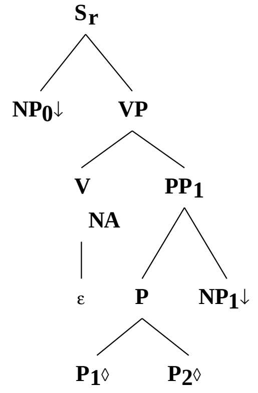


Figure 6.36: Declarative PP Small Clause tree with two-word preposition, where both words are prepositions: αnx0PPnx1

Other available trees: wh-moved subject, wh-moved object of PP, relative clause on subject with overt and covert extracted *wh*-NP's, relative clause on object of PP with and without comp/with PP pied-piping, imperative, wh-moved object of PP, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, PRO subject, NP gerund with PRO subject.

6.37 PP Small Clause, with Prep and Noun anchors: Tnx0PNaPnx1

Description: This family is selected by multi-word prepositions that can occur in small clause constructions. In particular, this family is selected by three-word prepositions. The first and third words are always prepositions, and the middle word is a noun. The noun is marked for null adjunction since it cannot be modified by noun modifiers. All three components of the multi-word preposition are anchors. For more information on small clause constructions, see section 9.3. 9 multi-word preposition select this tree family.

Examples: *in back of*, *in line with*, *on top of*
The red plaid box should be in back of the plain black box .
The evidence is in line with my newly concocted theory .
She is on top of the world .
**She is on direct top of the world .*

Declarative tree: See Figure 6.37.

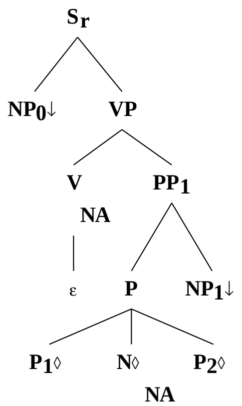


Figure 6.37: Declarative PP Small Clause tree with three-word preposition, where the middle noun is marked for null adjunction: αnx0PNaPnx1

Other available trees: wh-moved subject, wh-moved object of PP, relative clause on subject with overt and covert extracted *wh*-NP's, relative clause on object of PP with overt and covert extracted *wh*-NP's/with PP pied-piping, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping, imperative, NP Gerund, PRO subject, NP gerund with PRO subject.

6.38 PP Small Clause with Sentential Subject, and Adv and Prep anchors: Ts0ARBPnx1

Description: This tree family is selected by multi-word prepositions that take sentential subjects. In particular, this family is selected by two-word prepositions, where the first word is an adverb, the second word a preposition. Both components of the multi-word preposition are anchors. The sentential subject can be indicative or infinitive. Small clauses are explained in much greater detail in section 9.3. 2 prepositions select this tree family.

Examples: *due to*, *contrary to*
that David slept until noon is due to the fact that he never sleeps during the week .
that Michael's joke was funny is contrary to the usual status of his comic attempts .

Declarative tree: See Figure 6.38.

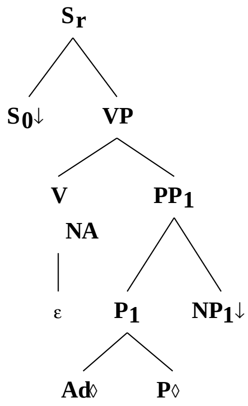


Figure 6.38: Declarative PP Small Clause with Sentential Subject Tree, with two word preposition, where the first word is an adverb, and the second word is a preposition: αs0ARBPnx1

Other available trees: wh-moved subject, relative clause on object of the PP with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.39 PP Small Clause with Sentential Subject, and Adj and Prep anchors: Ts0APnx1

Description: This tree family is selected by multi-word prepositions that take sentential subjects. In particular, this family is selected by two-word prepositions, where the first word is an adjective, the second word a preposition. Both components of the multi-word preposition are anchors. The sentential subject can be indicative or infinitive. Small clauses are explained in much greater detail in section 9.3. 4 prepositions select this tree family.

Examples: *devoid of*, *according to*

that he could walk out on her is devoid of all reason .

that the conversation erupted precisely at that moment was according to my theory .

Declarative tree: See Figure 6.39.

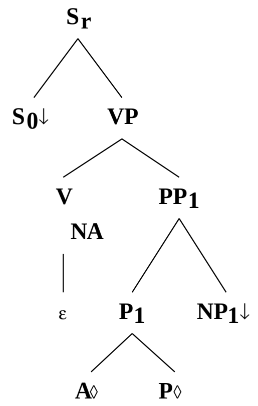


Figure 6.39: Declarative PP Small Clause with Sentential Subject Tree, with two word preposition, where the first word is an adjective, and the second word is a preposition: αs0APnx1

Other available trees: wh-moved subject, relative clause on object of the PP with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.40 PP Small Clause with Sentential Subject, and Noun and Prep anchors: Ts0NPnx1

Description: This tree family is selected by multi-word prepositions that take sentential subjects. In particular, this family is selected by two-word prepositions, where the first word is a noun, the second word a preposition. Both components of the multi-word preposition are anchors. The sentential subject can be indicative or infinitive. Small clauses are explained in much greater detail in section 9.3. 1 preposition selects this tree family.

Examples: *thanks to*

that she is worn out is thanks to a long day in front of the computer terminal .

Declarative tree: See Figure 6.40.

Other available trees: wh-moved subject, relative clause on object of the PP with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

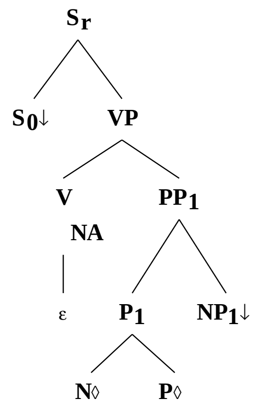


Figure 6.40: Declarative PP Small Clause with Sentential Subject Tree, with two word preposition, where the first word is a noun, and the second word is a preposition: αs0NPnx1

6.41 PP Small Clause with Sentential Subject, and Prep anchors: Ts0PPnx1

Description: This tree family is selected by multi-word prepositions that take sentential subjects. In particular, this family is selected by two-word prepositions, where both words are prepositions. Both components of the multi-word preposition are anchors. The sentential subject can be indicative or infinitive. Small clauses are explained in much greater detail in section 9.3. 3 prepositions select this tree family.

Examples: *outside of*
that Mary did not complete the task on time is outside of the scope of this discussion .

Declarative tree: See Figure 6.41.

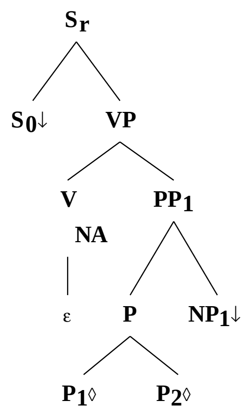


Figure 6.41: Declarative PP Small Clause with Sentential Subject Tree, with two word preposition, where both words are prepositions: αs0PPnx1

Other available trees: wh-moved subject, relative clause on object of the PP with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

6.42 PP Small Clause with Sentential Subject, and Prep and Noun anchors: Ts0PNaPnx1

Description: This tree family is selected by multi-word prepositions that take sentential subjects. In particular, this family is selected by three-word prepositions. The first and third words are always prepositions, and the middle word is a noun. The noun is marked for null adjunction since it cannot be modified by noun modifiers. All three components of the multi-word preposition are anchors. Small clauses are explained in much greater detail in section 9.3. 4 prepositions select this tree family.

Examples: *on account of*, *in support of*
that Joe had to leave the beach was on account of the hurricane .
that Maria could not come is in support of my theory about her .
**that Maria could not come is in direct/strict/desparate support of my theory about her .*

Declarative tree: See Figure 6.42.

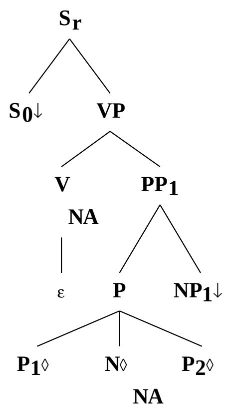


Figure 6.42: Declarative PP Small Clause with Sentential Subject Tree, with three word preposition, where the middle noun is marked for null adjunction: αs0PNaPnx1

Other available trees: wh-moved subject, relative clause on object of the PP with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with covert extracted *wh*-NP/with PP pied-piping.

certain.

6.43 Sentential Subject with Small Clause Complement: Ts0Vs1

Description: This tree family is selected by verbs that take a sentential subject and a small clause complement. This tree family is selected by *make*.

Examples: *make that Max drives a Jaguar makes Bill jealous .*

Declarative tree: See Figure 6.43.

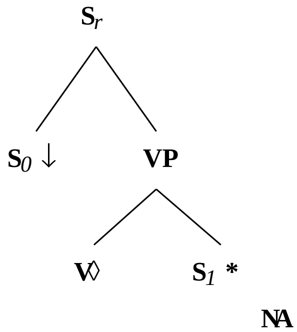


Figure 6.43: Sentential Subject with Small Clause Complement: αs0Vs1

Other available trees: wh-moved subject.

6.44 Locative Small Clause with Ad anchor: Tnx0nx1ARB

Description: These trees are not anchored by verbs, but by adverbs that are part of locative adverbial phrases. Locatives are explained in much greater detail in the section on the locative modifier trees (see section 21.7). The only remarkable aspect of this tree family is the wh-moved locative tree, αW1nx0nx1ARB, shown in Figure 6.45. This is the only tree family with this type of transformation, in which the entire adverbial phrase is wh-moved but not all elements are replaced by wh items (as in *how many city blocks away is the record store?*). Locatives that consist of just the locative adverb or the locative adverb and a degree adverb (see Section 21.7 for details) are treated as exhaustive PPs and therefore select that tree family (Section 6.29) when used predicatively. For an extensive description of small clauses, see Section 9.3. 26 adverbs select this tree family.

Examples: *ahead*, *offshore*, *behind*
the crash is three blocks ahead
the naval battle was many kilometers offshore
how many blocks behind was Max?

Declarative tree: See Figure 6.44.

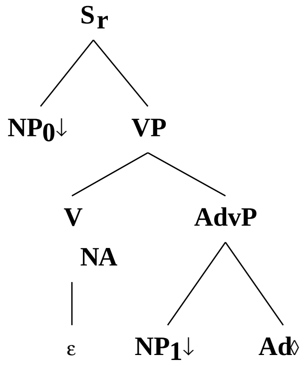


Figure 6.44: Declarative Locative Adverbial Small Clause Tree: αnx0nx1ARB

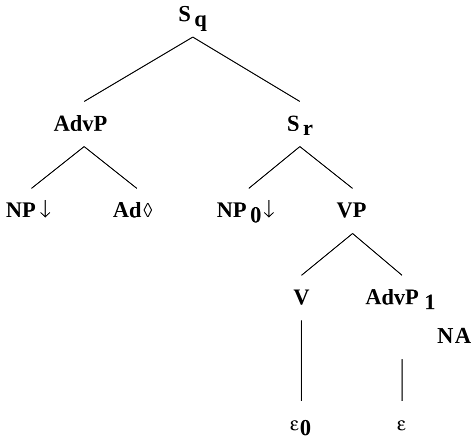


Figure 6.45: Wh-moved Locative Small Clause Tree: αW1nx0nx1ARB

Other available trees: wh-moved subject, relative clause on subject with overt and covert extracted *wh*-NP's, wh-moved locative, imperative, NP gerund, PRO subject, NP gerund with PRO subject.

6.45 Exceptional Case Marking: TXnx0Vs1

Description: This tree family is selected by verbs that are classified as exceptional case marking, meaning that the verb asssigns accusative case to the subject of the sentential complement. This is in contrast to verbs in the Tnx0Vnx1s2 family (section 6.7), which assign accusative case to a NP which is not part of the sentential complement. ECM verbs take sentential complements which are either an infinitive or a "bare" infinitive. As with the Tnx0Vs1 family (section 6.13), the declarative and other trees in the Xnx0Vs1 family are auxiliary trees, as opposed to the more common initial trees. These auxiliary trees adjoin onto an S node in an existing tree of the type specified by the sentential complement. This is the mechanism by which TAGs are able to maintain long-distance dependencies (see Chapter 15), even over multiple embeddings (e.g. *Who did Bill expect to eat beans?*)

or *who did Bill expect Mary to like?* See section 8.6.4 for details on this family. 21 verbs select this tree family.

Examples: *expect*, *see*

Van expects Bob to talk . Bob sees the harmonica fall .

Declarative tree: See Figure 6.46.

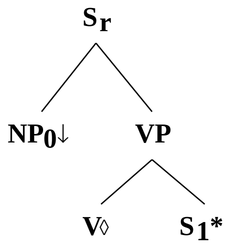


Figure 6.46: ECM Tree: βXnx0Vs1

Other available trees: wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP's, adjunct (gap-less) relative clause with overt and covert extracted *wh*-NP's/with PP pied-piping, imperative, NP gerund, PRO subject.

6.46 Idiom with V, D, and N anchors: Tnx0VDN1

Description: This tree family is selected by idiomatic phrases in which the verb, determiner, and NP are all frozen (as in *He kicked the bucket.*). Only a limited number of transformations are allowed, as compared to the normal transitive tree family (see section 6.3). Other idioms that have the same structure as *kick the bucket*, and that are limited to the same transformations would select this tree, while different tree families are used to handle other idioms. Note that *John kicked the bucket* is actually ambiguous, and would result in two parses - an idiomatic one (meaning that John died), and a compositional transitive one (meaning that there is an physical bucket that John hit with his foot). 21 idioms select this family.

Examples: *kick the bucket*, *bury the hatchet*, *break the ice*

Nixon kicked the bucket. The opponents finally buried the hatchet. The group activity really broke the ice.

Declarative tree: See Figure 6.47.

Other available trees: subject relative clause with and without comp, declarative, wh-moved subject, imperative, NP gerund, adjunct gapless relative with covert extracted *wh*-NP/with PP pied-piping, passive, w/wo by-phrase, wh-moved object of by-phrase, wh-moved by-phrase, relative (with overt and covert extracted *wh*-NP's) on subject of passive, PP relative, PRO subject, NP gerund with PRO subject.

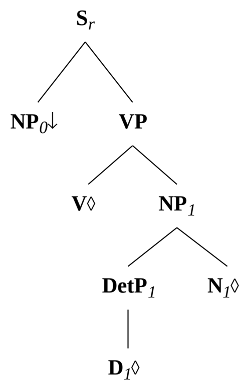


Figure 6.47: Declarative Transitive Idiom Tree: αnx0VDN1

6.47 Idiom with V, D, A, and N anchors: Tnx0VDAN1

Description: This tree family is selected by transitive idioms that are anchored by a verb, determiner, adjective, and noun. 2 idioms select this family.

Examples: *have a green thumb*, *sing a different tune*
Martha might have a green thumb, but it's uncertain after the death of all the plants.
After his conversion John sang a different tune.

Declarative tree: See Figure 6.48.

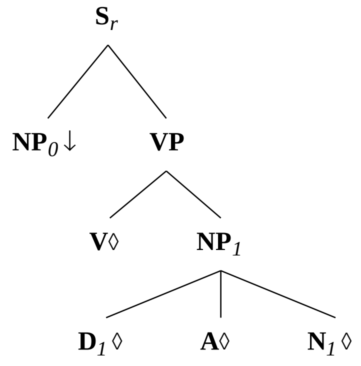


Figure 6.48: Declarative Idiom with V, D, A, and N Anchors Tree: αnx0VDAN1

Other available trees: Subject relative clause with overt and covert extracted *wh*-NP's, adjunct relative clause with covert extracted *wh*-NP/with PP pied-piping, wh-moved subject, imperative, NP gerund, passive without *by* phrase, passive with *by* phrase, passive with wh-moved object of *by* phrase, passive with wh-moved *by phrase*, passive with relative on object of *by* phrase with overt and covert extracted *wh*-NP's, PRO subject, NP gerund with PRO subject.

6.48 Idiom with V and N anchors: Tnx0VN1

Description: This tree family is selected by transitive idioms that are anchored by a verb and noun. 8 idioms select this family.

Examples: *draw blood*, *cry wolf*
Graham's retort drew blood.
The neglected boy cried wolf.

Declarative tree: See Figure 6.49.

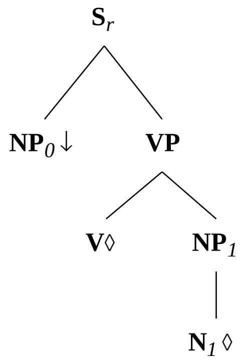


Figure 6.49: Declarative Idiom with V and N Anchors Tree: αnx0VN1

Other available trees: Subject relative clause with overt and covert extracted *wh*-NP's, adjunct relative clause with covert extracted *wh*-NP/with PP pied-piping, wh-moved subject, imperative, NP gerund, passive without *by* phrase, passive with *by* phrase, passive with wh-moved object of *by* phrase, passive with wh-moved *by phrase*, passive with relative on object of *by* phrase with overt and covert extracted *wh*-NP's, PRO subject, NP gerund with PRO subject.

6.49 Idiom with V, A, and N anchors: Tnx0VAN1

Description: This tree family is selected by transitive idioms that are anchored by a verb, adjective, and noun. 2 idioms select this family.

Examples: *break new ground*, *cry bloody murder*
The avant-garde film breaks new ground.
The investors cried bloody murder after the suspicious takeover.

Declarative tree: See Figure 6.50.

Other available trees: Subject relative clause with overt and covert extracted *wh*-NP's, adjunct relative clause with covert extracted *wh*-NP/with PP pied-piping, wh-moved subject, imperative, NP gerund, passive without *by* phrase, passive with *by* phrase, passive

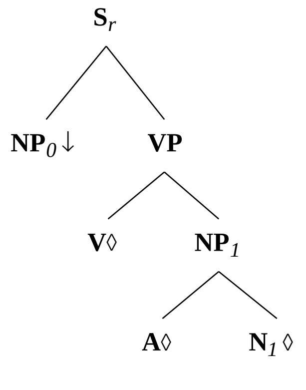


Figure 6.50: Declarative Idiom with V, A, and N Anchors Tree: αnx0VAN1

with wh-moved object of *by* phrase, passive with wh-moved *by phrase*, passive with relative on object of *by* phrase with overt and covert extracted *wh*-NP's, PRO subject, NP gerund with PRO subject.

6.50 Idiom with V, D, A, N, and Prep anchors: Tnx0VDAN1Pnx2

Description: This tree family is selected by transitive idioms that are anchored by a verb, determiner, adjective, noun, and preposition. 2 idioms select this family.

Examples: *make a big deal about*, *make a great show of*
John made a big deal about a miniscule dent in his car.
The company made a big show of paying generous dividends.

Declarative tree: See Figure 6.51.

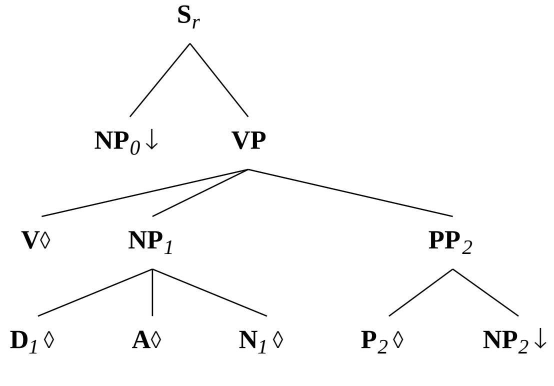


Figure 6.51: Declarative Idiom with V, D, A, N, and Prep Anchors Tree: αnx0VDAN1Pnx2

Other available trees: Subject relative clause with overt and covert extracted *wh*-NP's, adjunct relative clause with covert extracted *wh*-NP/with PP pied-piping, wh-moved subject, imperative, NP gerund, passive without *by* phrase, passive with *by* phrase, passive

with wh-moved object of *by* phrase, passive with wh-moved *by phrase*, outer passive without *by* phrase, outer passive with *by* phrase, outer passive with wh-moved *by* phrase, outer passive with wh-moved object of *by* phrase, outer passive without *by* phrase with relative on the subject with overt and covert extracted *wh*-NP's, outer passive with *by* phrase with relative on subject with overt and covert extracted *wh*-NP's, PRO subject, passive with PRO subject with and without the *by* phrase, NP gerund with PRO subject.

6.51 Idiom with V, A, N, and Prep anchors: Tnx0VAN1Pnx2

Description: This tree family is selected by transitive idioms that are anchored by a verb, adjective, noun, and preposition. 1 idiom selects this family.

Examples: *make short work of*
John made short work of the glazed ham.

Declarative tree: See Figure 6.52.

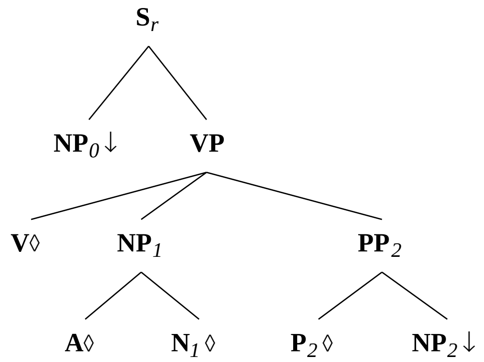


Figure 6.52: Declarative Idiom with V, A, N, and Prep Anchors Tree: αnx0VAN1Pnx2

Other available trees: Subject relative clause with overt and covert extracted *wh*-NP's, adjunct relative clause with covert extracted *wh*-NP/with PP pied-piping, wh-moved subject, imperative, NP gerund, passive without *by* phrase, passive with *by* phrase, passive with wh-moved object of *by* phrase, passive with wh-moved *by phrase*, outer passive without *by* phrase, outer passive with *by* phrase, outer passive with wh-moved *by* phrase, outer passive with wh-moved object of *by* phrase, outer passive without *by* phrase with relative on the subject with overt and covert extracted *wh*-NP's, outer passive with *by* phrase with relative on subject with overt and covert extracted *wh*-NP's, PRO subject, passive with PRO subject with and without the *by* phrase, NP gerund with PRO subject.

6.52 Idiom with V, N, and Prep anchors: Tnx0VN1Pnx2

Description: This tree family is selected by transitive idioms that are anchored by a verb, noun, and preposition. 6 idioms select this family.

Examples: *look daggers at*, *keep track of*
Maria looked daggers at her ex-husband across the courtroom.
The company kept track of its inventory.

Declarative tree: See Figure 6.53.

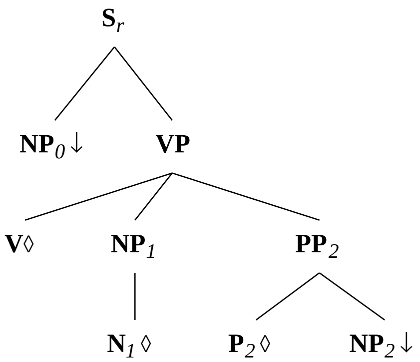


Figure 6.53: Declarative Idiom with V, N, and Prep Anchors Tree: αnx0VN1Pnx2

Other available trees: Subject relative clause with overt and covert extracted *wh*-NP's, adjunct relative clause with covert extracted *wh*-NP/with PP pied-piping, wh-moved subject, imperative, NP gerund, passive without *by* phrase, passive with *by* phrase, passive with wh-moved object of *by* phrase, passive with wh-moved *by phrase*, outer passive without *by* phrase, outer passive with *by* phrase, outer passive with wh-moved *by* phrase, outer passive with wh-moved object of *by* phrase, outer passive without *by* phrase with relative on the subject with overt and covert extracted *wh*-NP's, outer passive with *by* phrase with relative on subject with overt and covert extracted *wh*-NP's, PRO subject, passive with PRO subject with and without the *by* phrase, NP gerund with PRO subject.

6.53 Idiom with V, D, N, and Prep anchors: Tnx0VDN1Pnx2

Description: This tree family is selected by transitive idioms that are anchored by a verb, determiner, noun, and preposition. 9 idioms select this family.

Examples: *make a mess of*, *keep the lid on*
John made a mess of his new suit.
The tabloid didn't keep a lid on the imminent celebrity nuptials.

Declarative tree: See Figure 6.54.

Other available trees: Subject relative clause with overt and covert extracted *wh*-NP's, adjunct relative clause with covert extracted *wh*-NP/with PP pied-piping, wh-moved subject, imperative, NP gerund, passive without *by* phrase, passive with *by* phrase, passive

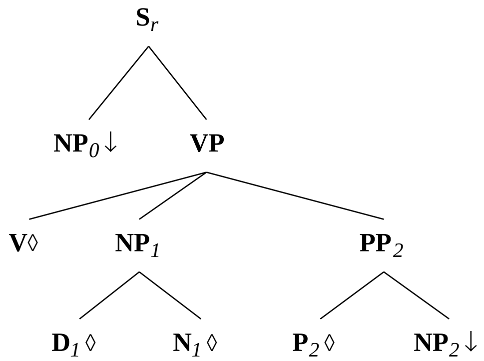


Figure 6.54: Declarative Idiom with V, D, N, and Prep Anchors Tree: αnx0VDN1Pnx2

with wh-moved object of *by* phrase, passive with wh-moved *by phrase*, outer passive without *by* phrase, outer passive with *by* phrase, outer passive with wh-moved *by* phrase, outer passive with wh-moved object of *by* phrase, outer passive without *by* phrase with relative on the subject with overt and covert extracted *wh*-NP's, outer passive with *by* phrase with relative on subject with overt and covert extracted *wh*-NP's, PRO subject, passive with PRO subject with and without the *by* phrase, NP gerund with PRO subject.

6.54 Resultatives with V (transitive and intransitive), A anchors: TRnx0Vnx1A2

Description: This tree family by transitive and intransitive verbs that form a complex predicate with adjectives. 55 multi-word anchors select this family.

Example: *hit unconscious*, *hammer flat*
Bill hit the boy unconscious. Miranda hammered the metal flat.

Declarative tree: See Figure 6.55.

Other available trees: passive with and without *by* phrase, wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP, wh-moved object, object relative clause with covert extracted *wh*-NP, passive with and without *by* phrase on wh-moved object, passive with and without *by* phrase on object relative clause with covert extracted *wh*-NP, imperative, wh-moved subject on passive with and without *by* phrase, wh-question on object of *by* phrase in subject extracted relative clauses with overt and covert extracted *wh*-NP/with PP pied-piping, multi anchored participial modifiers, relative clause on PP adjunct with overt and covert extracted *wh*-NP's, relative clause on PP adjunct in passives with and without *by* phrase with overt and covert extracted *wh*-NP's, NP gerund with and without *by* phrase, wh-moved adjective complement, passive on wh-moved adjective complement with and without *by* phrase.

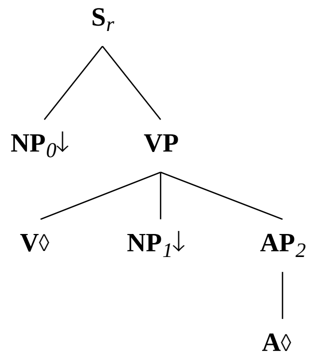


Figure 6.55: Resultative multi-anchored by transitive and intransitive verbs and adjectives, αRnx0Vnx1A2

6.55 Resultatives with V (transitive and intransitive), P anchors: TRnx0Vnx1Pnx2

Description: This tree family by transitive and intransitive verbs that form a complex predicate with prepositions. 11 multi-word anchors select this family.

Example: *grind into*, *beat to*
Bill ground the wheat into the dough. Miranda beat the box to a pulp.

Declarative tree: See Figure 6.56.

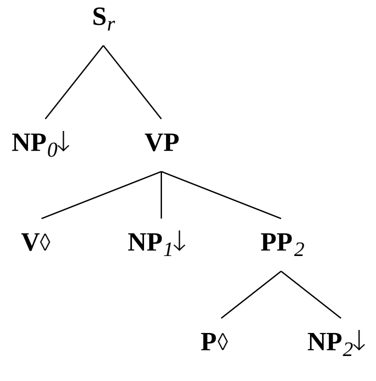


Figure 6.56: Resultative multi-anchored by transitive and intransitive verbs and prepositions, αRnx0Vnx1Pnx2

Other available trees: passive with and without *by* phrase, wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP, wh-moved object, object relative clause with covert extracted *wh*-NP, passive with and without *by* phrase on wh-moved object, passive with and without *by* phrase on object relative clause with covert extracted *wh*-NP, imperative, wh-moved subject on passive with and without *by* phrase, wh-question on object of *by* phrase in subject extracted relative clauses with overt and covert extracted *wh*-NP/with PP pied-piping, relative clause on PP adjunct with overt and covert extracted *wh*-NP's, relative clause on PP adjunct in passives with and without *by* phrase with overt and covert extracted *wh*-NP's, NP gerund with and without *by* phrase, wh-moved adjective complement, passive on wh-moved adjective complement with and without *by* phrase.

6.56 Resultatives with V (ergative), A anchors: TREnx1VA2

Description: This tree family by ergative verbs that form a complex predicate with adjectives. 14 multi-word anchors select this family.

Example: *freeze solid*, *break open*
The milk froze solid. The chest broke open.

Declarative tree: See Figure 6.57.

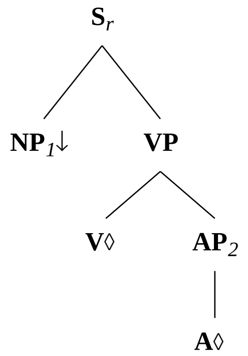


Figure 6.57: Resultative multi-anchored by transitive and intransitive verbs and adjectives, αREnx1VA2

Other available trees: wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP's, wh-moved object, imperative, subject extracted relative clauses with PP pied-piping, relative clause on PP adjunct with overt and covert extracted *wh*-NP's, NP gerund, wh-moved adjective complement.

6.57 Resultatives with V (ergative), P anchors: TREnx1VPnx2

Description: This tree family by ergative verbs that form a complex predicate with prepositions. 6 multi-word anchors select this family.

Example: *bend into*, *melt into*
The iron rod bent into a curve. The icecream melted into milk.

Declarative tree: See Figure 6.58.

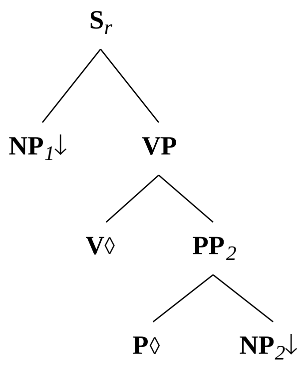


Figure 6.58: Resultative multi-anchored by transitive and intransitive verbs and adjectives, αREnx1VPnx2

Other available trees: wh-moved subject, subject relative clause with overt and covert extracted *wh*-NP's, wh-moved object, object relative clause with overt and covert extracted *wh*-NP's, imperative, relative clauses with PP pied-piping, relative clause on PP adjunct with overt and covert extracted *wh*-NP's, NP gerund, wh-moved adjective complement.

Chapter 7

Ergatives

Verbs in English that we will call *ergative* [1] display the kind of alternation shown in the sentences in 9 and 10 below.

(9) The sun melted the ice .

(10) The ice melted .

The object of the transitive sentence in 9 corresponds to the subject of the intransitive sentence in 10.

7.1 Various Approaches

The literature discussing such pairs as 9 and 10 is based largely on syntactic models that involve movement, particularly GB. Within that framework two basic approaches are discussed:

- **Derived Intransitive**
  The intransitive member of the ergative pair is derived through processes of movement and deletion from an underlying transitive structure [Burzio, 1986; Hale and Keyser, 1986; Hale and Keyser, 1987].

- **Pure Intransitive**
  The intransitive member is intransitive at all levels of the syntax and the lexicon and is not related to the transitive member syntactically or lexically [Napoli, 1988].

7.2 Xtag Analysis

Although XTAG does not have derivational movement, the relationships between the two arguments can be translated into the FB-LTAG framework. In the XTAG grammar the difference between these two approaches is not a matter of movement but rather a question of tree family

[1] The terminology is from [Burzio, 1986]. See also [Perlmutter, 1978] and [Rosen, 1981] for discussion within the Relational Grammar framework.

selection. The relation between sentences represented in terms of movement in other frameworks is represented in XTAG by membership in the same tree family or selection of multiple families which preserve the argument relations across families. Wh-questions and their indicative counterparts are one example of the former, whereas ergatives and ditransitives with PP shift exemplify the latter. Adopting the Pure Intransitive approach suggested by [Napoli, 1988] would mean placing the intransitive ergatives in a tree family with other intransitive verbs and separate from the transitive variants of the same verbs. This would result in a grammar that represented intransitive ergatives as more closely related to other intransitives than to their transitive counterparts. The only hint of the relation between the intransitive ergatives and the transitive ergatives would be that ergative verbs would select both tree families. While this is a workable solution, it is an unattractive one for the English XTAG grammar because semantic coherence is lost. In particular, constancy in thematic role is represented by constancy in node names across sentence types within a tree family. For example, if the object of a declarative tree is NP_1 the subject of the passive tree(s) in that family will also be NP_1.

The analysis that has been implemented in the English XTAG grammar is an adaptation of the Derived Intransitive approach. The ergatives select the two families Tnx0Vnx1 and TEnx1V. TEnx1V is quite similar to the regular intransitive family Tnx0V, except that it encodes a different type of semantic function—the syntactic subject of TEnx1V is given the index of '1' to represent that it is the logical object of the verb. The two families Tnx0Vnx1 and TEnx1V create the two possibilities needed to account for the data.

- **intransitive ergative/transitive alternation.** These verbs have transitive and intransitive variants as shown in sentences 11 and 12.

  (11) The sun melted the ice .

  (12) The ice melted .

  In the English XTAG grammar, verbs with this behavior select family $\{$Tnx0Vnx1 $\cup$ TEnx1V$\}$. The first family in the union handles sentences such as 11 and the second covers 12.

- **transitive only.** Verbs of this type do not allow the intransitive ergative variants, as can be seen in the pattern shown in sentences 13 and 14.

  (13) Elmo borrowed a book .

  (14) $*$A book borrowed .

  Such verbs select only the family Tnx0Vnx1, and thus do not have the ergative trees in TEnx1V available to them.

The declarative ergative tree is shown in Figure 7.1. Note that the index of the subject NP indicates that it originated as the object of the verb.

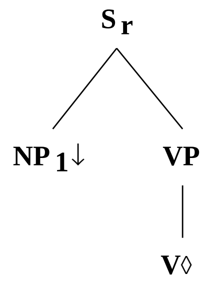


Figure 7.1: Ergative Tree: αEnx1V

Chapter 8

Sentential Subjects and Sentential Complements

In the XTAG grammar, arguments of a lexical item, including subjects, appear in the elementary tree anchored by that lexical item. A sentential argument appears as an S node in the appropriate position within an elementary tree anchored by the lexical item that selects it. This is the case for sentential complements of verbs, prepositions and nouns and for sentential subjects. The distribution of complementizers in English is intertwined with the distribution of embedded sentences. A successful analysis of complementizers in English must handle both the cooccurrence restrictions between complementizers and various types of clauses, and the distribution of the clauses themselves, in both subject and complement positions.

8.1 S or VP complements?

Two comparable grammatical formalisms, Generalized Phrase Structure Grammar (GPSG) [Gazdar *et al.*, 1985] and Head-driven Phrase Structure Grammar (HPSG) [Pollard and Sag, 1994], have rather different treatments of sentential complements (S-comps). Both treat embedded sentences as VP's with subjects, generating the correct structures but missing the generalization that S's behave similarly in both matrix and embedded environments, and that VP's behave quite differently. Neither account has PRO subjects of infinitival clauses– they have subjectless VP's instead. GPSG has a complete complementizer system, which appears to cover the same range of data as our analysis. It is not clear what sort of complementizer analysis could be implemented in HPSG.

Following the standard GB approach, the English XTAG grammar does not allow VP complements but treats verb-anchored structures without overt subjects as having PRO subjects. Thus, indicative clauses, infinitives and gerunds all have a uniform treatment as embedded clauses using the same trees under this approach. Furthermore, our analysis is able to preserve the selectional and distributional distinction between S's and VP's, in the spirit of GB theories, but without having to posit 'extra' empty categories, such as empty complementizers.[1] Consider the alternation between *that* and the null COMP,[2] shown in sentences 15 and 16.

[1]We do have PRO and NP traces in the grammar.

[2]Although we will continue to refer to 'null' complementizers, in our analysis this is actually the absence of

(15) He hopes ∅ Muriel wins .

(16) He hopes that Muriel wins .

In GB both *Muriel wins* in 15 and *that Muriel wins* in 16 are CPs even though there is no overt complementizer to head the phrase in 15. Our grammar does not distinguish by category label between the phrases that would be labeled in GB as IP and CP. We label both of these phrases S. The difference between these two levels is the presence or absence of the complementizer (or extracted WH constituent), and is represented in our system as a difference in feature values (here, of the <**comp**> feature), and the presence of the additional structure contributed by the complementizer or extracted constituent. This illustrates an important distinction in XTAG, that between features and node labels. Because we have a sophisticated feature system, we are able to make fine-grained distinctions between nodes with the same label which in another system might have to be realized by using distinguishing node labels.

8.2 Complementizers and Embedded Clauses in English: The Data

Verbs selecting sentential complements place restrictions on their complements, in particular, on the form of the embedded verb phrase.[3] Furthermore, complementizers are constrained to appear with certain types of clauses, again, based primarily on the form of the embedded VP. For example, *hope* selects both indicative and infinitival complements. With an indicative complement, it may only have *that* or null as possible complementizers; with an infinitival complement, it may only have a null complementizer. Verbs that allow wh+ complementizers, such as *ask*, can take *whether* and *if* as complementizers. The possible combinations of complementizers and clause types is summarized in Table 8.1.

As can be seen in Table 8.1, sentential subjects differ from sentential complements in requiring the complementizer *that* for all indicative and subjunctive clauses. In sentential complements, *that* often varies freely with a null complementizer, as illustrated in 17-22.

(17) Christy hopes that Mike wins .

(18) Christy hopes Mike wins .

(19) Dania thinks that Newt is a liar .

(20) Dania thinks Newt is a liar .

(21) That Helms won so easily annoyed me .

(22) ∗Helms won so easily annoyed me .

a complementizer.

[3]Other considerations, such as the relationship between the tense/aspect of the matrix clause and the tense/aspect of a complement clause are also important but are not currently addressed in the current English XTAG grammar.

Complementizer:		that	whether	if	for	null
Clause type						
indicative	subject	Yes	Yes	No	No	No
	complement	Yes	Yes	Yes	No	Yes
infinitive	subject	No	Yes	No	Yes	Yes
	complement	No	Yes	No	Yes	Yes
subjunctive	subject	Yes	No	No	No	No
	complement	Yes	No	No	No	Yes
gerundive[4]	complement	No	No	No	No	Yes
base	complement	No	No	No	No	Yes
small clause	complement	No	No	No	No	Yes

Table 8.1: Summary of Complementizer and Clause Combinations

Another fact which must be accounted for in the analysis is that in infinitival clauses, the complementizer *for* must appear with an overt subject NP, whereas a complementizer-less infinitival clause never has an overt subject, as shown in 23-26. (See section 8.5 for more discussion of the case assignment issues relating to this construction.)

(23) To lose would be awful .

(24) For Penn to lose would be awful .

(25) ∗For to lose would be awful .

(26) ∗Penn to lose would be awful .

In addition, some verbs select **<wh>=+** complements (either questions or clauses with *whether* or *if*) [Grimshaw, 1990]:

(27) Jesse wondered who left .

(28) Jesse wondered if Barry left .

(29) Jesse wondered whether to leave .

(30) Jesse wondered whether Barry left .

(31) ∗Jesse thought who left .

(32) ∗Jesse thought if Barry left .

(33) ∗Jesse thought whether to leave .

(34) ∗Jesse thought whether Barry left .

[4]Most gerundive phrases are treated as NP's. In fact, all gerundive subjects are treated as NP's, and the only gerundive complements which receive a sentential parse are those for which there is no corresponding NP parse. This was done to reduce duplication of parses. See Chapter 19 for further discussion of gerunds.

8.3 Features Required

As we have seen above, clauses may be <**wh**>=+ or <**wh**>=–, may have one of several complementizers or no complementizer, and can be of various clause types. The XTAG analysis uses three features to capture these possibilities: <**comp**> for the variation in complementizers, <**wh**> for the question vs. non-question alternation and <**mode**>[5] for clause types. In addition to these three features, the <**assign-comp**> feature represents complementizer requirements of the embedded verb. More detailed discussion of the <**assign-comp**> feature appears below in the discussions of sentential subjects and of infinitives. The four features and their possible values are shown in Table 8.2.

Feature	Values
<**comp**>	that, if, whether, for, rel, nil
<**mode**>	ind, inf, subjnt, ger, base, ppart, nom/prep
<**assign-comp**>	that, if, whether, for, rel, ind_nil, inf_nil
<**wh**>	+,–

Table 8.2: Summary of Relevant Features

8.4 Distribution of Complementizers

Like other non-arguments, complementizers anchor an auxiliary tree (shown in Figure 8.1) and adjoin to elementary clausal trees. The auxiliary tree for complementizers is the only alternative to having a complementizer position 'built into' every sentential tree. The latter choice would mean having an empty complementizer substitute into every matrix sentence and a complementizerless embedded sentence to fill the substitution node. Our choice follows the XTAG principle that initial trees consist only of the arguments of the anchor[6] – the S tree does not contain a slot for a complementizer, and the βCOMP tree has only one argument, an S with particular features determined by the complementizer. Complementizers select the type of clause to which they adjoin through constraints on the <**mode**> feature of the S foot node in the tree shown in Figure 8.1. These features also pass up to the root node, so that they are 'visible' to the tree where the embedded sentence adjoins/substitutes.

The grammar handles the following complementizers: *that*, *whether*, *if*, *for*, and no complementizer, and the clause types: indicative, infinitival, gerundive, past participial, subjunctive and small clause (**nom/prep**). The <**comp**> feature in a clausal tree reflects the value of the complementizer if one has adjoined to the clause.

The <**comp**> and <**wh**> features receive their root node values from the particular complementizer which anchors the tree. The βCOMPs tree adjoins to an S node with the feature <**comp**>=**nil**; this feature indicates that the tree does not already **have** a complementizer adjoined to it.[7] We ensure that there are no stacked complementizers by requiring the foot node of βCOMPs to have <**comp**>=**nil**.

[5]<**mode**> actually conflates several types of information, in particular verb form and mood.

[6]See section 4.3 for a discussion of the difference between complements and adjuncts in the XTAG grammar.

[7]Because root S's cannot have complementizers, the parser checks that the root S has <**comp**>=**nil** at the end of the derivation, when the S is also checked for a tensed verb.

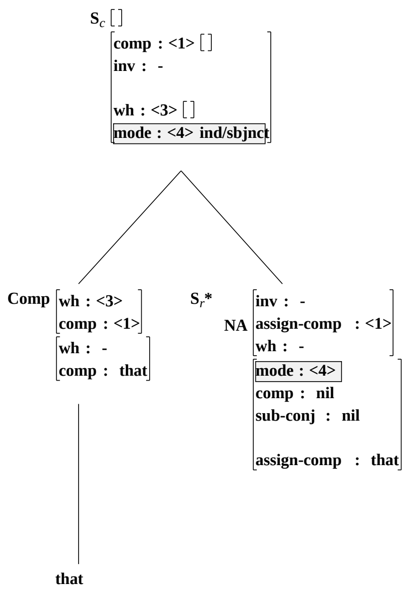


Figure 8.1: Tree βCOMPs, anchored by *that*

8.5 Complementizer *for* and Case Assignment of the Subject

The <**assign-comp**> feature is used to represent the requirements of particular types of clauses for particular complementizers. So while the <**comp**> feature represents constraints originating from the VP dominating the clause, the <**assign-comp**> feature represents constraints originating from the highest VP in the clause. <**assign-comp**> is used to control the appearance of subjects in infinitival clauses (see discussion of ECM constructions in 8.6.4), to block bare indicative sentential subjects (bare infinitival subjects are allowed), and to block 'that-trace' violations.

Examples 36, 37 and 38 show that an accusative case subject is obligatory in an infinitive clause if the complementizer *for* is present. The infinitive clause in 35 is analyzed in the English XTAG grammar as having a PRO subject.

(35) Christy wants to pass the exam .

(36) Mike wants for her to pass the exam .

(37) ∗Mike wants for she to pass the exam .

(38) ∗Christy wants for to pass the exam .

It is commonly accepted that *for* behaves as a case-assigning complementizer in this construction. It can assign accusative case to the embedded subject since the infinitival verb can not assign (nominative) case to this position. In 37 there is a feature clash between the nominative case subject *she* and the accusative case assigning complementizer, thus accounting for its ungrammaticality. Similarly, the sentence in 38 is ruled out because PRO has a feature <**case**>=**none** which is coindexed with with the <**assign-case**> feature on S. This feature clashes with the <**assign-case**>=**acc** feature in the *for* auxiliary tree.

8.6 Sentential Complements of Verbs

Tree families: Tnx0Vs1, Tnx0Vnx1s2, TItVnx1s2, TItVpnx1s2, TItVad1s2.

8.6.1 Long Distance Extraction

Verbs that select sentential complements restrict the <**mode**> and <**comp**> values for those complements. Since with very few exceptions long distance extraction is possible from sentential complements,[8] the S complement nodes are adjunction nodes. Figure 8.2 shows the declarative tree for a sentential complement-taking bridge verb, anchored by *think*.

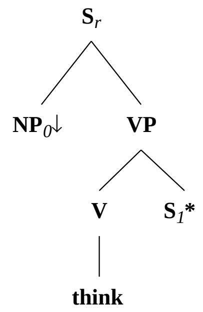


Figure 8.2: Sentential complement tree: βnx0Vs1

The need for an adjunction node rather than a substitution node at S_1 may not be obvious until one considers the derivation of sentences with long distance extractions. For example, the declarative in 39 is derived by adjoining the auxiliary tree in Figure 8.3(b) to the S_r node of the tree in Figure 8.3(a). Since there are no bottom features on S_1 in the auxiliary tree, the same final result could have been achieved with a substitution node at S_1.

(39) The emu thinks that the aardvark smells terrible .

However, adjunction is crucial in deriving sentences with long distance extraction, as in sentences 40 and 41.

(40) Who does the emu think smells terrible ?

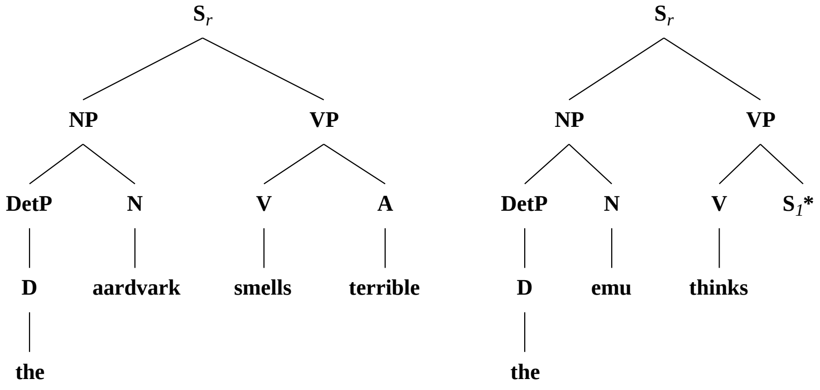
(41) Who did the elephant think the panda heard the emu say smells terrible ?

The example in (40) is derived from the trees for *who smells terrible?* shown in Figure 8.4 and *the emu thinks* S shown in Figure 8.3(b), by adjoining the latter at the S_r node of the former.[9] This process is recursive, allowing sentences like (41). Such a representation has been shown by [Kroch and Joshi, 1985] to be well-suited for describing unbounded dependencies.

In English, a complementizer may not appear on a complement with an extracted subject (the '*that*-trace' configuration). This phenomenon is illustrated in 42-44:

[8] Although see the discussion of non-bridge verbs in the next section.

[9] See Chapter 22 for a discussion of do-support.



(a) (b)

Figure 8.3: Trees for *The emu thinks that the aardvark smells terrible* .

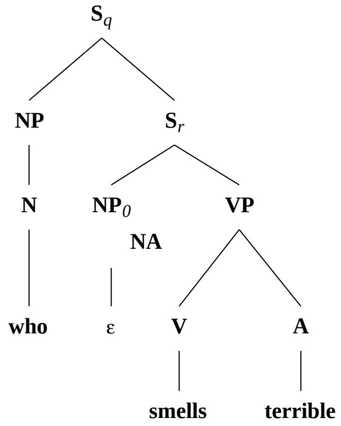


Figure 8.4: Tree for *Who smells terrible?*

(42) Which animal did the giraffe say that he likes ?

(43) ∗Which animal did the giraffe say that likes him ?

(44) Which animal did the giraffe say likes him ?

These sentences are derived in XTAG by adjoining the tree for *did the giraffe say* S at the S_r node of the tree for either *which animal likes him* (to yield sentence 44) or *which animal he likes* (to yield sentence 42). That-trace violations are blocked by the presence of the feature **<assign-comp>=inf_nil/ind_nil/ecm** feature on the bottom of the S_r node of trees with extracted subjects (W0), i.e. those used in sentences such as 43 and 44. If a complementizer tree, βCOMPs, adjoins to a subject extraction tree at S_r, its **<assign-comp> = that/whether/for/if** feature will clash and the derivation will fail. If there is no complementizer, there is no feature clash, and this will permit the derivation of sentences like 44, or of ECM constructions, in which case the ECM verb will have **<assign-comp>=ecm** (see section 8.6.4 for more discussion of the ECM case). Complementizers may adjoin normally to object extraction trees such as those used in sentence 42, and so object extraction trees have no value for the <**assign-comp**> feature.

8.6.2 Bridge vs. Non-bridge Verbs

There is a class of *non-bridge verbs* (such as the manner-of-speaking verbs, factives, and negative verbs) which do not allow extraction from their sentential complement. In contrast to a bridge verb *say* which allows extraction from its complement, as in 45, the non-bridge verb *whisper* does not allow this extraction, as shown in 46.

(45) Who did the elephant say that the emu saw ?

(46) $*$ Who did the elephant whisper that the emu saw ?

Similarly, adjunct extraction with a matrix bridge verb yields an ambiguity. In 47, the adjunct wh-expression *when* can be interpreted as modifying either the matrix predicate or the embedded predicate. In the non-bridge example 48, there is no such ambiguity. The wh-expression can only be construed as modifying the matrix predicate.

(47) When did Laura say she would be back ?

(48) When did Laura whisper she would be back ?

At this time, however, we do not account for the bridge/non-bridge distinction.

8.6.3 Subjacency Violations: Multiple Wh-extraction

In the case of indirect questions, subjacency follows from the principle that a given tree cannot contain more than one wh-element. Extraction out of an indirect question is ruled out because a sentence like 49 would have to be derived from the adjunction of *do you wonder* into who_i who_j e_j *loves* e_i, which is an ill-formed elementary tree.[10]

(49) $*$ Who$_i$ do you wonder who$_j$ e$_j$ loves e$_i$?

8.6.4 Exceptional Case Marking Verbs and Bare Infinitives

Tree family: TXnx0Vs1, Ts0Vs1

Exceptional Case Marking verbs are those which assign accusative case to the subject of the sentential complement. This is in contrast to verbs in the Tnx0Vnx1s2 family (section 6.7), which assign accusative case to an NP which is not part of the sentential complement.

The subject of an ECM infinitive complement is assigned accusative case in a manner analogous to that of a subject in a *for-to* construction, as described in section 8.5. As in the *for-to* case, the ECM verb assigns accusative case into the subject of the lower infinitive,

[10] This does not mean that elementary trees with more than one gap should be ruled out across the grammar. Such trees might be required for dealing with parasitic gaps or gaps in coordinated structures.

and so the infinitive uses the *to* which has no value for **<assign-case>** and has **<assign-comp>=for/ecm**. The ECM verb has **<assign-comp>=ecm** and **<assign-case>=acc** on its foot. The former allows the **<assign-comp>** features of the ECM verb and the *to* tree to unify, and so be used together, and the latter assigns the accusative case to the lower subject.

Figure 8.5 shows the declarative tree for the TXnx0Vs1 family, in this case anchored by *expects*. Figure 8.6 shows a parse for *Van expects Bob to talk*

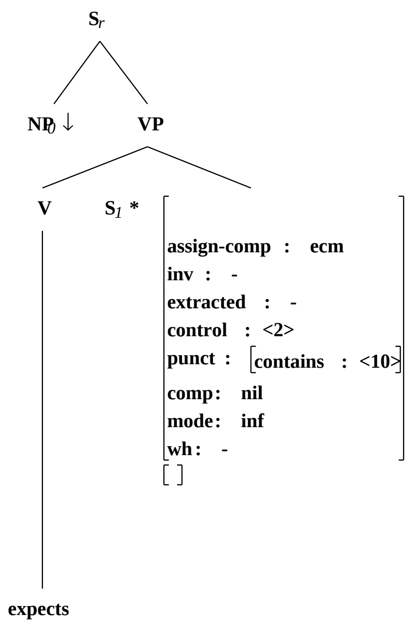


Figure 8.5: ECM tree: βXnx0Vs1

The ECM and *for-to* cases are analogous in how they are used together with the correct infinitival *to* to assign accusative case to the subject of the lower infinitive. However, they are different in that *for* is blocked along with other complementizers in subject extraction contexts, as discussed in section 8.6, as in 50, while subject extraction is compatible with ECM cases, as in 51.

(50) $*$What child did the giraffe ask for to leave ?

(51) Who did Bill expect to eat beans ?

Sentence 50 is ruled out by the **<assign-comp>= inf_nil/ind_nil/ecm** feature on the subject extraction tree for *ask*, since the **<assign-comp>=for** feature from the *for* tree will fail to unify. However, 51 will be allowed since **<assign-comp>=ecm** feature on the *expect* tree will unify with the foot of the ECM verb tree. The use of features allows the ECM and *for-to* constructions to act the same for exceptional case assignment, while also being distinguished for *that*-trace violations.

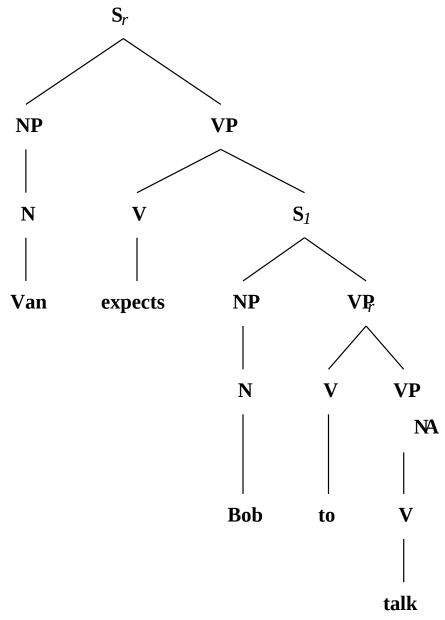


Figure 8.6: Sample ECM parse

8.6.4.1 ECM Passives

Passivized ECM verbs are treated as raising verbs, meaning that they are auxiliary trees recursive on VP. Since the subject of the infinitive is not thematically selected by the ECM verb, it is not part of the ECM verb's tree, and so it cannot be part of the passive tree. Therefore, the passive acts as a raising verb, and so for example, the sentence *John is believed to be happy* would be derived by adjoining *believed* in as a raising verb. For further discussion, see section 9.3.1.

8.6.4.2 Bare Infinitives

Verbs that take bare infinitives, as in 52, are also treated as ECM verbs, the only difference being that their foot feature has **<mode>=base** instead of **<mode>=inf**. Since the complement does not have *to*, there is no question of using the *to* tree for allowing accusative case to be assigned. Instead, verbs with **<mode>=base** allow either accusative or nominative case to be assigned to the subject. The foot of the ECM bare infinitive tree forces the subject to be accusative by its **<assign-case>=acc** value at its foot node which unifies with the **<assign-case>=nom/acc** value of the bare infinitive clause.

(52) Bob sees the harmonica fall .

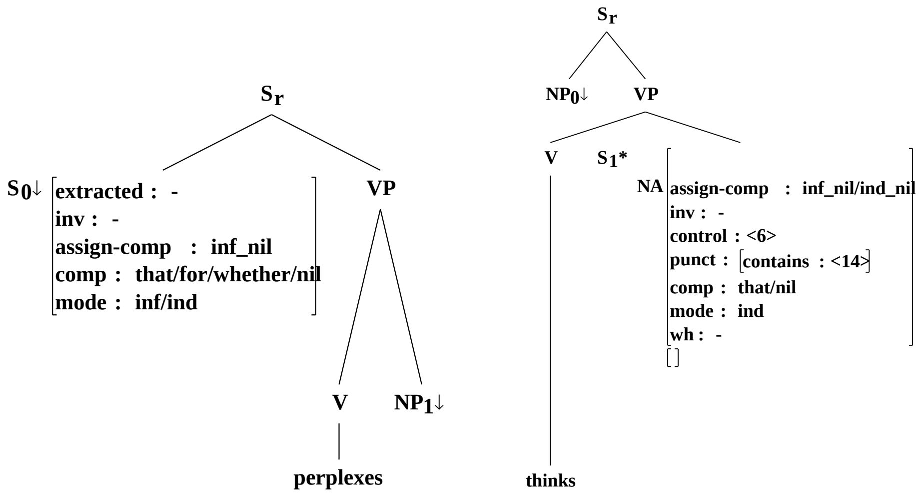
Verbs taking a bare infinitive complement and an NP subject select the TXnx0Vs1 family. Verbs taking a bare infinitive complement and a sentential subject (*make* and *let*) select the Ts0Vs1 family (see section 6.43). These verbs (*make* and *let*) also take a predicative small clause (see section 9.3), and so the foot node has value **<mode>=nom/prep/base**, thus allowing either a bare infinitive or a **nom/prep** complement.

8.7 Sentential Subjects

Tree families: Ts0Vnx1, Ts0Ax1, Ts0N1, Ts0Pnx1, Ts0ARBPnx1, Ts0PPnx1, Ts0PNaPnx1, Ts0V, Ts0Vtonx1, Ts0NPnx1, Ts0APnx1, Ts0Vs1.

Verbs that select sentential subjects anchor trees that have an S node in the subject position rather than an NP node. Since extraction is not possible from sentential subjects, they are implemented as substitution nodes in the English XTAG grammar. Restrictions on sentential subjects, such as the required *that* complementizer for indicatives, are enforced by feature values specified on the S substitution node in the elementary tree.

Sentential subjects behave essentially like sentential complements, with a few exceptions. In general, all verbs which license sentential subjects license the same set of clause types. Thus, unlike sentential complement verbs which select particular complementizers and clause types, the matrix verbs licensing sentential subjects merely license the S argument. Information about the complementizer or embedded verb is located in the tree features, rather than in the features of each verb selecting that tree. Thus, all sentential subject trees have the same <**mode**>, <**comp**> and <**assign-comp**> values shown in Figure 8.7(a).



(a) (b)

Figure 8.7: Comparison of <**assign-comp**> values for sentential subjects: αs0Vnx1 (a) and sentential complements: βnx0Vs1 (b)

The major difference in clause types licensed by S-subjs and S-comps is that indicative S-subjs obligatorily have a complementizer (see examples in section 8.2). The <**assign-comp**> feature is used here to license a null complementizer for infinitival but not indicative clauses. <**assign-comp**> has the same possible values as <**comp**>, with the exception that the **nil** value is 'split' into **ind_nil** and **inf_nil**. This difference in feature values is illustrated in Figure 8.7.

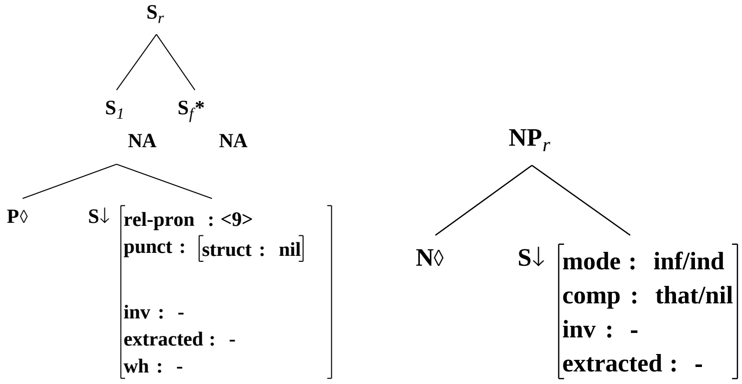
Another minor difference is that *whether* but not *if* is grammatical with S-subjs.[11] Thus, *if* is not among the <**comp**> values allowed in S-subjs. The final difference from S-comps is that

[11] Some speakers also find *if* as a complementizer only marginally grammatical in S-comps.

there are no S-subjs with **<mode>=ger**. As noted in footnote 4 of this chapter, gerundive complements are only allowed when there is no corresponding NP parse. In the case of gerundive S-subjs, there is always an NP parse available.

8.8 Nouns and Prepositions taking Sentential Complements

Trees: αNXNs, βvxPs, βPss, βnxPs, Tnx0N1s1, Tnx0A1s1.



(a) (b)

Figure 8.8: Sample trees for preposition: βPss (a) and noun: αNXNs (b) taking sentential complements

Prepositions and nouns can also select sentential complements, using the trees listed above. These trees use the **<mode>** and **<comp>** features as shown in Figure 8.8. For example, the noun *claim* takes only indicative complements with *that*, while the preposition *with* takes small clause complements, as seen in sentences 53-56.

(53) Beth's claim that Clove was a smart dog ....

(54) ∗Beth's claim that Clove a smart dog ....

(55) Dania wasn't getting any sleep with Doug sick .

(56) ∗Dania wasn't getting any sleep with Doug was sick .

8.9 PRO control

8.9.1 Types of control

In the literature on control, two types are often distinguished: obligatory control, as in sentences 57, 58, 59, and 60 and optional control, as in sentence 61.

(57) Srini$_i$ promised Mickey [PRO$_i$ to leave] .

(58) Srini persuaded Mickey$_i$ [PRO$_i$ to leave] .

(59) Srini$_i$ wanted [PRO$_i$ to leave] .

(60) Christy$_i$ left the party early [PRO$_i$ to go to the airport] .

(61) [PRO$_{arb/i}$ to dance] is important for Bill$_i$.

At present, an analysis for obligatory control into complement clauses (as in sentences 57, 58, and 59) has been implemented. An analysis for cases of obligatory control into adjunct clauses and optional control exists and can be found in [Bhatt, 1994].

8.9.2 A feature-based analysis of PRO control

The analysis for obligatory control involves co-indexation of the control feature of the NP anchored by PRO to the control feature of the controller. A feature equation in the tree anchored by the control verb co-indexes the control feature of the controlling NP with the foot node of the tree. All sentential trees have a co-indexed control feature from the root S to the subject NP.

When the tree containing the controller adjoins onto the complement clause tree containing the PRO, the features of the foot node of the auxiliary tree are unified with the bottom features of the root node of the complement clause tree containing the PRO. This leads to the control feature of the controller being co-indexed with the control feature of the PRO.

Depending on the choice of the controlling verb, the control propagation paths in the auxiliary trees are different. In the case of subject control (as in sentence 57), the subject NP and the foot node have co-indexed control features, while for object control (e.g. sentence 58, the object NP and the foot node are co-indexed for control. Among verbs that belong to the Tnx0Vnx1s2 family, i.e. verbs that take an NP object and a clausal complement, subject-control verbs form a distinct minority, *promise* being the only commonly used verb in this class.

Consider the derivation of sentence 58. The auxiliary tree for *persuade*, shown in Figure 8.9, has the following feature equation 62.

(62) NP$_1$:<**control**> = S$_2$.t:<**control**>

The auxiliary tree adjoins into the tree for *leave*, shown in Figure 8.10, which has the following feature equation 63.

(63) S$_r$.b:<**control**> = NP$_0$.t:<**control**>

Since the adjunction takes place at the root node (S$_r$) of the *leave* tree, after unification, NP$_1$ of the *persuade* tree and NP$_0$ of the *leave* tree share a control feature. The resulting derived and derivation trees are shown in Figures 8.11 and 8.12.

8.9.3 The nature of the control feature

The control feature does not have any value and is used only for co-indexing purposes. If two NPs have their control features co-indexed, it means that they are participating in a relationship of control; the c-commanding NP controls the c-commanded NP.

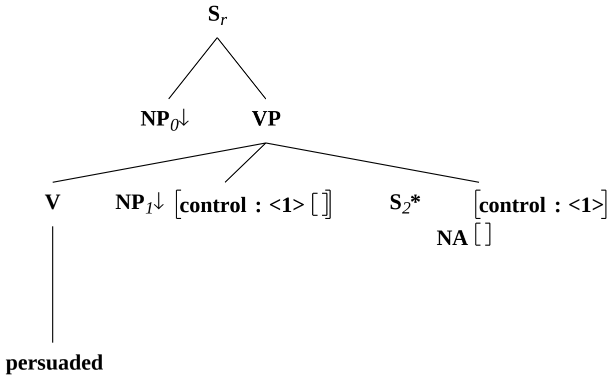


Figure 8.9: Tree for *persuaded*

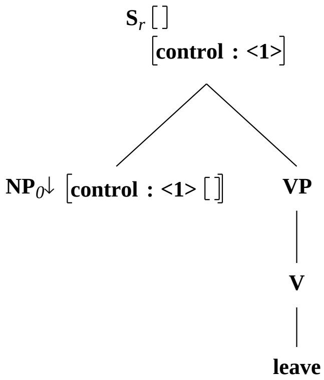


Figure 8.10: Tree for *leave*

8.9.4 Long-distance transmission of control features

Cases involving embedded infinitival complements with PRO subjects such as 64 can also be handled.

(64) John$_i$ wants [PRO$_i$ to want [PRO$_i$ to dance]] .

The control feature of 'John' and the two PRO's all get co-indexed. This treatment might appear to lead to a problem. Consider 65:

(65) John$_{*i}$ wants [Mary$_i$ to want [PRO$_i$ to dance]] .

If both the *want* trees have the control feature of their subject co-indexed to their foot nodes, we would have a situtation where the PRO is co-indexed for control feature with *John*, as well as with *Mary*. Note that the higher *want* in 64 is $want_{ECM}$ - it assigns case to the subject of the lower clause while the lower *want* in 64 is not. Subject control is restricted to non-ECM (Exceptional Case Marking) verbs that take infinitival complements. Since the two *wants* in 64 are different with respect to their control (and other) properties, the control feature of PRO stops at *Mary* and is not transmitted to the higher clause.

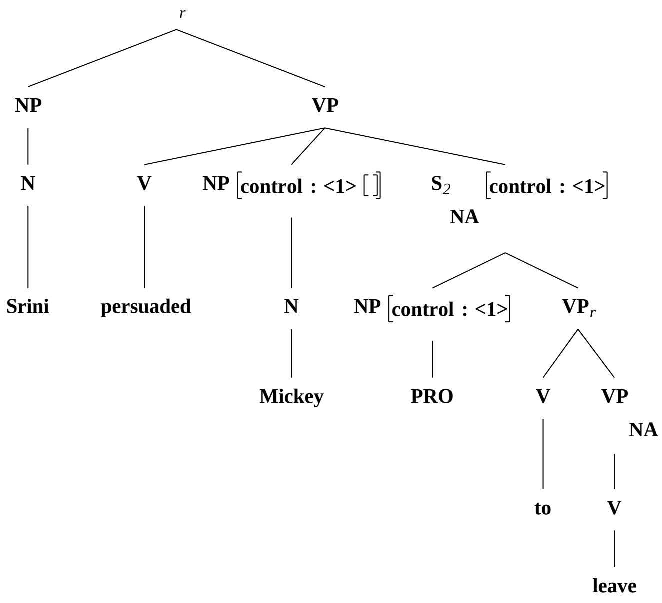


Figure 8.11: Derived tree for *Srini persuaded Mickey to leave*

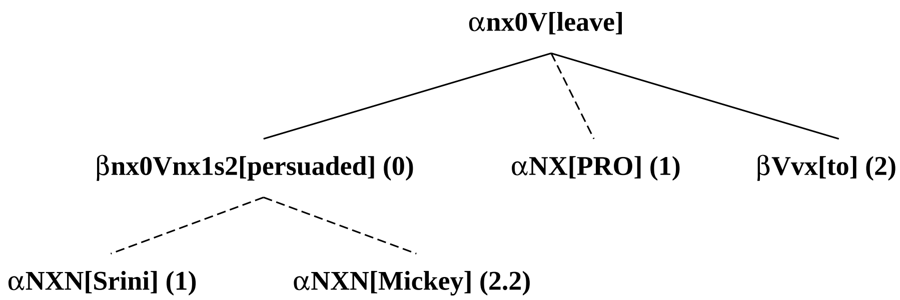


Figure 8.12: Derivation tree for *Srini persuaded Mickey to leave*

8.9.5 Locality constraints on control

PRO control obeys locality constraints. The controller for PRO has to be in the immediately higher clause. Consider the ungrammatical sentence 66 (66 is ungrammatical only with the co-indexing indicated below).

(66) * John$_i$ wants [PRO$_i$ to persuade Mary$_i$ [PRO$_i$ to dance]]

However, such a derivation is ruled out automatically by the mechanisms of a TAG derivation and feature unification. Suppose it was possible to first compose the *want* tree with the *dance* tree and then insert the *persuade* tree. (This is not possible in the XTAG grammar because of the convention that auxiliary trees have NA (Null Adjunction) constraints on their foot nodes.) Even then, at the end of the derivation the control feature of the subject of *want* would end up co-indexed with the PRO subject of *persuade* and the control feature of *Mary* would be co-indexed with the PRO subject of *dance* as desired. There is no way to generate the illegal co-indexing in 65. Thus the locality constraints on PRO control fall out from the mechanics of TAG derivation and feature unification.

8.10 Reported speech

Reported speech is handled in the XTAG grammar by having the reporting clause adjoin into the quote. Thus, the reporting clause is an auxiliary tree, anchored by the reporting verb. See [Doran, 1998] for details of the analysis. There are trees in both the Tnx0Vs1 and Tnx0nx1s2 families to handle reporting clauses which precede, follow and come in the middle of the quote.

Chapter 9

The English Copula, Raising Verbs, and Small Clauses

The English copula, raising verbs, and small clauses are all handled in the English XTAG grammar by a common analysis based on sentential clauses headed by non-verbal elements. Since there are a number of different analyses in the literature of how these phenomena are related (or not), we will present first the data for all three phenomena, then various analyses from the literature, finishing with the analysis used in the English XTAG grammar.[1]

9.1 Usages of the copula, raising verbs, and small clauses

9.1.1 Copula

The verb *be* as used in sentences 67-69 is often referred to as the COPULA. It can be followed by a noun, an adjective, or a prepositional phrase.

(67) Carl is a jerk .

(68) Carl is upset .

(69) Carl is in a foul mood .

Although the copula may look like a main verb at first glance, its syntactic behavior follows the auxiliary verbs rather than main verbs. In particular,

- Copula *be* inverts with the subject.

  (70) is Beth writing her dissertation ?
  is Beth upset ?
  ∗wrote Beth her dissertation ?

- Copula *be* occurs to the left of the negative marker *not*.

[1] This chapter is strongly based on [Heycock, 1991]. Sections 9.1 and 9.2 are greatly condensed from her paper, while the description of the XTAG analysis in section 9.3 is an updated and expanded version.

(71) Beth is not writing her dissertation .
Beth is not upset .
∗Beth wrote not her dissertation .

- Copula *be* can contract with the negative marker *not*.

(72) Beth isn't writing her dissertation .
Beth isn't upset .
∗Beth wroten't her dissertation .

- Copula *be* can contract with pronominal subjects.

(73) She's writing her dissertation .
She's upset .
∗She'ote her dissertation .

- Copula *be* occurs to the left of adverbs in the unmarked order.

(74) Beth is often writing her dissertation .
Beth is often upset .
∗Beth wrote often her dissertation .

Unlike all the other auxiliaries, however, copula *be* is not followed by a verbal category (by definition) and therefore must be the rightmost verb. In this respect, it is like a main verb.

The semantic behavior of the copula is also unlike main verbs. In particular, any semantic restrictions or roles placed on the subject come from the complement phrase (NP, AP, PP) rather than from the verb, as illustrated in sentences (75) and (76). Because the complement phrases predicate over the subject, these types of sentences are often called PREDICATIVE sentences.

(75) The bartender was garrulous .

(76) ?The cliff was garrulous .

9.1.2 Raising Verbs

Raising verbs are the class of verbs that share with the copula the property that the complement, rather than the verb, places semantic constraints on the subject.

(77) Carl seems a jerk .
Carl seems upset .
Carl seems in a foul mood .

(78) Carl appears a jerk .
Carl appears upset .
Carl appears in a foul mood .

The raising verbs are similar to auxiliaries in that they order with other verbs, but they are unique in that they can appear to the left of the infinitive, as seen in the sentences in (79). They cannot, however, invert or contract like other auxiliaries (80), and they appear to the right of adverbs (81).

(79) Carl seems to be a jerk .
Carl seems to be upset .
Carl seems to be in a foul mood .

(80) *seems Carl to be a jerk ?
*Carl seemn't to be upset .
*Carl'ems to be in a foul mood .

(81) Carl often seems to be upset .
*Carl seems often to be upset .

9.1.3 Small Clauses

One way of describing small clauses is as predicative sentences without the copula. Since matrix clauses require tense, these clausal structures appear only as embedded sentences. They occur as complements of certain verbs, each of which may allow certain types of small clauses but not others, depending on its lexical idiosyncrasies.

(82) I consider [Carl a jerk] .
I consider [Carl upset] .
?I consider [Carl in a foul mood] .

(83) I prefer [Carl in a foul mood] .
??I prefer [Carl upset] .

9.1.4 Raising Adjectives

Raising adjectives are the class of adjectives that share with the copula and raising verbs the property that the complement, rather than the verb, places semantic constraints on the subject.

They appear with the copula in a matrix clause, as in (84). However, in other cases, such as that of small clauses (85), they do not have to appear with the copula.

(84) Carl is likely to be a jerk .
Carl is likely to be upset .
Carl is likely to be in a foul mood .
Carl is likely to perjure himself .

(85) I consider Carl likely to perjure himself .

9.2 Various Analyses

9.2.1 Main Verb Raising to INFL + Small Clause

In [Pollock, 1989] the copula is generated as the head of a VP, like any main verb such as *sing* or *buy*. Unlike all other main verbs[2], however, *be* moves out of the VP and into Infl in a

[2] with the exception of *have* in British English. See footnote 1 in Chapter 22.

tensed sentence. This analysis aims to account for the behavior of *be* as an auxiliary in terms of inversion, negative placement and adverb placement, while retaining a sentential structure in which *be* heads the main VP at D-Structure and can thus be the only verb in the clause.

Pollock claims that the predicative phrase is not an argument of *be*, which instead he assumes to take a small clause complement, consisting of a node dominating an NP and a predicative AP, NP or PP. The subject NP of the small clause then raises to become the subject of the sentence. This accounts for the failure of the copula to impose any selectional restrictions on the subject. Raising verbs such as *seem* and *appear*, presumably, take the same type of small clause complement.

9.2.2 Auxiliary + Null Copula

In [Lapointe, 1980] the copula is treated as an auxiliary verb that takes as its complement a VP headed by a passive verb, a present participle, or a null verb (the true copula). This verb may then take AP, NP or PP complements. The author points out that there are many languages that have been analyzed as having a null copula, but that English has the peculiarity that its null copula requires the co-presence of the auxiliary *be*.

9.2.3 Auxiliary + Predicative Phrase

In GPSG ([Gazdar *et al.*, 1985], [Sag *et al.*, 1985]) the copula is treated as an auxiliary verb that takes an X^2 category with a + value for the head feature [PRD] (predicative). AP, NP, PP and VP can all be [+PRD], but a Feature Co-occurrence Restriction guarantees that a [+PRD] VP will be headed by a verb that is either passive or a present participle.

GPSG follows [Chomsky, 1970] in adopting the binary valued features [V] and [N] for decomposing the verb, noun, adjective and preposition categories. In that analysis, verbs are [+V,−N], nouns are [−V,+N], adjectives [+V,+N] and prepositions [−V,−N]. NP and AP predicative complements generally pattern together; a fact that can be stated economically using this category decomposition. In neither [Sag *et al.*, 1985] nor [Chomsky, 1970] is there any discussion of how to handle the complete range of complements to a verb like *seem*, which takes AP, NP and PP complements, as well as infinitives. The solution would appear to be to associate the verb with two sets of rules for small clauses, leaving aside the use of the verb with an expletive subject and sentential complement.

9.2.4 Auxiliary + Small Clause

In [Moro, 1990] the copula is treated as a special functional category - a lexicalization of tense, which is considered to head its own projection. It takes as a complement the projection of another functional category, Agr (agreement). This projection corresponds roughly to a small clause, and is considered to be the domain within which predication takes place. An NP must then raise out of this projection to become the subject of the sentence: it may be the subject of the AgrP, or, if the predicate of the AgrP is an NP, this may raise instead. In addition to occurring as the complement of *be*, AgrP is selected by certain verbs such as *consider*. It follows from this analysis that when the complement to *consider* is a simple AgrP, it will always consist of a subject followed by a predicate, whereas if the complement contains the verb *be*, the predicate of the AgrP may raise to the left of *be*, leaving the subject of the AgrP to the right.

(86) John$_i$ is $[_{AgrP}$ t_i the culprit $]$.

(87) The culprit$_i$ is $[_{AgrP}$ John t_i $]$.

(88) I consider $[_{AgrP}$ John the culprit$]$.

(89) I consider [John$_i$ to be $[_{AgrP}$ t_i the culprit]] .

(90) I consider [the culprit$_i$ to be $[_{AgrP}$ John t_i]] .

Moro does not discuss a number of aspects of his analysis, including the nature of Agr and the implied existence of sentences without VP's.

9.3 XTAG analysis

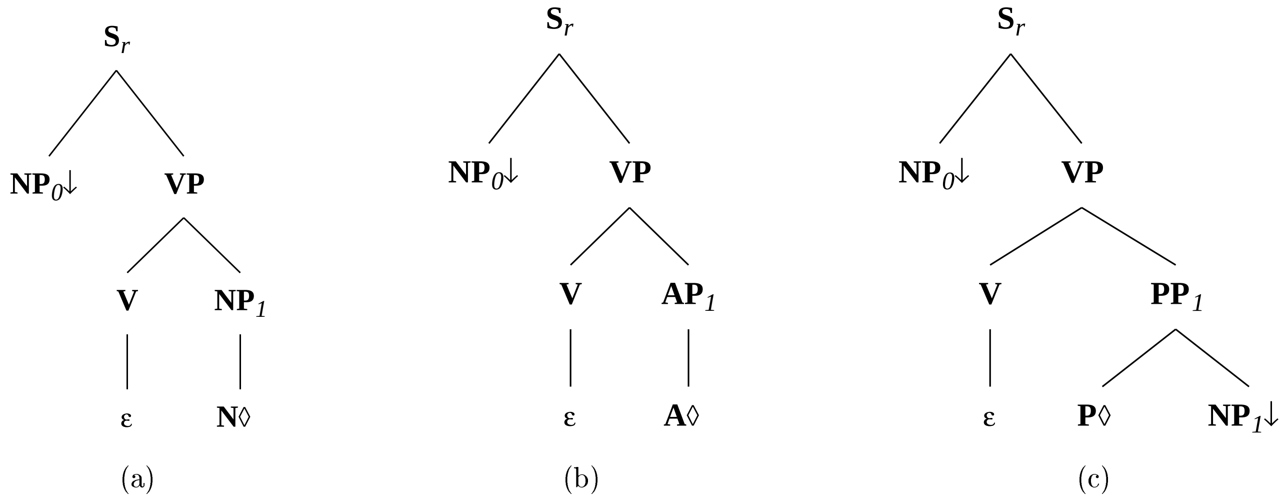


Figure 9.1: Predicative trees: αnx0N1 (a), αnx0Ax1 (b) and αnx0Pnx1 (c)

The XTAG grammar provides a uniform analysis for the copula, raising verbs and small clauses by treating the maximal projections of lexical items that can be predicated as predicative clauses, rather than simply noun, adjective and prepositional phrases. The copula adjoins in for matrix clauses, as do the raising verbs. Certain other verbs (such as *consider*) can take the predicative clause as a complement, without the adjunction of the copula, to form the embedded small clause.

The structure of a predicative clause, then, is roughly as seen in (91)-(93) for NP's, AP's and PP's. The XTAG trees corresponding to these structures[3] are shown in Figures 9.1(a), 9.1(b), and 9.1(c), respectively.

[3]There are actually two other predicative trees in the XTAG grammar. Another predicative noun phrase tree is needed for noun phrases without determiners, as in the sentence *They are firemen*, and another prepositional phrase tree is needed for exhaustive prepositional phrases, such as *The workers are below.*

(91) $[_S$ NP $[_{VP}$ N $\ldots]]$

(92) $[_S$ NP $[_{VP}$ A $\ldots]]$

(93) $[_S$ NP $[_{VP}$ P $\ldots]]$

The copula *be* and raising verbs all get the basic auxiliary tree as explained in the section on auxiliary verbs (section 22.1). Unlike the raising verbs, the copula also selects the inverted auxiliary tree set. Figure 9.2 shows the basic auxiliary tree anchored by the copula *be*. The <**mode**> feature is used to distinguish the predicative constructions so that only the copula and raising verbs adjoin onto the predicative trees.

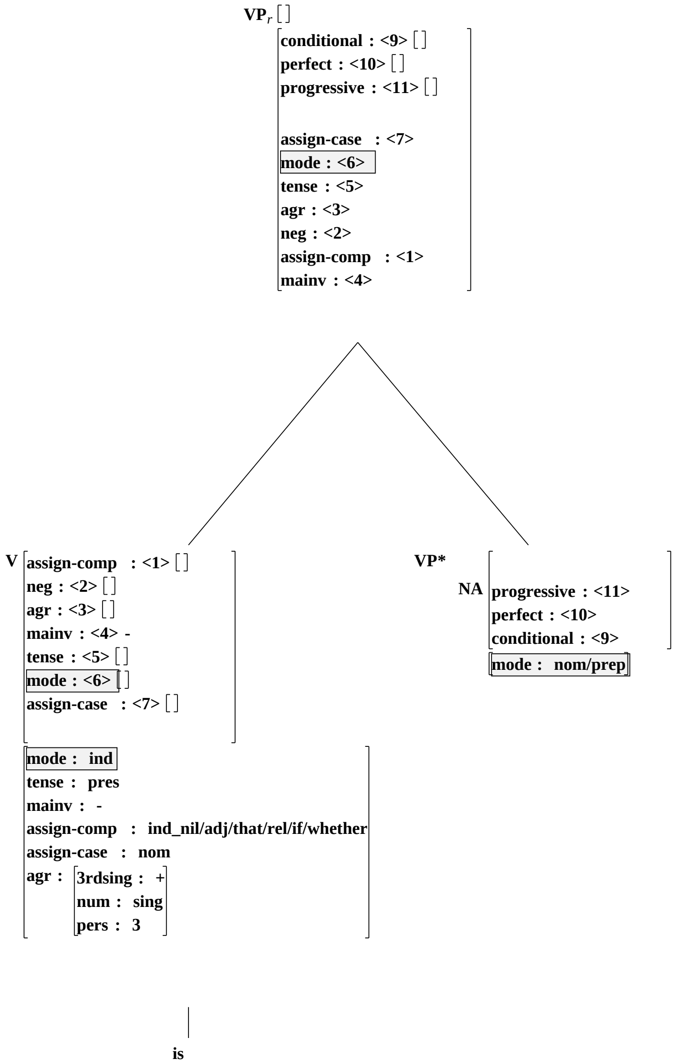


Figure 9.2: Copula auxiliary tree: βVvx

There are two possible values of <**mode**> that correspond to the predicative trees, **nom** and **prep**. They correspond to a modified version of the four-valued [N,V] feature described in section 9.2.3. The **nom** value corresponds to [N+], selecting the NP and AP predicative clauses. As mentioned earlier, they often pattern together with respect to constructions using predicative clauses. The remaining prepositional phrase predicative clauses, then, correspond to the **prep** mode.

Figure 9.3 shows the predicative adjective tree from Figure 9.1(b) now anchored by *upset* and with the features visible. As mentioned, <**mode**>=**nom** on the VP node prevents auxiliaries other than the copula or raising verbs from adjoining into this tree. In addition, it prevents the predicative tree from occurring as a matrix clause. Since all matrix clauses in XTAG must be mode indicative (**ind**) or imperative (**imp**), a tree with <**mode**>=**nom** or <**mode**>=**prep** must have an auxiliary verb (the copula or a raising verb) adjoin in to make it <**mode**>=**ind**.

The distribution of small clauses as embedded complements to some verbs is also managed through the mode feature. Verbs such as *consider* and *prefer* select trees that take a sentential complement, and then restrict that complement to be <**mode**>=**nom** and/or <**mode**>=**prep**, depending on the lexical idiosyncrasies of that particular verb. Many verbs that don't take small clause complements do take sentential complements that are <**mode**>=**ind**, which includes small clauses with the copula already adjoined. Hence, as seen in sentence sets (94)-(96), *consider* takes only small clause complements, *prefer* takes both **prep** (but not **nom**) small clauses and indicative clauses, while *feel* takes only indicative clauses.

(94) She considers Carl a jerk .
?She considers Carl in a foul mood .
∗She considers that Carl is a jerk .

(95) ∗She prefers Carl a jerk .
She prefers Carl in a foul mood .
She prefers that Carl is a jerk .

(96) ∗She feels Carl a jerk .
∗She feels Carl in a foul mood .
She feels that Carl is a jerk .

Figure 9.4 shows the tree anchored by *consider* that takes the predicative small clauses.

Raising verbs such as *seems* work essentially the same as the auxiliaries, in that they also select the basic auxiliary tree, as in Figure 9.2. The only difference is that the value of <**mode**> on the VP foot node might be different, depending on what types of complements the raising verb takes. Also, two of the raising verbs take an additional tree, βVpxvx, shown in Figure 9.5, which allows for an experiencer argument, as in *John seems to me to be happy.*

Raising adjectives, such as *likely*, take the tree shown in Figure 9.6. This tree combines aspects of the auxiliary tree βVvx and the adjectival predicative tree shown in Figure 9.1(b). As with βVvx, it adjoins in as a VP auxiliary tree. However, since it is anchored by an adjective, not a verb, it is similar to the adjectival predicative tree in that it has an ϵ at the V node, and a feature value of <**mode**>=**nom** which is passed up to the VP root indicates that it is an adjectival predication. This serves the same purpose as in the case of the tree in Figure 9.3, and forces another auxiliary verb, such as the copula, to adjoin in to make it <**mode**>=**ind**.

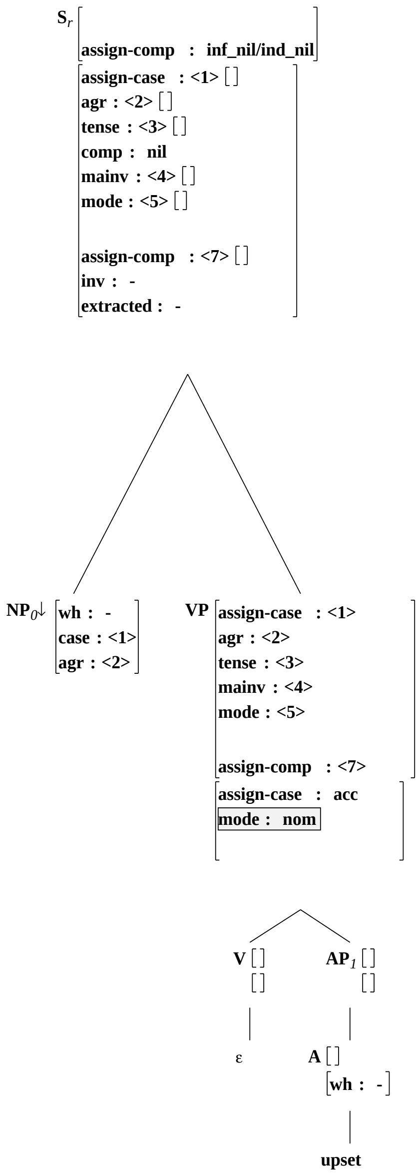


Figure 9.3: Predicative AP tree with features: αnx0Ax1

9.3.1 Raising Passives

As discussed in Section 8.6.4, the passives of Exceptional Case Marking verbs are also treated as raising verbs. The passive therefore also selects the Vvx tree, and like other raising verbs such as *seems* will select for an infinitival complement. However, unlike *seems*, the <**mode**> feature is set to **ppart**, which therefore forces an auxiliary to also adjoin, as with other passives, as described in Chapter 14. For example, to derive (97), the Vvx tree for *was* adjoins at the root of the tree for *expected* in in Figure 9.7(a), which adjoins into a derivation for *Bob to talk* at the VP node.

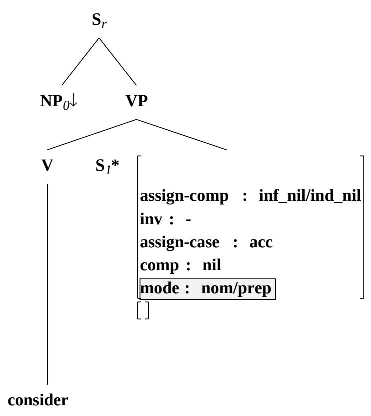


Figure 9.4: *Consider* tree for embedded small clauses

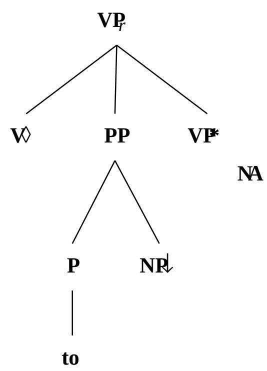


Figure 9.5: Raising verb with experiencer tree: βVpxvx

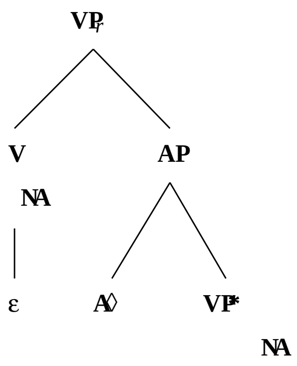


Figure 9.6: Raising adjective tree: βVvx-adj

(97) Bob was expected to talk .

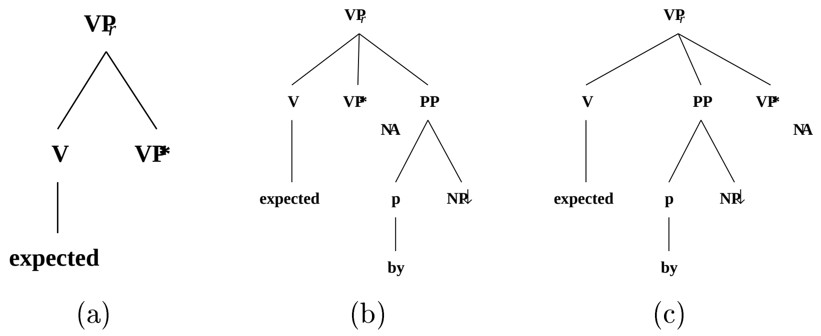


Figure 9.7: ECM raising passive trees: βVvx (a), βVvxbynx (b), βVbynxvx (c)

Also, the *by* phrase associated with the raising passives can appear to the left or right of the infinitival complement, as in (98) and (99).

(98) Bob was expected by Bill to talk .

(99) Bob was expected to talk by Bill .

To handle these cases, the raising passives such as *expect* can also select the trees shown in figures 9.7(b) and 9.7(c). ECM verbs such as *expect* therefore select the ECM tree family and also, separately, the three trees in figure 9.7.

Also, it has long been noted that passives of both full and bare infinitive ECM constructions are full infinitives, as in (100) and (101).

(100) Bob sees the harmonica fall .

(101) The harmonica was seen to fall .

(102) $*$The harmonica was seen fall .

Under the TAG ECM analysis, this fact is easy to implement. The foot node of the ECM passive tree is simply set to have <**mode**>=**inf**, which prevents the derivation of (102). Therefore, while verbs selecting full and bare infinitives will differ in the <**mode**> specified for the active form when selecting the ECM tree family, they all use <**mode**>=**inf** for the passive tree.

Selecting the passive tree separately from the ECM tree family for the active trees has the advantage of allowing verbs which only have the passive to not select the active trees. There are a number of such cases, as in (103), where the raising passive exists but not its active counterpart. For more discussion of this issue, see [Kroch and Joshi, 1985].

(103) John is said to be a crook .

There are also some cases for which the raising passive complement does not take <**mode**>=**inf**, but rather a **nom/prep** complement, as in (104) and (105):

(104) John is considered a crook .

(105) The Americans are believed involved in the coup .

(106) *They believe the Americans involved in the coup .

Such cases are handled by selecting the Vvx, Vbynxvx, and Vvxbynx trees with the appropriate <**mode**> on the foot node, instead of just **inf**. In some cases, as in (106), the corresponding active sentence sounds quite bad, and this is prohibited by not selecting the ECM family for the active cases with that mode. For example, *believe* selects the ECM family with <**mode**>**=inf** and the raising passive trees with <**mode**>**=inf/nom/prep**.

9.4 Non-predicative *BE*

The examples with the copula that we have given seem to indicate that *be* is always followed by a predicative phrase of some sort. This is not the case, however, as seen in sentences such as (107)-(112). The noun phrases in these sentences are not predicative. They do not take raising verbs, and they do not occur in embedded small clause constructions.

(107) my teacher is Mrs. Wayman .

(108) Doug is the man with the glasses .

(109) *My teacher seems Mrs. Wayman .

(110) *Doug appears the man with the glasses .

(111) *I consider [my teacher Mrs. Wayman] .

(112) *I prefer [Doug the man with the glasses] .

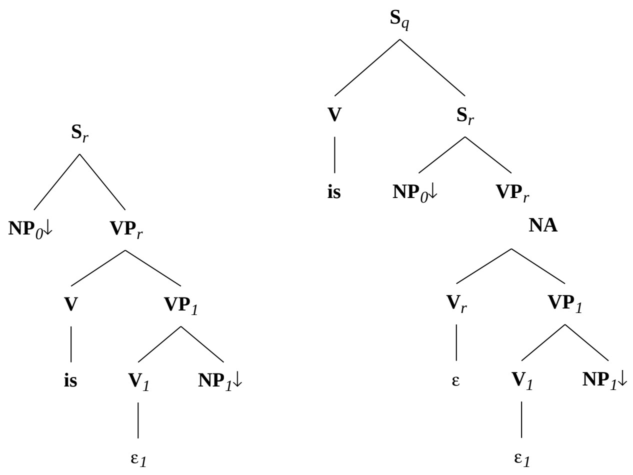
In addition, the subject and complement can exchange positions in these type of examples but not in sentences with predicative *be*. Sentence (113) has the same interpretation as sentence (108) and differs only in the positions of the subject and complement NP's. Similar sentences, with a predicative *be*, are shown in (114) and (115). In this case, the sentence with the exchanged NP's (115) is ungrammatical.

(113) The man with the glasses is Doug .

(114) Doug is a programmer .

(115) *A programmer is Doug .

The non-predicative *be* in (107) and (108), also called EQUATIVE BE, patterns differently, both syntactically and semantically, from the predicative usage of *be*. Since these sentences are clearly not predicative, it is not desirable to have a tree structure that is anchored by the NP, AP, or PP, as we have in the predicative sentences. In addition to the conceptual problem, we would also need a mechanism to block raising verbs from adjoining into these sentences (while



(a) (b)

Figure 9.8: Equative *BE* trees: αnx0BEnx1 (a) and αInvnx0BEnx1 (b)

allowing them for true predicative phrases), and prevent these types of sentence from being embedded (again, while allowing them for true predicative phrases).

Although non-predicative *be* is not a raising verb, it does exhibit the auxiliary verb behavior set out in section 9.1.1. It inverts, contracts, and so forth, as seen in sentences (116) and (117), and therefore can not be associated with any existing tree family for main verbs. It requires a separate tree family that includes the tree for inversion. Figures 9.8(a) and 9.8(b) show the declarative and inverted trees, respectively, for equative *be*.

(116) is my teacher Mrs. Wayman ?

(117) Doug isn't the man with the glasses .

Chapter 10

Ditransitive constructions and dative shift

Verbs such as *give* and *put* that require two objects, as shown in examples (118)-(121), are termed ditransitive.

(118) Christy gave a cannoli to Beth Ann .

(119) ∗Christy gave Beth Ann .

(120) Christy put a cannoli in the refrigerator .

(121) ∗Christy put a cannoli .

The indirect objects *Beth Ann* and *refrigerator* appear in these examples in the form of PP's. Within the set of ditransitive verbs there is a subset that also allow two NP's as in (122). As can be seen from (122) and (123) this two NP, or double-object, construction is grammatical for *give* but not for *put*.

(122) Christy gave Beth Ann a cannoli .

(123) ∗Christy put the refrigerator the cannoli .

The alternation between (118) and (122) is known as dative shift.[1] In order to account for verbs with dative shift the English XTAG grammar includes structures for both variants in the two tree families Tnx0Vnx1Pnx2 and Tnx0Vnx2nx1. For an alternating verb such as *give*, there are two anchoring instances: *give to* selects Tnx0Vnx1Pnx2 and *give* selects Tnx0Vnx2nx1. The declarative trees for the shifted and non-shifted alternations are shown in Figure 10.1.

The indexing of nodes in these two trees represents the fact that the semantic role of the indirect object (NP_2) in Figure 10.1(a) is the same as that of the direct object (NP_2) in Figure 10.1(b) (and vice versa). This use of indexing is consistent with our treatment of other constructions such as passive and ergative.

[1] In languages similar to English that have overt case marking indirect objects would be marked with dative case. It has also been suggested that for English the preposition *to* serves as a dative case marker.

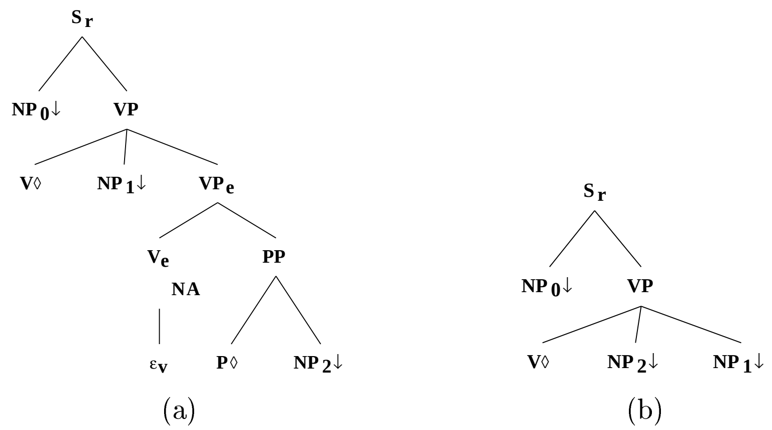


Figure 10.1: Dative shift trees: αnx0Vnx1Pnx2 (a) and αnx0Vnx2nx1 (b)

Verbs that do not show this alternation and have only the NP PP structure (e.g. *put*) do not select the tree family Tnx0Vnx2nx1. Non-alternating verbs that select for particular prepositions take Tnx0Vnx1Pnx2 and those which take a wider range of prepositions take Tnx0Vnx1pnx2 (see Chapter 11 for more exposition on the XTAG treatment of these constructions).

Other verbs such as *ask* allow only the NP NP structure as shown in (124) and (125). These verbs select only the tree family Tnx0Vnx2nx1.

(124) Beth Ann asked Srini a question .

(125) ∗Beth Ann asked a question to Srini .

There are other apparent cases of dative shift with *for*, such as the so called benefectives, (126) and (127), that we have taken to be structurally distinct from the cases with *to*.

(126) Beth Ann baked Dusty a biscuit .

(127) Beth Ann baked a biscuit for Dusty .

[McCawley, 1988] notes:

> A "*for-dative*" expression in underlying structure is external to the V with which it is combined, in view of the fact that the latter behaves as a unit with regard to all relevant syntactic phenomena.

In other words, the *for* PP's that appear to undergo dative shift are actually adjuncts, not complements. Examples (128) and (129) demonstrate that PP's with *for* are optional while ditransitive *to* PP's are not.

(128) Beth Ann made dinner .

(129) ∗Beth Ann gave dinner .

Other evidence in support of the view that *for* PP's are adjuncts comes from the *do so* test. *Do so* substitutes for an entire VP. This is illustrated in (130), where *do so* substitutes for *ate a banana.* The VP for which *do so* substitutes can exclude adjuncts (which are adjoined to a VP) as in (131), but not arguments, as shown in (132).

(130) John ate a banana and Mary did so too . (Mary ate a banana also)

(131) John ate a banana in the morning and Mary did so in the evening. (Mary ate a banana also and she ate it in the evening)

(132) *John ate a banana and Mary did so an apple.

The dative *to* phrase acts like an argument according to this test (133), but the *for* phrase does not (134).

(133) John gave a book to Mary and Bill did so too .

(134) *John gave a book to Mary and Bill did so to Janet .

(135) John baked a cake for Mary and Bill did so too .

(136) John baked a cake for Mary and Bill did so for Janet .

Since *do so* substitutes for an entire VP, the fact that (136) is grammatical indicates that the *for* PP is an adjunct (i.e., there exists a VP node lower than the *for* PP which *do so* can substitute for).

Consequently, in the XTAG grammar the apparent dative shift with *for* PP's is treated as Tnx0Vnx2nx1 for the NP NP structure, and as a transitive plus an adjoined adjunct PP for the NP PP structure. To account for the ditransitive *to* PP's, the preposition *to* co-anchors the tree family Tnx0Vnx1Pnx2 with the alternating verb.

[McCawley, 1988] also notes that the *to* and *for* cases differ with respect to passivization; the indirect objects with *to* may be the subjects of corresponding passives, as seen in (137)-(138), while the alleged indirect objects with *for* cannot, as in sentences (139)-(140). Note that the passivisation examples are for NP NP structures of verbs that take *to* or *for* PP's.

(137) Beth Ann gave Clove dinner .

(138) Clove was given dinner (by Beth Ann) .

(139) Beth Ann made Clove dinner .

(140) ?Clove was made dinner (by Beth Ann) .

However, we believe that this picture is inaccurate, and that the indirect objects in the *for* case are, in fact, allowed to be the subjects of passives, as in sentences (141)-(142). The apparent strangeness of sentence (140) is caused by interference from other interpretations of *Clove was made dinner .*

(141) Dania baked Doug a cake .

(142) Doug was baked a cake by Dania .

Chapter 11

PP Complement Verbs

Just as some verbs require noun phrase complements, others require preposition phrase complements. For an intransitive verb like *sleep*, a PP which follows it is an adjunct, as can be seen in (143) and (144), but for a verb such as *venture*, the PP is required, as the ungrammaticality of (146) attests.

(143) The spelunker slept in the cave .

(144) The spelunker slept .

(145) The spelunker ventured into the cave .

(146) *The spelunker ventured .

We will call verbs such as *venture* transitive PP complement verbs. There are also ditransitive PP complement verbs, such as *put*.

(147) Bill put the cigar on the table .

(148) *Bill put the cigar .

(a) PP substitition (b) Expanded PP

Figure 11.1: Two analyses of the PP complement

Since the PP in examples like (145) and (147) is an argument of the verb, PP complement verbs such as *venture* and *put* must provide a substitution site for the PP. There are two possible structures for allowing this in the grammar: the PP substitution node structure (shown in Figure 11.1a) and the expanded PP structure (shown in Figure 11.1b).

The structure chosen is that of the expanded PP in Figure 11.1b. The choice was motivated by two reasons. The first is that it is possible to extract the NP object of the P, as shown in (149), and expanding the PP allows the NP node to be available to metarules which create the trees for extraction.

(149) What did Bill put the cigar on ?

The second reason has to do with the inherent semantics of the elementary trees of such verbs. Currently, we treat the verb as taking the NP, the object of the preposition, as its argument rather than the whole PP. In order to have access to this NP for the semantic representation, it must be explicitly represented in the elementary tree.

However, the current XTAG grammar does not parse sentences like (150) in which the verb takes exhaustive PP's like *there* (and also *somewhere*, *where* etc.) as its argument.

(150) Bill put the cigar there .

Since the existing analysis does not allow an exhaustive PP substitution, these cases are currently a problem.

There is another class of PP complement verbs which make it necessary to expand the PP structure. These are the verbs which require a particular preposition to head their PP complement, as illustrated in the following examples:

(151) Calvin depends on Hobbes .

(152) *Calvin depends with Hobbes .

(153) Calvin thought of the transmogrifier .

(154) *Calvin thought on the transmogrifier .[1]

(155) The expedition leader geared the hikers for the trip .

(156) *The expedition leader geared the hikers into the trip .

(157) The speaker subjected the audience to his dull wit .

For such verbs, the preposition is intrinsic to the construction. Note that sentence (153) is ambiguous between two interpretations, one in which the PP is an adjunct and the other in which it is a complement. The adjunct interpretation of this sentence is something like "Calvin was thinking and the topic of this thought was the transmogrifier" while the complement interpretation is "Calvin created the idea for the transmogrifier". The preposition correlated with the verb thus conveys a noncompositional meaning, which motivates us to analyze these constructions as being multiply-anchored by a verb and a corresponding preposition. Having the P as one of the anchors necessitates the expansion of the PP complement for these constructions.

We have thus determined one important motivation for the suggested basic structure of PP complement verbs. Another important aspect of these verbs is that, contrary to cases where verbs take an NP complement, certain adjuncts are found to occur relatively freely between the verb and the PP complement. NP complements cannot be separated from the verb by modifiers unless they are heavy, as shown in examples (158-161).

[1]This is fine as an adjunct, but not as a complement.

(158) *Bill borrowed quickly the car .

(159) *Bill borrowed for many days the car .

(160) Bill borrowed for many days the car which his father had bought from his uncle two weeks ago .

(161) *John gave Mary often the book .

PP complements, however, allow several types of adjuncts in this position. For example, adverbs (162-164), certain prepositional phrases (165-167), and nominal adjuncts (168-170) are able to intercede the verb and the preposition:

(162) John depends heavily on his staff .

(163) John thinks often of new ideas .

(164) John bases his opinions extensively on public attitudes .

(165) John depends in many ways on his staff .

(166) John thought for many hours of new ideas .

(167) John based his opinions for more than ten years on public attitudes .

(168) John depends quite a bit on his staff .

(169) John thought all the time of new ideas .

(170) John bases his opinions a good deal on public attitudes .

Some restrictions, however, do occur. The locative PP's in 171-173 and the adjunct clauses in 174-176 cannot intervene between the verb and its PP complement.

(171) *John depends from Honolulu on his staff .

(172) *John thought in the kitchen of new ideas .

(173) *John based his opinions in Honolulu on public attitudes .

(174) *John depends because he needs help on his wonderful and caring staff .

(175) *John thought because he was so creative of new ideas that provided the basis of many present technologies .

(176) *John bases his opinions because he cannot think for himself on the public attitudes that prevail today .

There are several ways in which we could handle these adjuncts. One of these is to allow only certain adjuncts to adjoin to V. However, this is unappealing because it would cause right-edge ambiguities (V vs. VP adjunction), and also because it cannot handle examples like (164), (167), (170), and (173) where there is an NP between the verb and the preposition.

Another option is to make the claim that the adjuncts get into the intervening position because of heavy-shift. With such an analysis, the structures of the trees for prepositional complements would not provide a site for adjunction between the verb and the preposition. However, there is no sense of heaviness in the examples (162-173), and we would expect (174),(175), and (176) to be fine since the PP's are quite heavy here.

The analysis we provide is structural, and is given in Figure 11.2 for the multi-anchor PP complement verbs. The analysis suggests short verb movement, and is reminiscent of Larsonian shells. However, the most important aspect of these trees for the purposes of the English XTAG grammar is that it provides a site for left adjunction to VP that intervenes between the verb and the preposition. This immediately rules out the adjunct clauses as desired, since such clauses only right adjoin to VP in the grammar. Most prepositional phrases are not good in the intervening position either, and this fits in nicely with the fact that the grammar does not have left VP adjoining prepositional phrases. For the restricted class of prepositions which are permitted in the intervening position, the grammar currently does not parse them, but entries which allow such left adjuction can be easily added later. Adverbs, however, already left-adjoin to VP's in the current grammar, so the structure accomodates this without need of any further modification.

The comparatives also provide motivation for the structure that we have adopted. Comparative adverb phrases such as *more heavily* may either left- or right-adjoin, and can thus adjoin in the intervening position. However, the *than*-clause portion of a comparative can only right adjoin. Together, these two adjoining behaviors work with our PP complement analysis to correctly capture the distinctions seen in the following examples:

(177) John depends more heavily on his staff than Mary .

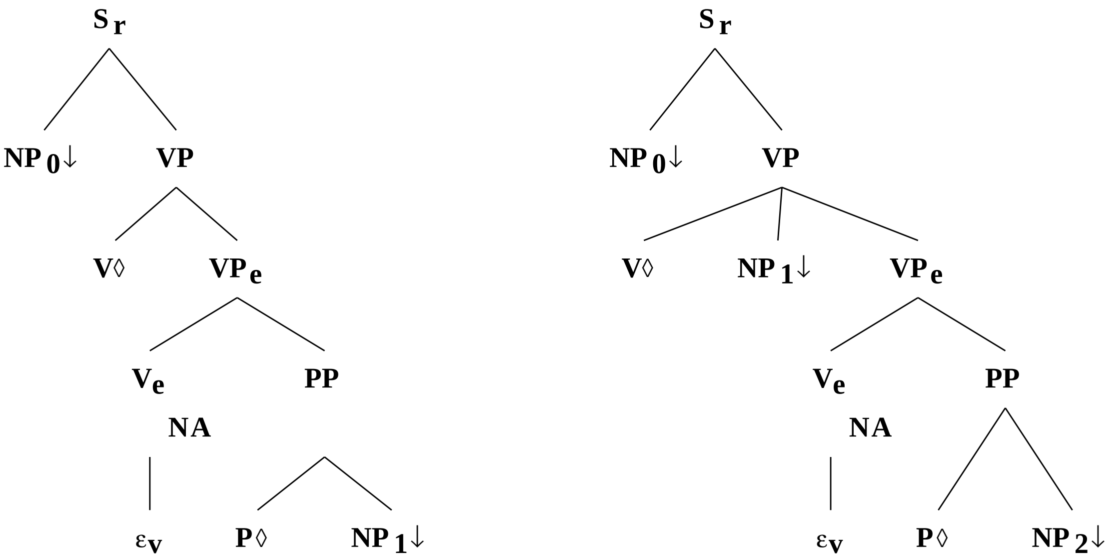
(178) John depends on his staff more heavily than Mary .

(179) *John depends more heavily than Mary on his staff .

In (177), the comparative adverb phrase has left-adjoined to the lower VP and the *than*-clause has right adjoined to it. In (178), both are right adjoined. The ungrammaticality of (179) is captured by the fact that the *than*-clause can only right-adjoin, and there is no right-adjoinable VP node available between the verb and the preposition.

This basic schema of short verb movement for PP complement families captures the adjunction possibilities directly and appears to partition the different types of adjoining elements in the appropriate way, with very little modification to the grammar. The analysis is operative in the families Tnx0Vpnx1, Tnx0Vnx1pnx2, Tnx0VPnx1, and Tnx0Vnx1Pnx2.

Notice that having two VP nodes along the right spine of the trees will engender right edge ambiguities. In other structures where this ambiguity arises (for example, sentences with auxiliaries), VP right-adjoining modifiers are restricted by the feature **mainv** $=+$ on their footnodes to avoid right-edge ambiguities. This restriction is extended to the present case where right VP adjunction is allowed to occur only on the higher VP by giving the lower VP the feature **mainv** $=$ **-**.



(a) αnx0VPnx1 (b) αnx0Vnx1Pnx2

Figure 11.2: Declarative trees for multi-anchor PP complement families

Chapter 12

It-clefts

There are several varieties of it-clefts in English. All the it-clefts have four major components:

- **the dummy subject:** *it*,
- **the main verb:** *be*,
- **the clefted element:** A constituent (XP) compatible with any gap in the clause,
- **the clause:** A clause (e.g. S) with or without a gap.

Examples of it-clefts are shown in (180)-(183).

(180) it was $[_{XP}$ here $_{XP}]$ $[_S$ that the ENIAC was created . $_S]$

(181) it was $[_{XP}$ at MIT $_{XP}]$ $[_S$ that colorless green ideas slept furiously . $_S]$

(182) it is $[_{XP}$ happily $_{XP}]$ $[_S$ that Seth quit Reality . $_S]$

(183) it was $[_{XP}$ there $_{XP}]$ $[_S$ that she would have to enact her renunciation . $_S]$

The clefted element can be of a number of categories, for example NP, PP or adverb. The clause can also be of several types. The English XTAG grammar currently has a separate analysis for only a subset of the 'specificational' it-clefts[1], in particular the ones without gaps in the clause (e.g. (182) and (183)). It-clefts that have gaps in the clause, such as (180) and (181) are currently handled as relative clauses. Although arguments have been made against treating the clefted element and the clause as a constituent ([Delahunty, 1984]), the relative clause approach does capture the restriction that the clefted element must fill the gap in the clause, and does not require any additional trees.

In the 'specificational' it-cleft without gaps in the clause, the clefted element has the role of an adjunct with respect to the clause. For these cases the English XTAG grammar requires additional trees. These it-cleft trees are in separate tree families because, although some researchers (e.g. [Akmajian, 1970]) derived it-clefts through movement from other sentence types, most current researchers (e.g. [Delahunty, 1984], [Knowles, 1986], [Gazdar *et al.*, 1985], [Delin,

[1] See e.g. [Ball, 1991], [Delin, 1989] and [Delahunty, 1984] for more detailed discussion of types of it-clefts.

1989] and [Sornicola, 1988]) favor base-generation of the various cleft sentences. Placing the it-cleft trees in their own tree families is consistent with the current preference for base generation, since in the XTAG English grammar, structures that would be related by transformation in a movement-based account will appear in the same tree family. Like the base-generated approaches, the placement of it-clefts in separate tree families makes the claim that there is no derivational relation between it-clefts and other sentence types.

The three it-cleft tree families are virtually identical except for the category label of the clefted element. Figure 12.1 shows the declarative tree and an inverted tree for the PP It-cleft tree family.

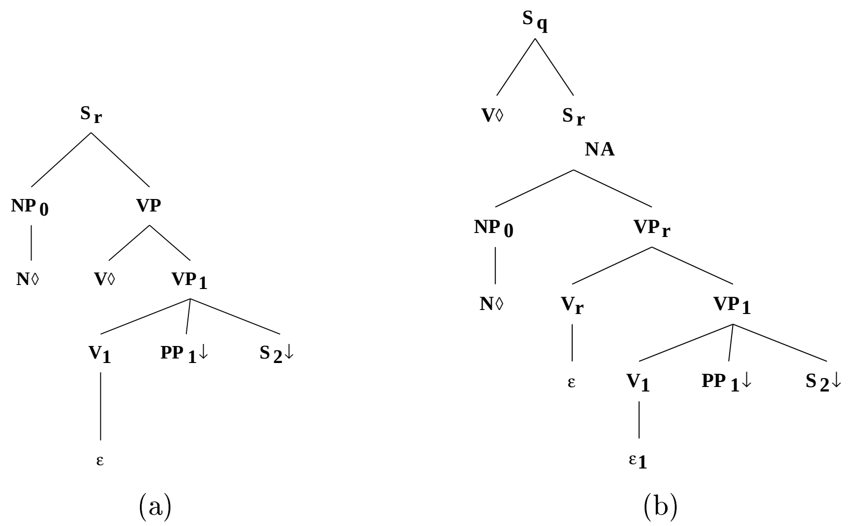


Figure 12.1: It-cleft with PP clefted element: αItVpnx1s2 (a) and αInvItVpnx1s2 (b)

The extra layer of tree structure in the VP represents that, while *be* is a main verb rather than an auxiliary in these cases, it retains some auxiliary properties. The VP structure for the equative/it-cleft-*be* is identical to that obtained after adjunction of predicative-*be* into small-clauses.[2] The inverted tree in Figure 12.1(b) is necessary because of *be*'s auxiliary-like behavior. Although *be* is the main verb in it-clefts, it inverts like an auxiliary. Main verb inversion cannot be accomplished by adjunction as is done with auxiliaries and therefore must be built into the tree family. The tree in Figure 12.1(b) is used for yes/no questions such as (184).

(184) was it in the forest that the wolf talked to the little girl ?

[2]For additional discussion of equative or predicative-*be* see Chapter 9.

Chapter 13

Resultatives

Resultatives constructions in English XTAG consist of verb-preposition and verb-adjective combinations.[1] The verbs that can form the resultatives are transitive, intransitive and ergative verbs that allow for either a prepositional or adjectival secondary predicate which expresses the result of an event denoted by the verb. In the examples in 187 – 189, the (a) sentences show the non-resultative declarative construction, whereas the (b) and (c) sentences show the resultative counterparts with adjectival and prepositional predicates respectively.

- **Transitive:**

  (187) John pounded the dough.

  (188) John pounded the dough flat.

  (189) John pounded the dough into a flat shape.

- **Intransitive:**

  (190) The joggers ran (*their sneakers).

  (191) The joggers ran their sneakers threadbare.

  (192) The joggers ran their sneakers into pieces.

- **Ergative:**

  (193) The river froze.

  (194) The river froze solid.

  (195) The river froze into one big piece of ice.

[1] Currently, we do not handle possible verb-noun combinations, as in examples 185 and 186,

(185) She painted the barn a weird shade of red.

(186) They ran their sneakers a dingy shade of grey.

The resultative construction shows the following properties:

- The selection by the verb of the result AP/PP is semantically restricted, so that in a sentence like 188, the verb and the result adjective predicate seem to form a complex predicate. As can be seen from the oddness of 197, the not any adjective can appear as the result predicate.

  (196) #John *pounded* the dough *smooth*

- The post-verbal NP behaves like an argument, at least in the case of transitive resultatives [Carrier and Randall, 1992].[2] It receives the same theta-role and obeys the same selectional restrictions as the direct object in the non resultative counterpart. For example, middle formation and adjectival passive formation which can only apply to verbs that have a direct internal argument also apply to resultatives.

  – **Middle Formation:**

    (197) New seedlings water flat easily.

  – **Adjectival Passive Formation:**

    (198) The smashed-open safe.

Two different analyses follow from the research on resultatives. One analysis provides a treatment in which the entire predicate forms a small clause argument of the verb (e.g. [Hoekstra, 1988]). The second analysis gives a ternary branching structure to the resultatives in which both the post verbal NP and the result XP are sisters to the verb ([Carrier and Randall, 1992]).

The XTAG treatment of resultatives is essentially identical to the ternary analysis. The properties of the resultative construction seem to be best captured by the ternary analysis, which explains the argument-like behavior of the post-verbal NP and the semantic relation between the verb and result predicate. Furthermore, a small clause analysis with the post verbal NP as the small clause subject seems problematic in view of the fact that the thematic relationship of the verb with the post verbal NP in resultatives is the same as with the direct object in the non-resultatives.

There are four tree families in the XTAG grammar for representing the resultatives. The tree families (with their corresponding derived constructions) are selected by verb/adjective and verb/preposition pairs to capture the lexical restrictions that hold between the verb and the adjectival/prepositional predicate. The trees, therefore, are multi-anchored.[3] The four tree families are listed below.

- **TRnx0Vnx1Pnx2:** selected by transitive/intransitive verbs with prepositional result predicates.

- **TRnx0Vnx1A2:** selected by transitive/intransitive verbs with adjectival result predicates.

[2] The data is less decisive for the case of intransitive resultatives.

[3] Even though the meaning of the resultative construction may be constructed compositionally in many cases, the set of allowed verb/adjective and verb/preposition combinations is highly restricted.

- **TREnx1VA2:** selected by ergative verbs with adjectival result predicates.
- **TREnx1VPnx2:** selected by ergative verbs with prepositional result predicates.

Figures 13.1(a) and 13.1(b) show the tree selected by the transitive/intransitive verbs with adjectival result predicates and the tree selected by ergative verbs with prepositional result predicates, respectively.

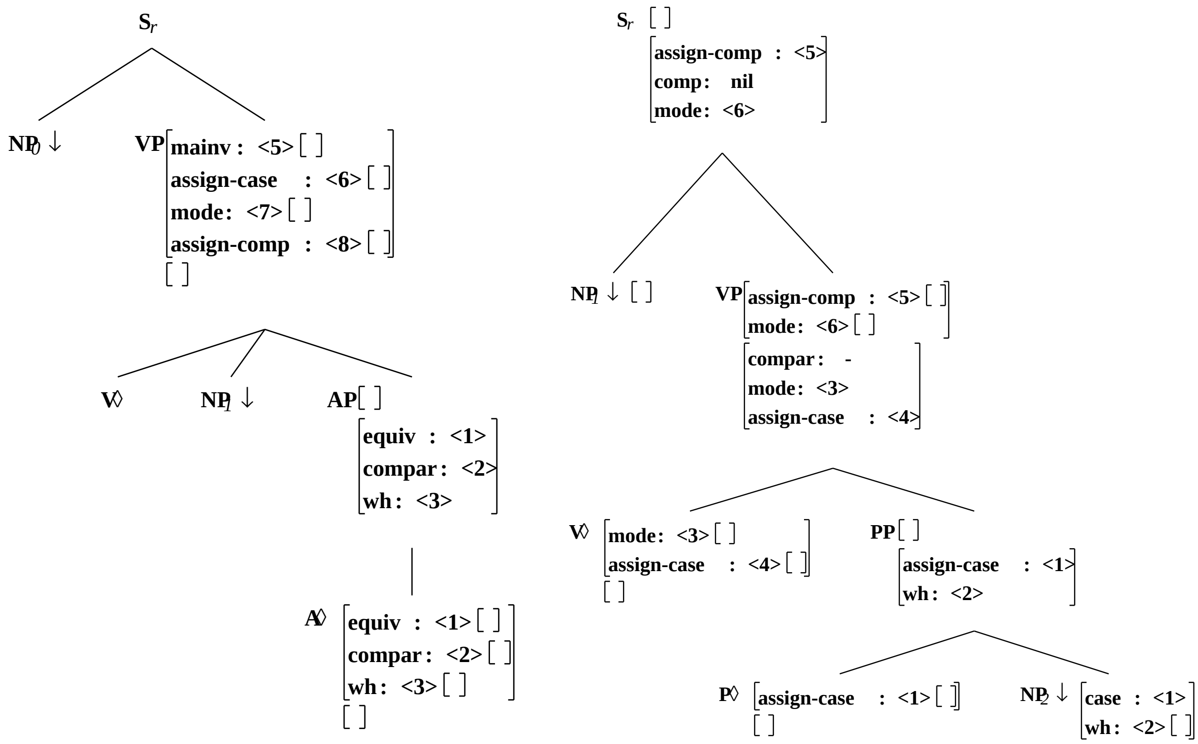


Figure 13.1: Resultative trees with adjectival result predicates, αRnx0Vnx1A2 (a) and prepositional result predicates, αREnx1VPnx2 (b)

Part III

Sentence Types

Chapter 14

Passives

In passive constructions such as (199), the subject NP is interpreted as having the same role as the direct object NP in the corresponding active declarative (200).

(199) **An airline buy-out bill** was approved by the House. (WSJ)

(200) The House approved **an airline buy-out bill**.

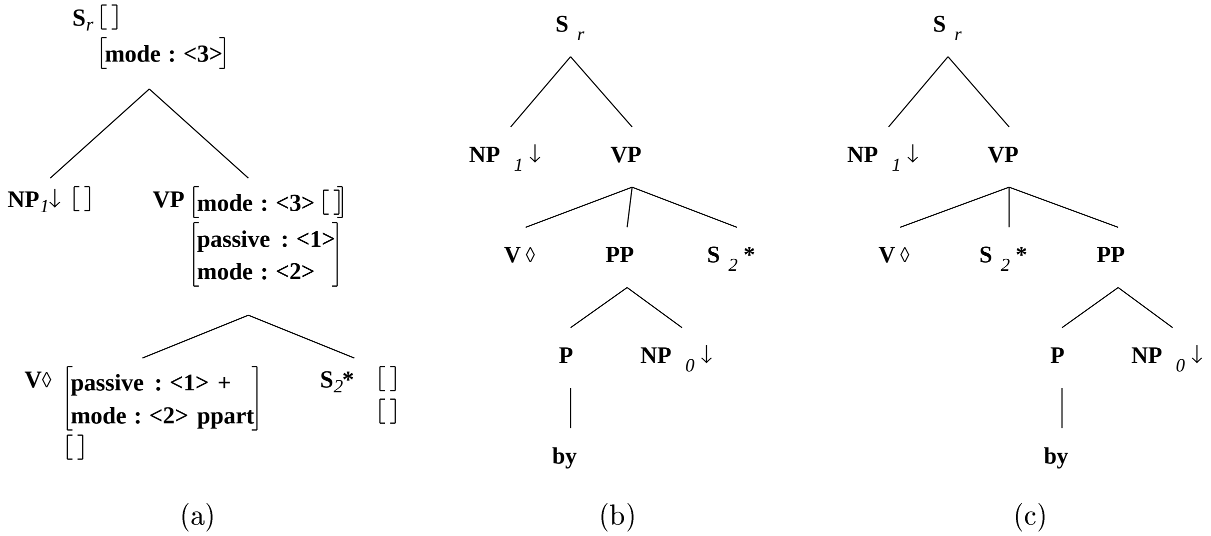


Figure 14.1: Passive trees in the Sentential Complement with NP tree family: βnx1Vs2 (a), βnx1Vbynx0s2 (b) and βnx1Vs2bynx0 (c)

In a movement analysis, the direct object is said to have moved to the subject position. The original declarative subject is either absent in the passive or is in a *by* headed PP (*by* phrase). In the English XTAG grammar, passive constructions are handled by having separate trees within the appropriate tree families. Passive trees are found in most tree families that have a direct object in the declarative tree (the light verb tree families, for instance, do not contain passive trees). Passive trees occur in pairs - one tree with the *by* phrase, and another without it. Variations in the location of the *by* phrase are possible if a subcategorization includes other arguments such as a PP or an indirect object. Additional trees are required for these variations. For example, the Sentential Complement with NP tree family has three passive trees, shown in Figure 14.1: one without the *by*-phrase (Figure 14.1(a)), one with the *by* phrase before the sentential complement (Figure 14.1(b)), and one with the *by* phrase after the sentential complement (Figure 14.1(c)).

Figure 14.1(a) also shows the feature restrictions imposed on the anchor[1]. Only verbs with **<mode>=ppart** (i.e. verbs with passive morphology) can anchor this tree. The **<mode>** feature is also responsible for requiring that passive *be* adjoin into the tree to create a matrix sentence. Since a requirement is imposed that all matrix sentences must have **<mode>=ind/imp**, an auxiliary verb that selects **<mode>=ppart** and **<passive>=+** (such as *was*) must adjoin (see Chapter 22 for more information on the auxiliary verb system).

[1]A reduced set of features are shown for readability.

Chapter 15

Extraction

The discussion in this chapter covers constructions that are analyzed as having wh-movement in GB, in particular, wh-questions and topicalization. Relative clauses, which could also be considered extractions, are discussed in Chapter 16.

Extraction involves a constituent appearing in a linear position to the left of the clause with which it is interpreted. One clause argument position is empty. For example, the position filled by *frisbee* in the declarative in sentence (201) is empty in sentence (202). The wh-item *what* in sentence (202) is of the same syntactic category as *frisbee* in sentence (201) and fills the same role with respect to the subcategorization.

(201) Clove caught a frisbee.

(202) What$_i$ did Clove catch ϵ_i?

The English XTAG grammar represents the connection between the extracted element and the empty position with co-indexing (as does GB). The <**trace**> feature is used to implement the co-indexing. In extraction trees in XTAG, the 'empty' position is filled with an ϵ. The extracted item always appears in these trees as a sister to the S$_r$ tree, with both dominated by a S$_q$ root node. The S$_r$ subtrees in extraction trees have the same structure as the declarative tree in the same tree family. The additional structure in extraction trees of the S$_q$ and NP nodes roughly corresponds to the CP and Spec of CP positions in GB.

All sentential trees with extracted components (this does not include relative clause trees) are marked <**extracted**>=+ at the top S node, while sentential trees with no extracted components are marked <**extracted**>=–. Items that take embedded sentences, such as nouns, verbs and some prepositions can place restrictions on whether the embedded sentence is allowed to be extracted or not. For instance, sentential subjects and sentential complements of nouns and prepositions are not allowed to be extracted, while certain verbs may allow extracted sentential complements and others may not (e.g. sentences (203)-(206)).

(203) The jury wondered [who killed Nicole].

(204) The jury wondered [who Simpson killed].

(205) The jury thought [Simpson killed Nicole].

(206) $*$The jury thought [who did Simpson kill]?

The <**extracted**> feature is also used to block embedded topicalization in infinitival complement clauses as exemplified in (207).

(207) * John wants [Bill$_i$ [PRO to see t$_i$]]

Verbs such as *want* that take non-*wh* infinitival complements specify that the <**extracted**> feature of their complement clause (i.e. of the foot S node) is $-$. Clauses that involve topicalization have $+$ as the value of their <**extracted**> feature (i.e. of the root S node). Sentences like (207) are thus ruled out.

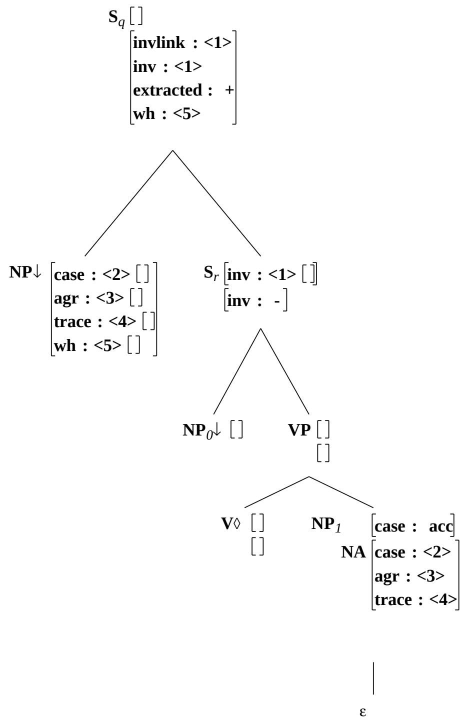


Figure 15.1: Transitive tree with object extraction: αW1nx0Vnx1

The tree that is used to derive the embedded sentence in (205) in the English XTAG grammar is shown in Figure 15.1[1]. The important features of extracted trees are:

- The subtree that has S$_r$ as its root is identical to the declarative tree or a non-extracted passive tree, except for having one NP position in the VP filled by ϵ.
- The root S node is S$_q$, which dominates NP and S$_r$.
- The <**trace**> feature of the ϵ filled NP is co-indexed with the <**trace**> feature of the NP daughter of S$_q$.

[1]Features not pertaining to this discussion have been taken out to improve readability.

- The <**case**> and <**agr**> features are passed from the empty NP to the extracted NP. This is particularly important for extractions from subject NP's, since <**case**> can continue to be assigned from the verb to the subject NP position, and from there be passed to the extracted NP.

- The <**inv**> feature of S_r is co-indexed to the <**wh**> feature of NP through the use of the <**invlink**> feature in order to force subject-auxiliary inversion where needed (see section 15.1 for more discussion of the <**inv**>/<**wh**> co-indexing and the use of these trees for topicalization).

15.1 Topicalization and the value of the <inv> feature

Our analysis of topicalization uses the same trees as wh-extraction. For any NP complement position a single tree is used for both wh-questions and for topicalization from that position. Wh-questions have subject-auxiliary inversion and topicalizations do not. This difference between the constructions is captured by equating the values of the S_r's <**inv**> feature and the extracted NP's <**wh**> feature. This means that if the extracted item is a wh-expression, as in wh-questions, the value of <**inv**> will be + and an inverted auxiliary will be forced to adjoin. If the extracted item is a non-wh, <**inv**> will be – and no auxiliary adjunction will occur. An additional complication is that inversion only occurs in matrix clauses, so the values of <**inv**> and <**wh**> should only be equated in matrix clauses and not in embedded clauses. In the English XTAG grammar, appropriate equating of the <**inv**> and <**wh**> features is accomplished using the <**invlink**> feature and a restriction imposed on the root S of a derivation. In particular, in extraction trees that are used for both wh-questions and topicalizations, the value of the <**inv**> feature for the top of the S_r node is co-indexed to the value of the <**inv**> feature on the bottom of the S_q node. On the bottom of the S_q node the <**inv**> feature is co-indexed to the <**invlink**> feature. The <**wh**> feature of the extracted NP node is co-indexed to the value of the <**wh**> feature on the bottom of S_q. The linking between the value of the S_q <**wh**> and the <**invlink**> features is imposed by a condition on the final root node of a derivation (i.e. the top S node of a matrix clause) requires that <**invlink**>=<**wh**>. For example, the tree in Figure 15.1 is used to derive both (208) and (209).

(208) John, I like.

(209) Who do you like?

For the question in (209), the extracted item *who* has the feature value <**wh**>=+, so the value of the <**inv**> feature on VP is also + and an auxiliary, in this case *do*, is forced to adjoin. For the topicalization (208) the values for *John*'s <**wh**> feature and for S_q's <**inv**> feature are both – and no auxiliary adjoins.

15.2 Extracted subjects

The extracted subject trees provide for sentences like (210)-(212), depending on the tree family with which it is associated.

(210) Who left?

(211) Who wrote the paper?

(212) Who was happy?

Wh-questions on subjects differ from other argument extractions in not having subject-auxiliary inversion. This means that in subject wh-questions the linear order of the constituents is the same as in declaratives so it is difficult to tell whether the subject has moved out of position or not (see [Heycock and Kroch, 1993] for arguments for and against moved subject).

The English XTAG treatment of subject extractions assumes the following:

- Syntactic subject topicalizations don't exist; and
- Subjects in wh-questions are extracted rather than in situ.

The assumption that there is no syntactic subject topicalization is reasonable in English since there is no convincing syntactic evidence and since the interpretability of subjects as topics seems to be mainly affected by discourse and intonational factors rather than syntactic structure. As for the assumption that wh-question subjects are extracted, these questions seem to have more similarities to other extractions than to the two cases in English that have been considered in situ wh: multiple wh questions and echo questions. In multiple wh questions such as sentence (213), one of the wh-items is blocked from moving sentence initially because the first wh-item already occupies the location to which it would move.

(213) Who ate what?

This type of 'blocking' account is not applicable to subject wh-questions because there is no obvious candidate to do the blocking. Similarity between subject wh-questions and echo questions is also lacking. At least one account of echo questions ([Hockey, 1994]) argues that echo questions are not ordinary wh-questions at all, but rather focus constructions in which the wh-item is the focus. Clearly, this is not applicable to subject wh-questions. So it seems that treating subject wh-questions similarly to other wh-extractions is more justified than an in situ treatment.

Given these assumptions, there must be separate trees in each tree family for subject extractions. The declarative tree cannot be used even though the linear order is the same because the structure is different. Since topicalizations are not allowed, the <**wh**> feature for the extracted NP node is set in these trees to +. The lack of subject-auxiliary inversion is handled by the absence of the <**invlink**> feature. Without the presence of this feature, the <**wh**> and <**inv**> are never linked, so inversion can not occur. Like other wh-extractions, the S_q node is marked <**extracted**>=+ to constrain the occurrence of these trees in embedded sentences. The tree in Figure 15.2 is an example of a subject wh-question tree.

15.3 Wh-moved NP complement

Wh-questions can be formed on every NP object or indirect object that appears in the declarative tree or in the passive trees, as seen in sentences (214)-(219). A tree family will contain

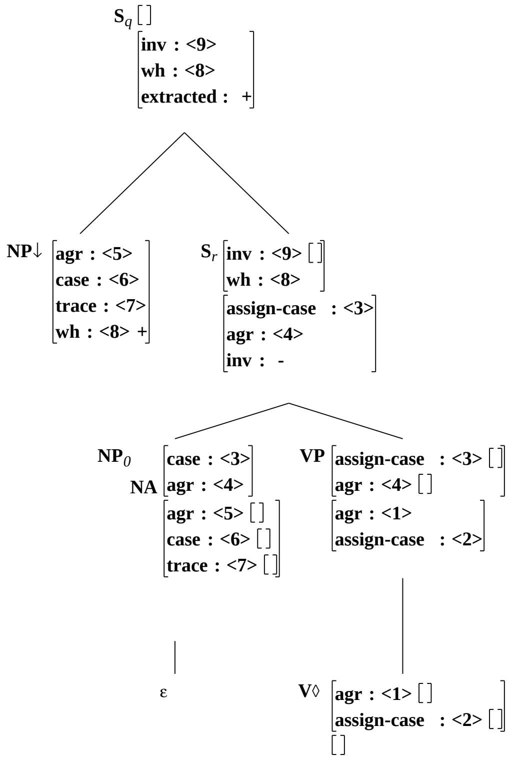


Figure 15.2: Intransitive tree with subject extraction: αW0nx0V

one tree for each of these possible NP complement positions. Figure 15.3 shows the two extraction trees from the ditransitive tree family for the extraction of the direct (Figure 15.3(a)) and indirect object (Figure 15.3(b)).

(214) Dania asked Beth a question.

(215) Who$_i$ did Dania ask ϵ_i a question?

(216) What$_i$ did Dania ask Beth ϵ_i?

(217) Beth was asked a question by Dania.

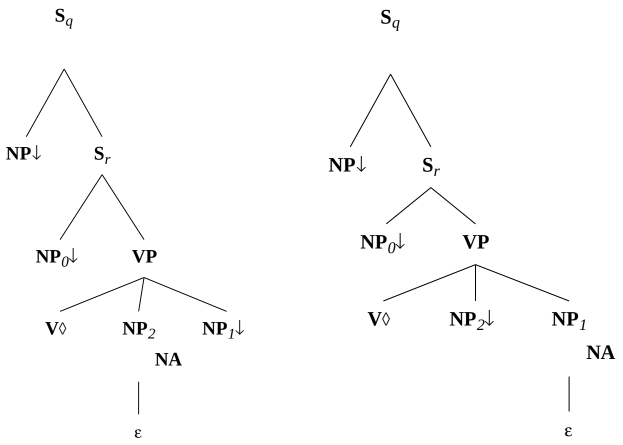
(218) Who$_i$ was Beth asked a question by ϵ_i??

(219) What$_i$ was Beth asked ϵ_i? by Dania?

15.4 Wh-moved object of a P

Wh-questions can be formed on the NP object of a complement PP as in sentence (220).

(220) [Which dog]$_i$ did Beth Ann give a bone to ϵ_i?



(a) (b)

Figure 15.3: Ditransitive trees with direct object: αW2nx0Vnx2nx1 (a) and indirect object extraction: αW1nx0Vnx2nx1 (b)

The *by* phrases of passives behave like complements and can undergo the same type of extraction, as in (221).

(221) [Which dog]$_i$ was the frisbee caught by ϵ_i?

Tree structures for this type of sentence are very similar to those for the wh-extraction of NP complements discussed in section 15.3 and have the identical important features related to tree structure and trace and inversion features. The tree in Figure 15.4 is an example of this type of tree. Topicalization of NP objects of prepositions is handled the same way as topicalization of complement NP's.

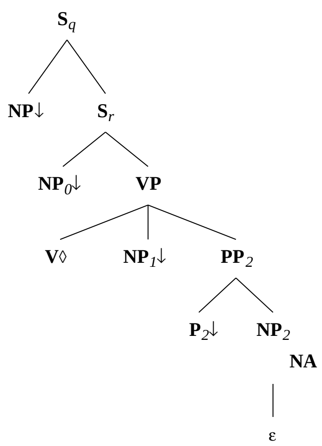


Figure 15.4: Ditransitive with PP tree with the object of the PP extracted: αW2nx0Vnx1pnx2

15.5 Wh-moved PP

Like NP complements, PP complements can be extracted to form wh-questions, as in sentence (222).

(222) [To which dog]$_i$ did Beth Ann throw the frisbee ϵ_i?

As can be seen in the tree in Figure 15.5, extraction of PP complements is very similar to extraction of NP complements from the same positions.

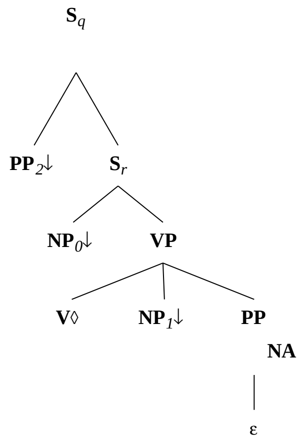


Figure 15.5: Ditransitive with PP with PP extraction tree: αpW2nx0Vnx1pnx2

The PP extraction trees differ from NP extraction trees in having a PP rather than an NP left daughter node under S$_q$ and in having the ϵ fill a PP rather than an NP position in the VP. In other respects these PP extraction structures behave like the NP extractions, including being used for topicalization.

15.6 Wh-moved S complement

Except for the node label on the extracted position, the trees for wh-questions on S complements look exactly like the trees for wh-questions on NP's in the same positions. This is because there is no separate wh-lexical item for clauses in English, so the item *what* is ambiguous between representing a clause or an NP. To illustrate this ambiguity notice that the question in (223) could be answered by either a clause as in (224) or an NP as in (225). The extracted NP in these trees is constrained to be <**wh**>=+, since sentential complements can not be topicalized.

(223) What does Clove want?

(224) for Beth Ann to play frisbee with her

(225) a biscuit

15.7 Wh-moved Adjective complement

In subcategorizations that select an adjective complement, that complement can be questioned in a wh-question, as in sentence (226).

(226) How$_i$ did he feel ϵ_i?

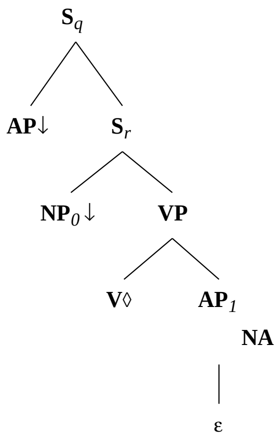


Figure 15.6: Predicative Adjective tree with extracted adjective: αWA1nx0Vax1

The tree families with adjective complements include trees for such adjective extractions that are very similar to the wh-extraction trees for other categories of complements. The adjective position in the VP is filled by an ϵ and the trace feature of the adjective complement and of the adjective daughter of S$_q$ are co-indexed. The extracted adjective is required to be **<wh>**=+[2], so no topicalizations are allowed. An example of this type of tree is shown in Figure 15.6.

[2] *How* is the only **<wh>**=+ adjective currently in the XTAG English grammar.

Chapter 16

Relative Clauses

Relative clauses are NP modifiers involving extraction of an argument or an adjunct. The NP head (the portion of the NP being modified by the relative clause) is not directly related to the extracted element. For example in 227, *the person* is the head NP and is modified by the relative clause *whose mother ϵ likes Chris.* However, *the person* is not interpreted as the subject of the relative clause which is missing an overt subject. In other cases, such as 228, the relationship between the head NP *export exhibitions* may seem to be more direct but even here we assume that there are two independent relationships: one between the entire relative clause and the (head) NP it modifies, and another between the extracted element and its trace inside the relative clause. The extracted element can further be overt, as in 227, or covert, as in 228.

(227) [[the person$_i$] [whose$_i$ mother$_j$ [ϵ_j likes Chris]]]

(228) [[export exhibitions$_i$] [ϵ_i that [ϵ_i included high-tech items]]]

The XTAG analysis of relative clauses is essentially identical to the GB analysis. We represent relative clauses as auxiliary trees that adjoin to NP's. These trees are anchored by the verb of the relative clause and appear in the appropriate tree families representing various verb subcategorizations. Each family has a group of relative clause trees based on the declarative tree and each passive tree in that family. Furthermore, within each of these groups, there is a separate relative clause tree corresponding to each argument that can be extracted from the clause. The relationship between the relative clause and the head NP is not represented directly. Rather, it is treated as a semantic relationship which could be provided by any reasonable compositional theory. The relationship between the extracted element, NP$_w$ (which can be covert) and the position from which it was extracted is captured by co-indexing the <**trace**> features of the two positions/nodes. If for example, it is **NP**$_1$ that is extracted (object extraction), we have the following feature equations (see Figure 16.1(a)):

- $\mathbf{NP}_w\mathbf{.t:}\langle$ **trace** $\rangle =\mathbf{NP}_1\mathbf{.t:}\langle$ **trace** $\rangle$
- $\mathbf{NP}_w\mathbf{.t:}\langle$ **case** $\rangle =\mathbf{NP}_1\mathbf{.t:}\langle$ **case** $\rangle$
- $\mathbf{NP}_w\mathbf{.t:}\langle$ **agr** $\rangle =\mathbf{NP}_1\mathbf{.t:}\langle$ **agr** $\rangle$

One aspect of the implementation of relative clauses is to allow a covert + <**wh**> NP and/or a covert COMP. For example, 229 has a covert + <**wh**> NP and overt COMP, 230 has a covert COMP and overt + <**wh**>, and 231 has both a covert + <**wh**> NP and a covert COMP.

(229) export exhibitions [[$_{NP_w}$ ϵ]$_i$ that [ϵ_i included high-tech items]]

(230) the export exhibition [which$_i$ [Muriel planned ϵ_i]]

(231) the export exhibition [[$_{NP_w}$ ϵ]$_i$ [Muriel planned ϵ_i]]

Consequently, our treatment of relative clauses has different trees to handle relative clauses with an overt extracted *wh*-NP (Section 16.1) and relative clauses with a covert extracted *wh*-NP (Section 16.2). Covert and overt COMP's are handled by adjunction with the already existing auxiliary trees for complementizers, that are used for sentential complementation (see Chapter 8).

16.1 Relative Clauses with overt extracted *wh*-phrases

Relative clauses with an overt extracted *wh*-NP (Figure 16.1(a)) involve substitution of a +<**wh**> NP into the (extracted) NP$_w$ node. The feature equation **NP$_w$.t:<wh>**= + allows only *wh*-phrases to substitute into this node, such as *whose mother*, *who*, *whom*, *which* (but not *when* and *where*, which are treated as exhaustive +*wh* PPs (Figure 16.1(b))).

Complementizers can never cooccur with the overt extracted + <**wh**> NP (cf. **I saw the man who$_i$ that Muriel saw* ϵ_i). Consequently, the auxiliary βCOMPs trees are prevented from adjoining at the **S**$_r$ node in these trees by the equation

- **S$_r$.t:$\langle$comp$\rangle$ = nil**

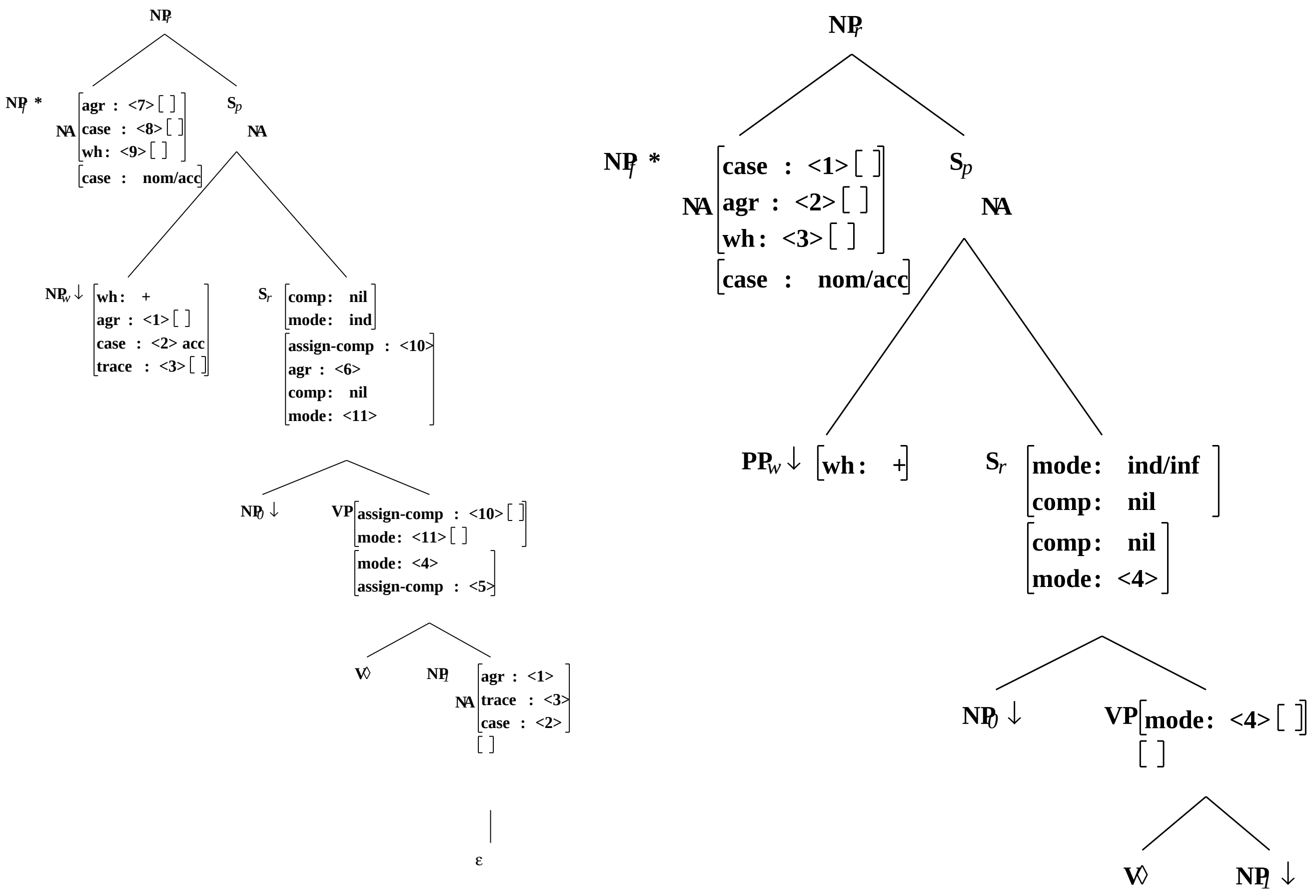
in the relative clause tree, which will always fail to unify with the (non-**nil**) values of the <**comp**> feature in the βCOMPs trees (see Figure 16.3). Examples 232 and 233 are examples for which the tree in Figure 16.1(a) is used. Cases of PP pied-piping, as in 234, are handled in a similar fashion by building in a PP$_w$ substitution node (Figure 16.1(b)).[1]

(232) the man who Muriel likes

(233) the man whose mother Muriel likes

(234) the bowl in which Miriam ate her cereal

[1]Adjunct traces are not represented in the XTAG analysis of adjunct extraction. Since relative clauses on adjuncts also do not have traces, feature equations showing the trace coindexation are not present in such trees. See Section 16.4.4 for more discussion of adjunct relative clauses.



(a) (b)

Figure 16.1: Relative clause trees with overt *wh*-phrases in the transitive tree family: (a) object extraction tree βN1nx0Vnx1, (*the man who Muriel loved*), and (b) adjunct relative clause tree with PP pied-piping βNpxnx0Vnx1, *the bowl in which Miranda ate her cereal.*

16.1.1 Contraints on the mode of the relative clause

Relative clause trees that have $\mathbf{NP}_w$ as a substitution node have the feature equation given below. The examples in 236–240 provide the rationale for this feature setting.

- $\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ind**

(235) the man [[whose wife]$_i$ [ϵ_i cooked this meal]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ind**)

(236) *the girl [[who]$_i$ [ϵ_i to win the prize]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**inf**)

(237) *the candidate [[who]$_i$ [ϵ_i defeated in the elections]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ppart**)

(238) *the boy [[whose dog]$_i$ [ϵ_i chasing the cat]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ = **ger**)

(239) the boy [[whose mother]$_i$ [Bill believes [ϵ_i to be beautiful]]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ind**)

Trees with a $\mathbf{PP}_w$ substitution node have the feature equation given below with the rationale provided by examples in 241–244.[2]

- $\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ = **ind/inf**

(240) the person [[by whom]$_i$ [this machine was invented ϵ_i]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ind**)

(241) a company [[in which]$_i$ [to invest ϵ_i]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**inf**)

(242) *the fork [[with which]$_i$ (Geoffrey) eaten the pudding ϵ_i]] ($\mathbf{S}_r$**.t:**$\langle$ **mode**$\rangle$ =**ppart**)

(243) *the person [[by whom]$_i$ [(this machine) inventing ϵ_i]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ger**)

16.2 Relative Clauses with Covert Extracted *wh*-NP

Relative clauses with a covert extracted *wh*-NP (Figure 16.2) have a NP_w node headed by ϵ_w, which is built into the trees. Complementizers can adjoin in at the $\mathbf{S}_r$ node in a manner parallel to sentential complementation (see Chapter 8). The examples in 244-247 are handled by this tree.

(244) the cake that Muriel said Steven ate

(245) the book for Miranda to read

(246) the man looking at Muriel

(247) the librarian to check out the book

There are two aspects to the implementation of these trees. Firstly, in a manner parallel to the relative clause trees with overt NP_w, there are constraints on the mode of the relative clause for these trees, which is realized with the <**mode**> feature (Section 16.2.1). Secondly, there are cooccurrence constraints between the mode of the relative clause and the complementizers that can adjoin in – these are realized with the <**assign-comp**> and <**comp**> features (Section 16.2.2). The implementation of the cooccurrence constraints is entirely parallel to sentential complementation, except in one respect: the occurrence of the null COMP (which in the case of the relative clauses, is represented by disallowing the adjunction of any COMP – namely, the βCOMPs auxiliary tree – altogether) is subject to further constraints, which we realize with the <**nocomp-mode**> feature (Section 16.2.3).

16.2.1 Constraints on the mode of the relative clause

The mode of relative clause varies depending on which argument has been extracted. For example, subject extraction can occur only in the indicative, infinitive, and gerundive modes, as can be seen from examples like 248–251. Object extraction can only occur in the indicative and infinitive modes, as shown in examples 252–255. This restriction is implemented by setting the $\mathbf{S}_r$**.t:**<**mode**> feature to the appropriate values, such as **ind**, **inf**, **ger**, etc.. Figure 16.2 shows this restriction implemented for the relative clause tree with subject extraction in the transitive tree family.

[2]As is the case for $\mathbf{NP}_w$ substitution, any +<**wh**>PP can substitute under PP_w. This is implemented by the following equation: $\mathbf{PP}_w$**.t:**$\langle$**wh**$\rangle$ = +.

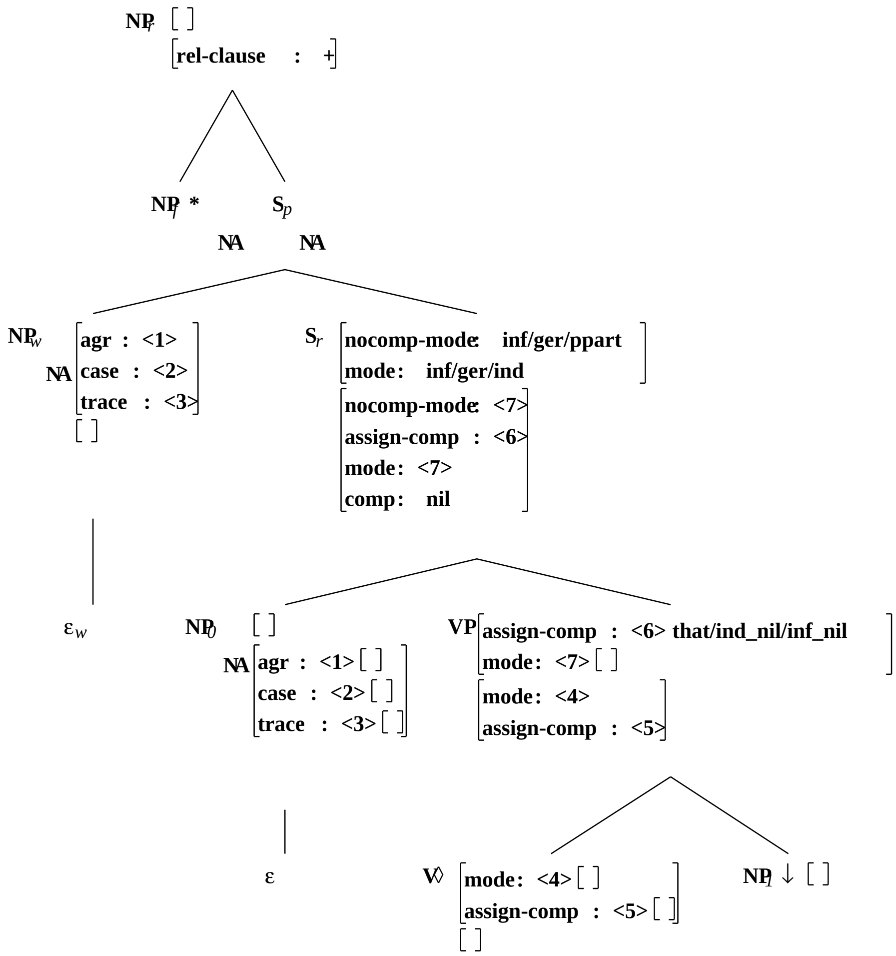


Figure 16.2: Subject extraction tree in the transitive tree family with non-overt *wh*-phrase, βNc0nx0Vnx1

(248) the dog [ϵ_w that [ϵ ate the cake]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ind**)

(249) the team [ϵ_w [ϵ to win the gold medal]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**inf**)

(250) the girl [ϵ_w [ϵ reading the book]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ger**)

(251) *the woman [ϵ_w [ϵ seen the sight]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ppart**)

(252) the toy [ϵ_w that [Miranda likes ϵ]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ind**)

(253) the guava [ϵ_w for [Susan to eat ϵ]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**inf**)

(254) *the toy [ϵ_w [Laura liking ϵ]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ger**)

(255) *the book [ϵ_w [Danny torn ϵ]] ($\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ppart**)

The full set of mode restrictions on the different relative clause trees is as follows:

- For all non-passive cases of subject extraction, $\mathbf{S}_r$**.t:**⟨**mode**⟩ = **ind/ger/inf** (see 256–259):

  (256) the girl [ϵ_w [ϵ reading the magazine]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ger**)

  (257) the cowboy [ϵ_w [ϵ to win the fight]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**inf**)

  (258) the man [ϵ_w that [ϵ loaded the gun]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ind**)

  (259) *the child [ϵ_w (that/for) [ϵ eaten the cake]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ppart**)

- For all passive cases of subject extraction, $\mathbf{S}_r$**.t:**⟨**mode**⟩ = **ind/ger/ppart/inf** (see 260–263):

  (260) the toy [ϵ_w that [ϵ was broken by the child]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ind**)

  (261) the man [ϵ_w [ϵ being arrested by the officer]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ger**)

  (262) the food [ϵ_w [ϵ to be served during dinner]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**inf**)

  (263) the candidates [ϵ_w [ϵ elected by the people]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ppart**)

- Finally, for all cases of non-subject extraction, $\mathbf{S}_r$**.t:**⟨**mode**⟩ = **ind/inf** (see 264–267):

  (264) the book [ϵ_w [John will read ϵ]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ind**)

  (265) the candidate [ϵ_w for [people to tear ϵ]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**inf**)

  (266) *the ring [ϵ_w (that/for) [Miranda tearing ϵ]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ger**)

  (267) *the table [ϵ_w (that/for) [Danny broken ϵ]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**ppart**)

Relative clause formation with **<mode>=nom/prep** (for adjectival, nominal and prepositional predicates) are also allowed, but only with a covert $_{NP_w}$ and an covert COMP. Furthermore, they can be formed only on the subject of the clause. Some families that have these additional modes are Tnx0APnx1 268, Tnx0ARBPnx1 269, Tnx0nx1ARB 270.

(268) the accused [ϵ_w [ϵ void of all hope]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**prep**)

(269) the dog [ϵ_w [ϵ next to the tree]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**prep**)

(270) the road [ϵ_w [ϵ seven miles away]] ($\mathbf{S}_r$**.t:**⟨**mode**⟩ =**nom**)

16.2.2 Complementizer Selection

The **VP.t:<assign-comp>** feature in the relative clause is assigned values which represent constraints on COMP selection by the highest verb in the clause. The feature values are passed up to the $\mathbf{S}_r$ node of the relative clause by the equation,

- $\mathbf{S}_r$**.b:⟨assign-comp⟩ =VP.t:⟨assign-comp⟩**

This ensures proper selection of the appropriate COMP since the auxiliary tree anchored by each complementizer also has the **<assign-comp>** feature with a value appropriate to the particular complementizer in question (see for example the βCOMPs anchored by *that* in Figure 16.3). Adjunction of any complementizer can therefore succeed only if the **<assign-comp>** features in the COMP tree and the relative clause tree have the same value.

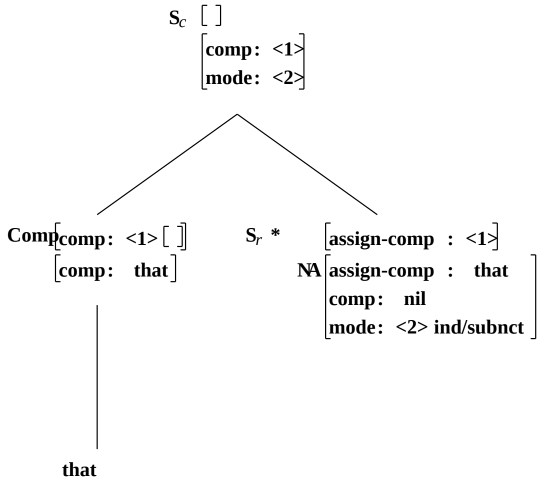


Figure 16.3: Tree βCOMPs, anchored by *that*

So, while the subject extraction tree in Figure 16.2 allows *that* to adjoin, it prevents *for* from adjoining because the $\mathbf{S}_r$**.b:<assign-comp>=for** equation in the βCOMPs tree anchored by *for* will fail to unify with the **VP.t:<assign-comp>= that/ind_nil/inf_nil/ecm** equation, which is coindexed with the $\mathbf{S}_r$**.b:<assign-comp>** feature in the relative clause tree.

16.2.3 Further Constraints on the Null COMP

In our analysis, the *null* complementizer is not represented in the structure of the relative clause at all – realization of the null COMP implies preventing any COMP from adjoining. However, this requires an additional set of constraints, both for distributional and implementational reasons. For example, the null COMP is not permitted in cases of subject extraction with **<mode>=ind** unless there is an intervening clause. The evidence can be seen in 271-274, especially in the contrast between 271 and 272.

(271) *the girl [ϵ_w [ϵ likes Danny]]] (**<mode>=ind**)

(272) the man [ϵ_w [Mary said [ϵ likes Dafna]]] (<**mode**>=**ind**)

(273) the boy [ϵ_w [ϵ eating the guava]] (<**mode**>=**ger**)

(274) the man [ϵ_w [ϵ_i defeated by the cowboy]]] (<**mode**>=**ppart**)

(275) the boy [ϵ_w [ϵ_i to win the gold medal]]] (<**mode**>=**inf**)

(276) the snake [ϵ_w [ϵ_i next to the tree]]] (<**mode**>=**prep**)

(277) the town [ϵ_w [ϵ_i seven miles away]]] (<**mode**>=**nom**)

To model this paradigm, the feature ⟨**nocomp-mode**⟩ is used in conjunction with the following equations.[3]

- **S**$_r$**.t:**⟨**nocomp-mode**⟩ = **inf/ger/ppart** (in relative clause trees with subject extraction)
- **S**$_r$**.b:**⟨**nocomp-mode**⟩ = **S**$_r$**.b:**⟨**mode**⟩

Given the two equations above, successful unification of the **S**$_r$**.t:**<**nocomp-mode**> and **S**$_r$**.b:**<**nocomp-mode**> features implies realization of the null COMP, which, in the subject extracted relative clauses (see Figure 16.2), is possible only if the relative clause is in the **inf**, **ger**, or **ppart** mode (see examples above). Since the *that* βCOMPs tree selects a clause in the indicative mode (See Figure 16.3), *that* will never be able to adjoin to a relative clause with the subject extracted. However, if a clause adjoins first to the relative clause, as would be the case in 272, this adjunction puts the **S**$_r$**.t:**<**nocomp-mode**> and **S**$_r$**.b:**<**nocomp-mode**> in different nodes, thus preventing a feature clash between the mode of the relative clause and the values of the <nocomp-mode> feature that are specified in the subject extracted relative clause tree.

The above feature equations also permit the mode of the relative clause to be **ind** just in case there is an intervening clause, as in 272. Adjunction of the clause puts the **S**$_r$**.t:**<**nocomp-mode**> and **S**$_r$**.b:**<**nocomp-mode**> in different nodes, thus preventing a unification failure. Note, however, that the feature mismatch induced by the above equations is not remedied by adjunction of just any S-adjunct since all other S-adjuncts are transparent to the ⟨**nocomp-mode**⟩ feature because of the following equation,

- **S**$_m$**.b:**⟨**nocomp-mode**⟩ = **S**$_f$**.t:**⟨**nocomp-mode**⟩

where **S**$_f$**.t** is in the foot node of the adjoining adjunct.

The obligatory adjunction of complementizers implemented above for subject extracted relative clauses (in the indicative <**mode**>) contrasts with what we do with COMP adjunction in subject extracted questions, where we disallow COMP from adjoining to the embedded S (**Who did Miranda say that likes Zed?*). We are thus able to capture the facts related to *that-trace* constraints in English.[4]

[3]The **S**$_r$**.t:**⟨**nocomp-mode**⟩ value given here appears in the relative clause trees with subject extraction. Trees with other constituents extracted will have different values for this feature. For example, in object extraction trees, this feature has the value **ind**.

[4]See Chapter 8 for a more detailed discussion related to *that-trace* constraints.

16.3 External syntax

A relative clause can combine with the NP it modifies in at least the following two ways:

(278) [the [toy [ϵ_w [Dafna likes ϵ_i]]]]

(279) [[the toy] [ϵ_w [Dafna likes ϵ_i]]]

Based on cases like 280 and 281, which are problematic for the structure in 278, the structure in 279 is adopted.

(280) [[the man and the woman] [who met on the bus]]

(281) [[the man and the woman] [who like each other]]

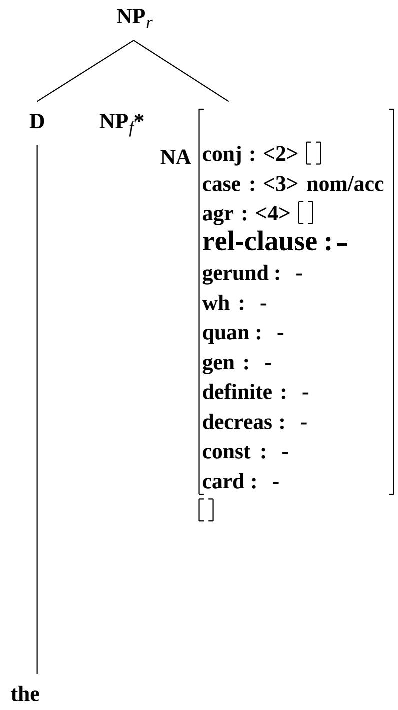


Figure 16.4: Determiner tree with <**rel-clause**> feature: βDnx

As it stands, the relative clause analysis sketched so far will combine in two ways with the Determiner tree shown in Figure (16.4),[5] giving us both the possiblities shown in 278 and 279. In order to block the structure exemplified in 278, the feature $\langle$**rel-clause**$\rangle$ is used in combination with the following equations.

[5]The determiner tree shown has the <**rel-clause**> feature built in. The relative clause analysis would give two parses in the absence of this feature.

- **NP**$_r$**.b:⟨rel-clause⟩** = + (on the Relative Clause)
- **NP**$_f$**.t:⟨rel-clause⟩** = − (on the Determiner tree)

Together, these equations block introduction of the determiner above the relative clause.

16.4 Other Issues

16.4.1 Reduced Relatives

The analysis presented above accounts for reduced relatives (which are commonly treated as derived from relative clauses through deletion of the relative pronoun and if there is a *be*, then deletion of that also). Reduced relatives are permitted only in cases of subject-extraction. Past participial reduced relatives are only permitted on passive clauses. See 282-289.

(282) the gypsy [ϵ_w [ϵ playing the banjo]]]

(283) *the instrument [ϵ_w [Amis playing ϵ]]]

(284) *the day [ϵ_w [Amis playing the banjo]]]

(285) the apple [ϵ_w [ϵ eaten by Dafna]]]

(286) *the child [ϵ_w [the apple eaten by ϵ]]]

(287) *the day [ϵ_w [Amis eaten the apple]]]

(288) *the apple [ϵ_w [Dafna eaten ϵ]]]

(289) *the child [ϵ_w [ϵ eaten the apple]]]

These restrictions are built into the <**mode**> specifications of **S**$_r$**.t**, as explained in Section 16.1.1.

16.4.1.1 Restrictive vs. Non-restrictive relatives

The English XTAG grammar does not contain any syntactic distinction between restrictive and non-restrictive relatives because we believe this to be a semantic and/or pragmatic difference.

16.4.2 Stacking of Complementizers

Complementizers are prevented from stacking, as in example 290, just as in sentential complementation.

(290) *the book [ϵ_w [that [that [Muriel wrote ϵ]]]]

16.4.3 Adjunction on PRO

Adjunction on PRO, which would yield the ungrammatical 291 is blocked.

(291) *I want [[PRO [who Muriel likes] to read a book]].

This is done by setting the <**case**> feature of $\mathbf{NP}_f$ (foot node of the relative clause treee) to be **nom/acc**. The <**case**> feature of PRO is **none**. This leads to a feature clash and blocks adjunction of relative clauses onto PRO.

16.4.4 Adjunct relative clauses

Two types of trees to handle adjunct relative clauses exist in the XTAG grammar: one in which there is $\mathbf{PP}_w$ substitution and one in which there is a null $\mathbf{NP}_w$ built in and a **COMP** adjoins in. There is no $\mathbf{NP}_w$ substitution tree. This is because of the contrast between 292 and 293.

(292) the horse [[on whose back] [Muriel rode away]]]

(293) *the horse [[whose back] [Muriel rode away]]]

In general, adjunct relatives are not possible with an overt $\mathbf{NP}_w$. We do not consider 294 and 295 to be counterexamples to the above statements because we consider *where* and *when* to be exhaustive **PP**s that head a **PP** initial tree.

(294) the place [where [Muriel wrote her first book]]]

(295) the time [when [Muriel lived in Bryn Mawr]]]

16.4.5 ECM

Cases where *for* assigns exceptional case (cf. 296, 297) are handled, again parallel to the way ECM is done in sentential complementation.

(296) a book [ϵ_w [for [him to read ϵ]]]

(297) the time [ϵ_w [for [her to leave Haverford]]]

The assignment of case by *for* is implemented by a combination of the following equations:

- $\mathbf{S}_r$**.b:⟨assign-case⟩=acc** (in the *for* βCOMPs tree)
- $\mathbf{S}_r$**.b:⟨assign-case⟩** = $\mathbf{NP}_0$**.t:⟨case⟩** (in the Relative clause tree)

16.5 Cases not handled

16.5.1 Partial treatment of free-relatives

Free relatives are only partially handled. All free relatives in non-subject positions and some free relatives on subject positions are handled. The structure assigned to free relatives treats the extracted *wh*-NP as the head NP of the relative clause. The remaining relative clause modifies this extracted *wh*-NP (cf. 298-300).

(298) what(ever) [ϵ_{w_i} [Mary likes ϵ_i]]]

(299) where(ever) [ϵ_w [Mary lives]]]

(300) who(ever) [ϵ_{w_i} [Muriel thinks [ϵ_i likes Mary]]]]

However, simple subject extractions without further embedding are not handled (cf. 301).

(301) who(ever) [ϵ_{w_i} [ϵ_i likes Bill]]]

This is because 301 is treated exactly like the ungrammatical 302.

(302) *the person [ϵ_{w_i} [ϵ_i likes Bill]]]

16.5.2 Adjunct P-stranding

The following cases of adjunct preposition stranding are not handled (cf. 303, 304).

(303) the pen Muriel wrote this letter with

(304) the street Muriel lives on

Adjuncts are not built into elementary trees in XTAG. So there is no clean way to represent adjunct preposition stranding. A better solution might, probably, be available if we make use of multi-component adjunction.

16.5.3 Overgeneration

The following types of ungrammatical examples are currently accepted by the XTAG grammar. This is because no clean and conceptually attractive way of ruling them out is obvious to us yet.

16.5.3.1 *how* as *wh*-NP

In standard American English, *how* is not acceptable as a relative pronoun (cf. 305).

(305) *the way [how [PRO to solve this problem]]]

However, 305 is accepted by the current grammar. The only way to rule 305 out would be to introduce a special feature devoted to this purpose. This is unappealing. Further, there exist speech registers/dialects of English, where 305 is acceptable.

16.5.3.2 Internal head constraint

Relative clauses in English (and in an overwhelming number of languages) obey a 'no internal head' constraint. This constraint is exemplified in the contrast between 306 and 307.

(306) the person [who$_i$ Muriel likes ϵ_i]]

(307) *the person [[which person]$_i$ Muriel likes ϵ_i]]

We know of no good way to rule 307 out, while still ruling 308 in.

(308) the person [[whose mother]$_i$ Muriel likes ϵ_i]]

Dayal (1996) suggests that 'full' NPs such as *which person* and *whose mother* are R-expressions while *who* and *whose* are pronouns. R-expressions, unlike pronouns, are subject to Condition C. 306 is, then, ruled out as a violation of Condition C since *the person* and *which person* are co-indexed and *the person* c-commands *which person*. If we accept Dayal's argument, we have a principled reason for allowing overgeneration of relative clauses that violate the internal head constraint, the reason being that the XTAG grammar does generate binding theory violations.

16.5.3.3 Overt COMP constraint on stacked relatives

Stacked relatives of the kind in 309 are handled.

(309) [[the book [that Bill likes]] [which Mary wrote]]

However, there is a constraint on stacked relatives: all but the relative clause closest to the head-NP must have either an overt **COMP** or an overt $\mathbf{NP}_w$. Thus 310 is ungrammatical.

(310) *[[the book [that Bill likes]] [Mary wrote]]

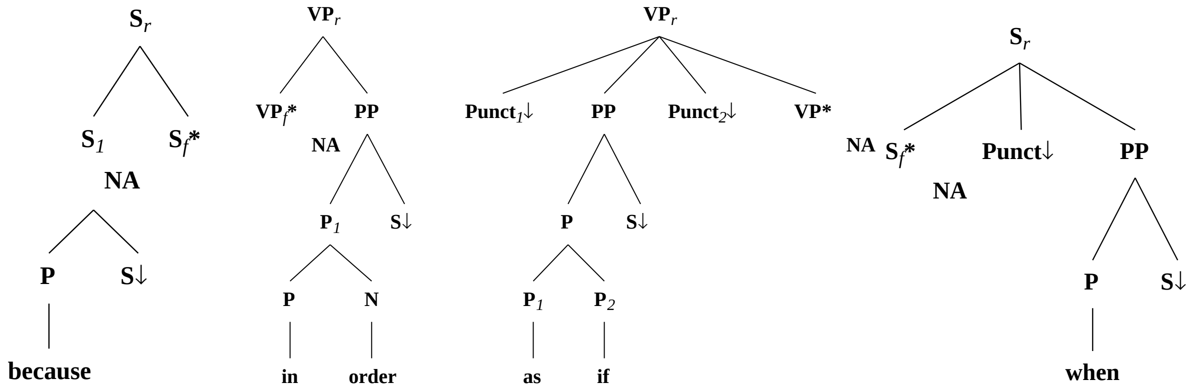
We currently know of no good way of handling this constraint, and 310 is incorrectly accepted by XTAG.

Chapter 17

Adjunct Clauses

Adjunct clauses include subordinate clauses (i.e. those with overt subordinating conjunctions), purpose clauses and participial adjuncts.

Subordinating conjunctions each select four trees, allowing them to appear in four different positions relative to the matrix clause. The positions are (1) before the matrix clause, (2) after the matrix clause, (3) before the VP, surrounded by two punctuation marks, and (4) after the matrix clause, separated by a punctuation mark. Each of these trees is shown in Figure 17.1.



(1) βPss (2) βvxPNs (3) βpuPPspuvx (4) βspuPs

Figure 17.1: Auxiliary Trees for Subordinating Conjunctions

Sentence-initial adjuncts adjoin at the root S of the matrix clause, while sentence-final adjuncts adjoin at a VP node. In this, the XTAG analysis follows the findings on the attachment sites of adjunct clauses for conditional clauses ([Iatridou, 1991]) and for infinitival clauses ([Browning, 1987]). One compelling argument is based on Binding Condition C effects. As can be seen from examples (311)-(313) below, no Binding Condition violation occurs when the adjunct is sentence initial, but the subject of the matrix clause clearly governs the adjunct clause when it is in sentence final position and co-indexation of the pronoun with the subject of the adjunct clause is impossible.

(311) Unless she$_i$ hurries, Mary$_i$ will be late for the meeting.

(312) *She$_i$ will be late for the meeting unless Mary$_i$ hurries.

(313) Mary$_i$ will be late for the meeting unless she$_i$ hurries.

We had previously treated subordinating conjunctions as a subclass of *conjunction*, but are now assigning them the POS *preposition*, as there is such clear overlap between words that function as prepositions (taking NP complements) and subordinating conjunctions (taking clausal complements). While there are some prepositions which only take NP complements and some which only take clausal complements, many take both as shown in examples (314)-(317), and it seems to be artificial to assign them two different parts-of-speech.

(314) Helen left before the party.

(315) Helen left before the party began.

(316) Since the election, Bill has been elated.

(317) Since winning the election, Bill has been elated.

Each subordinating conjunction selects the values of the <**mode**> and <**comp**> features of the subordinated S. The <**mode**> value constrains the types of clauses the subordinating conjunction may appear with and the <**comp**> value constrains the complementizers which may adjoin to that clause. For instance, indicative subordinate clauses may appear with the complementizer *that* as in (318), while participial clauses may not have any complementizers (319).

(318) Midge left that car so that Sam could drive to work.

(319) *Since that seeing the new VW, Midge could think of nothing else.

17.0.4 Multi-word Subordinating Conjunctions

We extracted a list of multi-word conjunctions, such as *as if*, *in order*, and *for all (that)*, from [Quirk *et al.*, 1985]. For the most part, the components of the complex are all anchors, as shown in Figures 17.2(a). In one case, *as ADV as*, there is a great deal of latitude in the choice of adverb, so this is a substitution site (Figures 17.2(b)). This multi-anchor treatment is very similar to that proposed for idioms in [Abeillé and Schabes, 1989], and the analysis of light verbs in the XTAG grammar (see section 6.16).

17.1 "Bare" Adjunct Clauses

"Bare" adjunct clauses do not have an overt subordinating conjunction, but are typically parallel in meaning to clauses with subordinating conjunctions. For this reason, we have elected to handle them using the same trees shown above, but with null anchors. They are selected at the same time and in the same way the *PRO* tree is, as they all have *PRO* subjects. Three values of <**mode**> are licensed: **inf** (infinitive), **ger** (gerundive) and **ppart** (past participal).[1] They interact with complementizers as follows:

[1]We considered allowing bare indicative clauses, such as *He died that others may live*, but these were considered too archaic to be worth the additional ambiguity they would add to the grammar.

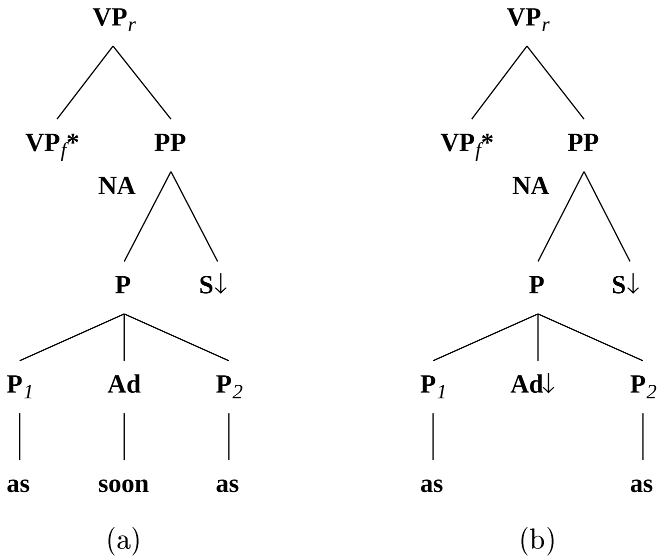


Figure 17.2: Trees Anchored by Subordinating Conjunctions: βvxPARBPs and βvxParbPs

- Participial complements do not license any complementizers:[2]

  (320) [Destroyed by the fire], the building still stood.

  (321) The fire raged for days [destroying the building].

  (322) ∗[That destroyed by the fire], the building still stood.

- Infinitival adjuncts, including purpose clauses, are licensed both with and without the complementizer *for*.

  (323) Harriet bought a Mustang [to impress Eugene].

  (324) [To impress Harriet], Eugene dyed his hair.

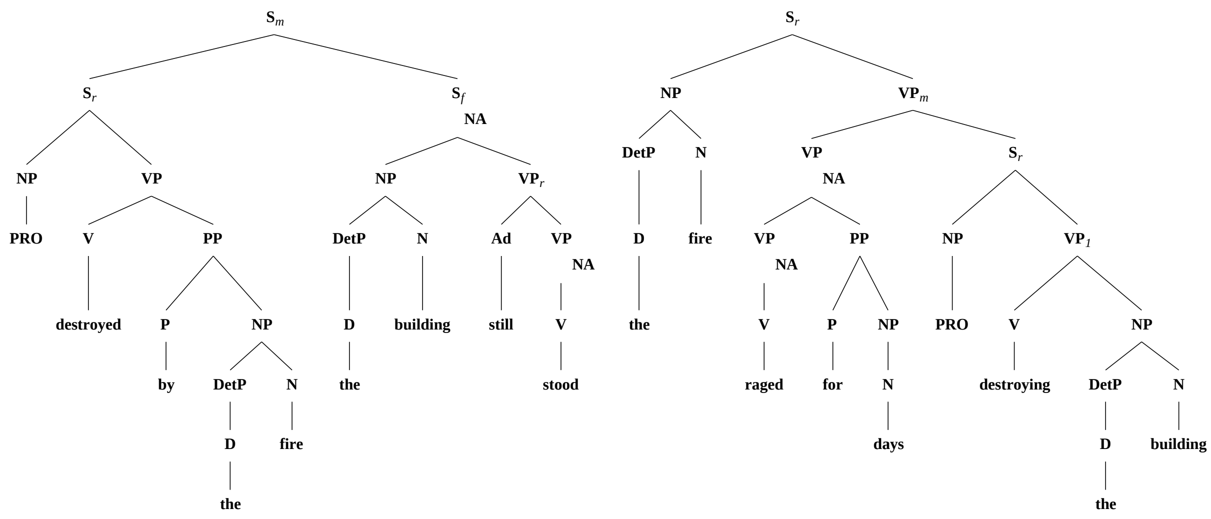
  (325) Traffic stopped [for Harriet to cross the street].

17.2 Discourse Conjunction

The CONJs auxiliary tree is used to handle 'discourse' conjunction, as in sentence (326). Only the coordinating conjunctions (*and, or* and *but*) are allowed to adjoin to the roots of matrix sentences. Discourse conjunction with *and* is shown in the derived tree in Figure 17.4.

(326) And Truffula trees are what everyone needs! [Seuss, 1971]

[2]While these sound a bit like extraposed relative clauses (see [Kroch and Joshi, 1987]), those move only to the right and adjoin to S; as these clauses are equally grammatical both sentence-initially and sentence-finally, we are analyzing them as adjunct clauses.



(a) (b)

Figure 17.3: Sample Participial Adjuncts

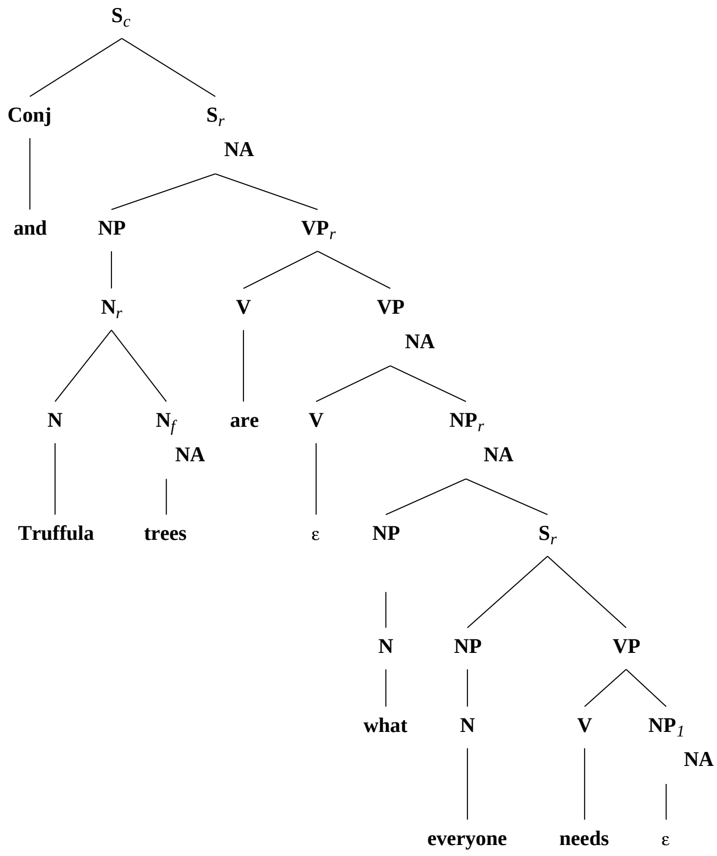


Figure 17.4: Example of discourse conjunction, from Seuss' *The Lorax*

Chapter 18

Imperatives

18.1 Agreement, mode, and the null subject

Imperatives in English do not require overt subjects. These subjects, whether overt or not, are generally interpreted as second person, as is clear from the verbal agreement and the interpretation. Imperatives with non-overt subjects are handled by the imperative trees discussed in this section. Imperatives with overt subjects are not currently handled. More discussion on imperatives with overt subjects is given in Sections 18.4.1 and 18.4.2.

The imperative tree(s) in each tree family in the English XTAG grammar are identical to the declarative tree(s) of that family except that the NP_0 subject position is filled by an ϵ, the NP_0 <**agr pers**> feature is set to the value **2nd**, and the <**mode**> feature on the root node has the value **imp** (see equations 327 – 328). Hardwiring the <**agr pers**> feature into the tree ensures the proper verbal agreement for an imperative. The <**mode**> value of **imp** on the root node is recognized as a valid mode for a matrix clause.[1] The **imp** value for <**mode**> also prevents imperatives from appearing as embedded clauses. Figure 18.1 shows the imperative tree in the transitive tree family.

(327) $\mathbf{NP_0.t{:}\langle agr\ pers\rangle = 2}$

(328) $\mathbf{S_r.b{:}\langle mode\rangle = imp}$

Moreover, the <**mode**> feature on the anchor is unspecified, and the <**mode**> feature on the top feature structure associated with the VP has the value **base** (see equation in 329).

(329) $\mathbf{VP.t{:}\langle mode\rangle = base}$

This allows the lexical verb of the imperative to be any type of verb, as long as the leftmost verb has <**mode**> = **base**. For instance, in a simple transitive imperative as in 330, the verb *eat*, which is specified with <**mode**> = **base**, anchors the imperative tree, unifying with **VP.t:**<**mode**> = **base**. In an imperative with auxiliary *be* as in 331, the verb *waiting*, which is specified with <**mode**> = **ger**, anchors the imperative tree, and the auxiliary *be*, which is specified with <**mode**> = **base**, adjoins onto the VP, unifying with **VP.t:**<**mode**> = **base**.

[1] The other valid <**mode**> for a matrix clause is **ind**.

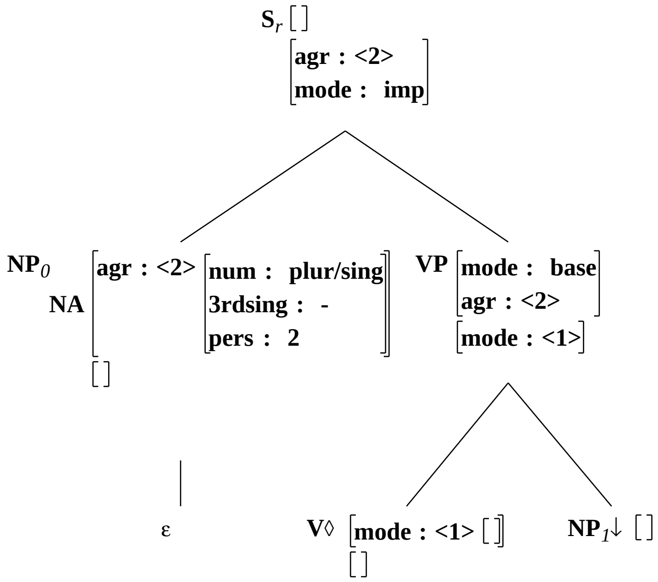


Figure 18.1: Transitive imperative tree: αInx0Vnx1

(330) Eat the cake!

(331) Be waiting for me!

18.2 Negative Imperatives

18.2.1 *Don't* imperatives

All Negative imperatives in English require *do*-support, even those that are formed with *be* and auxiliary *have*.

(332) Don't leave!

(333) *Not leave!

(334) Do not eat the cake!

(335) *Not eat the cake!

(336) Don't have eaten everything when the guests arrive!

(337) *Not have eaten everything when the guests arrive!

(338) *Have not eaten everything when the guests arrive!

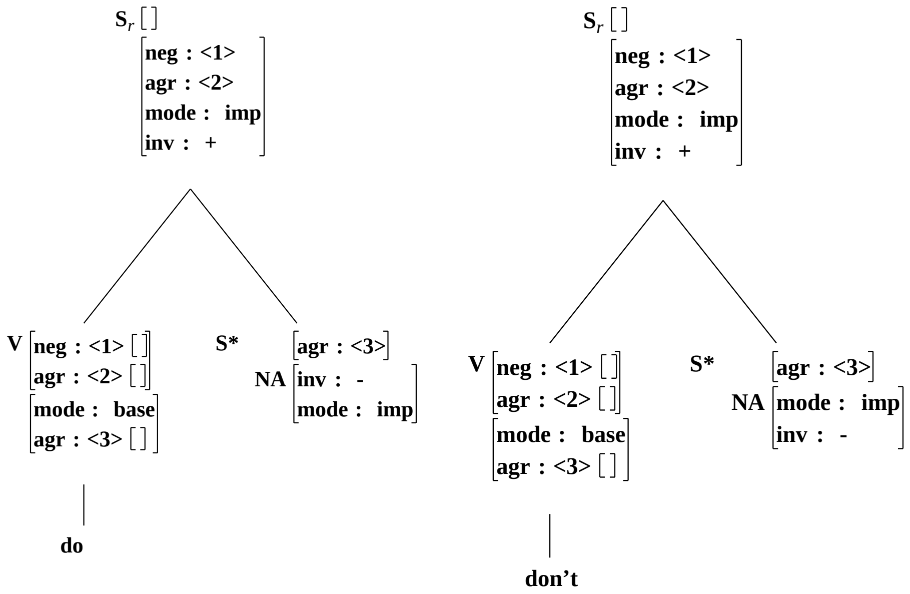
In English XTAG grammar, negative imperatives receive a similar structural analysis to *yes-no* questions, as in [Potsdam, 1997] and [Han, 1999]. That is, *do* and *don't* in negative imperatives are treated as an instance of *do*-support and adjoin to a clause. The crucial structural evidence for this analysis is that when there is an overt subject in negative imperatives formed with *don't*, the subject must follow *don't*, just as it does in *yes-no* questions.

(339) Don't you move!

(340) Don't you like carrots?

Do-support in negative imperatives is handled by the elementary tree βIVs anchored by *do* and *don't*, as shown in Figure 18.2. This tree adjoins onto the root node of the imperative tree. The feature <**mode**> = **imp** on the S foot node restricts this tree to adjoin only to imperative trees. Furthermore, the S root node of βIVs is specified with <**mode**> = **imp**, which prevents imperatives with *do*-support from appearing as embedded clauses.

In negative imperatives formed with *don't*, the βIVs[don't] tree in Figure 18.2 adjoins to the root node of the imperative tree. The derived tree for the negative imperative *Don't leave!* is given in Figure 18.3.



βIVs[do] βIVs[don't]

Figure 18.2: Trees anchored by imperative *do* and *don't*

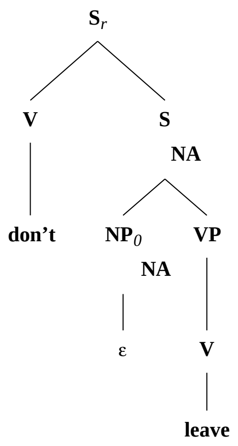


Figure 18.3: Derived tree for *Don't leave!*

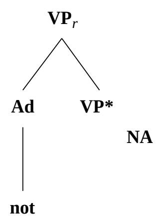


Figure 18.4: Tree anchored by *not*

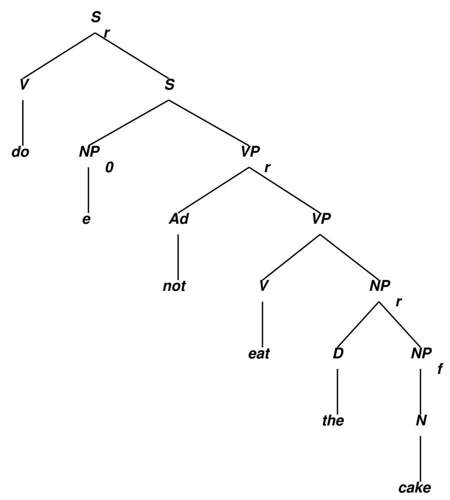


Figure 18.5: Derived trees for *Do not eat the cake!*

18.2.2 *Do not* imperatives

In negative imperatives formed with *do not*, the βIVs[do] tree in Figure 18.2 adjoins to the root node of the imperative tree and the βNEGvx tree that anchors *not*, shown in Figure 18.4, adjoins to the VP node of the imperative tree.

The derived tree for the negative imperative *Do not eat the cake!* is given in Figure 18.5.

If *do* in negative imperatives is in the same position as *do* in *yes-no* questions, the fact that an overt subject cannot intervene between *do* and *not* is puzzling.

(341) Do not open the window!

(342) *Do you not open the window!

We adopt the account given in [Akmajian, 1984] that this fact is not due to syntax but due to an intonational constraint in imperatives. He argues that (i) when an imperative sentence has an overt subject, the subject must be the only intonation center preceding the verb phrase, and (ii) in negative imperatives with *do* and *not*, either *do* or *not* must be the intonation center. These two contraints conspire to rule out *do not* imperatives with an overt subject. Since the constraint ruling out 342 is not taken to be a syntactic one, the sentence is not ruled out by the XTAG grammar.

18.2.3 Negative Imperatives with *be* and *have*

Another puzzling fact that needs to be explained is that negative imperatives require *do*-support even with *be* and auxiliary *have*, unlike negative declaratives 343-344 and negative questions 345-346 where *do* cannot co-occur with *have* or *be*.

(343) He isn't talking loud.

(344) *He doesn't be talking loud.

(345) Isn't he talking loud?

(346) *Doesn't he be talking loud?

[Han, 1999] accounts for this fact in terms of V to I movement. She notes that while declaratives and questions are tensed, imperatives are not. She argues that this tense difference is responsible for why negative imperatives require *do*-support even for *be* and auxiliary *have*. Assuming a clause structure in which CP dominates IP and IP dominates VP, she argues that it is the tense features in I^0 that attract *be* and auxiliary *have* in declaratives and questions. In declaratives, *be* or auxiliary *have* moves to and stays in I^0, and in questions, once *be* or auxiliary *have* moves to I^0, it can further move to C^0. Moreover, main verbs cannot move at all to I^0 in the overt syntax. Instead, they undergo movement at LF. But negation blocks LF movement and so as a last resort *do* is inserted in I^0 to support INFL. In imperatives, I^0 does not have tense features and so it cannot attract *be* and auxiliary *have*. Thus, *be* and auxiliary *have* as well as main verbs undergo movement at LF in imperatives. And so in negative imperatives, since negation blocks LF verb movement, *do* is inserted in I^0 as a last resort device even for *be* and auxiliary *have* and it further moves to C^0 in the overt syntax.

In the XTAG grammar we do not represent V to I movement or LF movement, and so our analysis cannot directly follow that of [Han, 1999]. Instead, we derive the facts by having a separate entry for *do* (which anchors a separate auxiliary tree βIVs) which is used only with imperatives. Unlike the other entry for *do*, it does not require the verb below it to be <**mainv** = + > . Thus, this *do* can co-occur with the verbs *have* and *be* which have the feature <**mainv** = − >

18.3 Emphatic Imperatives

Another case where imperatives have *do*-support is emphatic imperatives.

(347) Do open the window!

(348) Do show up for the lecture!

In English XTAG grammar, *do* in emphatic imperatives is treated just as *do* in negative imperatives. It is adjoined to the S node of an imperative clause with an empty subject. Again, the crucial evidence for this analysis comes from word order facts. When emphatic imperatives have an overt subject, it must follow *do*.

(349) Do somebody bring me some water!

(350) Do at least some of you show up for the lecture!

18.4 Cases not handled

18.4.1 Overt subjects before *do/don't*

Given our analsis of negative imperatives, if the subject precedes *do* or *don't*, we are forced to treat it as a vocative rather than as a true subject. Vocatives are considered to be outside the clause structure and do not have any structural relation with any element in the clause.

(351) You don't drink the water. = (You! Don't drink the water!)

(352) You do not leave the room. = (You! Do not leave the room!)

Given the fact that the imperatives in 351 and 352 seem to be degraded unless there is an intonational break between *you* and the rest of the sentence, treating *you* as a vocative seems to be the correct approach. Currently, however, the XTAG grammar does not handle vocatives.

18.4.2 Overt subjects after *do/don't*

One remaining task for imperatives is to handle those with overt subjects such as the examples in 349 and 350. The type of overt subjects allowed in imperatives is restricted to 2nd person pronouns and some quantified noun phrases. Currently, the English XTAG grammar only has imperative trees with empty subjects.

18.4.3 Passive Imperatives

Passive imperatives such as 353 are currently not handled.

(353) Don't be defeated at the race today!

One way to account for these would be to make separate passive imperative trees in each tree family, as is done for the declaratives and otehr clause types in each family.

Another way to allow for passive imperatives would be to remove the equation **V.t:<passive>=** – from the imperative trees. However, this option may affect the consistency of the treatment of passives. A decision in this respect will have to be made before implementing imperative passives.

18.4.4 Overgeneration

As discussed above, imperatives can also be formed with auxiliaries like *have* and *be*, as in 354 and 355.

(354) Don't have fallen asleep when I come back!

(355) Be waiting for me when I return!

The auxiliary *be* can form affirmative as well as negative imperatives. However, *have* can only form a negative imperative, as can be seen from 355 and the ungrammaticality of 356:

(356) * Have eaten your meal by the time I return!

The current analysis of imperative, however, does not rule out 356.

Chapter 19

Gerund NP's

There are two types of gerunds identified in the linguistics literature. One is the class of *derived nominalizations* (also called *nominal gerundives* or *action nominalizations*) exemplified in (357), which instantiates the direct object within an *of* PP. The other is the class of so-called *sentential* or *VP gerundives* exemplified in (358). In the English XTAG grammar, the derived nominalizations are termed **determiner gerunds**, and the sentential or VP gerunds are termed **NP gerunds**.

(357) Some think that **the selling of bonds** is beneficial.

(358) Are private markets approving of **Washington bashing Wall Street**?

Both types of gerunds exhibit a similar distribution, appearing in most places where NP's are allowed.[1] The bold face portions of sentences (359)–(361) show examples of gerunds as a subject and as the object of a preposition.

(359) **Avoiding such losses** will take a monumental effort.

(360) **Mr. Nolen's wandering** doesn't make him a weirdo.

(361) Are private markets approving of **Washington bashing Wall Street**?

The motivation for splitting the gerunds into two classes is semantic as well as structural in nature. Semantically, the two gerunds are in sharp contrast with each other. NP gerunds refer to an action, i.e., a way of doing something, whereas determiner gerunds refer to a fact. Structurally, there are a number of properties (extensively discussed in [Lees, 1960]) that show that NP gerunds have the syntax of verbs, whereas determiner gerunds have the syntax of basic nouns. Firstly, the fact that the direct object of the determiner gerund can only appear within an *of* PP suggests that the determiner gerund, like nouns, is not a case assigner and needs insertion of the preposition *of* for assignment of case to the direct object. NP gerunds, like verbs, need no such insertion and can assign case to their direct object. Secondly, like nouns, only determiner gerunds can appear with articles (cf. example (362) and (363)). Thirdly, determiner gerunds, like nouns, can be modified by adjectives (cf. example (364)), whereas

[1] an exception being the NP positions in "equative BE" sentences, such as, *John is my father*.

NP gerunds, like verbs, resist such modification (cf. example (365)). Fourthly, nouns, unlike verbs, cannot co-occur with aspect (cf. example (366) and (367)). Finally, only NP gerunds, like verbs, can take adverbial modification (cf. example (368) and (369)).

(362) ...the proving of the theorem.... (det ger with article)

(363) * ...the proving the theorem.... (NP ger with article)

(364) John's rapid writing of the book.... (det ger with Adj)

(365) * John's rapid writing the book.... (NP ger with Adj)

(366) * John's having written of the book.... (det ger with aspect)

(367) John having written the book.... (NP ger with aspect)

(368) * His writing of the book rapidly.... (det ger with Adverb)

(369) His writing the book rapidly.... (NP ger with Adverb)

In English XTAG, the two types of gerunds are assigned separate trees within each tree family, but in order to capture their similar distributional behavior, both are assigned NP as the category label of their top node. The feature **gerund** $= +/-$ distinguishes gerund NP's from regular NP's where needed.[2] The determiner gerund and the NP gerund trees are discussed in section (19.1) and (19.2) respectively.

19.1 Determiner Gerunds

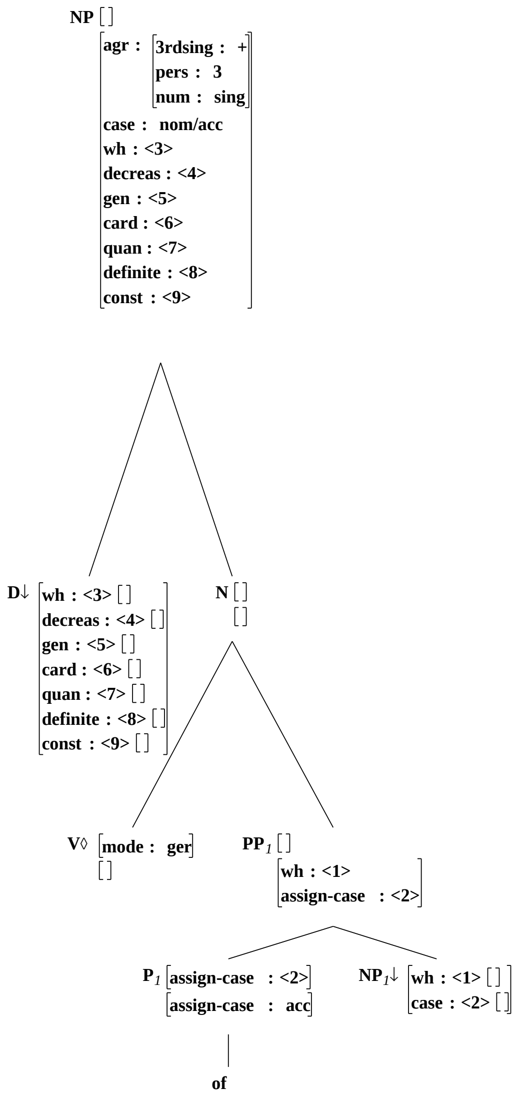
The determiner gerund tree in Figure 19.1 is anchored by a V, capturing the fact that the gerund is derived from a verb. The verb projects an N and instantiates the direct object as an *of* PP. The nominal category projected by the verb can now display all the syntactic properties of basic nouns, as discussed above. For example, it can be straightforwardly modified by adjectives; it cannot co-occur with aspect; and it can appear with articles. The only difference of the determiner gerund nominal with the basic nominals is that the former cannot occur without the determiner, whereas the latter can. The determiner gerund tree therefore has an initial D modifying the N.[3] It is used for gerunds such as the ones in bold face in sentences (370), (371) and (372).

The D node can take a simple determiner (cf. example (370)), a genitive pronoun (cf. example (371)), or a genitive NP (cf. example (372)).[4]

[2]This feature is also needed to restrict the selection of gerunds in NP positions. For example, the subject and object NP's in the "equative BE" tree (Tnx0BEnx1) cannot be filled by gerunds, and are therefore assigned the feature **gerund** $= -$, which prevents gerunds (which have the feature **gerund** $= +$) from substituting into these NP positions.

[3]Note that the determiner can adjoin to the gerund only from *within* the gerund tree. Adjunction of determiners to the gerund root node is prevented by constraining determiners to only select NP's with the feature **gerund** $= -$. This rules out sentences like *Private markets approved of (*the) [the selling of bonds].*

[4]The trees for genitive pronouns and genitive NP's have the root node labelled as D because they seem to function as determiners and as such, sequence with the rest of the determiners. See Chapter 20 for discussion on determiner trees.



αDnx0Vnx1

Figure 19.1: Determiner Gerund tree from the transitive tree family: αDnx0Vnx1

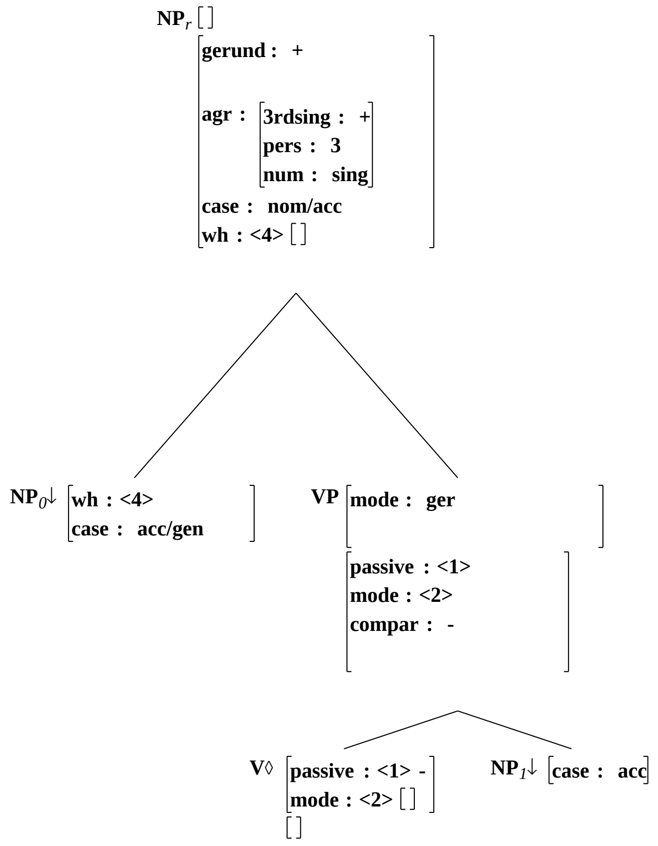
(370) Some think that **the selling of bonds** is beneficial.

(371) **His painting of Mona Lisa** is highly acclaimed.

(372) Are private markets approving of **Washington's bashing of Wall Street**?

19.2 NP Gerunds

NP gerunds show a number of structural peculiarities, the crucial one being that they have the internal properties of sentences. In the English XTAG grammar, we adopt a position similar to that of [Rosenbaum, 1967] and [Emonds, 1970] – that these gerunds are NP's exhaustively dominating a clause. Consequently, the tree assigned to the transitive NP gerund tree (cf. Figure 19.2) looks exactly like the declarative transitive tree for clauses except for the root node label and the feature values. The anchoring verb projects a VP. Auxiliary adjunction is allowed, subject to one constraint – that the highest verb in the verbal sequence be in gerundive form, regardless of whether it is a main or auxiliary verb. This constraint is implemented by requiring the topmost VP node to be <**mode**> = **ger**. In the absence of any adjunction, the anchoring verb itself is forced to be gerundive. But if the verbal sequence has more than one verb, then the sequence and form of the verbs is limited by the restrictions that each verb in the sequence imposes on the succeeding verb. The nature of these restrictions for sentential clauses, and the manner in which they are implemented in XTAG, are both discussed in Chapter 22. The analysis and implementation discussed there differs from that required for gerunds only in one respect – that the highest verb in the verbal sequence is required to be <**mode**> = **ger**.



αGnx0Vnx1

Figure 19.2: NP Gerund tree from the transitive tree family: αGnx0Vnx1

Additionally, the subject in the NP gerund tree can be in the accusative 373 or genitive form 374, or it can be a PRO, exhibiting null case 375.

(373) Mother disapproved of **me wearing such casual clothes**.

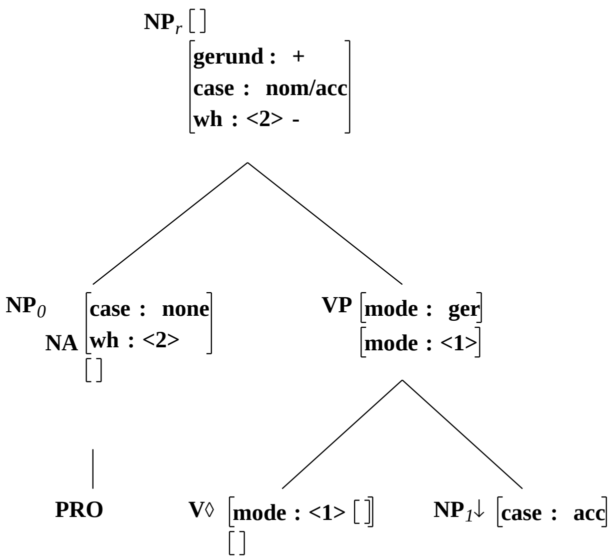
(374) Are private markets approving of **Washington's bashing Wall Street**?

(375) John does not like **wearing a hat**.

The gerunds with accusative and genitive subject NP's are handled by the following feature equation on the subject NP node

(376) **case : acc/gen**

which allows only accusative and genitive marked NPs to substitute into the subject position. The PRO subject gerunds are, however, treated differently. In the previous versions of the grammar, the distribution of PRO was handled in the same way as for NP's in other cases, i.e., through case assignment by the verb. However, this entailed treating PRO as a lexicalizing element. In the current version of the grammar, we have dispensed with the option of lexicalizing trees with PRO. Instead, the grammar now contains trees (corresponding to each construction type that can have a PRO subject) with the PRO built-in into the subject NP position. Since the NP gerund can also have a PRO subject, there is one tree in each family for the different NP gerund constructions. Figure 19.3 shows the NP gerund tree in the transtive family with the PRO built in. Null case in the subject NP is still represented as the feature equation **case : none**, even though the verb does not assign this case.[5]



αGnx0Vnx1

Figure 19.3: NP Gerund tree from the transitive tree family, with PRO built in: αGnx0Vnx1-PRO

One question that arises with respect to gerunds is whether there is anything special about their distribution as compared to other types of NP's. In fact, it appears that gerund NP's

[5]This, however, is not merely for reasons of consistency. The feature equation is needed to rule out a PRO clause being selected as the complement of an ECM predicate. Putting the feature equation on the subject NP in the PRO constructions leads to a feature clash with the accusative case feature equations of the ECM predicate.

can occur in any NP position. Some verbs might not seem to be very accepting of gerund NP arguments, as in (377) below, but we believe this to be a semantic incompatibility rather than a syntactic problem since the same structures are possible with other lexical items.

(377) ? [$_{NP}$John's tinkering$_{NP}$] ran.

(378) [$_{NP}$John's tinkering$_{NP}$] worked.

By having the root node of gerund trees be NP, the gerunds have the same distribution as any other NP in the English XTAG grammar without doing anything exceptional. The clause structure is captured by the form of the trees and by inclusion in the tree families.

19.3 Gerund Passives

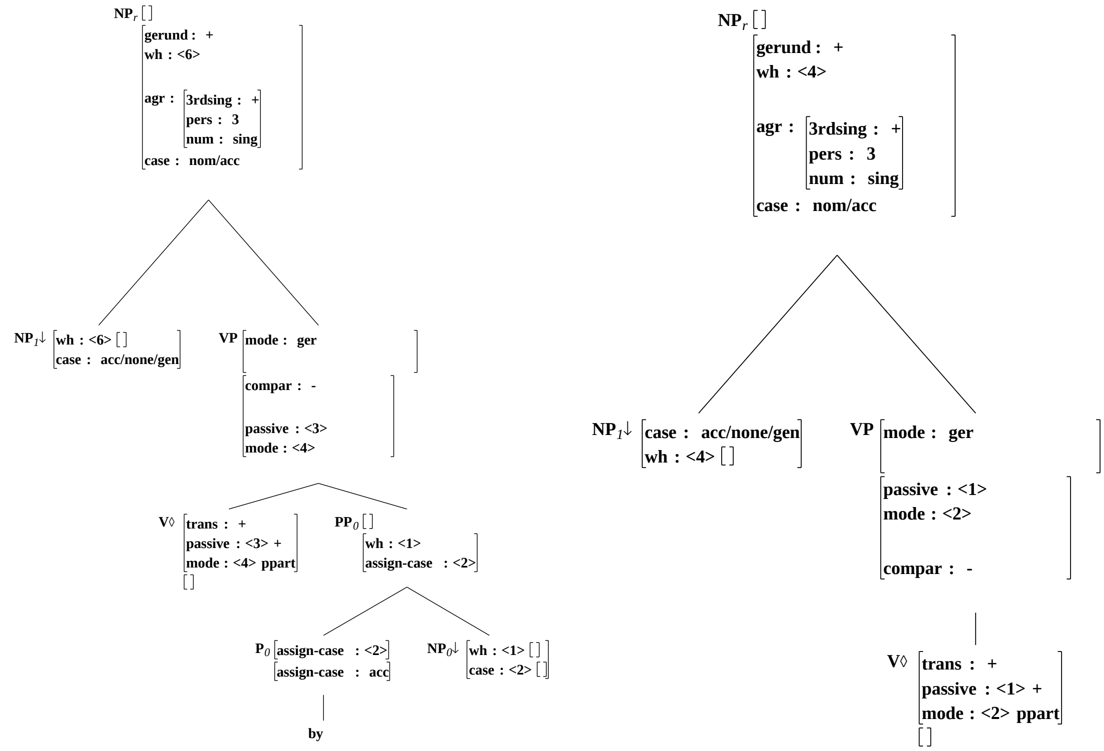
It was mentioned above that the NP gerunds display certain clausal properties. It is therefore not surprising that they too have their own set of transformationally related structures. For example, NP gerunds allow passivization just like their sentential counterparts 379.

(379) The lawyers objected to **the slanderous book being written by John**.

In the English XTAG grammar, gerund passives are treated in an almost exactly similar fashion to sentential passives, and are assigned separate trees within the appropriate tree families. The passives occur in pairs, one with the *by* phrase, and another without it. There are two feature restrictions imposed on the passive trees: (a) only verbs with <**mode**> = **ppart** (i.e., verbs with passive morphology) can be the anchors, and (b) the highest verb in the verb sequence is required to be <**mode**> = **ger**. The two restrictions, together, ensure the selection of only those sequences of auxiliary verb(s) that select <**mode**> = **ppart** and <**passive**> = + (such as *being* or *having been* but NOT *having*). The passive trees are assumed to be related to only the NP gerund trees (and not the determiner gerund trees), since passive structures involve movement of some object to the subject position (in a movement analysis). Like the sentential passives, gerund passives are found in most tree families that have a direct object in the declarative tree. Figure 19.4 shows the gerund passive trees for the transitive tree family.

Gerund passives can also have a PRO subject 380. As was mentioned above in the discussion of the declarative gerunds, there is a separate tree for such gerund passives (with and without the *by* phrase) with PRO built in.

(380) Susan could not forget **having been embarrassed by the vicar**.



(a) αGnx1Vbynx0 (b) αGnx1V

Figure 19.4: Passive Gerund trees from the transitive tree family: αGnx1Vbynx0 (a) and αGnx1V (b)

Part IV

Other Constructions

Chapter 20

Determiners and Noun Phrases

In our English XTAG grammar,[1] all nouns select the noun phrase (NP) tree structure shown in Figure 20.1. Common nouns do not require determiners in order to form grammatical NPs. Rather than being ungrammatical, singular countable nouns without determiners are restricted in interpretation and can only be interpreted as mass nouns. Allowing all nouns to head determinerless NPs correctly treats the individuation in countable NPs as a property of determiners. Common nouns have negative("-") values for determiner features in the lexicon in our analysis and can only acquire a positive("+") value for those features if determiners adjoin to them. Other types of NPs such as pronouns and proper nouns have been argued by Abney [Abney, 1987] to either be determiners or to move to the determiner position because they exhibit determiner-like behavior. We can capture this insight in our system by giving pronouns and proper nouns positive values for determiner features. For example pronouns and proper nouns would be marked as definite, a value that NPs containing common nouns can only obtain by having a definite determiner adjoin. In addition to the determiner features, nouns also have values for features such as reflexive (**refl**), case, pronoun (**pron**) and conjunction (**conj**).

A single tree structure is selected by simple determiners, an auxiliary tree which adjoins to NP. An example of this determiner tree anchored by the determiner *these* is shown in Figure 20.2. In addition to the determiner features the tree in Figure 20.2 has noun features such as **case** (see section 4.4.2), the **conj** feature to control conjunction (see Chapter 23), **rel-clause**− (see Chapter 16) and **gerund**− (see Chapter 19) which prevent determiners from adjoining on top of relative clauses and gerund NPs respectively.

Complex determiners such as genitives and partitives also anchor tree structures that adjoin to NP. They differ from the simple determiners in their internal complexity. Details of our treatment of these more complex constructions appear in Sections 20.3 and 20.4. Sequences of determiners, as in the NPs *all her dogs* or *those five dogs* are derived by multiple adjunctions of the determiner tree, with each tree anchored by one of the determiners in the sequence. The order in which the determiner trees can adjoin is controlled by features.

This treatment of determiners as adjoining onto NPs is similar to that of [Abeillé, 1990], and allows us to capture one of the insights of the DP hypothesis, namely that determiners select NPs as complements. In Figure 20.2 the determiner and its NP complement appear in the configuration that is typically used in LTAG to represent selectional relationships. That

[1] A more detailed discussion of this analysis can be found in [Hockey and Mateyak, 1998].

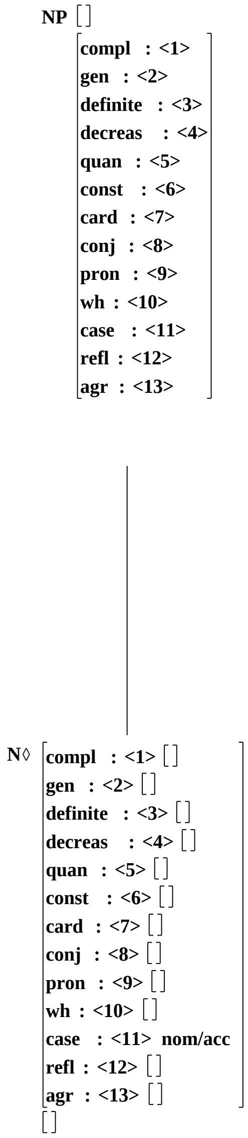


Figure 20.1: NP Tree

is, the head serves as the anchor of the tree and it's complement is a sister node in the same elementary tree.

The XTAG treatment of determiners uses nine features for representing their properties: definiteness (**definite**), quantity (**quan**), cardinality (**card**), genitive (**gen**), decreasing (**decreas**), constancy (**const**), **wh**, agreement (**agr**), and complement (**compl**). Seven of these features were developed by semanticists for their accounts of semantic phenomena ([Keenan and Stavi, 1986], [Barwise and Cooper, 1981], [Partee *et al.*, 1990]), another was developed for a semantic account of determiner negation by one of the authors of this determiner analysis

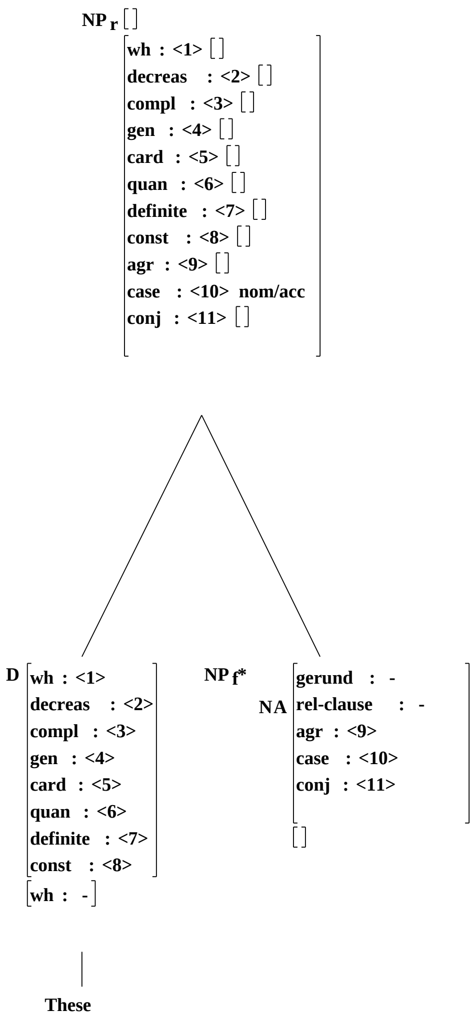


Figure 20.2: Determiner Trees with Features

([Mateyak, 1997]), and the last is the familiar agreement feature. When used together these features also account for a substantial portion of the complex patterns of English determiner sequencing. Although we do not claim to have exhaustively covered the sequencing of determiners in English, we do cover a large subset, both in terms of the phenomena handled and in terms of corpus coverage. The XTAG grammar has also been extended to include complex determiner constructions such as genitives and partitives using these determiner features.

Each determiner carries with it a set of values for these features that represents its own properties, and a set of values for the properties of NPs to which can adjoin. The features are crucial to ordering determiners correctly. The semantic definitions underlying the features are given below.

Definiteness: Possible Values [+/−].

A function f is definite iff f is non-trivial and whenever f(s) $\neq \emptyset$ then it is always the intersection of one or more individuals. [Keenan and Stavi, 1986]

Quantity: Possible Values [+/–].
If A and B are sets denoting an NP and associated predicate, respectively; E is a domain in a model M, and F is a bijection from M_1 to M_2, then we say that a determiner satisfies the constraint of quantity if $\mathrm{Det}_{E_1}\mathrm{AB} \leftrightarrow \mathrm{Det}_{E_2}\mathrm{F(A)F(B)}$. [Partee *et al.*, 1990]

Cardinality: Possible Values [+/–].
A determiner D is cardinal iff D $\in$ cardinal numbers ≥ 1.

Genitive: Possible Values [+/–].
Possessive pronouns and the possessive morpheme (*'s*) are marked **gen+**; all other nouns are **gen−**.

Decreasing: Possible Values [+/–].
A set of Q properties is decreasing iff whenever s$\leq$t and t$\in$Q then s$\in$Q. A function f is decreasing iff for all properties f(s) is a decreasing set.

A non-trivial NP (one with a Det) is decreasing iff its denotation in any model is decreasing. [Keenan and Stavi, 1986]

Constancy: Possible Values [+/–].
If A and B are sets denoting an NP and associated predicate, respectively, and E is a domain, then we say that a determiner displays constancy if $(A \cup B) \subseteq E \subseteq E'$ then $\mathrm{Det}_E\mathrm{AB} \leftrightarrow \mathrm{Det}_{E'}\mathrm{AB}$. Modified from [Partee *et al.*, 1990]

Complement: Possible Values [+/–].
A determiner Q is positive complement if and only if for every set X, there exists a continuous set of possible values for the size of the negated determined set, NOT(QX), and the cardinality of QX is the only aspect of QX that can be negated. (adapted from [Mateyak, 1997])

The **wh**-feature has been discussed in the linguistics literature mainly in relation to wh-movement and with respect to NPs and nouns as well as determiners. We give a shallow but useful working definition of the **wh**-feature below:

Wh: Possible Values [+/–].
Interrogative determiners are **wh+**; all other determiners are **wh−**.

The **agr** feature is inherently a noun feature. While determiners are not morphologically marked for agreement in English many of them are sensitive to number. Many determiners are semantically either singular or plural and must adjoin to nouns which are the same. For example, *a* can only adjoin to singular nouns (*a dog* vs ∗*a dogs* while *many* must have plurals (*many dogs* vs ∗*many dog*). Other determiners such as *some* are unspecified for agreement in our analysis because they are compatible with either singulars or plurals (*some dog*, *some dogs*). The possible values of agreement for determiners are: [3sg, 3pl, 3].

The determiner tree in Figure 20.2 shows the appropriate feature values for the determiner *these*, while Table 20.1 shows the corresponding feature values of several other common determiners.

Det	definite	quan	card	gen	wh	decreas	const	agr	compl
all	−	+	−	−	−	−	+	3pl	+
both	+	−	−	−	−	−	+	3pl	+
this	+	−	−	−	−	−	+	3sg	−
these	+	−	−	−	−	−	+	3pl	−
that	+	−	−	−	−	−	+	3sg	−
those	+	−	−	−	−	−	+	3pl	−
what	−	−	−	−	+	−	+	3	−
whatever	−	−	−	−	−	−	+	3	−
which	−	−	−	−	+	−	+	3	−
whichever	−	−	−	−	−	−	+	3	−
the	+	−	−	−	−	−	+	3	−
each	−	+	−	−	−	−	+	3sg	−
every	−	+	−	−	−	−	+	3sg	+
a/an	−	+	−	−	−	−	+	3sg	+
some_1	−	+	−	−	−	−	+	3	−
some_2	−	+	−	−	−	−	−	3pl	−
any	−	+	−	−	−	−	+	3sg	+
another	−	+	−	−	−	−	+	3sg	+
few	−	+	−	−	−	+	−	3pl	−
a few	−	+	−	−	−	−	+	3pl	−
many	−	+	−	−	−	−	−	3pl	+
many a/an	−	+	−	−	−	−	−	3sg	+
several	−	+	−	−	−	−	+	3pl	−
various	−	−	−	−	−	−	+	3pl	−
sundry	−	−	−	−	−	−	+	3pl	−
no	−	+	−	−	−	+	+	3	−
neither	−	−	−	−	−	+	+	3	−
either	−	−	−	−	−	−	+	3	−
GENITIVE	+	−	−	+	−	−	+	UN[2]	−
CARDINAL	−	+	+	−	−	−	+	3pl[3]	−[4]
PARTITIVE	−	+/-[5]	−	−	−	−	+	UN	+/-

Table 20.1: Determiner Features associated with D anchors

In addition to the features that represent their own properties, determiners also have features to represent the selectional restrictions they impose on the NPs they take as complements. The selectional restriction features of a determiner appear on the NP footnode of the auxiliary tree that the determiner anchors. The NP_f node in Figure 20.2 shows the selectional feature restriction imposed by *these*[6], while Table 20.2 shows the corresponding selectional feature restrictions of several other determiners.

20.1 The Wh-Feature

A determiner with a **wh+** feature is always the left-most determiner in linear order since no determiners have selectional restrictions that allow them to adjoin onto an NP with a +wh feature value. The presence of a wh+ determiner makes the entire NP wh+, and this is correctly represented by the coindexation of the determiner and root NP nodes' values for the wh-feature. Wh+ determiners' selectional restrictions on the NP foot node of their tree only allows them adjoin onto NPs that are **wh-** or unspecified for the wh-feature. Therefore ungrammatical sequences such as $*$*which what dog* are impossible. The adjunction of **wh +** determiners onto **wh+** pronouns is also prevented by the same mechanism.

20.2 Multi-word Determiners

The system recognizes the multi-word determiners *a few* and *many a*. The features for a multi-word determiner are located on the parent node of its two components (see Figure 20.3). We chose to represent these determiners as multi-word constituents because neither determiner retains the same set of features as either of its parts. For example, the determiner *a* is 3sg and *few* is decreasing, while *a few* is 3pl and increasing. Additionally, *many* is 3pl and *a* displays constancy, but *many a* is 3sg and does not display constancy. Example sentences appear in (381)-(382).

- Multi-word Determiners

  (381) **a few** teaspoons of sugar should be adequate .

  (382) **many a** man has attempted that stunt, but none have succeeded .

20.3 Genitive Constructions

There are two kinds of genitive constructions: genitive pronouns, and genitive NP's (which have an explicit genitive marker, *'s*, associated with them). It is clear from examples such as

[2] We use the symbol UN to represent the fact that the selectional restrictions for a given feature are unspecified, meaning the noun phrase that the determiner selects can be either positive or negative for this feature.

[3] Except *one* which is 3sg.

[4] Except *one* which is **compl+**.

[5] A partitive can be either **quan+** or **quan-**, depending upon the nature of the noun that anchors the partitive. If the anchor noun is modified, then the quantity feature is determined by the modifier's quantity value.

[6] In addition to this tree, *these* would also anchor another auxiliary tree that adjoins onto **card+** determiners.

[7] *one* differs from the rest of CARD in selecting singular nouns

Det	defin	quan	card	gen	wh	decreas	const	agr	compl	*e.g.*
	−	−	−	−	−	−	−	3pl	−	*dogs*
all	+	−	−	UN	−	UN	UN	3pl	−	*these dogs*
	UN	UN	+	UN	UN	UN	UN	3pl	UN	*five dogs*
	−	−	−	−	−	−	−	3pl	−	*dogs*
both	+	−	−	UN	−	UN	UN	3pl	−	*these dogs*
	−	−	−	−	−	−	−	3sg	−	*dog*
	−	+	UN	UN	−	+	−	3	UN	*few dogs*
this/that	−	+	UN	UN	−	−	−	3pl	+	*many dogs*
	UN	UN	+	UN	UN	UN	UN	3sg	UN	*five dogs*
	−	−	−	−	−	−	−	3pl	−	*dogs*
these/those	−	+	UN	UN	−	+	−	3pl	UN	*few dogs*
	UN	UN	+	UN	UN	UN	UN	3pl	UN	*five dogs*
what/which	−	−	−	−	−	−	−	3	−	*dog(s)*
whatever	−	+	UN	UN	−	+	−	3	UN	*few dogs*
whichever	UN	UN	+	UN	UN	UN	UN	3	UN	*many dogs*
	−	−	−	−	−	−	−	3	−	*dog(s)*
the	−	+	UN	UN	−	+	−	3	UN	*few dogs*
	+	−	−	−	−	−	−	UN	−	*the me*
	−	+	UN	UN	−	−	−	3pl	+	*many dogs*
	UN	UN	+	UN	UN	UN	UN	3	UN	*five dogs*
	−	−	−	−	−	−	−	3sg	−	*dog*
every/each	−	+	UN	UN	−	+	−	3	UN	*few dogs*
	UN	UN	+	UN	UN	UN	UN	3	UN	*five dogs*
a/an	−	−	−	−	−	−	−	3sg	−	*dog*
$\text{some}_{1,2}$	−	−	−	−	−	−	−	3	−	*dog(s)*
some_1	UN	UN	+	UN	UN	UN	UN	3pl	UN	*dogs*
	−	−	−	−	−	−	−	3sg	−	*dog*
any	−	+	UN	UN	−	+	−	3	UN	*few dogs*
	UN	UN	+	UN	UN	UN	UN	3	UN	*five dogs*
	−	−	−	−	−	−	−	3sg	−	*dog*
another	−	+	UN	UN	−	+	−	3	UN	*few dogs*
	UN	UN	+	UN	UN	UN	UN	3	UN	*five dogs*
few	−	−	−	−	−	−	−	3pl	−	*dogs*
a few	−	−	−	−	−	−	−	3pl	−	*dogs*
many	−	−	−	−	−	−	−	3pl	−	*dogs*
many a/an	−	−	−	−	−	−	−	3sg	−	*dog*
several	−	−	−	−	−	−	−	3pl	−	*dogs*
various	−	−	−	−	−	−	−	3pl	−	*dogs*
sundry	−	−	−	−	−	−	−	3pl	−	*dogs*
no	−	−	−	−	−	−	−	3	−	*dog(s)*
neither	−	−	−	−	−	−	−	3sg	−	*dog*
either	−	−	−	−	−	−	−	3sg	−	*dog*

Table 20.2: Selectional Restrictions Imposed by Determiners on the NP foot node

Det	definite	quan	card	gen	wh	decreas	const	agr	compl
	−	−	−	−	−	−	−	3	−
	−	+	UN	UN	−	+	−	3	UN
GENITIVE	−	+	UN	UN	−	−	−	3pl	+
	UN	UN	+	UN	UN	UN	UN	3	UN
	−	+	−	−	−	−	+	3pl	−
	−	−	−	−	−	−	+	3pl	−
CARDINAL	−	−	−	−	−	−	−	3pl[7]	−
PARTITIVE	UN	UN	UN	UN	−	UN	UN	UN	UN

Table 20.3: Selectional Restrictions Imposed by Groups of Determiners/Determiner Constructions

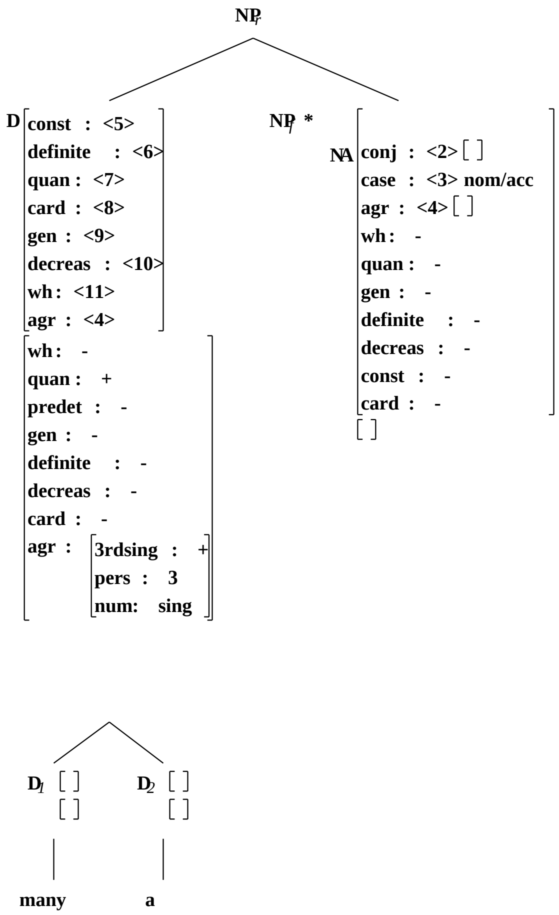


Figure 20.3: Multi-word Determiner tree: βDDnx

her dog returned home and *her five dogs returned home* vs ∗*dog returned home* that genitive pronouns function as determiners and as such, they sequence with the rest of the determiners. The features for the genitives are the same as for other determiners. Genitives are not required to agree with either the determiners or the nouns in the NPs that they modify. The value of the **agr** feature for an NP with a genitive determiner depends on the NP to which the genitive determiner adjoins. While it might seem to make sense to take *their* as 3pl, *my* as 1sg, and *Alfonso's* as 3sg, this number and person information only effects the genitive NP itself and bears no relationship to the number and person of the NPs with these items as determiners. Consequently, we have represented **agr** as unspecified for genitives in Table 20.1.

Genitive NP's are particularly interesting because they are potentially recursive structures. Complex NP's can easily be embedded within a determiner.

(383) [[[John]'s friend from high school]'s uncle]'s mother came to town.

There are two things to note in the above example. One is that in embedded NPs, the genitive morpheme comes at the end of the NP phrase, even if the head of the NP is at the beginning of the phrase. The other is that the determiner of an embedded NP can also be a genitive NP, hence the possibility of recursive structures.

In the XTAG grammar, the genitive marker, *'s*, is separated from the lexical item that it is attached to and given its own category (G). In this way, we can allow the full complexity of NP's to come from the existing NP system, including any recursive structures. As with the simple determiners, there is one auxiliary tree structure for genitives which adjoins to NPs. As can be seen in 20.4, this tree is anchored by the genitive marker *'s* and has a branching D node which accomodates the additional internal structure of genitive determiners. Also, like simple determiners, there is one initial tree structure (Figure 20.5) available for substitution where needed, as in, for example, the Determiner Gerund NP tree (see Chapter 19 for discussion on determiners for gerund NP's).

Since the NP node which is sister to the G node could also have a genitive determiner in it, the type of genitive recursion shown in (383) is quite naturally accounted for by the genitive tree structure used in our analysis.

20.4 Partitive Constructions

The deciding factor for including an analysis of partitive constructions(e.g. *some kind of*, *all of*) as a complex determiner constructions was the behavior of the agreement features. If partitive constructions are analyzed as an NP with an adjoined PP, then we would expect to get agreement with the head of the NP (as in (384)). If, on the other hand, we analyze them as a determiner construction, then we would expect to get agreement with the noun that the determiner sequence modifies (as we do in (385)).

(384) a *kind* [of these machines] *is* prone to failure.

(385) [a kind of] these *machines are* prone to failure.

In other words, for partitive constructions, the semantic head of the NP is the second rather than the first noun in linear order. That the agreement shown in (385) is possible suggests that the second noun in linear order in these constructions should also be treated as the syntactic head. Note that both the partitive and PP readings are usually possible for a particular NP. In the cases where either the partitive or the PP reading is preferred, we take it to be just that,

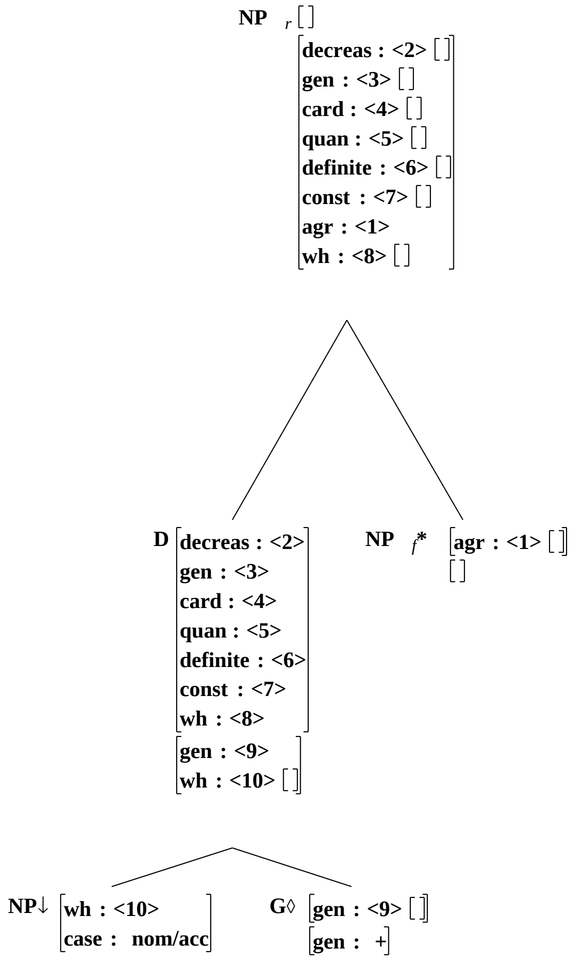


Figure 20.4: Genitive Determiner Tree

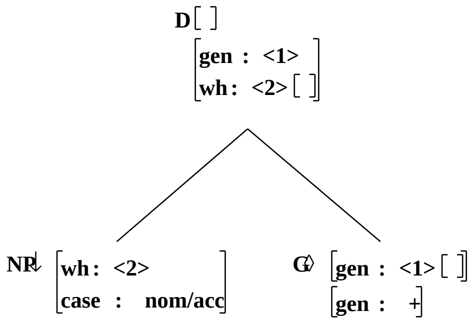


Figure 20.5: Genitive NP tree for substitution: αDnxG

a preference, most appropriately modeled not in the grammar but in a component such as the heuristics used with the XTAG parser for reducing the analyses produced to the most likely.

In our analysis the partitive tree in Figure 20.6 is anchored by one of a limited group of nouns that can appear in the determiner portion of a partitive construction. A rough semantic characterization of these nouns is that they either represent quantity (e.g. *part, half, most, pot, cup, pound* etc.) or classification (e.g. *type, variety, kind, version* etc.). In the absence of a more implementable characterization we use a list of such nouns compiled from a descriptive grammar [Quirk *et al.*, 1985], a thesaurus, and from online corpora. In our grammar the nouns on the list are the only ones that select the partitive determiner tree.

Like other determiners, partitives can adjoin to an NP consisting of just a noun (*'[a certain kind of] machine'*), or adjoin to NPs that already have determiners (*'[some parts of] these machines'*. Notice that just as for the genitives, the complexity and the recursion are contained below the D node and rest of the structure is the same as for simple determiners.

20.5 Adverbs, Noun Phrases, and Determiners

Many adverbs interact with the noun phrase and determiner system in English. For example, consider sentences (386)-(393) below.

(386) **Approximately** thirty people came to the lecture.

(387) **Practically** every person in the theater was laughing hysterically during that scene.

(388) **Only** John's crazy mother can make stuffing that tastes so good.

(389) **Relatively** few programmers remember how to program in COBOL.

(390) **Not** every martian would postulate that all humans speak a universal language.

(391) **Enough** money was gathered to pay off the group gift.

(392) **Quite** a few burglaries occurred in that neighborhood last year.

(393) I wanted to be paid **double** the amount they offered.

Although there is some debate in the literature as to whether these should be classified as determiners or adverbs, we believe that these items that interact with the NP and determiner system are in fact adverbs. These items exhibit a broader distribution than either determiners or adjectives in that they can modify many other phrasal categories, including adjectives, verb phrases, prepositional phrases, and other adverbs.

Using the determiner feature system, we can obtain a close approximation to an accurate characterization of the behavior of the adverbs that interact with noun phrases and determiners. Adverbs can adjoin to either a determiner or a noun phrase (see figure 20.7), with the adverbs restricting what types of NPs or determiners they can modify by imposing feature requirements on the foot D or NP node. For example, the adverb *approximately*, seen in (386) above, selects for determiners that are **card+**. The adverb *enough* in (391) is an example of an adverb that selects for a noun phrase, specifically a noun phrase that is not modified by a determiner.

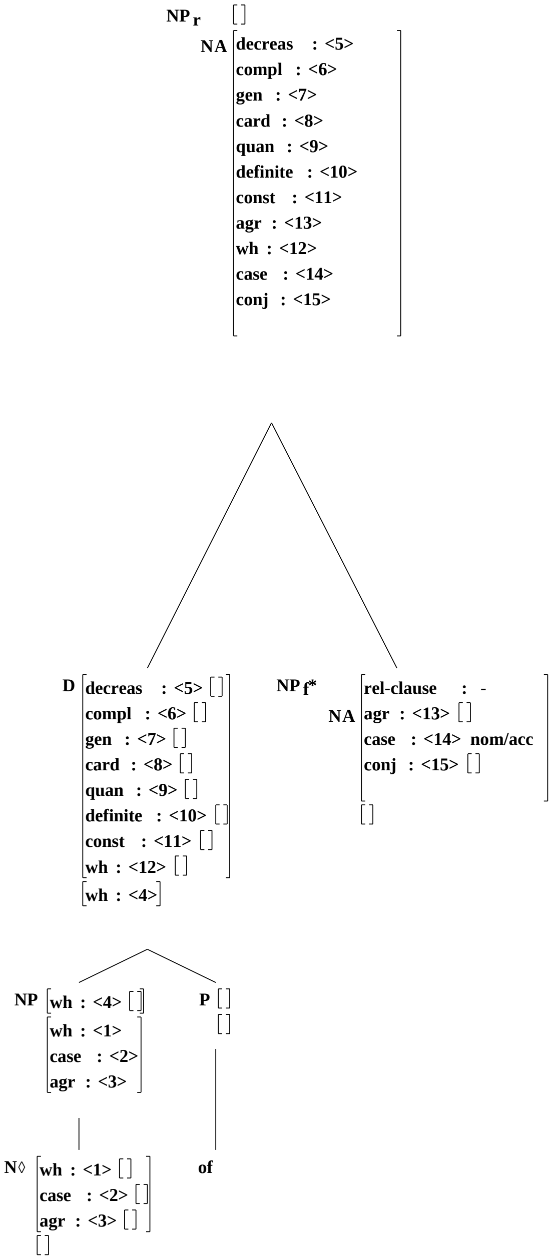
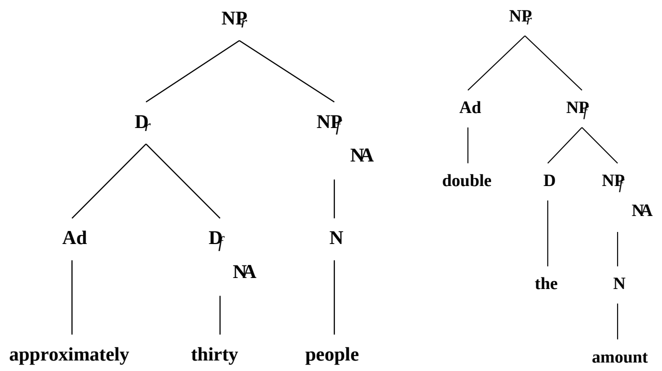


Figure 20.6: Partitive Determiner Tree

Most of the adverbs that modify determiners and NPs divide into six classes, with some minor variation within classes, based on the pattern of these restrictions. Three of the classes are adverbs that modify determiners, while the other three modify NPs.

The largest of the five classes is the class of adverbs that modify cardinal determiners. This class includes, among others, the adverbs *about*, *at most*, *exactly*, *nearly*, and *only*. These adverbs have the single restriction that they must adjoin to determiners that are **card+**. Another



(a) (b)

Figure 20.7: (a) Adverb modifying a determiner; (b) Adverb modifying a noun phrase

class of adverbs consists of those that can modify the determiners *every*, *all*, *any*, and *no*. The adverbs in this class are *almost*, *nearly*, and *practically*. Closely related to this class are the adverbs *mostly* and *roughly*, which are restricted to modifying *every* and *all*, and *hardly*, which can only modify *any*. To select for *every*, *all*, and *any*, these adverbs select for determiners that are [**quan+**, **card-**, **const+**, **compl+**], and to select for *no*, the adverbs choose a determiner that is [**quan+**, **decreas+**, **const+**]. The third class of adverbs that modify determiners are those that modify the determiners *few* and *many*, representable by the feature sequences [**quan+**, **decreas+**, **const-**] and [**quan+**, **decreas-**, **const-**, **3pl**, **compl+**], respectively. Examples of these adverbs are *awfully*, *fairly*, *relatively*, and *very*.

Of the three classes of adverbs that modify noun phrases, one actually consists of a single adverb *not*, that only modifies determiners that are **compl+**. Another class consists of the focus adverbs, *at least*, *even*, *only*, and *just*. These adverbs select NPs that are **wh-** and **card-**. For the NPs that are **card+**, the focus adverbs actually modify the cardinal determiner, and so these adverbs are also included in the first class of adverbs mentioned in the previous paragraph. The last major class that modify NPs consist of the adverbs *double* and *twice*, which select NPs that are [**definite+**] (i.e., *the*, *this/that/those/these*, and the genitives).

Although these restrictions succeed in recognizing the correct determiner/adverb sequences, a few unacceptable sequences slip through. For example, in handling the second class of adverbs mentioned above, *every*, *all*, and *any* share the features [**quan+**, **card-**, **const+**, **compl+**] with *a* and *another*, and so ∗*nearly a man* is acceptable in this system. In addition to this over-generation within a major class, the adverb *quite* selects for determiners and NPs in what seems to be a purely idiosyncratic fashion. Consider the following examples.

(394) a. **Quite** a few members of the audience had to leave.

b. There were **quite** many new participants at this year's conference.

c. **Quite** few triple jumpers have jumped that far.

d. Taking the day off was **quite** the right thing to do.

e. The recent negotiation fiasco is **quite** another issue.

f. Pandora is **quite** a cat!

In examples (394a)-(394c), *quite* modifies the determiner, while in (394d)-(394f), *quite* modifies the entire noun phrase. Clearly, it functions in a different manner in the two sets of sentences; in (394a)-(394c), *quite* intensifies the amount implied by the determiner, whereas in (394d)-(394f), it singles out an individual from the larger set to which it belongs. To capture the selectional restrictions needed for (394a)-(394c), we utilize the two sets of features mentioned previously for selecting *few* and *many*. However, *a few* cannot be singled out so easily; using the sequence [**quan+**, **card-**, **decreas-**, **const+**, **3pl**, **compl-**], we also accept the ungrammatical NPs ∗*quite several members* and ∗*quite some members* (where *quite* modifies *some*). In selecting *the* as in (d) with the features [**definite+**, **gen-**, **3sg**], *quite* also selects *this* and *that*, which are ungrammatical in this position. Examples (394e) and (394f) present yet another obstacle in that in selecting *another* and *a*, *quite* erroneously selects *every* and *any*.

It may be that there is an undiscovered semantic feature that would alleviate these difficulties. However, on the whole, the determiner feature system we have proposed can be used as a surprisingly efficient method of characterizing the interaction of adverbs with determiners and noun phrases.

Chapter 21

Modifiers

This chapter covers various types of modifiers: adverbs, prepositions, adjectives, and noun modifiers in noun-noun compounds.[1] These categories optionally modify other lexical items and phrases by adjoining onto them. In their modifier function these items are adjuncts; they are not part of the subcategorization frame of the items they modify. Examples of some of these modifiers are shown in (395)-(397).

(395) $[_{ADV}$ certainly $_{ADV}]$, the October 13 sell-off didn't settle any stomachs . (WSJ)

(396) Mr. Bakes $[_{ADV}$ previously $_{ADV}]$ had a turn at running Continental . (WSJ)

(397) most $[_{ADJ}$ foreign $_{ADJ}]$ $[_N$ government $_N]$ $[_N$ bond $_N]$ prices rose $[_{PP}$ during the week $_{PP}]$.

The trees used for the various modifiers are quite similar in form. The modifier anchors the tree and the root and foot nodes of the tree are of the category that the particular anchor modifies. Some modifiers, e.g. prepositions, select for their own arguments and those are also included in the tree. The foot node may be to the right or the left of the anchoring modifier (and its arguments) depending on whether that modifier occurs before or after the category it modifies. For example, almost all adjectives appear to the left of the nouns they modify, while prepositions appear to the right when modifying nouns.

21.1 Adjectives

In addition to being modifiers, adjectives in the XTAG English grammar can be also anchor clauses (see Adjective Small Clauses in Chapter 9). There is also one tree family, Intransitive with Adjective (Tnx0Vax1), that has an adjective as an argument and is used for sentences such as *Seth felt happy.* In that tree family the adjective substitutes into the tree rather than adjoining as is the case for modifiers.

As modifiers, adjectives anchor the tree shown in Figure 21.1. The features of the N node onto which the βAn tree adjoins are passed through to the top node of the resulting N. The null adjunction marker (NA) on the N foot node imposes right binary branching such that each

[1]Relative clauses are discussed in Chapter 16.

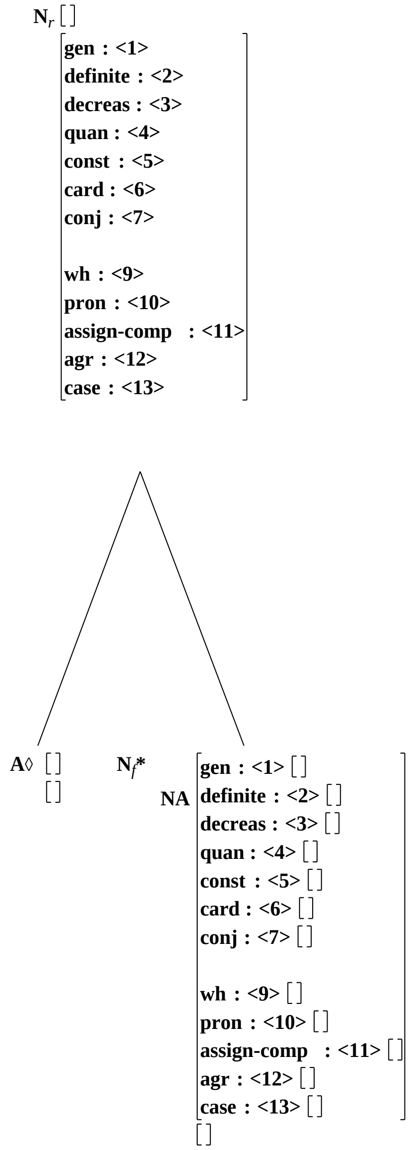


Figure 21.1: Standard Tree for Adjective modifying a Noun: βAn

subsequent adjective must adjoin on top of the leftmost adjective that has already adjoined. Due to the NA constraint, a sequence of adjectives will have only one derivation in the XTAG grammar. The adjective's morphological features such as superlative or comparative are instantiated by the morphological analyzer. See Chapter 24 for a description of how we handle comparatives. At this point, the treatment of adjectives in the XTAG English grammar does not include selectional or ordering restrictions. Consequently, any adjective can adjoin onto any noun and on top of any other adjective already modifying a noun. All of the modified noun phrases shown in (398)-(401) currently parse with the same structure shown for *colorless green ideas* in Figure 21.2.

(398) big green bugs

(399) big green ideas

(400) colorless green ideas

(401) ?green big ideas

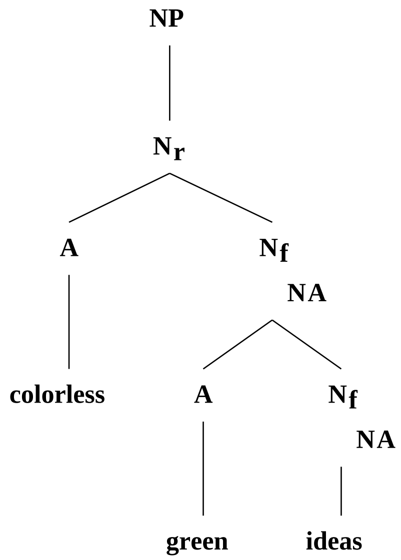


Figure 21.2: Multiple adjectives modifying a noun

While (399)-(401) are all semantically anomalous, (401) also suffers from an ordering problem that makes it seem ungrammatical in the absence of any licensing context. One could argue that the grammar should accept (398)-(400) but not (401). One of the future goals for the grammar is to develop a treatment of adjective ordering similar to that developed by [Hockey and Mateyak, 1998] for determiners[2]. An adequate implementation of ordering restrictions for adjectives would rule out (401).

21.2 Adjectival Passives and Adjectival Gerunds as Noun Modifiers

XTAG has an analysis of adjectival passives ($-ed$) (402-403) as well as adjectival gerunds ($-ing$) (404-405). The former are treated as derived from transitive verbs and are thus part of the transitive tree family Figure 21.3(a). The latter are derived from intransitives Figure 21.3(b) and ergatives (Figure 21.4)and thus have a tree in both the families.

[2]See Chapter 20 or [Hockey and Mateyak, 1998] for details of the determiner analysis.

(402) the murdered man .

(403) the devoured meal .

(404) the drinking man .

(405) the melting ice .

The feature equations **<mode>=ppart** in the βVtransn tree and the equation **<mode>=ger** in the βVintransn and the βVergativen trees forces only passive forms to be derived from the transitive verbs and only gerundive forms to be derived from the intransitive and ergative verbs.

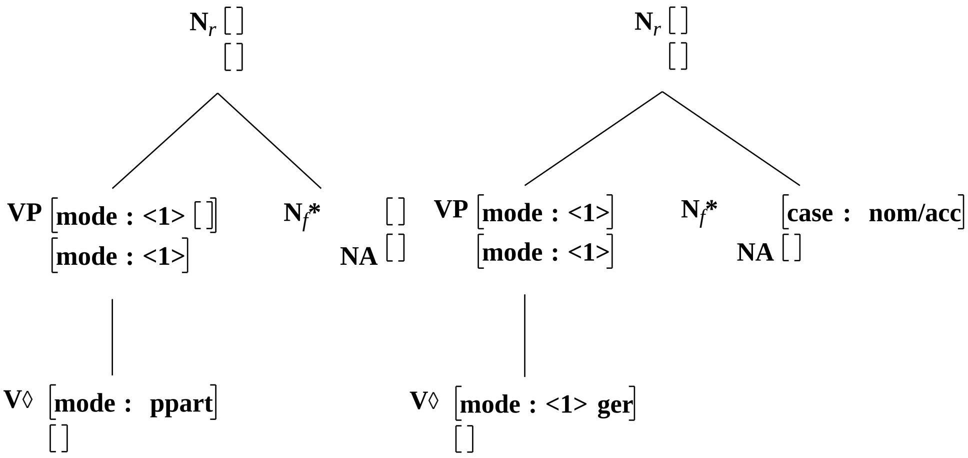


Figure 21.3: (a) βVtransn: Adjectival Passive Noun Modifiers (b) βVintransn: Adjectival Gerund Noun Modifiers

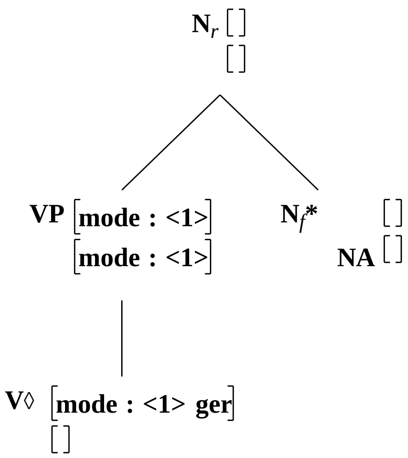


Figure 21.4: βVergativen: Adjectival Gerund Noun Modifiers

21.3 Noun-Noun Modifiers

Noun-noun compounding in the English XTAG grammar is very similar to adjective-noun modification. The noun modifier tree, shown in Figure 21.5, has essentially the same structure as the adjective modifier tree in Figure 21.1, except for the syntactic category label of the anchor.

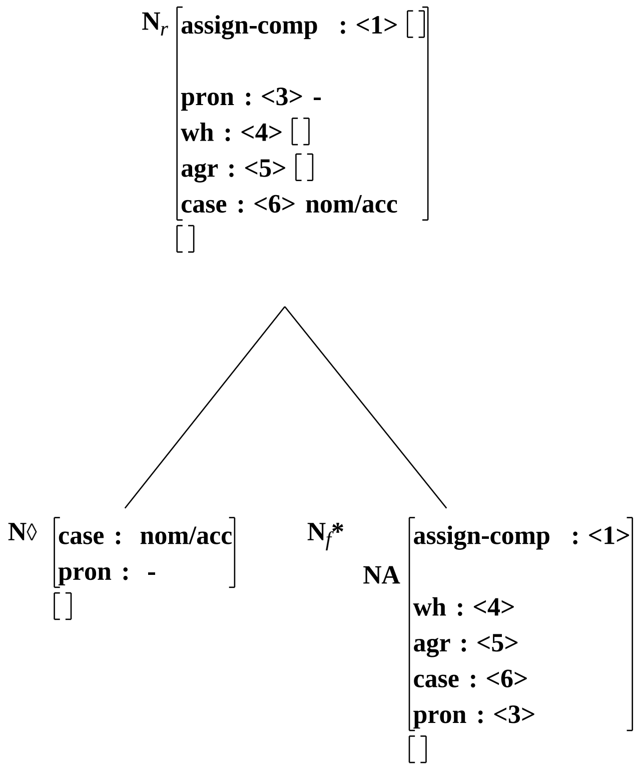


Figure 21.5: Noun-noun compounding tree: βNn (not all features displayed)

Noun compounds have a variety of scope possibilities not available to adjectives, as illustrated by the single bracketing possibility in (406) and the two possibilities for (407). This ambiguity is manifested in the XTAG grammar by the two possible adjunction sites in the noun-noun compound tree itself. Subsequent modifying nouns can adjoin either onto the N_r node or onto the N anchor node of that tree, which results in exactly the two bracketing possibilities shown in (407). This inherent structural ambiguity results in noun-noun compounds regularly having multiple derivations. However, the multiple derivations are not a defect in the grammar because they are necessary to correctly represent the genuine ambiguity of these phrases.

(406) $[_N$ big $[_N$ green design $_N]_N]$

(407) $[_N$ computer $[_N$ furniture design $_N]_N]$
$[_N$ $[_N$ computer furniture $_N]$ design $_N]$

Noun-noun compounds have no restriction on number. XTAG allows nouns to be either singular or plural as in (408)-(410).

(408) Hyun is taking an algorithms course .

(409) waffles are in the frozen foods section .

(410) I enjoy the dog shows .

21.4 Time Noun Phrases

Although in general NPs cannot modify clauses or other NPs, there is a class of NPs, with meanings that relate to time, that can do so.[3] We call this class of NPs "Time NPs". Time NPs behave essentially like PPs. Like PPs, time NPs can adjoin at four places: to the right of an NP, to the right and left of a VP, and to the left of an S.

Time NPs may include determiners, as in *this week* in example (411), or may be single lexical items as in *today* in example (412). Like other NPs, time NPs can also include adjectives, as in example (416).

(411) Elvis left the building this week

(412) Elvis left the building today

(413) It has no bearing on our work force today (WSJ)

(414) The fire yesterday claimed two lives

(415) Today it has no bearing on our work force

(416) Michael late yesterday announced a buy-back program

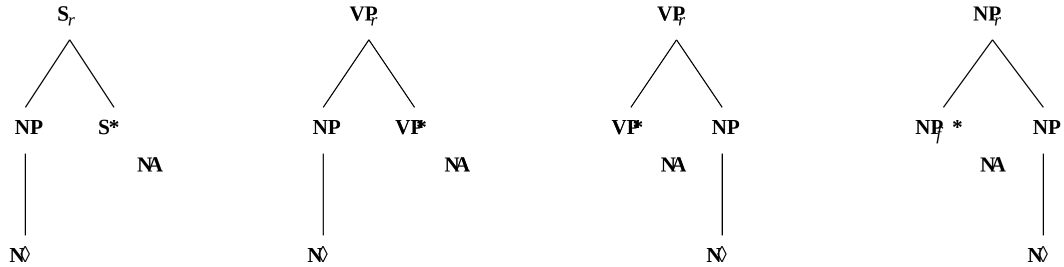
The XTAG analysis for time NPs is fairly simple, requiring only the creation of proper NP auxiliary trees. Only nouns that can be part of time NPs will select the relevant auxiliary trees, and so only these type of NPs will behave like PPs under the XTAG analysis. Currently, about 60 words select Time NP trees, but since these words can form NPs that include determiners and adjectives, a large number of phrases are covered by this class of modifiers.

Corresponding to the four positions listed above, time NPs can select one of the four trees shown in Figure 21.6.

Determiners can be added to time NPs by adjunction in the same way that they are added to NPs in other positions. The trees in Figure 21.7 show that the structures of examples (411) and (412) differ only in the adjunction of *this* to the time NP in example (411).

The sentence

(417) Esso said the Whiting field started production Tuesday (WSJ)



βNs βNvx βvxN βnxN

Figure 21.6: Time Phrase Modifier trees: βNs, βNvx, βvxN, βnxN

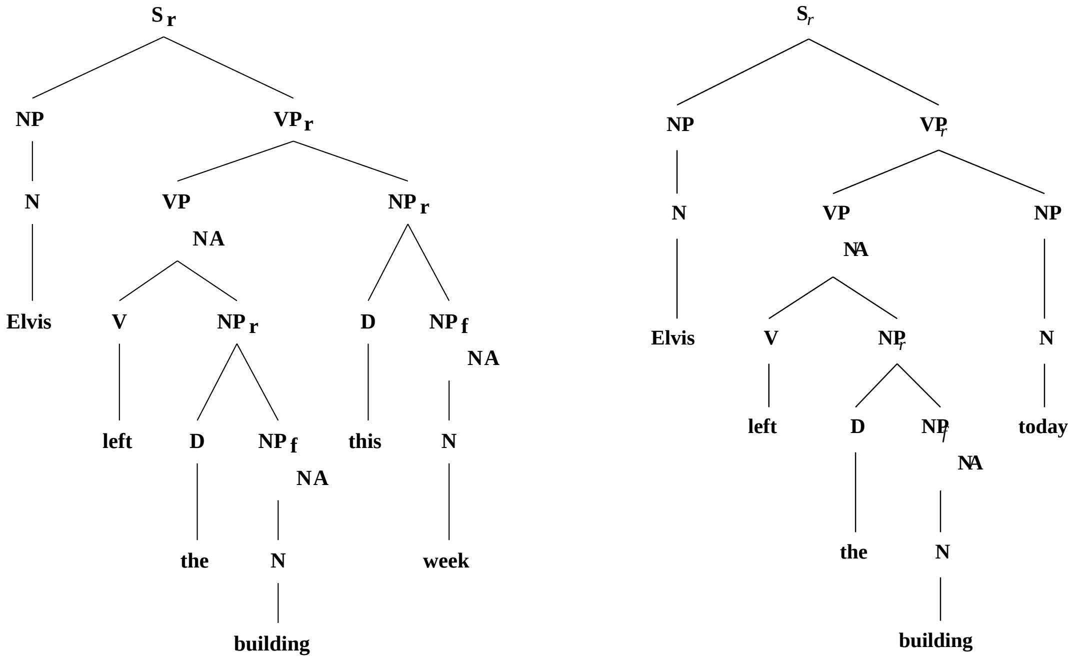


Figure 21.7: Time NPs with and without a determiner

has (at least) two different interpretations, depending on whether *Tuesday* attaches to *said* or to *started*. Valid time NP analyses are available for both these interpretations and are shown in Figure 21.8.

Derived tree structures for examples (413) – (416), which show the four possible time NP positions are shown in Figures 21.9 and 21.10. The derivation tree for example (416) is also shown in Figure 21.10.

[3]There may be other classes of NPs, such as directional phrases, such as *north, south* etc., which behave similarly. We have not yet analyzed these phrases.

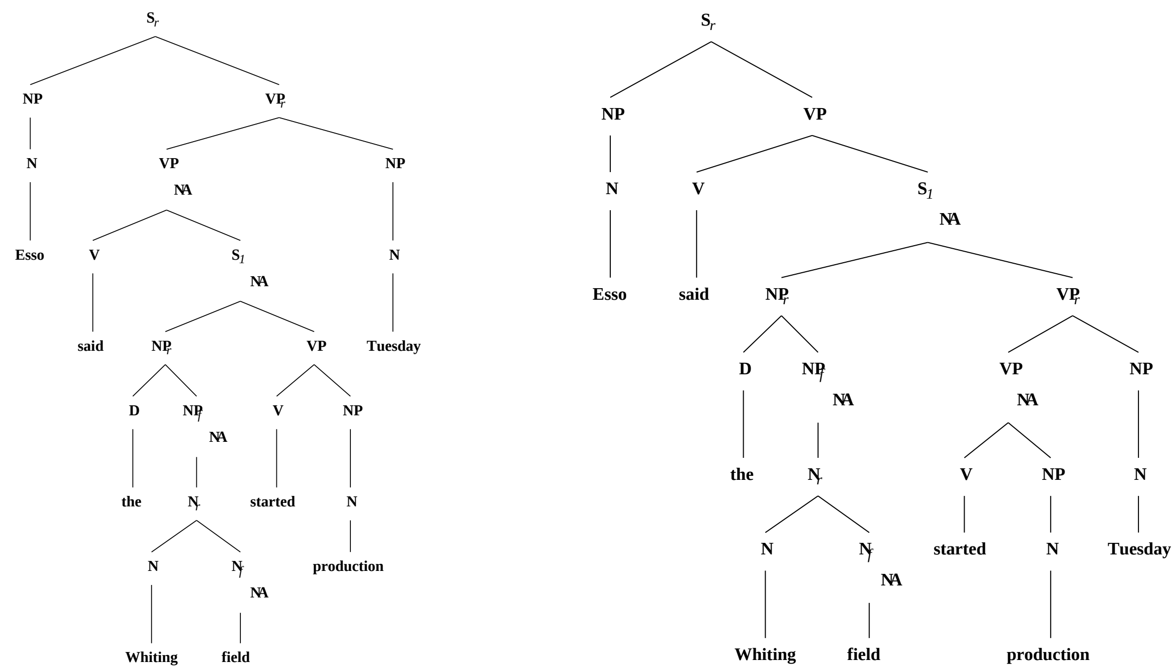


Figure 21.8: Time NP trees: Two different attachments

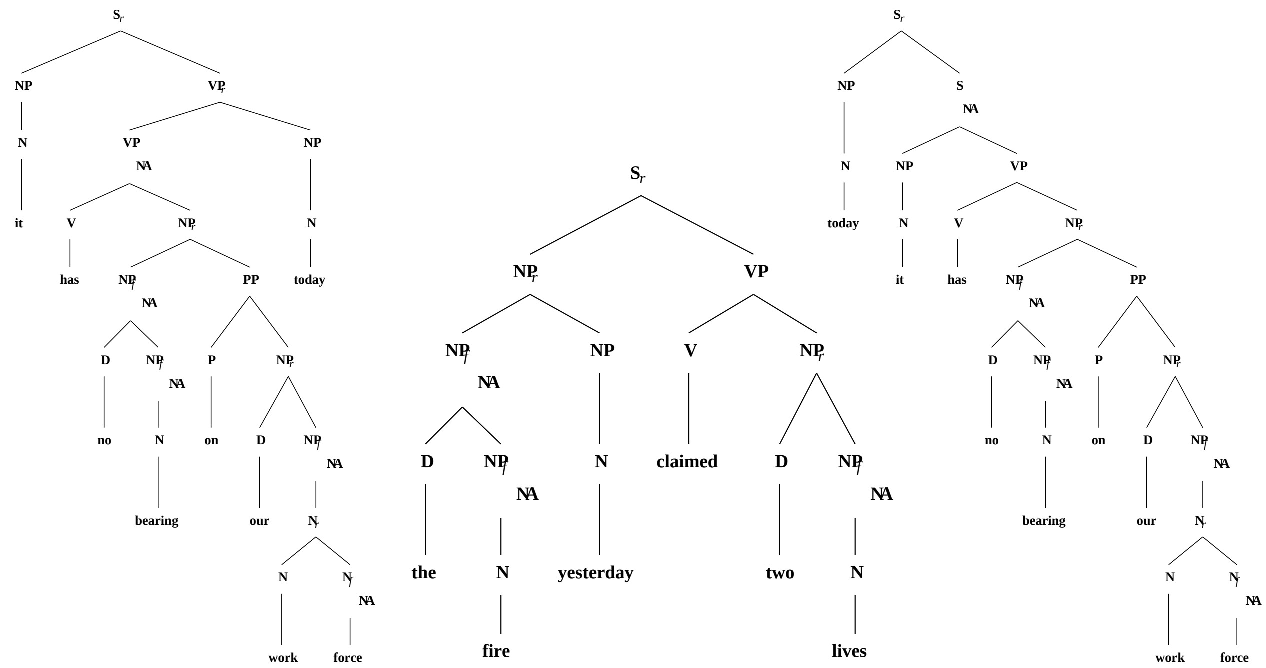


Figure 21.9: Time NPs in different positions (βvxN, βnxN and βNs)

21.5 Prepositions

There are three basic types of prepositional phrases, and three places at which they can adjoin. The three types of prepositional phrases are: Preposition with NP Complement, Preposition

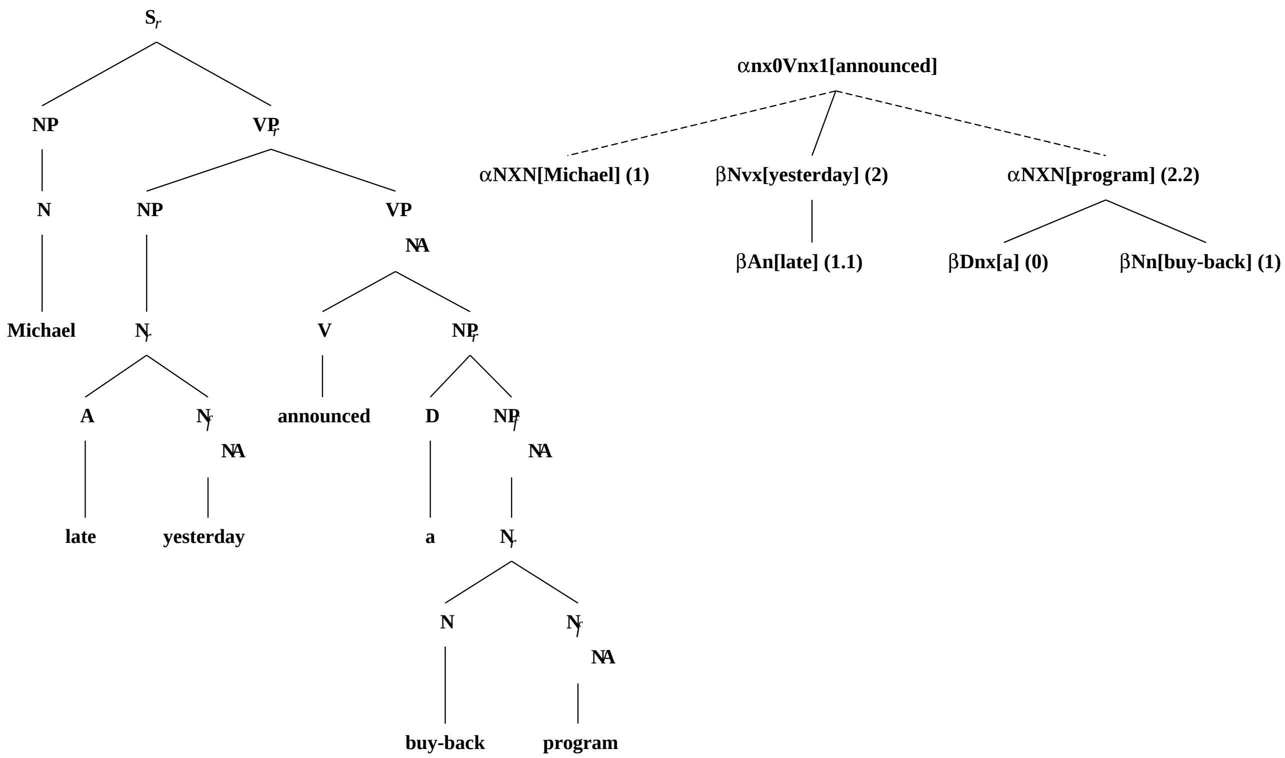


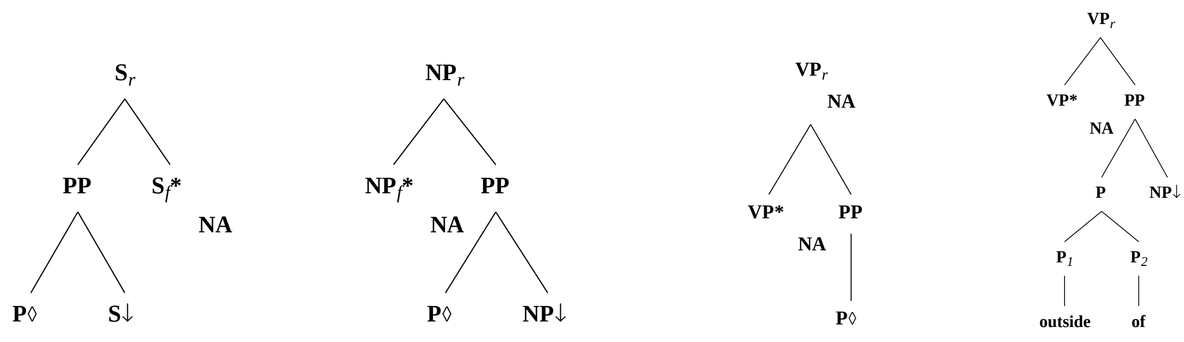
Figure 21.10: Time NPs: Derived tree and Derivation (βNvx position)

with Sentential Complement, and Exhaustive Preposition. The three places are to the right of an NP, to the right of a VP, and to the left of an S. Each of the three types of PP can adjoin at each of these three places, for a total of nine PP modifier trees. Table 21.1 gives the tree family names for the various combinations of type and location. (See Section 25.4.2 for discussion of the βspuPnx, which handles post-sentential comma-separated PPs.)

	position and category modified		
Complement type	pre-sentential S modifier	post-NP NP modifier	post-VP VP modifier
S-complement	βPss	βnxPs	βvxPs
NP-complement	βPnxs	βnxPnx	βvxPnx
no complement (exhaustive)	βPs	βnxP	βvxP

Table 21.1: Preposition Anchored Modifiers

The subset of preposition anchored modifier trees in Figure 21.11 illustrates the locations and the four PP types. Example sentences using the trees in Figure 21.11 are shown in (418)-(421). There are also more trees with multi-word prepositions as anchors. Examples of these are: *ahead of*, *contrary to*, *at variance with* and *as recently as*.



βPss βnxPnx βvxP βvxPPnx

Figure 21.11: Selected Prepositional Phrase Modifier trees: βPss, βnxPnx, βvxP and βvxPPnx

(418) [$_{PP}$ with Clove healthy $_{PP}$], the veterinarian's bill will be more affordable . (βPss[4])

(419) The frisbee [$_{PP}$ in the brambles $_{PP}$] was hidden . (βnxPnx)

(420) Clove played frisbee [$_{PP}$ outside $_{PP}$] . (βvxP)

(421) Clove played frisbee [$_{PP}$ outside of the house $_{PP}$] . (βvxPPnx)

Prepositions that take NP complements assign accusative case to those complements. Most prepositions take NP complements. Subordinating conjunctions are analyzed in XTAG as Preps (see Section 17 for details). Additionally, a few non-conjunction prepositions take S complements (see Section 8.8 for details).

21.6 Adverbs

In the English XTAG grammar, VP and S-modifying adverbs anchor the auxiliary trees βvxARB and βARBvx, βARBs, βsARB,[5] allowing pre and post modification of S's and VP's. Besides the VP and S-modifying adverbs, the grammar includes adverbs that modify other categories. Examples of adverbs modifying an adjective, an adverb, a PP, an NP, and a determiner are shown in (422)-(429). (See Sections 25.1.5 and 25.4.1 for discussion of the βpuARBpuvx and βspuARB, which handle pre-verbal parenthetical adverbs and post-sentential comma-separated adverbs.)

- Modifying an adjective

  (422) **extremely** good

  (423) **rather** tall

---

[4] *Clove healthy* is an adjective small clause

[5] In the naming conventions for the XTAG trees, ARB is used for <u>a</u>dve<u>rb</u>s. Because the letters in A, Ad, and Adv are all used for other parts of speech (<u>a</u>djective, <u>d</u>eterminer and <u>v</u>erb), ARB was chosen to eliminate ambiguity. Appendix 29 contains a full explanation of naming conventions.

(424) rich **enough**

- Modifying an adverb

  (425) oddly **enough**

  (426) **very** well

- Modifying a PP

  (427) **right** through the wall

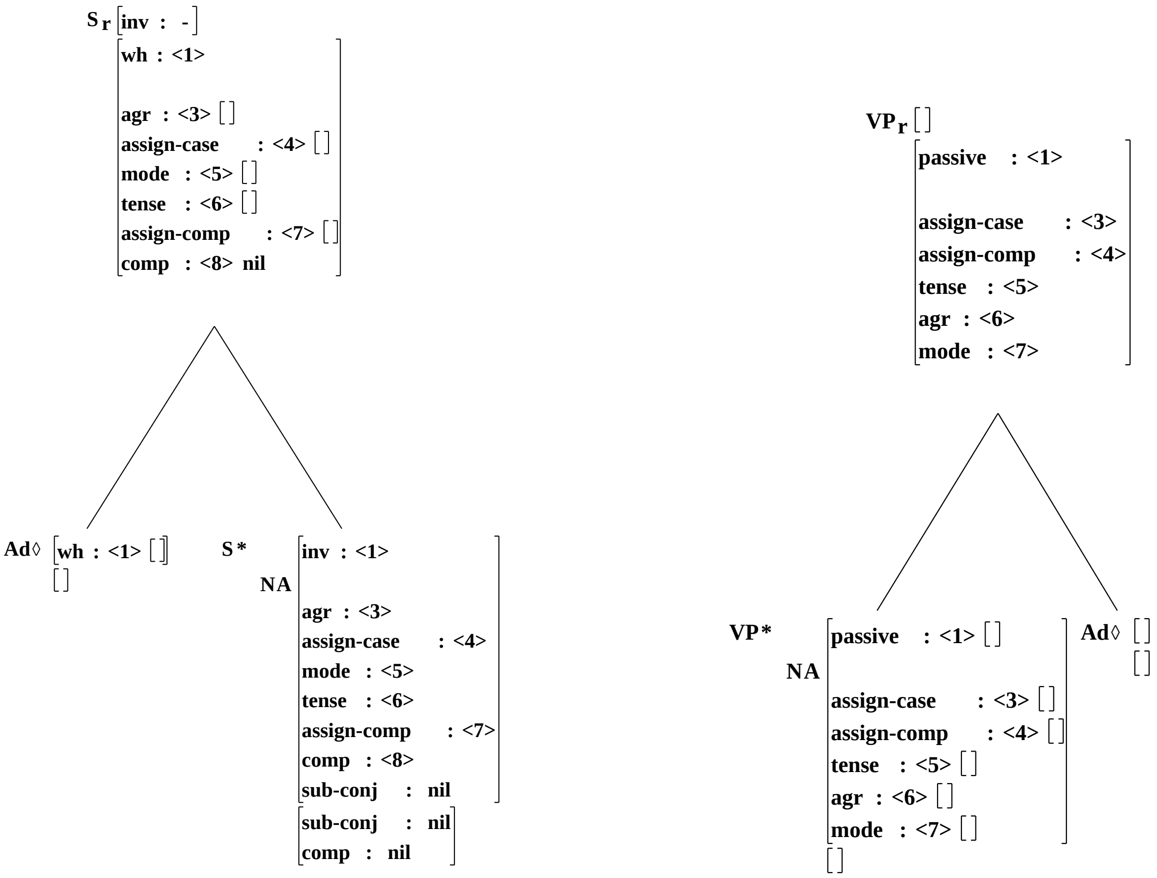
- Modifying a NP

  (428) **quite** some time

- Modifying a determiner

  (429) **exactly** five men

XTAG has separate trees for each of the modified categories and for pre and post modification where needed. The kind of treatment given to adverbs here is very much in line with the base-generation approach proposed by [Ernst, 1983], which assumes all positions where an adverb can occur to be base-generated, and that the semantics of the adverb specifies a range of possible positions occupied by each adverb. While the relevant semantic features of the adverbs are not currently implemented, implementation of semantic features is scheduled for future work. The trees for adverb anchored modifiers are very similar in form to the adjective anchored modifier trees. Examples of two of the basic adverb modifier trees are shown in Figure 21.12.



(a) (b)

Figure 21.12: Adverb Trees for pre-modification of S: βARBs (a) and post-modification of a VP: βvxARB (b)

Like the adjective anchored trees, these trees also have the NA constraint on the foot node to restrict the number of derivations produced for a sequence of adverbs. Features of the modified category are passed from the foot node to the root node, reflecting correctly that these types of properties are unaffected by the adjunction of an adverb. A summary of the categories modified and the position of adverbs is given in Table 21.2.

In the English XTAG grammar, no traces are posited for wh-adverbs, in-line with the base-generation approach ([Ernst, 1983]) for various positions of adverbs. Since convincing arguments have been made against traces for adjuncts of other types (e.g. [Baltin, 1989]), and since the reasons for wanting traces do not seem to apply to adjuncts, we make the general assumption in our grammar that adjuncts do not leave traces. Sentence initial wh-adverbs select the same auxiliary tree used for other sentence initial adverbs (βARBs) with the feature **<wh>=+**. Under this treatment, the derived tree for the sentence *How did you fall?* is as in Figure (21.13), with no trace for the adverb.

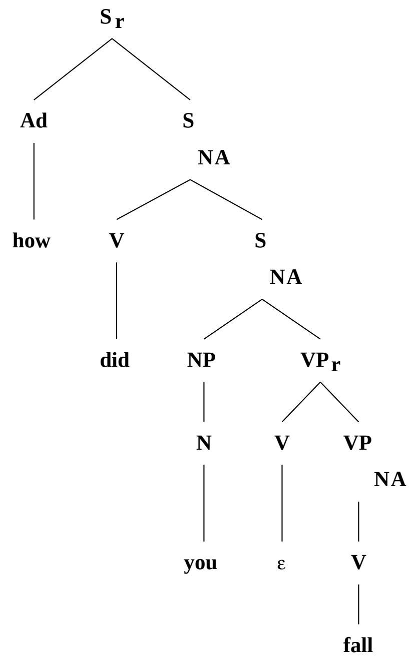


Figure 21.13: Derived tree for *How did you fall?*

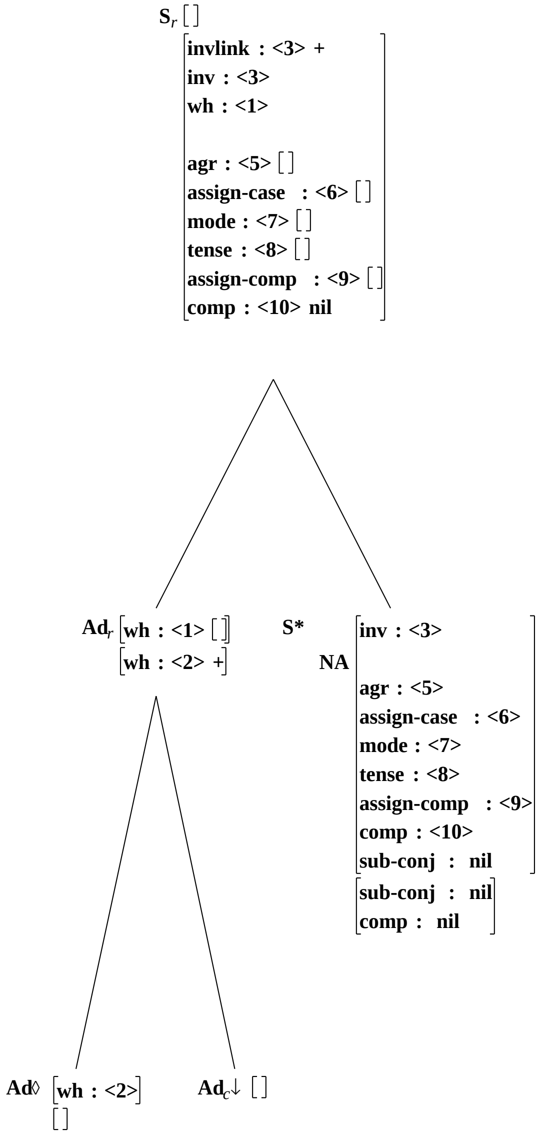


Figure 21.14: Complex adverb phrase modifier: βARBarbs

Category Modified	Position with respect to item modified	
	Pre	Post
S	βARBs	βsARB
VP	βARBvx,βpuARBpuvx	βvxARB
A	βARBa	βaARB
PP	βARBpx	βpxARB
ADV	βARBarb	βarbARB
NP	βARBnx	
Det	βARBd	

Table 21.2: Simple Adverb Anchored Modifiers

There is one more adverb modifier tree in the grammar which is not included in Table 21.2. This tree, shown in Figure 21.14, has a complex adverb phrase and is used for wh+ two-adverb phrases that occur sentence initially, such as in sentence (430). Since *how* is the only wh+ adverb, it is the only adverb that can anchor this tree.

(430) how quickly did Srini fix the problem ?

Focus adverbs such as *only*, *even*, *just* and *at least* are also handled by the system. Since the syntax allows focus adverbs to appear in practically any position, these adverbs select most of the trees listed in Table 21.2. It is left up to the semantics or pragmatics to decide the correct scope for the focus adverb for a given instance. In terms of the ability of the focus adverbs to modify at different levels of a noun phrase, the focus adverbs can modify either cardinal determiners or noun-cardinal noun phrases, and cannot modify at the level of noun. The tree for adverbial modification of noun phrases is in shown Figure 21.15(a).

In addition to *at least*, the system handles the other two-word adverbs, *at most* and *up to*, and the three-word *as-as* adverb constructions, where an adjective substitutes between the two occurrences of *as*. An example of a three-word *as-as* adverb is *as little as*. Except for the ability of *at least* to modify many different types of constituents as noted above, the multi-word adverbs are restricted to modifying cardinal determiners. Example sentences using the trees in Figure 21.15 are shown in (431)-(435).

- Focus Adverb modifying an NP

  (431) **only** a member of our crazy family could pull off that kind of a stunt .

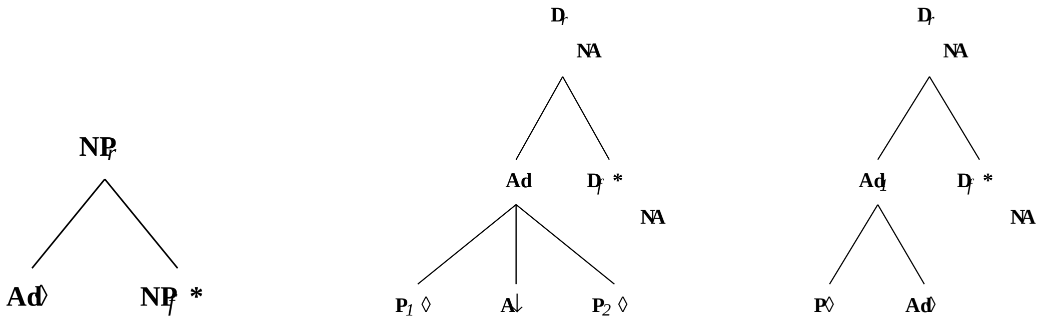
  (432) **even** a flying saucer sighting would seem interesting in comparison with your story .

  (433) The report includes a proposal for **at least** a partial impasse in negotiations .

- Multi-word adverbs modifying cardinal determiners

  (434) **at most** ten people came to the party .

  (435) They gave monetary gifts of **as little as** five dollars .



βARBnx
(a)

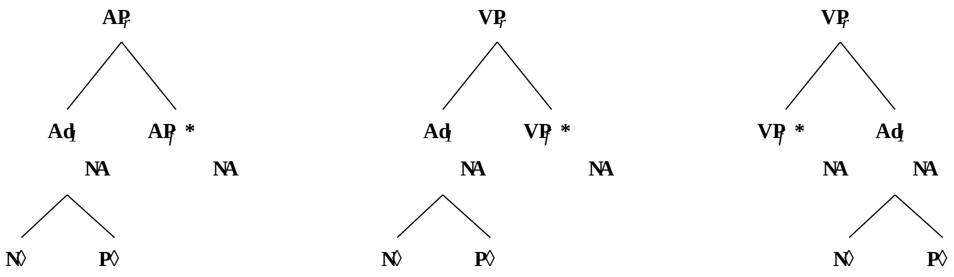
βPaPd
(b)

βPARBd
(c)

Figure 21.15: Selected Focus and Multi-word Adverb Modifier trees: βARBnx, βPaPd and βPARBd

The grammar also includes auxiliary trees anchored by multi-word adverbs like *a little*, *a bit*, *a mite*, *sort of*, *kind of*, etc..

Multi-word adverbs like *sort of* and *kind of* can pre- modify almost any non-clausal category. The only strict constraint on their occurrence is that they can't modify nouns (in which case an adjectival interpretation would obtain)[6]. The category which they scope over can be directly determined from their position, except for when they occur sentence finally in which case they are assumed to modify VP's. The complete list of auxiliary trees anchored by these adverbs are as follows: βNPax, βNPpx, βNPnx, βNPvx, βvxNP, βNParb. Selected trees are shown in Figure 21.16, and some examples are given in (436)-(439).



βNPax
(a)

βNPvx
(b)

βvxNP
(c)

Figure 21.16: Selected Multi-word Adverb Modifier trees (for adverbs like *sort of*, *kind of*): βNPax, βNPvx, βvxNP.

(436) John is **sort of** [$_{AP}$ tired].

(437) John is **sort of** [$_{PP}$ to the right].

[6]Note that there are semantic/lexical constraints even for the categories that these adverbs *can* modify, and no doubt invite a more in-depth analysis.

(438) John could have been **sort of** [$_{VP}$ eating the cake].

(439) John has been eating his cake **sort of** [$_{ADV}$ slowly].

There are some multi-word adverbs that are, however, not so free in their distribution. Adverbs like *a little*, *a bit*, *a mite* modify AP's in predicative constructions (sentences with the copula and small clauses, AP complements in sentences with raising verbs, and AP's when they are subcategorized for by certain verbs (e.g., *John felt angry*). They can also post-modify VP's and PP's, though not as freely as AP's[7]. Finally, they also function as downtoners for almost all adverbials[8]. Some examples are provided in (440)-(443).

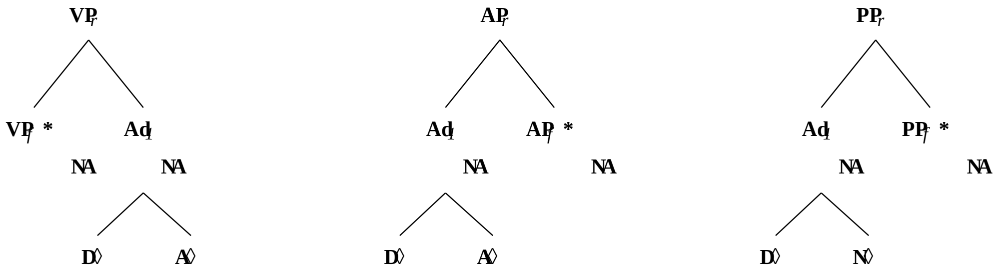
(440) Mickey is **a little** [$_{AP}$ tired].

(441) The medicine [$_{VP}$ has eased John's pain] **a little**.

(442) John is **a little** [$_{PP}$ to the right].

(443) John has been reading his book **a little** [$_{ADV}$ loudly].

Following their behavior as described above, the auxiliary trees they anchor are βDAax, βDApx, βvxDA, βDAarb, βDNax, βDNpx, βvxDN, βDNarb. Selected trees are shown in Figure 21.17).



βvxDA (a) βDAax (b) βDNpx (c)

Figure 21.17: Selected Multi-word Adverb Modifier trees (for adverbs like *a little*, *a bit*): βvxDA, βDAax, βDNpx.

21.7 Locative Adverbial Phrases

Locative adverbial phrases are multi-word adverbial modifiers whose meanings relate to spatial location. Locatives consist of a locative adverb (such as *ahead* or *downstream*) preceded by

[7] They can also appear before NP's, as in, "John wants *a little* sugar". However, here they function as multi-word determiners and should not be analyzed as adverbs.

[8] It is to be noted that this analysis, which allows these multiword adverbs to modify adjectival phrases as well as adverbials, will yield (not necessarily desirable) multiple derivations for a sentence like *John is a little unecessarily stupid.* In one derivation, *a little* modifies the AP and in the other case, it modifies the adverb.

an NP, an adverb, or nothing, as in Examples (444)–(446) respectively. The modifier as a whole describes a position relative to one previously specified in the discourse. The nature of the relation, which is usually a direction, is specified by the anchoring locative adverb(*behind, east*). If an NP or a second adverb is present in the phrase, it specifies the degree of the relation (for example: *three city blocks, many meters,* and *far*).

(444) The accident *three blocks ahead* stopped traffic

(445) The ship sank *far offshore*

(446) The trouble *ahead* distresses me

Locatives can modify NPs, VPs and Ss. They modify NPs only by right-adjoining post-positively, as in Example (444). Post-positive is also the more common position when a locative modifies either of the other categories. Locatives pre-modify VPs only when separated by balanced punctuation (commas or dashes). The trees locatives select when modifying NPs are shown in Figure 21.18.

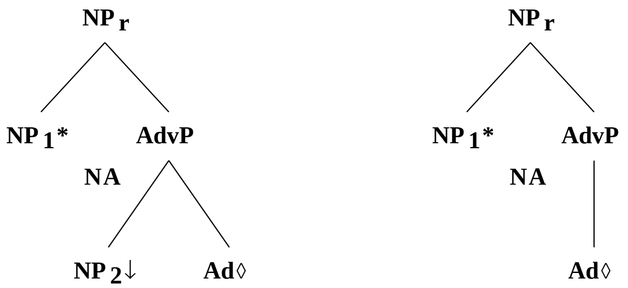


Figure 21.18: Locative Modifier Trees: βnxnxARB, βnxARB

When the locative phrase consists of only the anchoring locative adverb, as in Example (445), it uses the βnxARB tree, shown in Figure 21.18, and its VP analogue, βvxARB. In addition, these are the trees selected when the locative anchor is modified by an adverb expressing degree, as in Example 445. The degree adverb adjoins on to the anchor using the βARBarb tree, which is described in Section 21.6. Figure 21.19 shows an example of these trees in action.

One possible analysis of locative phrases with NPs might maintain that the NP is the head, with the locative adverb modifying the NP. This is initially attractive because of the similarity to time NPs, which also feature NPs that can modify clauses. This analysis seems insufficient, however, in light of the fact that virtually any NP can occur in locative phrases, as in example (447). Therefore, in the XTAG analysis the locative adverb anchors the locative phrase trees. A complete summary of all trees selected by locatives is contained in Table 21.3. 26[9] adverbs select the locative trees.

(447) I left my toupee and putter *three holes back*

[9]Though nearly all of these adverbs are spatial in nature, this number also includes a few temporal adverbs, such as *ago*, that also select these trees.

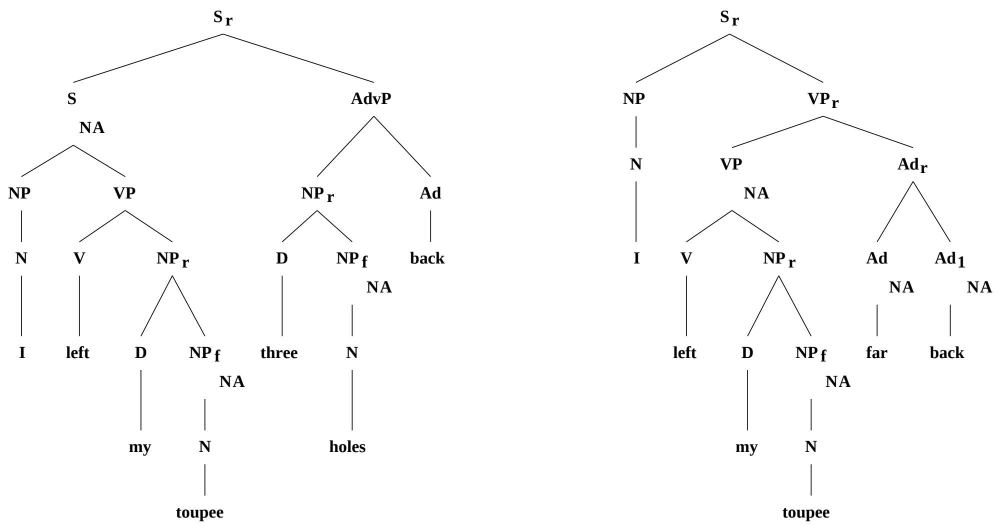


Figure 21.19: Locative Phrases featuring NP and Adverb Degree Specifications

Category Modified	Degree Phrase Type: NP	Degree Phrase Type: Ad/None
NP	βnxnxARB	βnxARB
VP (post)	βvxnxARB	βvxARB
VP (pre, punct-separated)	βpunxARBpuvx	βpuARBpuvx
S	βnxARBs	βARBs

Table 21.3: Locative Modifiers

Chapter 22

Auxiliaries

Although there has been some debate about the lexical category of auxiliaries, the English XTAG grammar follows [McCawley, 1988], [Haegeman, 1991], and others in classifying auxiliaries as verbs. The category of verbs can therefore be divided into two sets, main or lexical verbs, and auxiliary verbs, which can co-occur in a verbal sequence. Only the highest verb in a verbal sequence is marked for tense and agreement regardless of whether it is a main or auxiliary verb. Some auxiliaries (*be*, *do*, and *have*) share with main verbs the property of having overt morphological marking for tense and agreement, while the modal auxiliaries do not. However, all auxiliary verbs differ from main verbs in several crucial ways.

- Multiple auxiliaries can occur in a single sentence, while a matrix sentence may have at most one main verb.
- Auxiliary verbs cannot occur as the sole verb in the sentence, but must be followed by a main verb.
- All auxiliaries precede the main verb in verbal sequences.
- Auxiliaries do not subcategorize for any arguments.
- Auxiliaries impose requirements on the morphological form of the verbs that immediately follow them.
- Only auxiliary verbs invert in questions (with the sole exception in American English of main verb *be*[1]).
- An auxiliary verb must precede sentential negation (e.g. $*$*John not goes*).
- Auxiliaries can form contractions with subjects and negation (e.g. *he'll*, *won't*).

The restrictions that an auxiliary verb imposes on the succeeding verb limits the sequence of verbs that can occur. In English, sequences of up to five verbs are allowed, as in sentence (448).

[1]Some dialects, particularly British English, can also invert main verb *have* in yes/no questions (e.g. *have you any Grey Poupon ?*). This is usually attributed to the influence of auxiliary *have*, coupled with the historic fact that English once allowed this movement for all verbs.

(448) The music should have been being played [for the president] .

The required ordering of verb forms when all five verbs are present is:

modal base perfective progressive passive

The rightmost verb is the main verb of the sentence. While a main verb subcategorizes for the arguments that appear in the sentence, the auxiliary verbs select the particular morphological forms of the verb to follow each of them. The auxiliaries included in the English XTAG grammar are listed in Table 22.1 by type. The third column of Table 22.1 lists the verb forms that are required to follow each type of auxiliary verb.

TYPE	LEX ITEMS	SELECTS FOR
modals	*can*, *could*, *may*, *might*, *will*, *would*, *ought*, *shall*, *should* *need*	base form[2] (e.g. *will go*, *might come*)
perfective	*have*	past participle (e.g. *has gone*)
progressive	*be*	gerund (e.g. *is going*, *was coming*)
passive	*be*	past participle (e.g. *was helped by Jane*)
do support	*do*	base form (e.g. *did go*, *does come*)
infinitive to	*to*	base form (e.g. *to go*, *to come*)

Table 22.1: Auxiliary Verb Properties

22.1 Non-inverted sentences

This section and the sections that follow describe how the English XTAG grammar accounts for properties of the auxiliary system described above.

In our grammar, auxiliary trees are added to the main verb tree by adjunction. Figure 22.1 shows the adjunction tree for non-inverted sentences.

The restrictions outlined in column 3 of Table 22.1 are implemented through the features <**mode**>, <**perfect**>, <**progressive**> and <**passive**>. The syntactic lexicon entries for the auxiliaries give values for these features on the foot node (VP*) in Figure 22.1. Since the top features of the foot node must eventually unify with the bottom features of the node it adjoins onto for the sentence to be valid, this enforces the restrictions made by the auxiliary node. In

[2]There are American dialects, particularly in the South, which allow double modals such as *might could* and *might should*. These constructions are not allowed in the XTAG English grammar.

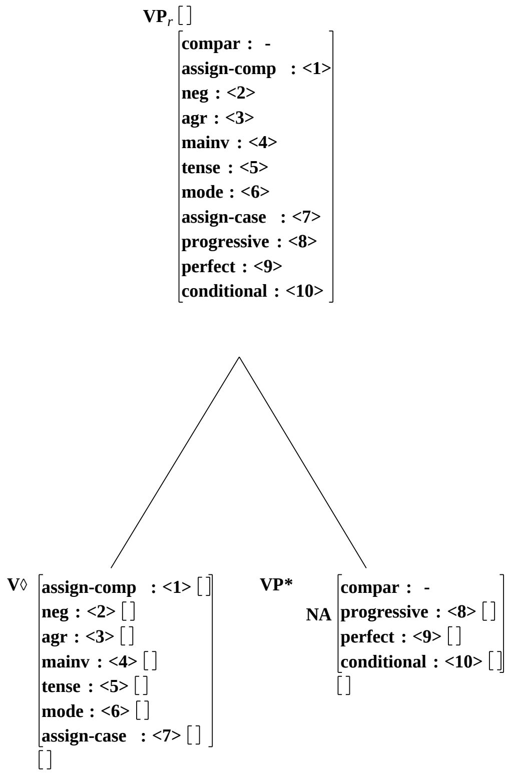


Figure 22.1: Auxiliary verb tree for non-inverted sentences: βVvx

addition to these feature values, each auxiliary also gives values to the anchoring node (V$\diamond$), to be passed up the tree to the root VP (VP$_r$) node; there they will become the new features for the top VP node of the sentential tree. Another auxiliary may now adjoin on top of it, and so forth. These feature values thereby ensure the proper auxiliary sequencing. Figure 22.2 shows the auxiliary trees anchored by the four auxiliary verbs in sentence (448). Figure 22.3 shows the final tree created for this sentence.

The general English restriction that matrix clauses must have tense (or be imperatives) is enforced by requiring the top S-node of a sentence to have **<mode>=ind/imp** (indicative or imperative). Since only the indicative and imperative sentences have tense, non-tensed clauses are restricted to occurring in embedded environments.

Noun-verb contractions are labeled NVC in their part-of-speech field in the morphological database and then undergo special processing to split them apart into the noun and the reduced verb before parsing. The noun then selects its trees in the normal fashion. The contraction, say *'ll* or *'d*, likewise selects the normal auxiliary verb tree, βVvx. However, since the contracted form, rather than the verb stem, is given in the morphology, the contracted form must also be listed as a separate syntactic entry. These entries have all the same features of the full form of the auxiliary verbs, with tense constraints coming from the morphological entry (e.g. *it's* is listed as IT 'S NVC 3SG PRES). The ambiguous contractions *'d* (*had/would*) and *'s* (*has/is*) behave like other ambiguous lexical items; there are simply multiple entries for those lexical

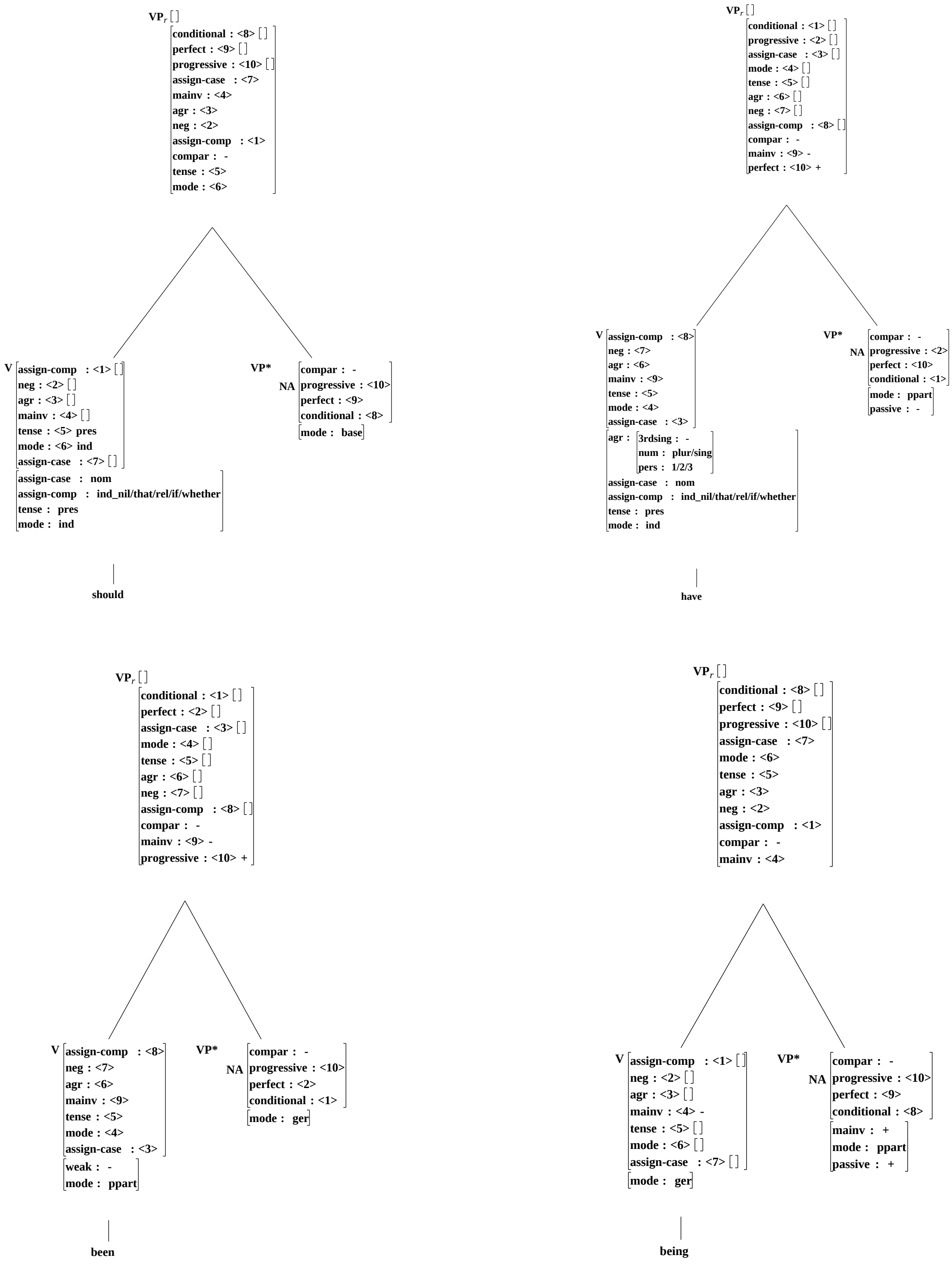


Figure 22.2: Auxiliary trees for *The music should have been being played .*

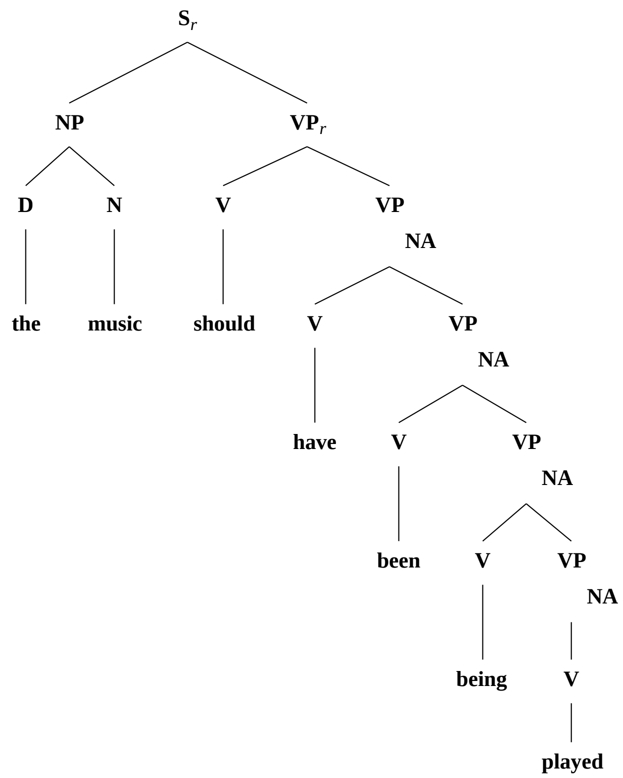


Figure 22.3: *The music should have been being played .*

items in the lexicon, each with different features. In the resulting parse, the contracted form is shown with features appropriate to the full auxiliary it represents.

22.2 Inverted Sentences

In main clause yes-no questions and non-subject-extracted wh-questions in English, the first (matrix) auxiliary appears before the subject, as shown in 449 and 450.

(449) Will John buy a backpack?

(450) What will John buy?

In this case, the tree anchored by the auxiliary verb will adjoin to S, rather than VP. The relevant tree is shown in Figure 22.4 Figure 22.5 shows this tree (anchored by *will*) and adjoined to the declarative transitive tree[3] which is anchored by main verb *buy*.

22.3 Do-Support

As discussed in the previous section, subject-auxiliary inversion targets the highest auxiliary in the clause. Similarly, negation always occurs after the highest auxiliary. When there is

[3]The declarative transitive tree was seen in section 6.3.

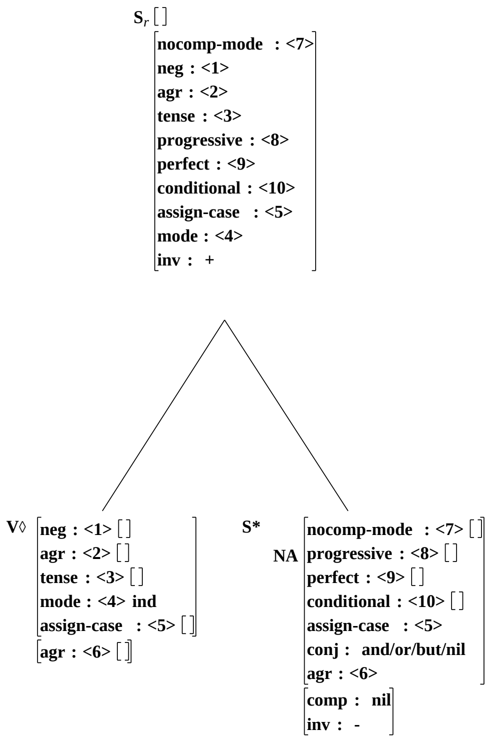


Figure 22.4: Tree for auxiliary verb inversion: βVs

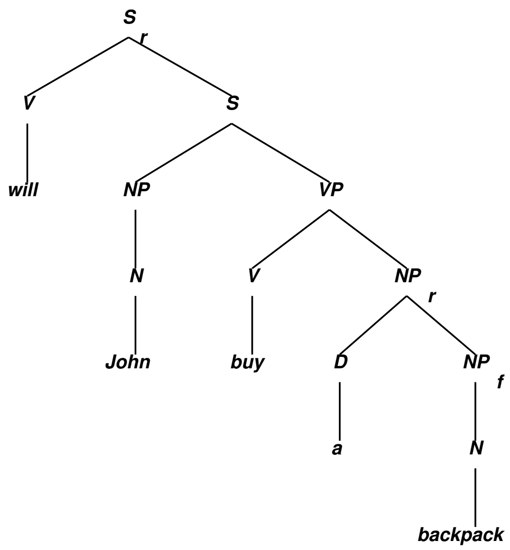


Figure 22.5: *will John buy a backpack ?*

no auxiliary verb in the clause, a dummy auxiliary *do* is inserted, a mechanism known as 'do-support'. It is this dummy auxiliary that undergoes inversion in questions and supports negation in negative sentences.

(451) John bought a backpack .

(452) ∗Bought John a backpack ?

(453) Did John buy a packpack ?

(454) ∗John bought not a backpack .

(455) ∗John not bought a backpack .

(456) John did not buy a backpack .

22.3.1 In negated sentences

The GB analysis of do-support in negated sentences hinges on the separation of the INFL and VP nodes (see [Chomsky, 1965], [Jackendoff, 1972] and [Chomsky, 1986]). The claim is that the presence of the negative morpheme blocks the main verb from getting tense from the INFL node, thereby forcing the addition of a verbal lexeme to carry the inflectional elements. If an auxiliary verb is present, then it carries tense, but if not, periphrastic or 'dummy', *do* is required. This seems to indicate that *do* and other auxiliary verbs would not co-occur, and indeed this is the case (see sentences (457)-(458)). Auxiliary *do* is allowed in English when no negative morpheme is present, but this usage is marked as emphatic. Emphatic *do* is also not allowed to co-occur with auxiliary verbs (sentences (459)-(462)).

(457) ∗We will have do bought a sleeping bag .

(458) ∗We do will have bought a sleeping bag .

(459) You **do** have a backpack, don't you ?

(460) I **do** want to go !

(461) ∗You **do** can have a backpack, don't you ?

(462) ∗I **did** have had a backpack !

In the XTAG grammar, *do* is prevented from co-occurring with other auxiliary verbs by a requirement that it adjoin only onto main verbs (<**mainv**> = +). It has indicative mode, so no other auxiliaries can adjoin above it.[4] The lexical item *not* is only allowed to adjoin onto a non-indicative (and therefore non-tensed) verb. Since all matrix clauses must be indicative (or imperative), a negated sentence will fail unless an auxiliary verb, either *do* or another auxiliary, adjoins somewhere above the negative morpheme, *not*. In addition to forcing adjunction of an auxiliary, this analysis of *not* allows it freedom to move around in the auxiliaries, as seen in the sentences (463)-(466).

(463) John will have had a backpack .

(464) ∗John not will have had a backpack .

(465) John will not have had a backpack .

(466) John will have not had a backpack .

22.3.2 In inverted yes/no questions

In inverted yes/no questions, *do* is required if there is no auxiliary verb to invert, as seen in sentences (454)-(456), replicated here as (467)-(469).

(467) do you want to leave home ?

(468) ∗want you to leave home ?

(469) will you want to leave home ?

(470) ∗do you will want to leave home ?

[4]Earlier, we said that indicative mode carries tense with it. Since only the topmost auxiliary carries the tense, any subsequent verbs must **not** have indicative mode.

In English, unlike other Germanic languages, the main verb cannot move to the beginning of a clause, with the exception of main verb *be*.[5] In a GB account of inverted yes/no questions, the tense feature is said to be in C^0 at the front of the sentence. Since main verbs cannot move, they cannot pick up the tense feature, and do-support is again required if there is no auxiliary verb to perform the role. Sentence (470) shows that *do* does not interact with other auxiliary verbs, even when in the inverted position.

In XTAG, trees anchored by a main verb that lacks tense are required to have an auxiliary verb adjoin onto them, whether at the VP node to form a declarative sentence, or at the S node to form an inverted question. *Do* selects the inverted auxiliary trees given in Figure 22.4, just as other auxiliaries do, so it is available to adjoin onto a tree at the S node to form a yes/no question. The mechanism described in section 22.3.1 prohibits *do* from co-occurring with other auxiliary verbs, even in the inverted position.

22.4 Infinitives

The infinitive *to* is considered an auxiliary verb in the XTAG system, and selects the auxiliary tree in Figure 22.1. *To*, like *do*, does not interact with the other auxiliary verbs, adjoining only to main verb base forms, and carrying infinitive mode. It is used in embedded clauses, both with and without a complementizer, as in sentences (471)-(473). Since it cannot be inverted, it simply does not select the trees in Figure 22.4.

(471) John wants to have a backpack .

(472) John wants Mary to have a backpack .

(473) John wants for Mary to have a backpack .

The usage of infinitival *to* interacts closely with the distribution of null subjects (PRO), and is described in more detail in section 8.5.

22.5 Semi-Auxiliaries

Under the category of semi-auxiliaries, we have placed several verbs that do not seem to closely follow the behavior of auxiliaries. One of these auxiliaries, *dare*, mainly behaves as a modal and selects for the base form of the verb. The other semi-auxiliaries all select for the infinitival form of the verb. Examples of this second type of semi-auxiliary are *used to*, *ought to*, *get to*, *have to*, *BE to* and *BE about to*.

22.5.1 Marginal Modal *dare*

The auxiliary *dare* is unique among modals in that it both allows DO-support and exhibits a past tense form. It clearly falls in modal position since no other auxiliary (except *do*) may precede it in linear order[6]. Examples appear below.

[5]The inversion of main verb *have* in British English was previously noted.

[6]Some speakers accept *dare* preceded by a modal, as in *I might dare finish this report today.* In the XTAG analysis, this particular double modal usage is accounted for. Other cases of double modal occurrence exist in some dialects of American English, although these are not accounted for in the system, as was mentioned earlier.

(474) she **dare** not have been seen .

(475) she does not **dare** succeed .

(476) Jerry **dared** not eat the strawberries .

(477) only models **dare** wear such extraordinary outfits .

(478) **dare** Dale tell Mary the secret ?

(479) *Louise had dared not tell a soul .

As mentioned above, auxiliaries are prevented from having DO-support within the XTAG system. To allow for DO-support in this case, we had to create a lexical entry for *dare* that allowed it to have the feature **mainv+** and to have **base** mode (this measure is what also allows *dare* to occur in double-modal sequences). A second lexical entry was added to handle the regular modal occurrence of *dare*. Additionally, all other modals are classified as being present tense, while *dare* has both present and past forms. To handle this behavior, *dare* was given similar features to the other modals in the morphology minus the specification for tense.

22.5.2 Other semi-auxiliaries

The other semi-auxiliaries all select for the infinitival form of the verb. Many of these auxiliaries allow for DO-support and can appear in both base and past participle forms, in addition to being able to stand alone (indicative mode). Examples of this type appear below.

(480) Alex **used** to attend karate workshops .

(481) Angelina might have **used** to believe in fate .

(482) Rich did not **used** to want to be a physical therapist .

(483) Mick might not **have** to play the game tonight .

(484) Singer **had** to have been there .

(485) Heather has **got** to finish that project before she goes insane .

The auxiliaries *ought to* and *BE to* may not be preceded by any other auxiliary.

(486) Biff **ought** to have been working tonight .

(487) *Carson does **ought** to have been working tonight .

(488) the party **is** to take place tonight .

(489) *the party had **been** to take place tonight .

The trickiest element in this group of auxiliaries is *used to*. While the other verbs behave according to standard inflection for auxiliaries, *used to* has the same form whether it is in mode base, past participle, or indicative forms. The only connection *used to* maintains with the infinitival form *use* is that occasionally, the bare form *use* will appear with DO-support. Since the three modes mentioned above are mutually exclusive in terms of both the morphology and the lexicon, *used* has three entries in each.

Semi-auxiliaries like *BE about to* are modal expressions derived from adverbs like *about*, which select for *BE* as the immediately preceding auxiliary and for infinitive *to* as the following one. This is illustrated in the following examples:

(490) The party cannot possibly *be about to* begin just yet .

(491) Miranda *is about to* steal the gun .

(492) I consider Miranda *about to* be upset .

In order to handle these expressions, we have auxiliary trees anchored by the word from which the modal expression is derived. So, for *BE about to*, *about* anchors the βVvx-arb tree with its <**mode**> value set to **nom** (Figure 22.6) and selects for the infinitive VP, *to*, with the following equation: **VP.b:<mode>=inf**.

In addition to the above constraint that restricts the form of the verb selected by the βVvx-arb tree to be infinitive, there is another constraint that needs to be accounted for, namely, the selection of *BE* as the immediately preceding auxiliary. This is especially needed to rule out examples like 493 and 494:

(493) ??John seems about to leave .

(494) *Mary could about to solve the problem .

Seems is a raising verb that can select for VP's which have <**mode**> = **nom** (*Miranda seems ready*) and will therefore be able to incorrectly adjoin to the root VP node of the βVvx-arb tree anchored by *about*.[7] We are able to rule 493 out because of the special status of raising verbs like *seems*. Though the raising verbs are treated like the other auxiliary verbs in that they select the same tree,βVvx, they are different in that the value for their <**mainv**> feature is +, whereas it is **-** for the other auxiliaires. We simply add the feature equation <**mainv**>**=-** in the top node of the root VP in the βVvx-arb tree. As a result only auxiliaries that come with the same binary feature value will be able to adjoin in.

To further disallow auxiliaries other than *BE* from selecting the βVvx-arb tree, as illustrated in 494 above, we have an additional lexical entry for *BE* which selects for a VP with the feature equation **VP.b:<mode>=nom**. Since no other auxiliary has this equation in the tree it anchors (except for raising verbs which are independently ruled out by the mechanism described above), the grammar correctly excludes examples like 494.

[7] Adjectival predicates project to a VP with the equation <**mode**>**=nom** on the VP node. See Chapter 9 for the XTAG analysis of adjectival predicative constructions and the associated feature structures and equations.

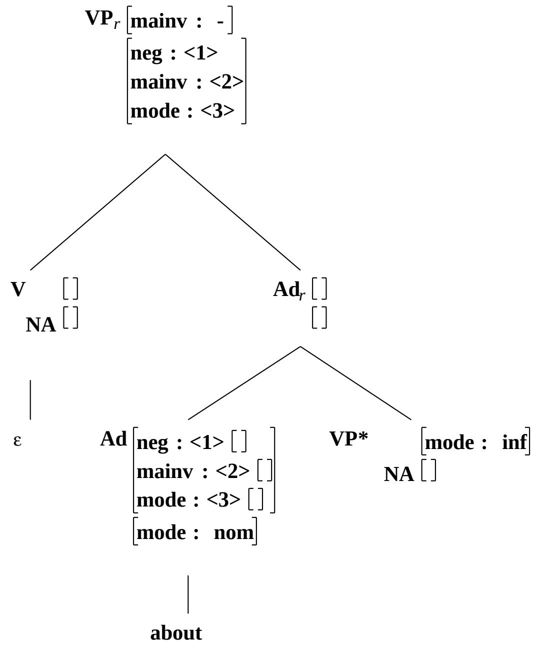


Figure 22.6: Auxiliary Tree for Semi-auxiliary anchored by *about*: βVvx-arb

22.5.3 Other Issues

There is a lingering problem with the auxiliaries that stems from the fact that there currently is no way to distinguish between the main verb and auxiliary verb behaviors for a given letter string within the morphology. This situation results in many unacceptable sentences being successfully parsed by the system. Examples of the unacceptable sentences are given below.

(495) the miller **cans** tell a good story . (vs the farmer **cans** peaches in July .)

(496) David **wills** have finished by noon . (vs the old man **wills** his fortune to me .)

(497) Sarah **needs** not leave . (vs Sarah **needs** to leave .)

(498) Jennifer **dares** not be seen . (vs the young woman **dares** him to do the stunt .)

(499) Lila **does use** to like beans . (vs Lila **does use** her new cookware .)

Chapter 23

Conjunction

23.1 Introduction

The XTAG system can handle sentences with conjunction of two constituents of the same syntactic category. The coordinating conjunctions which select the conjunction trees are *and*, *or* and *but*.[1] There are also multi-word conjunction trees, anchored by *either-or*, *neither-nor*, *both-and*, and *as well as*. There are eight syntactic categories that can be coordinated, and in each case an auxiliary tree is used to implement the conjunction. These eight categories can be considered as four different cases, as described in the following sections. In all cases the two constituents are required to be of the same syntactic category, but there may also be some additional constraints, as described below.

23.2 Adjective, Adverb, Preposition and PP Conjunction

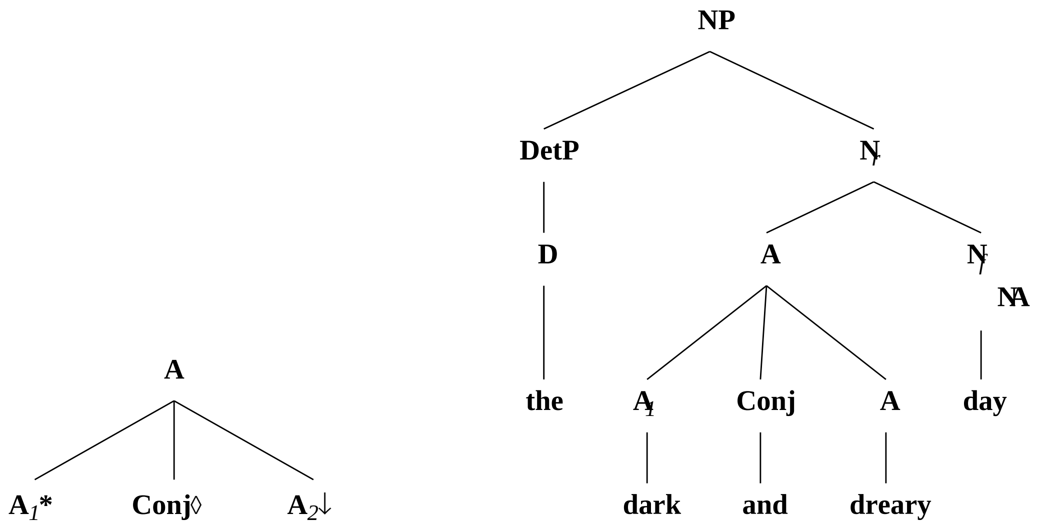
Each of these four categories has an auxiliary tree that is used for conjunction of two constituents of that category. The auxiliary tree adjoins into the left-hand-side component, and the right-hand-side component substitutes into the auxiliary tree.

Figure 23.1(a) shows the auxiliary tree for adjective conjunction, and is used, for example, in the derivation of the parse tree for the noun phrase *the dark and dreary day*, as shown in Figure 23.1(b). The auxiliary tree adjoins onto the node for the left adjective, and the right adjective substitutes into the right hand side node of the auxiliary tree. The analysis for adverb, preposition and PP conjunction is exactly the same and there is a corresponding auxiliary tree for each of these that is identical to that of Figure 23.1(a) except, of course, for the node labels.

23.3 Noun Phrase and Noun Conjunction

The tree for NP conjunction, shown in Figure 23.2(a), has the same basic analysis as in the previous section except that the <**wh**> and <**case**> features are used to force the two noun phrases to have the same <**wh**> and <**case**> values. This allows, for example, *he and she wrote the book together* while disallowing $*$*he and her wrote the book together.* Agreement is

[1]We believe that the restriction of *but* to conjoining only two items is a pragmatic one, and our grammars accepts sequences of any number of elements conjoined by *but*.



(a) (b)

Figure 23.1: Tree for adjective conjunction: βa1CONJa2 and a resulting parse tree

lexicalized, since the various conjunctions behave differently. With *and*, the root <**agr num**> value is <**plural**>, no matter what the number of the two conjuncts. With *or*, however, the root <**agr num**> is co-indexed with the <**agr num**> feature of the right conjunct. This ensures that the entire conjunct will bear the number of both conjuncts if they agree (Figure 23.2(b)), or of the most "recent" one if they differ (*Either the boys or John is going to help you.*). There is no rule per se on what the agreement should be here, but people tend to make the verb agree with the last conjunct (cf. [Quirk *et al.*, 1985], section 10.41 for discussion). The tree for N conjunction is identical to that for the NP tree except for the node labels. (The multi-word conjunctions do not select the N conjunction tree - **the both dogs and cats*).

23.4 Determiner Conjunction

In determiner coordination, all of the determiner feature values are taken from the left determiner, and the only requirement is that the <**wh**> feature is the same, while the other features, such as <**card**>, are unconstrained. For example, *which and what* and *all but one* are both acceptable determiner conjunctions, but $*$*which and all* is not.

(500) how many and which people camp frequently ?

(501) *some or which people enjoy nature .

23.5 Sentential Conjunction

The tree for sentential conjunction, shown in Figure 23.4, is based on the same analysis as the conjunctions in the previous two sections, with a slight difference in features. The <**mode**>

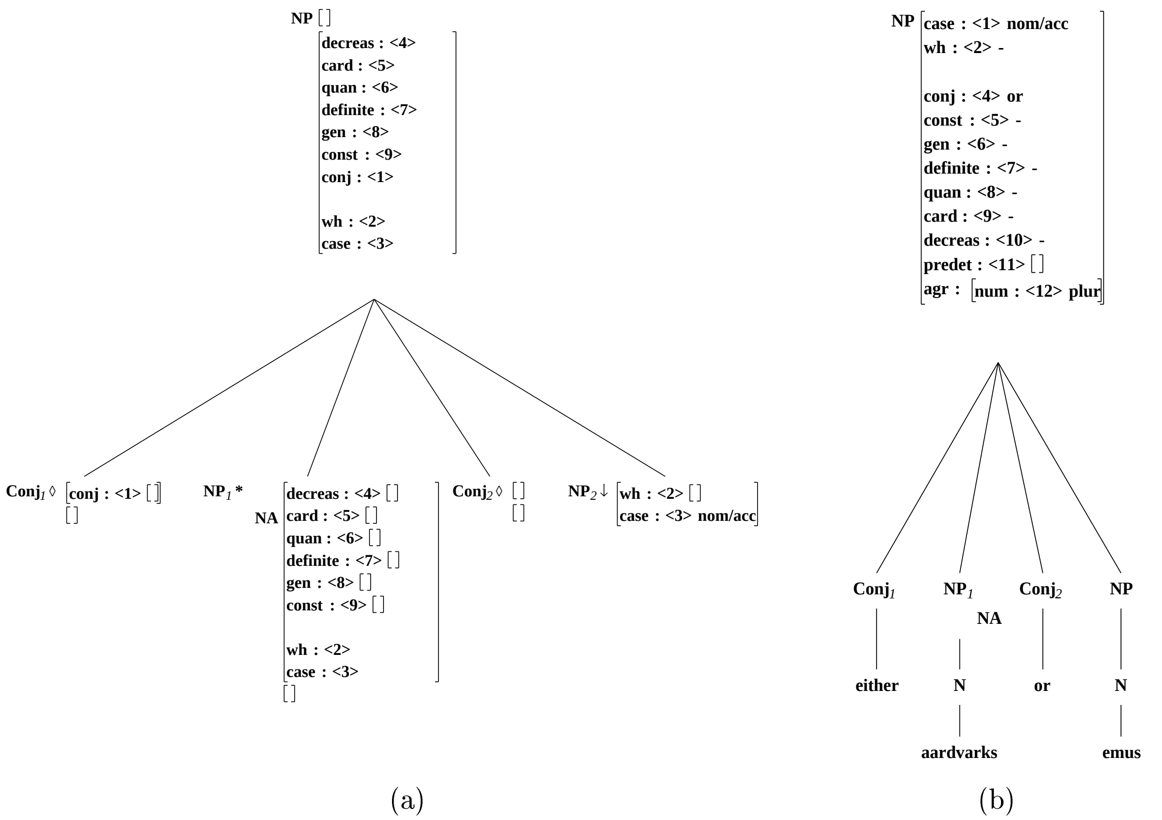


Figure 23.2: Tree for NP conjunction: βCONJnx1CONJnx2 and a resulting parse tree

feature[2] is used to constrain the two sentences being conjoined to have the same mode so that *the day is dark and the phone never rang* is acceptable, but ∗*the day dark and the phone never rang* is not. Similarly, the two sentences must agree in their <**wh**>, <**comp**> and <**extracted**> features. Co-indexation of the <**comp**> feature ensures that either both conjuncts have the same complementizer, or there is a single complementizer adjoined to the complete conjoined S. The <**assign-comp**> feature[3] feature is used to allow conjunction of infinitival sentences, such as *to read and to sleep is a good life.*

23.6 Comma as a conjunction

We treat comma as a conjunction in conjoined lists. It anchors the same trees as the lexical conjunctions, but is considerably more restricted in how it combines with them. The trees anchored by commas are prohibited from adjoining to anything but another comma conjoined element or a non-coordinate element. (All scope possibilities are allowed for elements coordinated with lexical conjunctions.) Thus, structures such as Tree 23.5(a) are permitted, with each element stacking sequentially on top of the first element of the conjunct, while structures such as Tree 23.5(b) are blocked.

[2]See section 8.3 for an explanation of the <**mode**> feature.

[3]See section 8.5 for an explanation of the <**assign-comp**> feature.

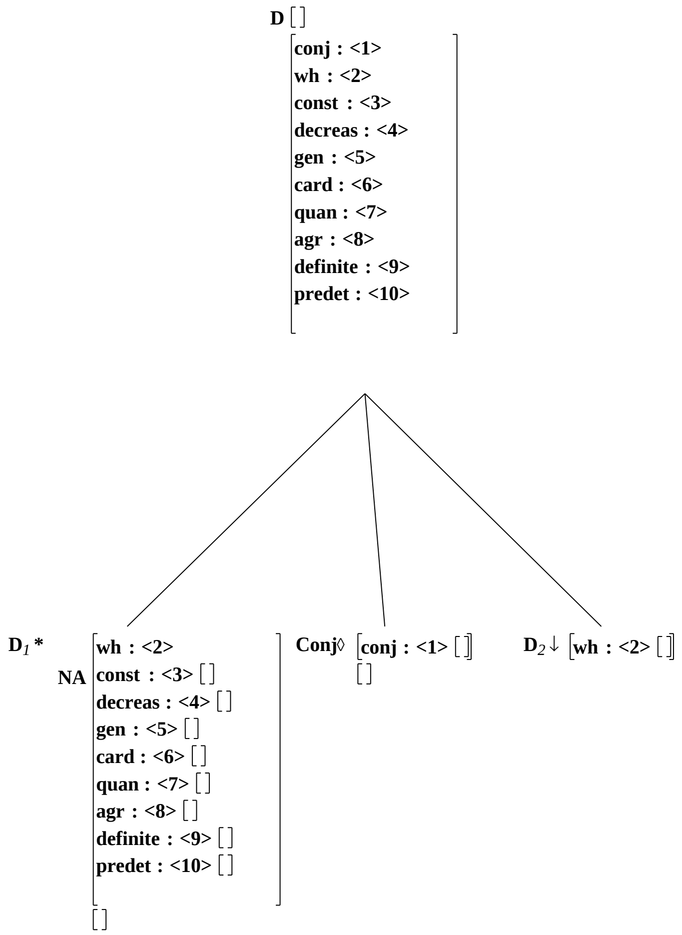


Figure 23.3: Tree for determiner conjunction: βd1CONJd2.ps

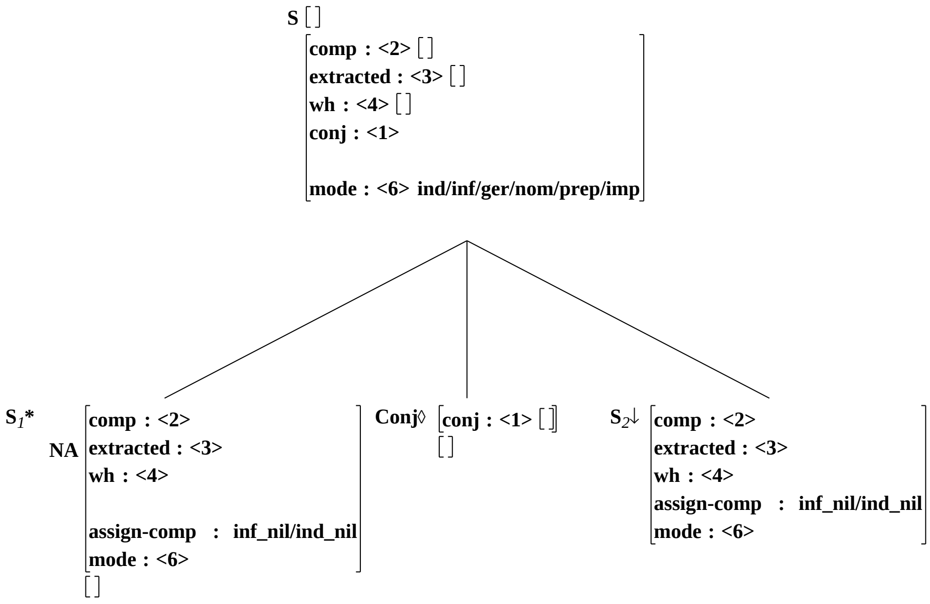
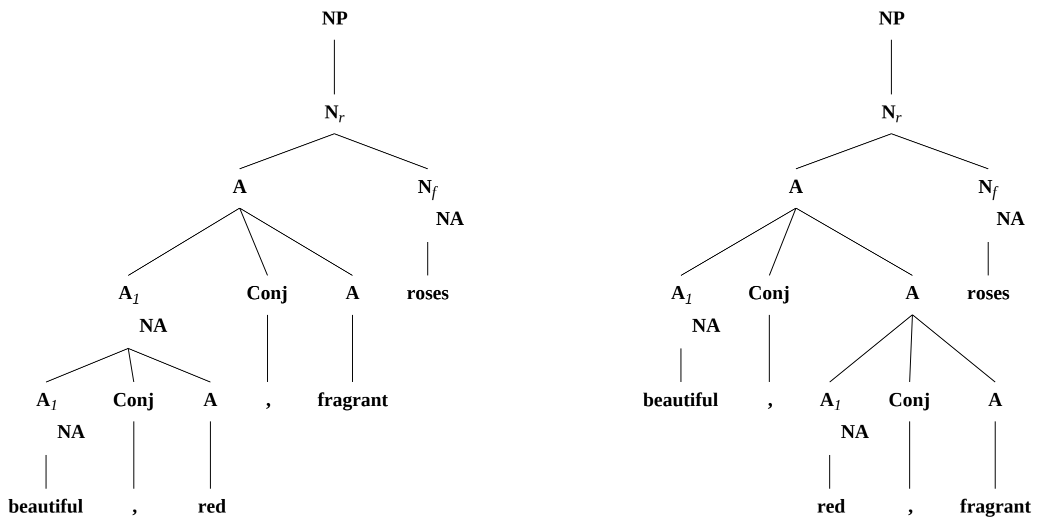


Figure 23.4: Tree for sentential conjunction: βs1CONJs2



(a) Valid tree with comma conjunction (b) Invalid tree

Figure 23.5:

This is accomplished by using the <**conj**> feature, which has the values **and/or/but** and **comma** to differentiate the lexical conjunctions from commas. The <**conj**> values for a comma-anchored tree and *and*-anchored tree are shown in Figure 23.6. The feature <**conj**> = **comma/none** on A_1 in (a) only allows comma conjoined or non-conjoined elements as the left-adjunct, and <**conj**> = **none** on A in (a) allows only a non-conjoined element as the right conjunct. We also need the feature <**conj**> = **and/or/but/none** on the right conjunct of the trees anchored by lexical conjunctions like (b), to block comma-conjoined elements from substituting there. Without this restriction, we would get multiple parses of the NP in Tree 23.5; with the restrictions we only get the derivation with the correct scoping, shown as (a).

Since comma-conjoined lists can appear without a lexical conjunction between the final two elements, as shown in example (502), we cannot force all comma-conjoined sequences to end with a lexical conjunction.

(502) So it is too with many other spirits which we all know: the spirit of Nazism or Communism, school spirit , the spirit of a street corner gang or a football team, the spirit of Rotary or the Ku Klux Klan. [Brown cd01]

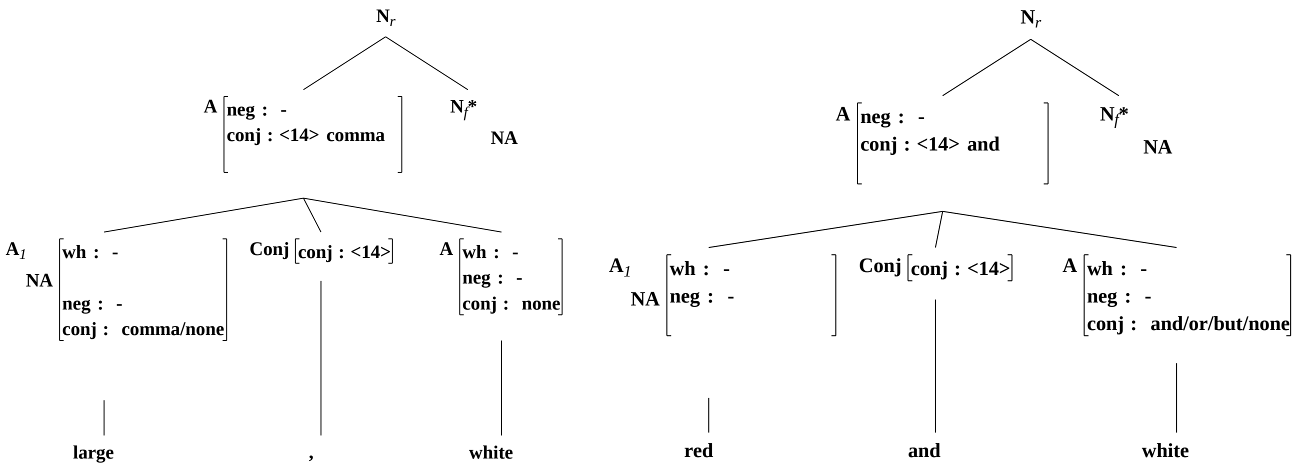


Figure 23.6: βa1CONJa2 (a) anchored by comma and (b) anchored by *and*

23.7 *But-not*, *not-but*, *and-not* and ϵ-*not*

We are analyzing conjoined structures such as *The women but not the men* with a multi-anchor conjunction tree anchored by the conjunction plus the adverb *not*. The alternative is to allow *not* to adjoin to any constituent. However, this is the only construction where *not* can freely occur onto a constituent other than a VP or adjective (cf. βNEGvx and βNEGa trees). It can also adjoin to some determiners, as discussed in Section 20. We want to allow sentences like (503) and rule out those like (504). The tree for the good example is shown in Figure 23.7. There are similar trees for *and-not* and ϵ-*not*, where ϵ is interpretable as either *and* or *but*, and a tree with *not* on the first conjunct for *not-but*.

(503) Beth grows basil in the house (but) not in the garden .

(504) *Beth grows basil (but) not in the garden .

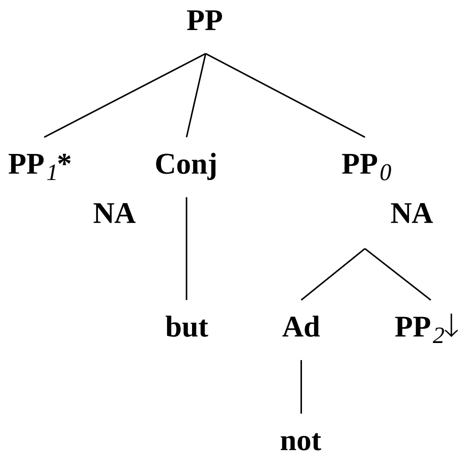


Figure 23.7: Tree for conjunction with but-not: βpx1CONJARBpx2

Although these constructions sound a bit odd when the two conjuncts do not have the same number, they are sometimes possible. The agreement information for such NPs is always that of the non-negated conjunct: *his sons, and not Bill, are in charge of doing the laundry* or *not Bill, but his sons, are in charge of doing the laundry* (Some people insist on having the commas here, but they are frequently absent in corpus data.) The agreement feature from the non-negated conjunct in passed to the root NP, as shown in Figure 23.8. Aside from agreement, these constructions behave just like their non-negated counterparts.

23.8 *To* as a Conjunction

To can be used as a conjunction for adjectives (Fig. 23.9) and determiners, when they denote points on a scale:

(505) two to three degrees

(506) high to very high temperatures

As far as we can tell, when the conjuncts are determiners they must be cardinal.

23.9 Predicative Coordination

This section describes the method for predicative coordination (including VP coordination of various kinds) used in XTAG. The description is derived from work described in ([Sarkar and Joshi, 1996]). It is important to say that this implementation of predicative coordination is not part of the XTAG release at the moment due massive parsing ambiguities. This is partly because of the current implementation and also the inherent ambiguities due to VP coordination that cause a combinatorial explosion for the parser. We are trying to remedy both of these limitations using a probability model for coordination attachments which will be included as part of a later XTAG release.

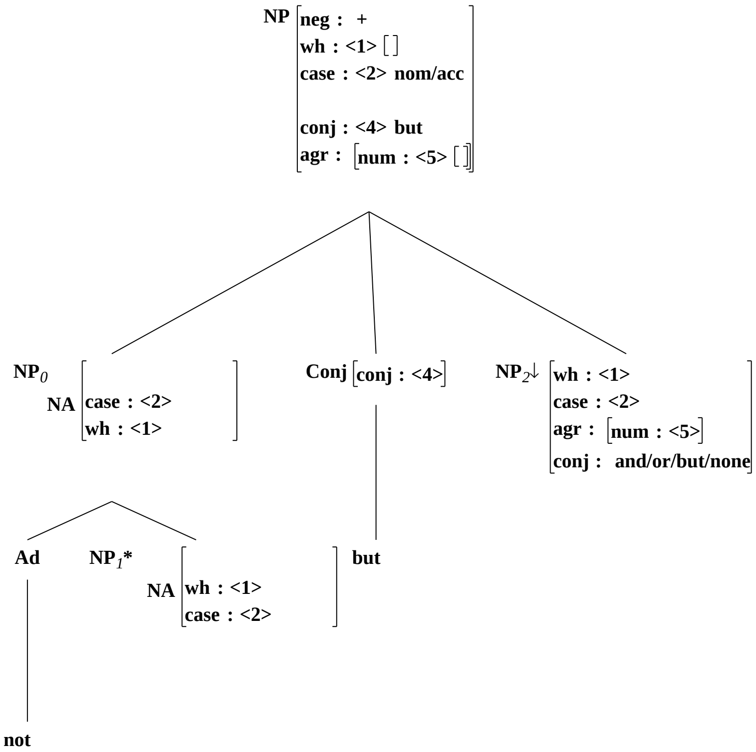


Figure 23.8: Tree for conjunction with not-but: βARBnx1CONJnx2

This extended domain of locality in a lexicalized Tree Adjoining Grammar causes problems when we consider the coordination of such predicates. Consider (507) for instance, the NP *the beans that I bought from Alice* in the Right-Node Raising (RNR) construction has to be shared by the two elementary trees (which are anchored by *cooked* and *ate* respectively).

(507) (((Harry cooked) and (Mary ate)) the beans that I bought from Alice)

We use the standard notion of coordination which is shown in Figure 23.10 which maps two constituents of *like type*, but with different interpretations, into a constituent of the same type.

We add a new operation to the LTAG formalism (in addition to substitution and adjunction) called *conjoin* (later we discuss an alternative which replaces this operation by the traditional operations of substitution and adjunction). While substitution and adjunction take two trees to give a derived tree, *conjoin* takes three trees and composes them to give a derived tree. One of the trees is always the tree obtained by specializing the schema in Figure 23.10 for a particular category. The tree obtained will be a lexicalized tree, with the lexical anchor as the conjunction: *and*, *but*, etc.

The conjoin operation then creates a *contraction* between nodes in the contraction sets of the trees being coordinated. The term *contraction* is taken from the graph-theoretic notion of edge contraction. In a graph, when an edge joining two vertices is contracted, the nodes are merged and the new vertex retains edges to the union of the neighbors of the merged vertices. The conjoin operation supplies a new edge between each corresponding node in the contraction set and then contracts that edge.

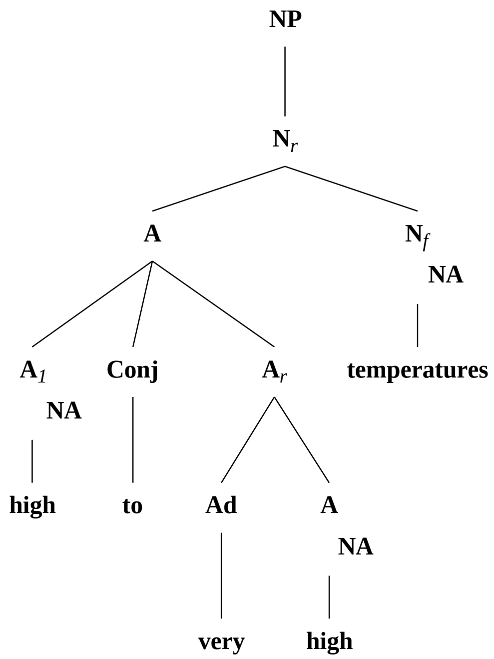


Figure 23.9: Example of conjunction with *to*

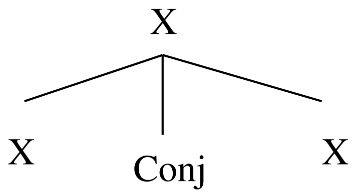


Figure 23.10: Coordination schema

For example, applying *conjoin* to the trees *Conj(and)*, $\alpha(eats)$ and $\alpha(drinks)$ gives us the derivation tree and derived structure for the constituent in 508 shown in Figure 23.11.

(508) ... eats cookies and drinks beer

Another way of viewing the conjoin operation is as the construction of an auxiliary structure from an elementary tree. For example, from the elementary tree $\alpha(drinks)$, the conjoin operation would create the auxiliary structure $\beta(drinks)$ shown in Figure 23.12. The adjunction operation would now be responsible for creating contractions between nodes in the contraction sets of the two trees supplied to it. Such an approach is attractive for two reasons. First, it uses only the traditional operations of substitution and adjunction. Secondly, it treats *conj X* as a kind of "modifier" on the left conjunct *X*. This approach reduces some of the parsing ambiguities introduced by the predicative coordination trees and forms the basis of the XTAG implementation.

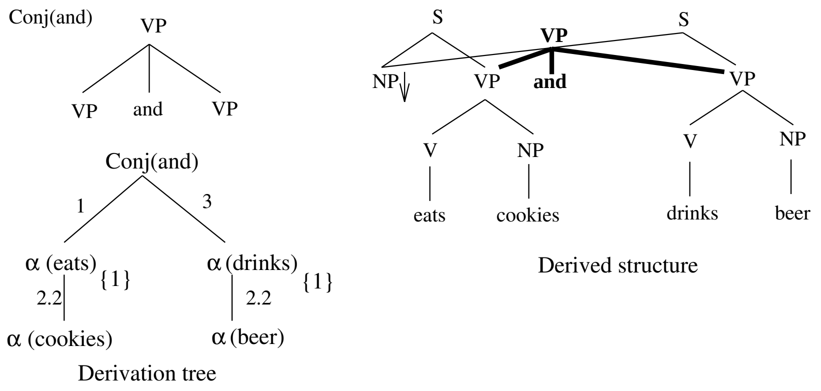


Figure 23.11: An example of the *conjoin* operation. {1} denotes a shared dependency.

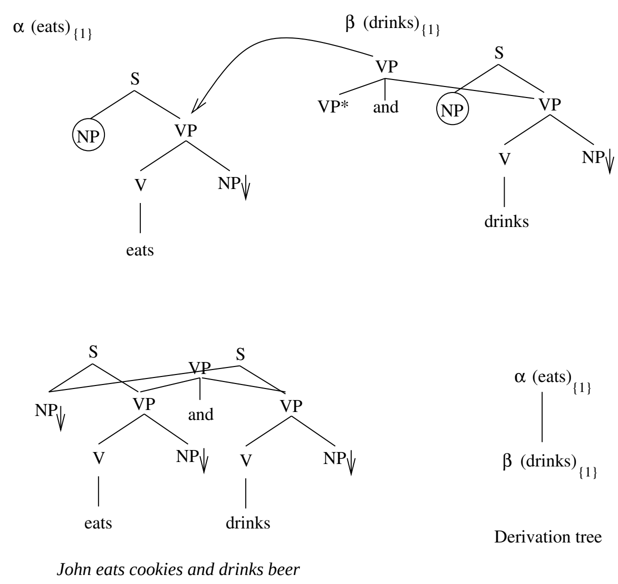


Figure 23.12: Coordination as adjunction.

More information about predicative coordination can be found in ([Sarkar and Joshi, 1996]), including an extension to handle gapping constructions.

23.10 Pseudo-coordination

The XTAG grammar does handle one sort of verb pseudo-coordination. Semi-idiomatic phrases such as 'try and' and 'up and' (as in 'they might try and come today') are handled as multi-anchor modifiers rather than as true coordination. These items adjoin to a V node, using the βVCONJv tree. This tree adjoins only to verbs in their base morphological (non-inflected) form. The verb anchor of the βVCONJv must also be in its base form, as shown in examples (509)-(511). This blocks 3rd-person singular derivations, which are the only person morphologically marked in the present, except when an auxiliary verb is present or the verb is in the infinitive.

(509) $*$He tried and came yesterday.

(510) They try and exercise three times a week.

(511) He wants to try and sell the puppies.

Chapter 24

Comparatives

24.1 Introduction

Comparatives in English can manifest themselves in many ways, acting on many different grammatical categories and often involving ellipsis. A distinction must be made at the outset between two very different sorts of comparatives—those which make a comparison between two propositions and those which compare the extent to which an entity has one property to a greater or lesser extent than another property. The former, which we will refer to as *propositional* comparatives, is exemplified in (512), while the latter, which we will call *metalinguistic* comparatives (following Hellan 1981), is seen in (513):

(512) Ronaldo is more angry than Romario.

(513) Ronaldo is more angry than upset.

In (512), the extent to which Ronaldo is angry is greater than the extent to which Romario is angry. Sentence (513) indicates that the extent to which Ronaldo is angry is greater than the extent to which he is upset.

Apart from certain of the elliptical cases, both kinds of comparatives can be handled straightforwardly in the XTAG system. Elliptical cases which are not presently covered include those exemplified by the following sentences, which would presumably be handled in the same way as other sorts of VP ellipsis would.

(514) Ronaldo is more angry than Romario is.

(515) Bill eats more broccoli than George eats.

(516) Bill eats more broccoli than George does.

We turn to the analysis of metalinguistic comparatives first.

24.2 Metalinguistic Comparatives

A metalinguistic comparison can be performed on basically all of the predicational categories—adjectives, verb phrases, prepositional phrases, and nouns—as in the following examples:

(517) The table is more long than wide. (AP)

(518) Clark more makes the rules than follows them. (VP)

(519) Calvin is more in the living room than in the kitchen. (PP)

(520) That unindentified amphibian in the bush is more frog than toad, I would say. (NP)

At present, we only deal with the adjectival metalinguistic comparatives as in (517). The analysis given here for these can be easily extended to prepositional phrases and nominal comparatives of the metalinguistic sort, but, as with coordination in XTAG, verb phrases will prove more difficult.

Adjectival comparatives appear to distribute with simple adjectives, as in the following examples:

(521) Herbert is more livid than angry.

(522) Herbert is more livid and furious than angry.

(523) The more innovative than conventional medication cured everyone in the sick ward.

(524) The elephant, more wobbly than steady, fell from the circus ball.

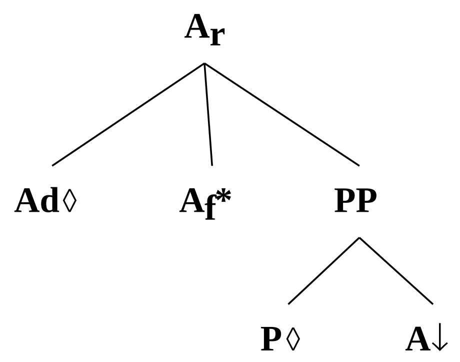


Figure 24.1: Tree for Metalinguistic Adjective Comparative: βARBaPa

This patterning indicates that we can give these comparatives a tree that adjoins quite freely onto adjectives, as in Figure 24.1. This tree is anchored by *more/less - than*. To avoid grammatically incorrect comparisons such as *more brighter than dark*, the feature **compar** is used to block this tree from adjoining onto morphologically comparative adjectives. The foot node is **compar-**, while *brighter* and its comparative siblings are **compar+**[1]. We also wish to block strings like *more brightest than dark*, which is accomplished with the feature **super**, indicating superlatives. This feature is negative at the foot node so that βARBaPa cannot adjoin to superlatives like *nicest*, which are specified as **super+** from the morphology. Furthermore, the root node is **super+** so that βARBaPa cannot adjoin onto itself and produce monstrosities such as (525):

[1]The analysis given later for adjectival propositional comparatives produces aggregated **compar+** adjectives such as *more bright*, which will also be incompatible (as desired) with βARBaPa.

(525) *Herbert is more less livid than angry than furious.

Thus, the use of the **super** feature is less to indicate superlativeness specifically, but rather to indicate that the subtree below a **super+** node contains a full-fleshed comparison. In the case of lexical superlatives, the comparison is against everything, implicitly.

A benefit of the multiple-anchor approach here is that we will never allow sentences such as (526), which would be permissible if we split the comparative component and the *than* component of metalinguistic comparatives into two separate trees.

(526) *Ronaldo is angrier than upset.

We also see another variety of adjectival comparatives of the form *more/less than X*, which indicates some property which is more or less extreme than the property X. In a sentence such as (527), some property is being said to hold of Francis such that it is of a kind with *stupid* and that it exceeds *stupid* on some scale (intelligence, for example). Quirk et al. also note that these constructions remark on the inadequacy of the lexical item. Thus, in (526), it could be that *stupid* is a starting point from which the speaker makes an approximation for some property which the speaker feels is beyond the range of the English lexicon, but which expresses the supreme lack of intellect of the individual it is predicated of.

(527) Francis is more than stupid.

(528) Romario is more than just upset.

Taking our inspiration from βARBaPa, we can handle these comparatives, which have the same distribution but contain an empty adjective, by using the tree shown in Figure 24.2.

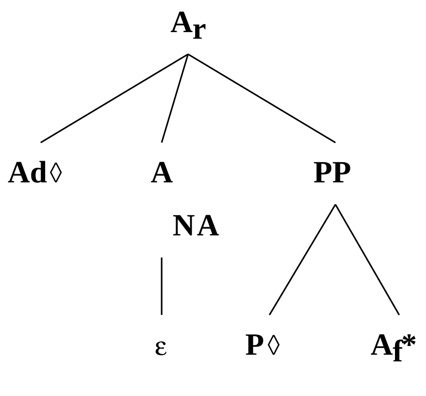


Figure 24.2: Tree for Adjective-Extreme Comparative: βARBPa

This sort of metalinguistic comparative also occurs with the verb phrase, prepositional phrase, and noun varieties.

(529) Clark more than makes the rules. (VP)

(530) Calvin's hands are more than near the cookie jar. (PP)

(531) That stuff on her face is more than mud. (NP)

Presumably, the analysis for these would parallel that for adjectives, though it has not yet been implemented.

24.3 Propositional Comparatives

24.3.1 Nominal Comparatives

Nominal comparatives are considered here to be those which compare the cardinality of two sets of entities denoted by nominal phrases. The following data lay out a basic distribution of these comparatives.

(532) More vikings than mongols eat spam.

(533) *More the vikings than mongols eat spam.

(534) Vikings eat less spaghetti than spam.

(535) More men that walk to the store than women who despise spam enjoyed the football game.

(536) More men than James hate scotch on the rocks.

(537) James knows fewer bunnies than rabbits.

Looking at these examples, we are tempted to produce a tree for this construction that is similar to βARBaPa. However, it is quite common for the *than* portion of these comparatives to be left out, as in the following sentences:

(538) More vikings eat spam.

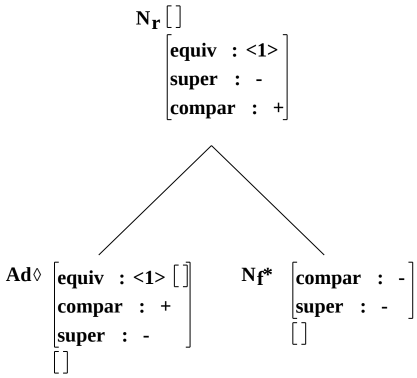
(539) Mongols eat less spam.

Furthermore, *than NP* cannot occur without *more*. These facts indicate that we can and should build up nominal comparatives with two separate trees. The first, which allows a comparative adverb to adjoin to a noun, is given in Figure 24.3(a). The second is the noun-phrase modifying prepositional tree. The tree βCARBn is anchored by *more/less/fewer* and βCnxPnx is anchored by *than*. The feature **compar** is used to ensure that only one βCARBn tree can adjoin to any given noun—its foot node is **compar-** and the root node is **compar+**. All nouns are **compar-**, and the **compar** value is passed up through all trees which adjoin to N or NP. In order to ensure that we do not allow sentences like **Vikings than mongols eat spam*, the **compar** feature is used. The NP foot node of βCnxPnx is **compar+**; thus, βCnxPnx will adjoin only to NP's which have been already modified by βCARBn (and thereby comparativized). In this way, we capture sentences like (538) en route to deriving sentences like (532), in a principled and simple manner.

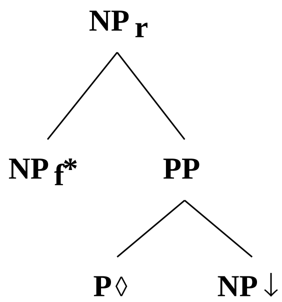
Further evidence for this approach comes from comparative clauses which are missing the noun phrase which is being compared against something, as in the following:

(540) The vikings ate more.[2]

[2]We ignore here the interpretation in which the comparison covers the eating event, focussing only on the one which the comparison involves the stuff being eaten.



(a) βCARBn tree



(b) βCnxPnx tree

Figure 24.3: Nominal comparative trees

(541) The vikings ate more than a boar.[3]

Sometimes the missing noun refers to an entity or set available in the prior discourse, while at other times it is a reference to some anonymous, unspecified set. The former is exemplified in a mini-discourse such as the following:

Calvin: "The mongols ate spam."
Hobbes: "The vikings ate more."

The latter can be seen in the following example:

Calvin: "The vikings ate a a boar."
Hobbes: "Indeed. But in fact, the vikings ate more than a boar."

Since the lone comparatives *more/less/fewer* have the same basic distribution as noun phrases, the tree in Figure 24.4 is employed to capture this fact. The root node of αCARB is **compar+**. Not only does this accord with our intuitions about what the **compar** feature is supposed to indicate, it also permits βnxPnx to adjoin, giving us strings such as *more than NP* for free.

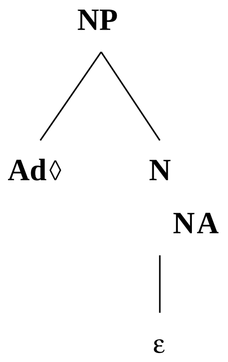


Figure 24.4: Tree for Lone Comparatives: αCARB

Thus, by splitting nominal comparatives into multiple trees, we make correct predictions about their distribution with a minimal number of simple trees. Furthermore, we now also get certain comparative coordinations for free, once we place the requirement that nouns and noun phrases must match for **compar** if they are to be coordinated. This yields strings such as the following:

(542) James eats more grapes and fewer boars than avocados.

(543) Were there more or less than fifty people (at the party)?

[3]This sentence differs from the metalinguistic comparison *That stuff on her face is more than mud* in that it involves a comment on the quantity and/or type of the compared NP, whereas the other expresses that the property denoted by the compared noun is an inadequate characterization of the thing being described.

The structures are given in Figure 24.5. Also, it will block strings like *more men and women than children* under the (impossible) interpretation that there are more men than children but the comparison of the quantity of women to children is not performed. Unfortunately, it will permit comparative clauses such as *more grapes and fewer than avocados* under the interpretation in which there are more grapes than avocados and fewer of some unspecified thing than avocados (see Figure 24.6).

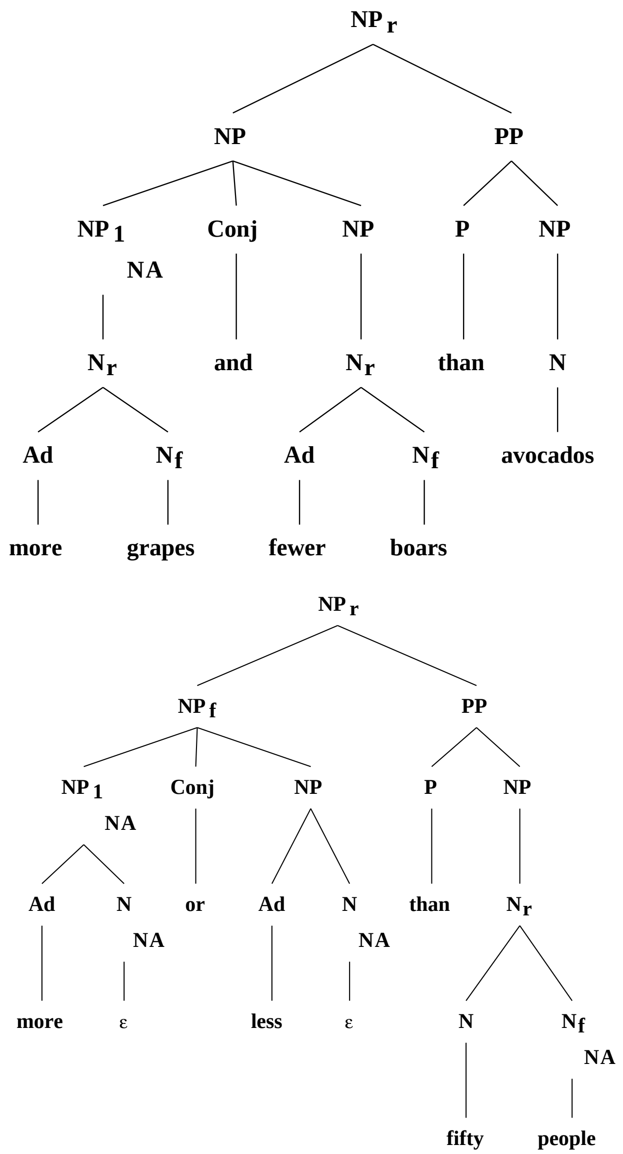


Figure 24.5: Comparative conjunctions.

One aspect of this analysis is that it handles the elliptical comparatives such as the following:

(544) Arnold kills more bad guys than Steven.

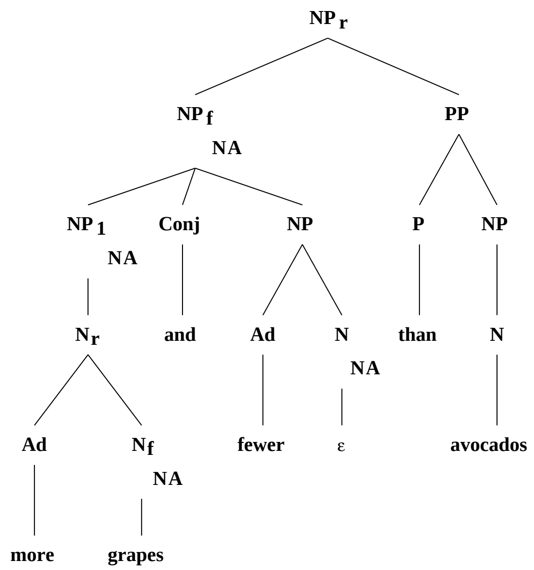


Figure 24.6: Comparative conjunctions.

In a sense, this is actually only simulating the ellipsis of these constructions indirectly. However, consider the following sentences:

(545) Arnold kills more bad guys than I do.

(546) Arnold kills more bad guys than I.

(547) Arnold kills more bad guys than me.

The first of these has a *pro*-verb phrase which has a nominative subject. If we totally drop the second verb phrase, we find that the second NP can be in either the nominative or the accusative case. Prescriptive grammars disallow accusative case, but it actually is more common to find accusative case—use of the nominative in conversation tends to sound rather stiff and unnatural. This accords with the present analysis in which the second noun phrase in these comparatives is the complement of *than* in βnxPnx, and receives its case-marking from *than*. This does mean that the grammar will not currently accept (546), and indeed such sentences will only be covered by an analysis which really deals with the ellipsis. Yet the fact that most speakers produce (547) indicates that some sort of restructuring has occured that results in the kind of structure the present analysis offers.

There is yet another distributional fact which falls out of this analysis. When comparative or comparativized adjectives modify a noun phrase, they can stand alone or occur with a *than* phrase; furthermore, they are obligatory when a *than*-phrase is present.

(548) Hobbes is a better teacher.

(549) Hobbes is a better teacher than Bill.

(550) A more exquisite horse launched onto the racetrack.

(551) A more exquisite horse than Black Beauty launched onto the racetrack.

(552) *Hobbes is a teacher than Bill.

Comparative adjectives such as *better* come from the lexicon as **compar+**. By having trees such as βAn transmit the **compar** value of the A node to the root N node, we can signal to βCnxPnx that it may adjoin when a comparative adjective has adjoined. An example of such an adjunction is given in Figure 24.7. Of course, if no comparative element is present in the lower part of the noun phrase, βnxPnx will not be able to adjoin since nouns themselves are **compar-**. In order to capture the fact that a comparative element blocks further modification to N, βAn must only adjoin to N nodes which are **compar-** in their lower feature matrix.

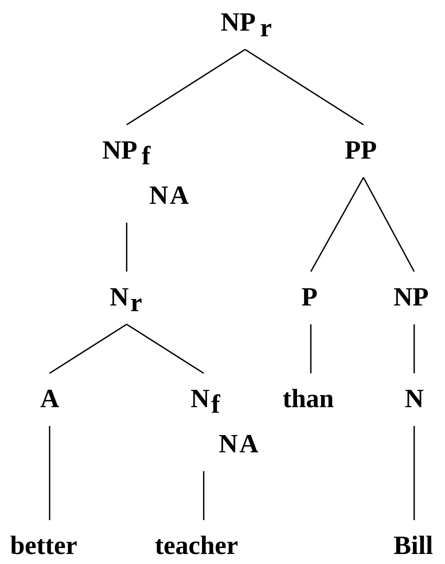


Figure 24.7: Adjunction of βnxPnx to NP modified by comparative adjective.

In order to obtain this result for phrases like *more exquisite horse*, we need to provide a way for *more* and *less* to modify adjectives without a *than*-clause as we have with βARBaPa. Actually, we need this ability independently for comparative adjectival phrases, as discussed in the next section.

24.3.2 Adjectival Comparatives

With nominal comparatives, we saw that a single analysis was amenable to both "pure" comparatives and elliptical comparatives. This is not possible for adjectival comparatives, as the following examples demonstrate:

(553) The dog is less stupid.

(554) The dog is less stupid than the cat.

(555) John is as stupid.

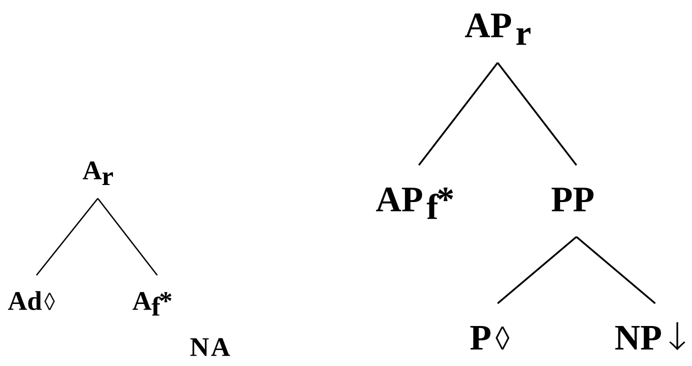
(556) John is as stupid as Mary.

(557) The less stupid dog waited quietly for its master.

(558) *The less stupid than the cat dog waited eagerly for its master.

The last example shows that comparative adjectival phrases cannot distribute quite as freely as comparative nominals.

The analysis of elliptical comparative adjectives follows closely to that of comparative nominals. We build them up by first adjoining the comparative element to the A node, which then signals to the AP node, via the **compar** feature, that it may allow a *than*-clause to adjoin. The relevant trees are given in Figure 24.8. βCARBa is anchored by *more, less* and *as*, and βaxPnx is anchored by both *than* and *as*.



(a) βCARBa tree (b) βaxPnx tree

Figure 24.8: Elliptical adjectival comparative trees

The advantages of this analysis are many. We capture the distribution exhibited in the examples given in (553) - (558). With βCARBa, comparative elements may modify adjectives wherever they occur. However, *than* clauses for adjectives have a more restricted distribution which coincides nicely with the distribution of AP's in the XTAG grammar. Thus, by making them adjoin to AP rather than A, ill-formed sentences like (558) are not allowed.

There are two further advantages to this analysis. One is that βCARBa interacts with βnxPnx to produce sequences like *more exquisite horse than Black Beauty*, a result alluded to at the end of Section 24.3.1. We achieve this by ensuring that the comparativeness of an adjective is controlled by a comparative adverb which adjoins to it. A sample derivation is given in Figure 24.9. The second advantage is that we get sentences such as (559) for free.

(559) Hobbes is better than Bill.

Since *better* comes from the lexicon as **compar+** and this value is passed up to the AP node, βaxPnx can adjoin as desired, giving us the derivation given in Figure 24.10.

Notice that the root AP node of Figure 24.10 is **compar-**, so we are basically saying that strings such as *better than Bill* are not "comparative." This accords with our use of the **compar** feature—a positive value for **compar** signals that the clause beneath it is to **be** compared

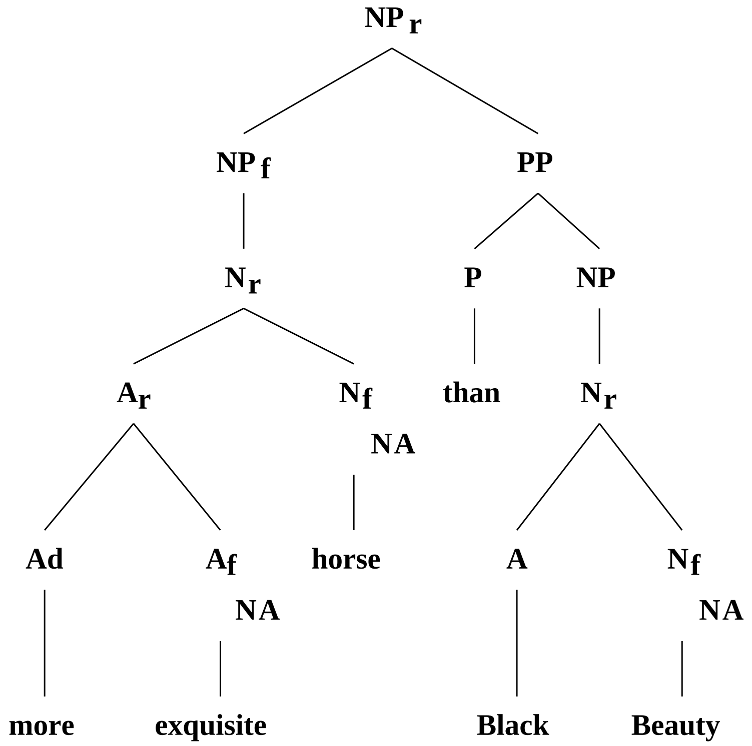


Figure 24.9: Comparativized adjective triggering βCnxPnx.

against something else. In the case of *better than Bill*, the comparison has been fulfilled, so we do not want it to signal for further comparisons. A nice result which follows is that βaxPnx cannot adjoin more than once to any given AP spine, and we have no need for the NA constraint on the tree's root node. Also, this treatment of the comparativeness of various strings proves important in getting the coordination of comparative constructions to work properly.

A note needs to be made about the analysis regarding the interaction of the equivalence comparative construction *as ... as* and the inequivalence comparative construction *more/less ... than*. In the grammar, *more, less*, and *as* all anchor βCARBa, and both *than* and *as* anchor βaxPnx. Without further modifications, this of course will give us sentences such as the following:

(560) *?Hobbes is as patient than Bill.

(561) *?Hobbes is more patient as Bill.

Such cases are blocked with the feature **equiv**: *more, less, fewer* and *than* are **equiv-** while *as* (in both adverbial and prepositional uses) is **equiv+**. The prepositional trees then require that their P node and the node to which they are adjoining match for **equiv**.

An interesting phenomena in which comparisons seem to be paired with an inappropriate *as/than*-clause is exhibited in (562) and (563).

(562) Hobbes is as patient or more patient than Bill.

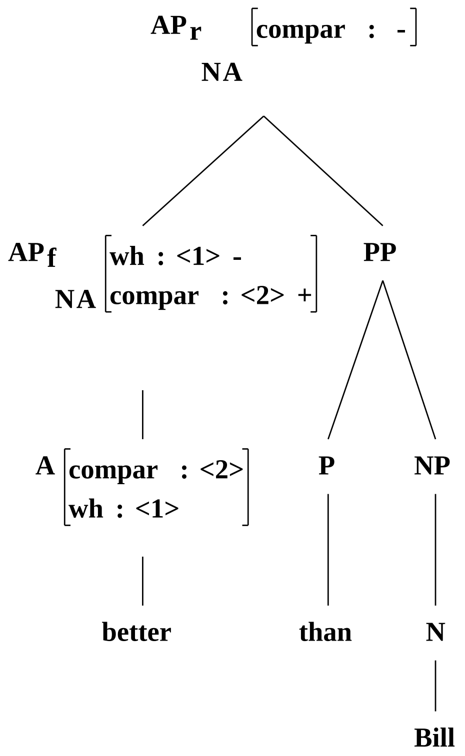


Figure 24.10: Adjunction of βaxPnx to comparative adjective.

(563) Hobbes is more patient or as patient as Bill.

Though prescriptive grammars disfavor these sentences, these are perfectly acceptable. We can capture the fact that the *as/than*-clause shares the **equiv** value with the latter of the comparison phrases by passing the **equiv** value for the second element to the root of the coordination tree.

24.3.3 Adverbial Comparatives

The analysis of adverbial comparatives encouragingly parallels the analysis for nominal and elliptical adjectival comparatives—with, however, some interesting differences. Some examples of adverbial comparatives and their distribution are given in the following:

(564) Albert works more quickly.

(565) Albert works more quickly than Richard.

(566) Albert works more.

(567) *Albert more works.

(568) Albert works more than Richard.

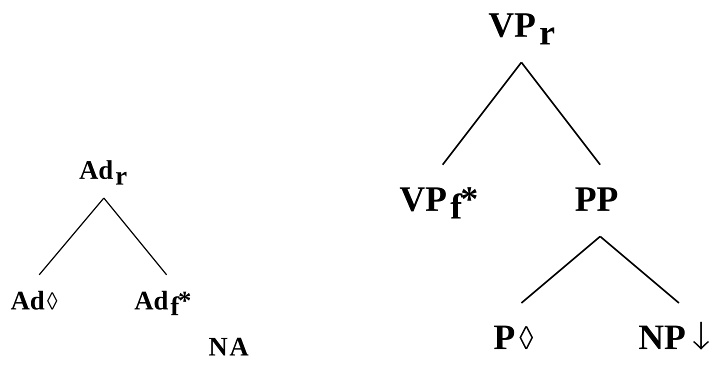
(569) Hobbes eats his supper more quickly than Calvin.

(570) Hobbes more quickly eats his supper than Calvin.

(571) *Hobbes more quickly than Calvin eats his supper.

When *more* is used alone as an adverb, it must also occur after the verb phrase. Also, it appears that adverbs modified by *more* and *less* have the same distribution as when they are not modified. However, the *than* portion of an adverbial comparative is restricted to post verb phrase positions.

The first observation can be captured by having *more* and *less* select only βvxARB from the set of adverb trees. Comparativization of adverbs looks very similar to that of other categories, and we follow this trend by giving the tree in Figure 24.11(a), which parallels the adjectival and nominal trees, for these instances. This handles the quite free distribution of adverbs which have been comparativized, while the tree in Figure 24.11(b), βvxPnx, allows the *than* portion of an adverbial comparative to occur only after the verb phrase, blocking examples such as (571).



(a) βCARBarb tree (b) βvxPnx tree

Figure 24.11: Adverbial comparative trees

The usage of the **compar** feature parallels that of the adjectives and nominals; however, trees which adjoin to VP are **compar-** on their root VP node. In this way, βvxPnx anchored by *than* or *as* (which must adjoin to a **compar+** VP) can only adjoin immediately above a comparative or comparativized adverb. This avoids extra parses in which the comparative adverb adjoins at a VP node lower than the *than*-clause.

A final note is that *as* may anchor βvxPnx non-comparatively, as in sentence (572). This means that there will be two parses for sentences such as (573).

(572) John works as a carpenter.

(573) John works as quickly as a carpenter.

This appears to be a legitimate ambiguity. One is that John works as quickly as a carpenter (works quickly), and the other is that John works quickly when he is acting as a carpenter (but maybe he is slow when he acting as a plumber).

24.4 Future Work

- Interaction with determiner sequencing (e.g., *several more men than women* but not **every more men than women*).
- Handle sentential complement comparisons (e.g., *Bill eats more pasta than Angus drinks beer*).
- Add partitives.
- Deal with constructions like *as many* and *as much*.
- Look at *so...as* construction.

Chapter 25

Punctuation Marks

Many parsers require that punctuation be stripped out of the input. Since punctuation is often optional, this sometimes has no effect. However, there are a number of constructions which must obligatorily contain punctuation and adding analyses of these to the grammar without the punctuation would lead to severe overgeneration. An especially common example is noun appositives. Without access to punctuation, one would have to allow every combinatorial possibility of NPs in noun sequences, which is clearly undesirable (especially since there is already unavoidable noun-noun compounding ambiguity). Aside from coverage issues, it is also preferable to take input "as is" and do as little editing as possible. With the addition of punctuation to the XTAG grammar, we need only do/assume the conversion of certain sequences of punctuation into the "British" order (this is discussed in more detail below in Section 25.2).

The XTAG POS tagger currently tags every punctuation mark as itself. These tags are all converted to the POS tag *Punct* before parsing. This allows us to treat the punctuation marks as a single POS class. They then have features which distinguish amongst them. Wherever possible we have the punctuation marks as anchors, to facilitate early filtering.

The full set of punctuation marks is separated into three classes: balanced, separating and terminal. The balanced punctuation marks are quotes and parentheses, separating are commas, dashes, semi-colons and colons, and terminal are periods, exclamation points and question marks. Thus, the <**punct**> feature is complex (like the <**agr**> feature), yielding feature equations like <**Punct bal = paren**> or <**Punct term = excl**>. Separating and terminal punctuation marks do not occur adjacent to other members of the same class, but may occasionally occur adjacent to members of the other class, e.g. a question mark on a clause which is separated by a dash from a second clause. Balanced punctuation marks are sometimes adjacent to one another, e.g. quotes immediately inside of parentheses. The <**punct**> feature allows us to control these local interactions.

We also need to control non-local interaction of punctuation marks. Two cases of this are so-called quote alternation, wherein embedded quotation marks must alternate between single and double, and the impossibility of embedding an item containing a colon inside of another item containing a colon. Thus, we have a fourth value for <**punct**>, <**contains colon/dquote/etc. +/-**>, which indicates whether or not a constituent contains a particular punctuation mark. This feature is percolated through all auxiliary trees. Things which may not embed are: colons under colons, semi-colons, dashes or commas; semi-colons under semi-colon or commas. Although it is rare, parentheses may appear inside of parentheses, say with a bibliographic reference inside a parenthesized sentence.

25.1 Appositives, parentheticals and vocatives

These trees handle constructions where additional lexical material is only licensed in conjunction with particular punctuation marks. Since the lexical material is unconstrained (virtually any noun can occur as an appositive), the punctuation marks are anchors and the other nodes are substitution sites. There are cases where the lexical material is restricted, as with parenthetical adverbs like *however*, and in those cases we have the adverb as the anchor and the punctuation marks as substitution sites.

When these constructions can appear inside of clauses (non-peripherally), they must be separated by punctuation marks on both sides. However, when they occur peripherally they have either a preceding or following punctuation mark. We handle this by having both peripheral and non-peripheral trees for the relevant constructions. The alternative is to insert the second (following) punctuation mark in the tokenization process (i.e. insert a comma before the period when an appositive appears on the last NP of a sentence). However, this is very difficult to do accurately.

25.1.1 βnxPUnxPU

The symmetric (non-peripheral) tree for NP appositives, anchored by: comma, dash or parentheses. It is shown in Figure 25.1 anchored by parentheses.

(574) The music here , Russell Smith's "Tetrameron " , sounded good . [Brown:cc09]

(575) ...cost 2 million pounds (3 million dollars)

(576) Sen. David Boren (D., Okla.)...

(577) ...some analysts believe the two recent natural disasters – Hurricane Hugo and the San Francisco earthquake – will carry economic ramifications.... [WSJ]

The punctuation marks are the anchors and the appositive NP is substituted. The appositive can be conjoined, but only with a lexical conjunction (not with a comma). Appositives with commas or dashes cannot be pronouns, although they may be conjuncts containing pronouns. When used with parentheses this tree actually presents an alternative rather than an appositive, so a pronoun is possible. Finally, the appositive position is restricted to having nominative or accusative case to block PRO from appearing here.

Appositives can be embedded, as in (578), but do not seem to be able to stack on a single NP. In this they are more like restrictive relatives than appositive relatives, which typically can stack.

(578) ...noted Simon Briscoe, UK economist for Midland Montagu, a unit of Midland Bank PLC.

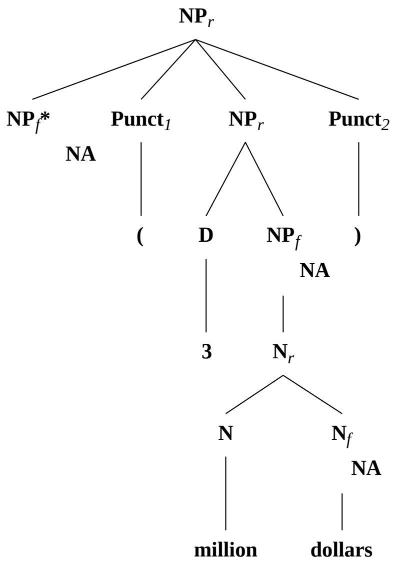


Figure 25.1: The βnxPUnxPU tree, anchored by parentheses

25.1.2 βnPUnxPU

The symmetric (non-peripheral) tree for N-level NP appositives, is anchored by comma. The modifier is typically an address. It is clear from examples such as (579) that these are attached at N, rather than NP. *Carrier* is not an appositive on *Menlo Park*, as it would be if these were simply stacked appositives. Rather, *Calif.* modifies *Menlo Park*, and that entire complex is compounded with *carrier*, as shown in the correct derivation in Figure 25.2. Because this distinction is less clear when the modifier is peripheral (e.g. ends the sentence), and it would be difficult to distinguish between NP and N attachment, we do not currently allow a peripheral N-level attachment.

(579) An official at Consolidated Freightways Inc., a Menlo Park, Calif., less-than-truckload carrier , said...

(580) Rep. Ronnie Flippo (D., Ala.), of the delegation, says...

25.1.3 βnxPUnx

This tree, which can be anchored by a comma, dash or colon, handles asymmetric (peripheral) NP appositives and NP colon expansions of NPs. Figure 25.3 shows this tree anchored by a dash and a colon. Like the symmetric appositive tree, βnxPUnxpu, the asymmetric appositive cannot be a pronoun, while the colon expansion can. Thus, this constraint comes from the syntactic entry in both cases rather than being built into the tree.

(581) the bank's 90% shareholder – Petroliam Nasional Bhd. [Brown]

(582) ...said Chris Dillow, senior U.K. economist at Nomura Research Institute .

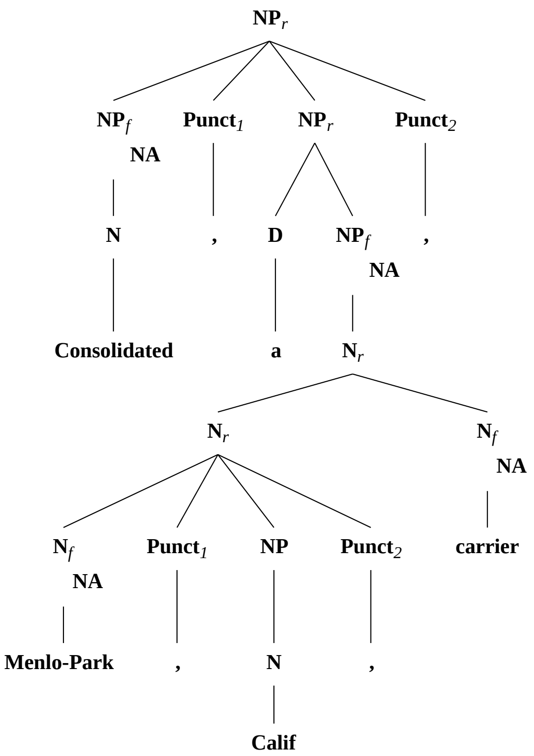


Figure 25.2: An N-level modifier, using the βnPUnx tree

(583) ...qualities that are seldom found in one work: Scrupulous scholarship, a fund of personal experience,... [Brown:cc06]

(584) I had eyes for only one person : him .

The colon expansion cannot itself contain a colon, so the foot S has the feature NP.t:$<$ $punctcontainscolon >= -$.

25.1.4 βPUpxPUvx

Tree for pre-VP parenthetical PP, anchored by commas or dashes -

(585) John , in a fit of anger , broke the vase

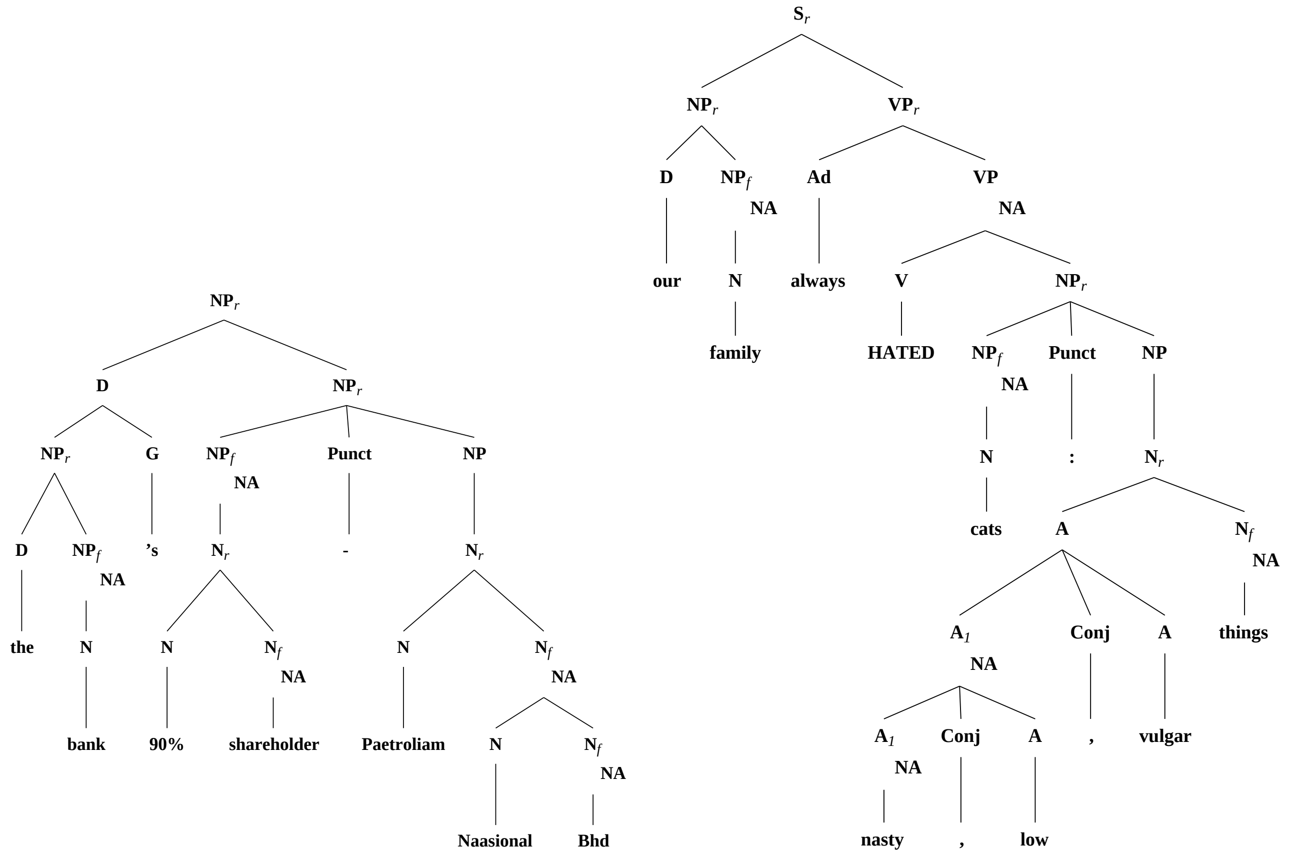
(586) Mary , just within the last year , has totalled two cars

These are clearly not NP modifiers.

Figures 25.4 and 25.5 show this tree alone and as part of the parse for (585).

25.1.5 βpuARBpuvx

Parenthetical adverbs - *however*, *though*, etc. Since the class of adverbs is highly restricted, this tree is anchored by the adverb and the punctuation marks substitute. The punctuation marks



(a) (b)

Figure 25.3: The derived trees for an NP with (a) a peripheral, dash-separated appositive and (b) an NP colon expansion (uttered by the Mouse in *Alice's Adventures in Wonderland*)

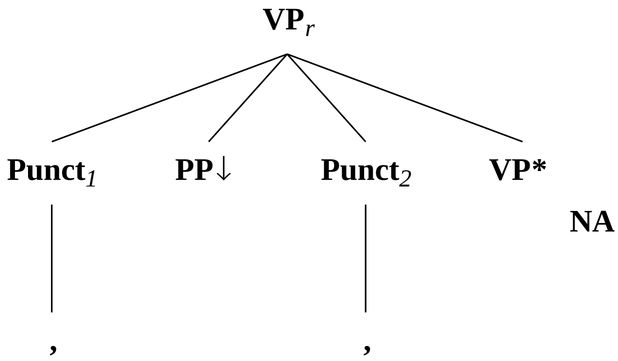


Figure 25.4: The βPUpxPUvx tree, anchored by commas

may be either commas or dashes. Like the parenthetical PP above, these are clearly not NP modifiers.

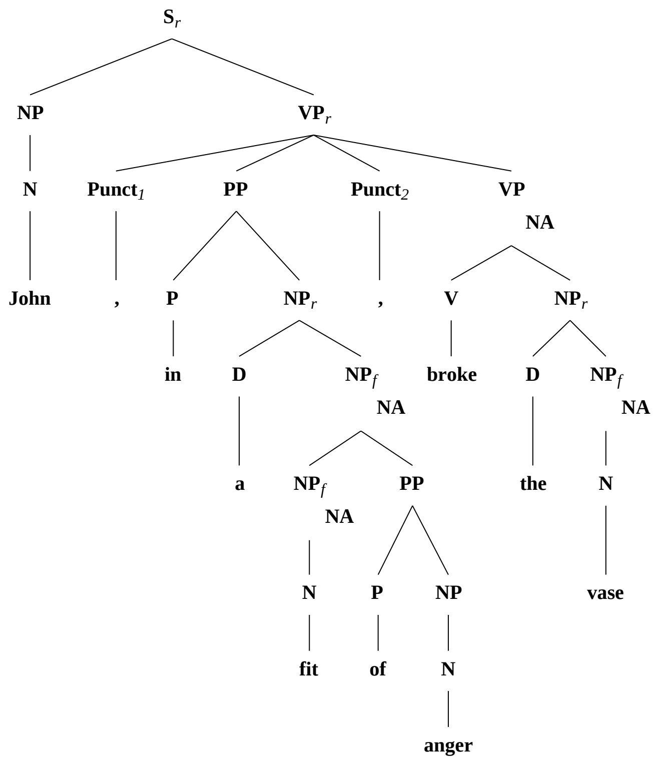


Figure 25.5: Tree illustrating the use of βPUpxPUvx

(587) The new argument over the notification guideline , however , could sour any atmosphere of cooperation that existed . [WSJ]

25.1.6 βsPUnx

Sentence final vocative, anchored by comma:

(588) You were there , Stanley/my boy .

Also, when anchored by colon, NP expansion on S. These often appear to be extraposed modifiers of some internal NP. The NP must be quite heavy, and is usually a list:

(589) Of the major expansions in 1960, three were financed under the R. I. Industrial Building Authority's 100% guaranteed mortgage plan: Collyer Wire, Leesona Corporation, and American Tube & Controls.

A simplified version of this sentence is shown in figure 25.6. The NP cannot be a pronoun in either of these cases. Both vocatives and colon expansions are restricted to appear on tensed clauses (indicative or imperative).

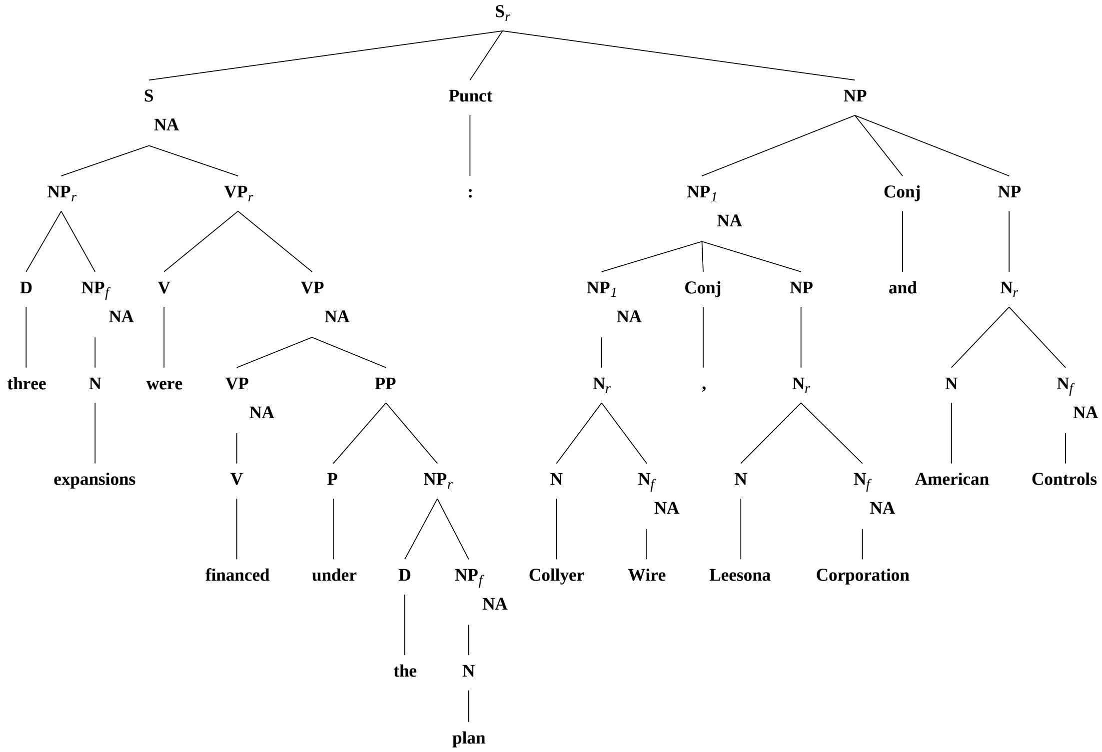


Figure 25.6: A tree illustrating the use of sPUnx for a colon expansion attached at S.

25.1.7 βnxPUs

Tree for sentence initial vocatives, anchored by a comma:

(590) Stanley/my boy , you were there .

The noun phrase may be anything but a pronoun, although it is most commonly a proper noun. The clause adjoined to must be indicative or imperative.

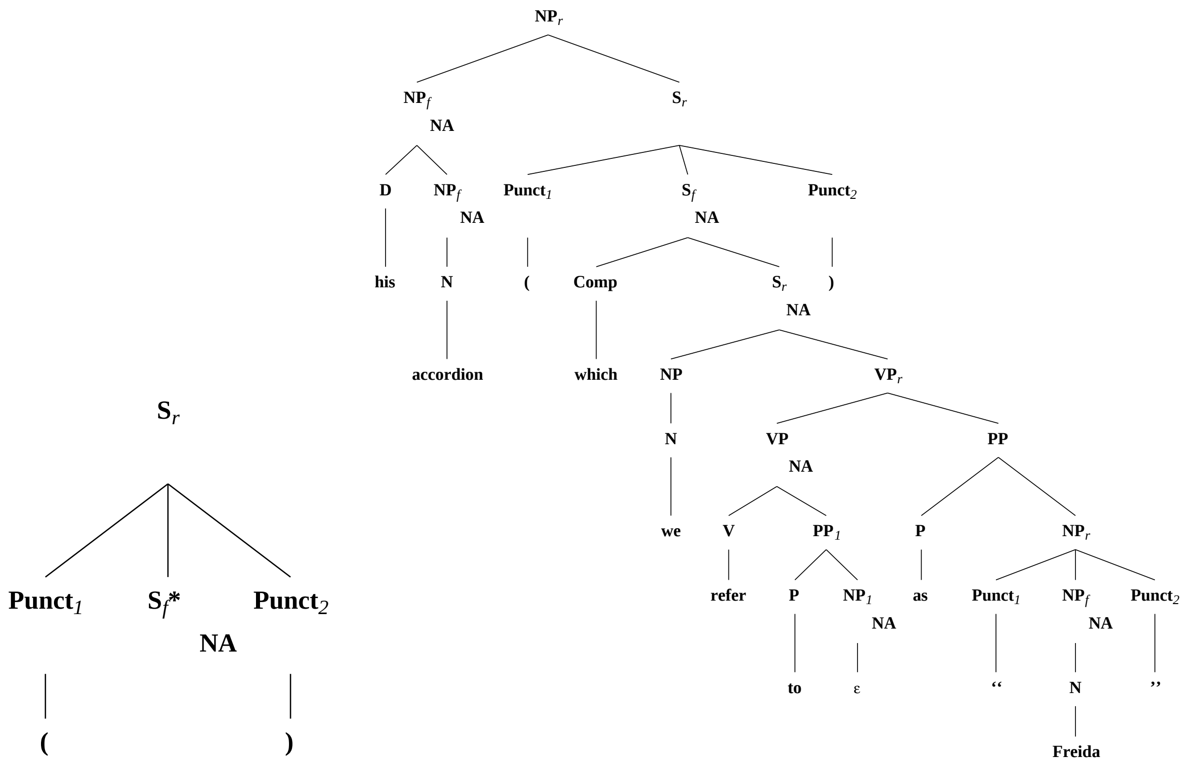
25.2 Bracketing punctuation

25.2.1 Simple bracketing

Trees: βPUsPU, βPUnxPU, βPUnPU, βPUvxPU, βPUvPU, βPUarbPU, βPUaPU, βPUdPU, βPUpxPU, βPUpPU
These trees are selected by parentheses and quotes and can adjoin onto any node type, whether a head or a phrasal constituent. This handles things in parentheses or quotes which are syntactically integrated into the surrounding context. Figure 25.7 shows the βPUsPU anchored by parentheses, and this tree along with βPUnxPU in a derived tree.

(591) Dick Carroll and his accordion (which we now refer to as “Freida”) held over at Bahia Cabana where “Sir” Judson Smith brings in his calypso capers Oct. 13 . [Brown:ca31]

(592) ...noted that the term “teacher-employee” (as opposed to, e.g., “maintenance employee”) was a not inapt description. [Brown:ca35]



(a) (b)

Figure 25.7: βPUsPU anchored by parentheses, and in a derivation, along with βPUnxPU

There is a convention in English that quotes embedded in quotes alternate between single and double; in American English the outermost are double quotes, while in British English they are single. The **contains** feature is used to control this alternation. The trees anchored by double quotation marks have the feature **punct contains dquote** = **-** on the foot node and the feature **punct contains dquote** = + on the root. All adjunction trees are transparent to the **contains** feature, so if any tree below the double quote is itself enclosed in double quotes the derivation will fail. Likewise with the trees anchored by single quotes. The quote trees in effect “toggle” the **contains Xquote** feature. Immediate proximity is handled by the **punct balanced** feature, which allows quotes inside of parentheses, but not vice-versa.

In addition, American English typically places/moves periods (and commas) inside of quotation marks when they would logically occur outside, as in example 593. The comma in the first part of the quote is not part of the quote, but rather part of the parenthetical quoting clause. However, by convention it is shifted inside the quote, as is the final period. British English does not do this. We assume here that the input has already been tokenized into the “British” format.

(593) “You can’t do this to us ,” Diane screamed . “We are Americans.”

The βPUsPU can handle quotation marks around multiple sentences, since the sPUs tree allows us to join two sentences with a period, exclamation point or question mark. Currently, however, we cannot handle the style where only an open quote appears at the beginning of a paragraph when the quotation extends over multiple paragraphs. We could allow a lone open quote to select the βPUs tree, if this is deemed desirable.

Also, the βPUsPU is selected by a pair of commas to handle non-peripheral appositive relative clauses, such as in example (594). Restrictive and appositive relative clauses are not syntactically differentiated in the XTAG grammar (cf. Chapter 16).

(594) This news , announced by Jerome Toobin , the orchestra's administrative director , brought applause ... [Brown:cc09]

The trees discussed in this section will only allow balanced punctuation marks to adjoin to constituents. We will not get them around non-constituents, as in (595).

(595) Mary asked him to leave (and he left)

25.2.2 βsPUsPU

This tree allows a parenthesized clause to adjoin onto a non-parenthesized clause.

(596) Innumerable motels from Tucson to New York boast swimming pools (" swim at your own risk " is the hospitable sign poised at the brink of most pools) . [Brown:ca17]

25.3 Punctuation trees containing no lexical material

25.3.1 αPU

This is the elementary tree for substitution of punctuation marks. This tree is used in the quoted speech trees, where including the punctuation mark as an anchor along with the verb of saying would require a new entry for every tree selecting the relevant tree families. It is also used in the tree for parenthetical adverbs (βpuARBpuvx), and for S-adjoined PPs and adverbs (βspuARB and βspuPnx).

25.3.2 βPUs

Anchored by comma: allows comma-separated clause initial adjuncts, (597-598).

(597) Here , as in "Journal" , Mr. Louis has given himself the lion's share of the dancing... [Brown:cc09]

(598) Choreographed by Mr. Nagrin, the work filled the second half of a program

To keep this tree from appearing on root Ss (i.e. , *sentence*), we have a root constraint that <**punct struct = nil**> (similar to the requirement that root Ss be tensed, i.e. <**mode = ind/imp**>). The <**punct struct**> = **nil** feature on the foot blocks stacking of multiple punctuation marks. This feature is shown in the tree in Figure 25.8.

This tree can be also used by adjuncts on embedded clauses:

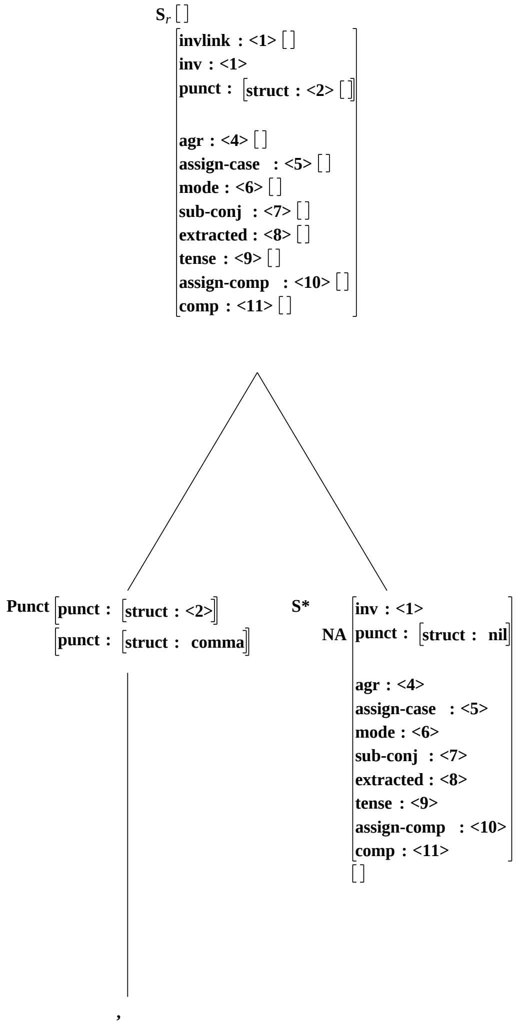


Figure 25.8: βPUs, with features displayed

(599) One might expect that in a poetic career of seventy-odd years, some changes in style and method would have occurred, some development taken place. [Brown:cj65]

These adjuncts sometimes have commas on both sides of the adjunct, or, like (599), only have them at the end of the adjunct.

Finally, this tree is also used for peripheral appositive relative clauses.

(600) Interest may remain limited into tomorrow's U.K. trade figures, which the market will be watching closely to see if there is any improvement after disappointing numbers in the previous two months.

25.3.3 βsPUs

This tree handles clausal "coordination" with comma, dash, colon, semi-colon or any of the terminal punctuation marks. The first clause must be either indicative or imperative. The second may also be infinitival with the separating punctuation marks, but must be indicative or imperative with the terminal marks; with a comma, it may only be indicative. The two clauses need not share the same mode. NB: Allowing the terminal punctuation marks to anchor this tree allows us to parse sequences of multiple sentences. This is not the usual mode of parsing; if it were, this sort of sequencing might be better handled by a higher level of processing.

(601) For critics , Hardy has had no poetic periods – one does not speak of early Hardy or late Hardy , or of the London or Max Gate period....

(602) Then there was exercise , boating and hiking , which was not only good for you but also made you more virile : the thought of strenuous activity left him exhausted.

This construction is one of the few where two non-bracketing punctuation marks can be adjacent. It is possible (if rare) for the first clause to end with a question mark or exclamation point, when the two clauses are conjoined with a semi-colon, colon or dash. Features on the foot node, as shown in Figure 25.9, control this interaction.

Complementizers are not permitted on either conjunct. Subordinating conjunctions sometimes appear on the right conjunct, but seem to be impossible on the left:

(603) Killpath would just have to go out and drag Gun back by the heels once an hour ; because he'd be damned if he was going to be a mid-watch pencil-pusher . [Brown:cl17]

(604) The best rule of thumb for detecting corked wine (provided the eye has not already spotted it) is to smell the wet end of the cork after pulling it : if it smells of wine , the bottle is probably all right ; if it smells of cork , one has grounds for suspicion. [Brown:cf27]

25.3.4 βsPU

This tree handles the sentence final punctuation marks when selected by a question mark, exclamation point or period. One could also require a final punctuation mark for all clauses, but such an approach would not allow non-periods to occur internally, for instance before a semi-colon or dash as noted above in Section 25.3.3. This tree currently only adjoins to indicative or imperative (root) clauses.

(605) He left !

(606) Get lost .

(607) Get lost ?

The feature **punct bal= nil** on the foot node ensures that this tree only adjoins inside of parentheses or quotes completely enclosing a sentence (608), but does not restrict it from adjoining to clause which ends with balanced punctuation if only the end of the clause is contained in the parentheses or quotes (609).

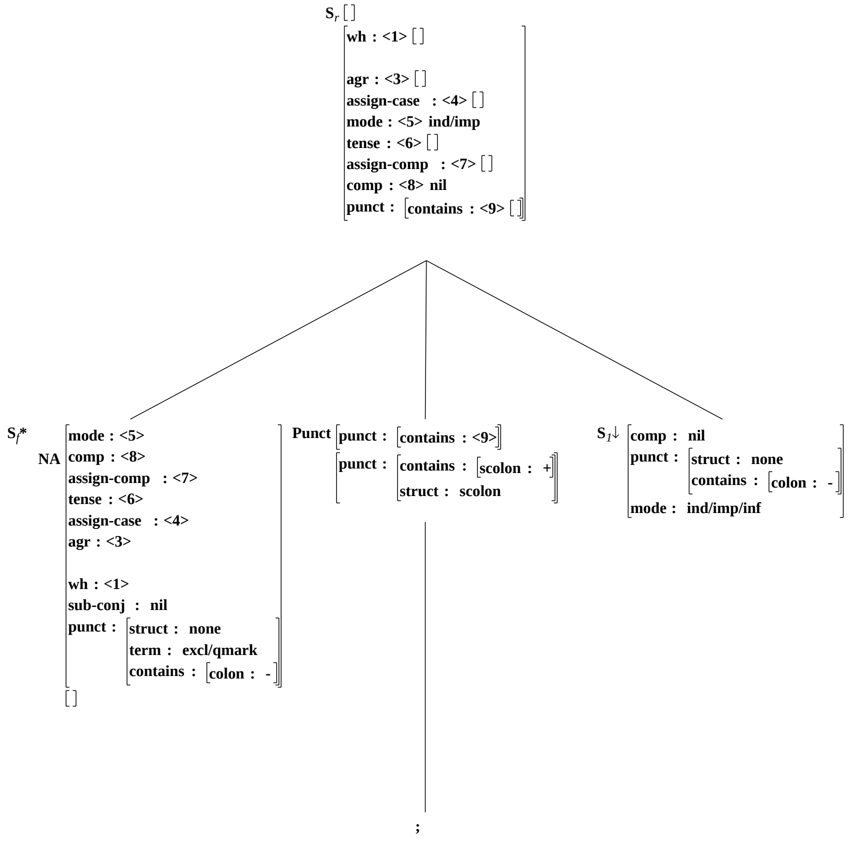


Figure 25.9: βsPUs, with features displayed

(608) (John then left .)

(609) (John then left) .

(610) Mary asked him to leave (immediately) .

This tree is also selected by the colon to handle a colon expansion after adjunct clause –

(611) Expressed differently : if the price for becoming a faithful follower... [Brown:cd02]

(612) Expressing it differently : if the price for becoming a faithful follower...

(613) To express it differently : if the price for becoming a faithful follower... [Brown:cd02]

This tree is only used after adjunct (untensed) clauses, which adjoin to the tensed clause using the adjunct clause trees (cf Section 17); the **mode** of the complete clause is that of the matrix rather than the adjunct. Indicative or imperative (i.e. root) clauses separated by a colon use the βsPUs tree (Section 25.3.3).

25.3.5 βvPU

This tree is anchored by a colon or a dash, and occurs between a verb and its complement. These typically are lists.

(614) Printed material Available , on request , from U.S. Department of Agriculture , Washington 25 , D.C. , are : Cooperative Farm Credit Can Assist...... [Brown:ch01]

25.3.6 βpPU

This tree is anchored by a colon or a dash, and occurs between a preposition and its complement. It typically occurs with a sequence of complements. As with the tree above, this typically occurs with a conjoined complement.

(615) ...and utilization such as : (A) the protection of forage...

(616) ...can be represented as : Af.

25.4 Other trees

25.4.1 βspuARB

In general, we attach post-clausal modifiers at the VP node, as you typically get scope ambiguity effects with negation (*John didn't leave today* – did he leave or not?). However, with post-sentential, comma-separated adverbs, there is no ambiguity - in *John didn't leave, today* he definitely did not leave. Since this tree is only selected by a subset of the adverbs (namely, those which can appear pre-sententially, without a punctuation mark), it is anchored by the adverb.

(617) The names of some of these products don't suggest the risk involved in buying them , either . [WSJ]

25.4.2 βspuPnx

Clause-final PP separated by a comma. Like the adverbs described above, these differ from VP adjoined PPs in taking widest scope.

(618) ...gold for current delivery settled at $367.30 an ounce , up 20 cents .

(619) It increases employee commitment to the company , with all that means for efficiency and quality control .

25.4.3 βnxPUa

Anchored by colon or dash, allows for post-modification of NPs by adjectives.

(620) Make no mistake , this Gorky Studio drama is a respectable import – aptly grave , carefully written , performed and directed .

Part V

Appendices

Chapter 26

Future Work

26.1 Adjective ordering

At this point, the treatment of adjectives in the XTAG English grammar does not include selectional or ordering restrictions.[1] Consequently, any adjective can adjoin onto any noun and on top of any other adjective already modifying a noun. All of the modified noun phrases shown in (621)-(624) currently parse.

(621) big green bugs

(622) big green ideas

(623) colorless green ideas

(624) ∗green big ideas

While (622)-(624) are all semantically anomalous, (624) also suffers from an ordering problem that makes it seem ungrammatical as well. Since the XTAG grammar focuses on syntactic constructions, it should accept (621)-(623) but not (624). Both the auxiliary and determiner ordering systems are structured on the idea that certain types of lexical items (specified by features) can adjoin onto some types of lexical items, but not others. We believe that an analysis of adjectival ordering would follow the same type of mechanism.

26.2 More work on Determiners

In addition to the analysis described in Chapter 20, there remains work to be done to complete the analysis of determiner constructions in English.[2] Although constructions such as determiner coordination are easily handled if overgeneration is allowed, blocking sequences such as *one and some* while allowing sequences such as *five or ten* still remains to be worked out. There are still a handful of determiners that are not currently handled by our system. We do not have

[1]This section is a repeat of information found in section 21.1.
[2]This section is from [Hockey and Mateyak, 1998].

an analysis to handle *most*, *such*, *certain*, *other* and *own*[3]. In addition, there is a set of lexical items that we consider adjectives (*enough*, *less*, *more* and *much*) that have the property that they cannot cooccur with determiners. We feel that a complete analysis of determiners should be able to account for this phenomenon, as well.

26.3 Removal of empty elements as anchors

In the current version of the grammar, we dispensed with some empty elements that were lexicalizing trees in the grammar. One was the null COMP used in relative clauses, the second was the empty verb anchoring the auxiliary verb tree used for multi-component adjunction, and the third was the non-lexical PRO which anchored the initial NP tree and substituted into the subject NP nodes of non-ECM infinitival clauses. However, there is still one more empty element remaining in the grammar which we also need to remove. This is the empty prepositional element anchoring the tree for subordinating conjunctions (see Chapter 17). These non-lexicalized trees are used to handle bare adjunct clauses like:

(625) The fire raged for days, destroying the building .

26.4 Verb selectional restrictions

Although we explicitly do not want to model semantics in the XTAG grammar, there is some work along the syntax/semantics interface that would help reduce syntactic ambiguity and thus decrease the number of semantically anomalous parses. In particular, verb selectional restrictions, particularly for PP arguments and adjuncts, would be quite useful. With the exception of the required *to* in the Ditransitive with PP Shift tree family (Tnx0Vnx1Pnx2), any preposition is allowed in the tree families that have prepositions as their arguments. In addition, there are no restrictions as to which prepositions are allowed to adjoin onto a given verb. The sentences in (626)-(628) are all currently accepted by the XTAG grammar. Their violations are stronger than would be expected from purely semantic violations, however, and the presence of verb selectional restrictions on PP's would keep these sentences from being accepted.

(626) #survivors walked of the street .

(627) #The man about the earthquake survived .

(628) #The president arranged on a meeting .

26.5 Thematic Roles

Elementary trees in TAGs capture several notions of locality, with the most primary of these being locality of θ-role assignment. Each elementary tree has associated with it the θ-roles

[3]The behavior of *own* is sufficiently unlike other determiners that it most likely needs a tree of its own, adjoining onto the right-hand side of genitive determiners.

assigned by the anchor of that elementary tree. In the current XTAG system, while the notion of locality of θ-role assignment within an elementary tree has been implicit, the θ-roles assigned by a head have not been explicitly represented in the elementary tree. Incorporating θ-role information will make the elementary trees more informative and will enable efficient pruning of spurious derivations when embedded into a specific context. In the case of a Synchronous TAG, θ-roles can also be used to automatically establish links between two elementary trees, one in the object language and one in the target language.

Chapter 27

Metarules

27.1 Introduction

The system of metarules is a collection of functions accessible from the XTAG user interface that help the user in the construction and maintenance of a tag tree-grammar. Here our primary purpose is to describe the facilities that exist for using metarules. For a discussion of metarules as a method for compact representation of the Lexicon see [Becker, 1993] and [Srinivas *et al.*, 1994].

The basic idea of using metarules is to take advantage of the similarities of the relations involving related pairs of XTAG elementary trees. For example, in the English grammar described in this technical report, comparing the XTAG trees for the basic form and the wh-subject moved form, the relation between this two trees for transitive verbs (αnx_0Vnx_1, $\alpha W_0nx_0Vnx_1$) is similar to the relation for the intransitive verbs (αnx_0V, αW_0nx_0V) and also to the relation for the ditransitives ($\alpha nx_0Vnx_1nx_2$, $\alpha W_0nx_0Vnx_1nx_2$). Hence, instead of generating by hand the six trees mentioned above, a more natural and robust way would be generating by hand only the basic trees for the intransitive, transitive and ditransitive cases, and letting the wh-subject moved trees to be automatically generated by the application of a unique transformation rule that would account exactly for the identical relation involved in each of the three pairs above.

Notice that the degree of generalization can be much higher than it might be thought in principle from the above paragraph. For example, once a rule for passivization is applied to the three different basic trees above, the wh-subject moved rule could be again applied to generate the wh-moved subject versions for the passive form. It is important to note that such recursive applications of metarules still always result in a finite number of trees in the grammar.

We still make here a point that the reduction of effort in grammar construction is not the only advantage of the approach. Robustness, reliability and maintainability of the grammar achieved by the use of metarules are equally or even more important.

In the next section we define a metarule in XTAG. Section 27.3 gives some linguistically motivated examples of metarules for the English grammar described in this technical report and their application. Section 27.4 describes the access through the user interface.

27.2 The definition of a metarule in XTAG

A metarule specifies a rule for transforming grammar rules into grammar rules. In XTAG the grammar rules are lexicalized trees. Hence an XTAG metarule **mr** is a pair **(lhs, rhs)** of XTAG trees, where:

- **lhs**, the *left-hand side* of the metarule, is a pattern tree, i.e., it is intended to present a specific pattern of tree to look for in the trees submitted to the application of the metarule.
- When a metarule **mr** is applied to an input tree **inp**, the first step is to verify if the input tree matches the pattern specified by the **lhs**. If there is no match, the application *fails*.
- **rhs**, the *right-hand side* of the metarule, specifies (together with **lhs**) the transformation that will be done in **inp**, in case of successful matching, thus generating the output tree of the metarule application[1].

27.2.1 Node names, variable instantiation, and matches

We will use the terms **lhs**, **rhs** and **inp** as introduced above to refer to the parts of a generic metarule being applied to an input tree.

The nodes at **lhs** can take three different forms: a constant node, a typed variable node, and a non-typed variable node. The naming conventions for these different classes of nodes is given below.

- **Constant Node:** Its name must not begin with a question mark ('?' character). They follow the same conventions used in normal XTAG trees; for instance, **inp** is expected to have only constant nodes. Some examples of constant nodes are NP, V, NP_0, NP_1, S_r. We will call the two parts that compose such names the *stem* and the *subscript*. In the examples above NP, V and S are stems and 0, 1, r are subscripts. Notice that the subscript part can also be empty.
- **Non-Typed Variable Node:** Its name begins with a question mark ('?'), followed by a sequence of digits (i.e., a number) which uniquely identifies the variable. Examples: ?1, ?3, ?3452.[2] There is no stem and no subscript in these names, i.e., '?' is just a meta-character to introduce a variable, and the number is the variable identifier.
- **Typed Variable Node:** Its name begins with a question mark ('?') followed by a sequence of digits, but is additionally followed by a *type specifiers definition*. A *type specifiers definition* is a sequence of one or more *type specifier* separated by a slash ('/'). A *type specifier* has the same form of a regular XTAG node name (like the constant nodes), except that the subscript can be also a question mark. Examples of typed variables are: ?1VP (a single type specifier with stem VP and no subscript), ?3NP_1/PP (two type specifiers, NP_1 and PP), ?1$NP_?$ (one type specifier, $NP_?$ with undetermined subscript).

[1]Actually more than one output tree can be generated from the successful application of a rule to an input tree, as will be seen later in this chapter

[2]Notice however that having the sole purpose of distinguishing between variables, a number like the one in the last example is not very likely to occur, and a metarule with more than three thousand variables can give you a place in the Guinness TagBook of Records.

Each type specifier represents an alternative for matching, and the presence of '?' in subscript position of a type specifier means that matching will only check for the stem [3].

During the process of matching, variables are associated (we use the term *instantiated*) with 'tree material'. According to its class a variable may be instantiated with different kinds of tree material:

- A typed variable will be instantiated with exactly one node of the input tree, which is in accordance to one of its type specifiers (The full rule is in the following subsection).
- A non-typed variable will be instantiated by a sequence of subtrees. These subtrees will be taken from one of the nodes of the input tree **inp**. Hence, there will be a node n in **inp**, with subtrees $n.t_1$, $n.t_2$, ..., $n.t_k$, in this order, where the variable will be instantiated with some subsequence of these subtrees (e.g., $n.t_2$, $n.t_3$, $n.t_4$). Note however, that some of these subtrees, may be incomplete, i.e., they may not go all the way to the bottom leaves. Entire subtrees may be removed. Actually for each child of the non-typed variable node, one subtree that matches this child subtree will be removed from some of the $n.t_i$(maybe an entire $n.t_i$), leaving in place a mark for inserting material during the substitution of occurrences at **rhs**.
  Notice still that the variable may be instantiated with a single tree and even with no tree.

We define a *match* to be a complete instantiation of all variables appearing in the metarule. In the process of matching, there may be several possible ways of instantiating the set of variables of the metarule, i.e., several possible matches. This is due to the presence of non-typed variables.

Now, we are ready to define what we mean by a successful matching. The process of matching is *successful* if the number of possible matches is greater then 0. When there is no possible match the process is said to *fail*. In addition to returning success or failure, the matching process also returns the set of all possible *matches*, which will be used for generating the output.

27.2.2 Structural Matching

The process of matching **lhs** with **inp** can be seen as a recursive procedure for matching trees, starting at their roots and proceeding in a top-down style along with their subtrees. In the explanation of this process that follows we use the term **lhs** not only to refer to the whole tree that contains the pattern but to any of its subtrees that is being considered in a given recursive step. The same applies to **inp**. By now we ignore feature equations, which will be accounted for in the next subsection.

The process described below returns at the end the set of matches (where an empty set means failure). We first give one auxiliary definition, of valid Mapping, and one recursive function Match, that matches lists of trees instead of trees, and then define the process of matching two trees as a special case of call to Match.

[3] This is different from not having a subscript, which is interpreted as checking that the matching node at the input tree has no subscript

Given a list $list_{lhs} = [lhs_1, lhs_2, ..., lhs_l]$ of nodes of **lhs** and a list $list_{inp} = [inp_1, inp_2, ..., inp_i]$ of nodes of **inp**, we define a *mapping* from $list_{lhs}$ to $list_{inp}$ to be a function $Mapping$, that for each element of $list_{lhs}$ assigns a list of elements of $list_{inp}$, defined by the following condition:

$$\textit{concatenation}\ (Mapping(lhs_1),\ Mapping(lhs_2),\ ...,\ Mapping(lhs_l))\ =\ list_{inp}$$

That is, the elements of $list_{inp}$ are split into sublists and assigned in order of appearance in the list to the elements of $list_{lhs}$.

We say that a mapping is a *valid mapping* if for all j, $1 \leq j \leq l$ (where l is the length of $list_{lhs}$), the following restrictions apply:

1. if lhs_j is a constant node, then $Mapping(lhs_j)$ must have a single element, say, $inp_{g(j)}$, and the two nodes must have the same name and agree on the markers (foot, substitution, head and NA), i.e., if lhs_j is NA, then $inp_{g(j)}$ must be NA, if lhs_j has no markers, then $inp_{g(j)}$ must have no markers, etc.

2. if lhs_j is a typed variable node, then $Mapping(lhs_j)$ must have a single element, say, $inp_{g(j)}$, and $inp_{g(j)}$ must be *marker-compatible* and *type-compatible* with lhs_j.
   $inp_{g(j)}$ is *marker-compatible* with lhs_j if any marker (foot, substitution, head and NA) present in lhs_j is also present in $inp_{g(j)}$[4].
   $inp_{g(j)}$ is *type-compatible* with lhs_j if at least one of the alternative type specifiers for the typed variable lhs_j satisfies the conditions below:

   - $inp_{g(j)}$ has the stem defined in the type specifier.
   - if the type specifier doesn't have a subscript, then $inp_{g(j)}$ must have no subscript.
   - if the type specifier has a subscript different from '?', then $inp_{g(j)}$ must have the same subscript as in the type specifier [5].

3. if lhs_j is a non-typed variable node, then there is no requirement: $Mapping(lhs_j)$ may have any length and even be empty.

The following algorithm, Match, takes as input a list of nodes of **lhs** and a list of nodes of **inp**, and returns the set of possible matches generated in the attempt of match this two lists. If the result is an empty set, this means that the matching failed.

Finally we can define the process of structurally matching **lhs** to **inp** as the evaluation of Match([root(**lhs**)], [root(**inp**)]. If the result is an empty set, the matching failed, otherwise the resulting set is the set of possible matches that will be used for generating the new trees (after being pruned by the feature equation matching).

[4]Notice that, unlike the case for the constant node, the inverse is not required, i.e., if lhs_j has no marker, $inp_{g(j)}$ is still allowed to have one.

[5]If the type specifier has a '?' subscript, there is no restriction, and that is exactly its function: to allow for the matching to be independent of the subscript.

```
Function Match (list_lhs, list_inp)
    Let MAPPINGS be the list of all valid mappings from list_lhs to list_inp
*   Make MATCHES = ∅
    For each mapping Mapping ∈ MAPPINGS do:
        Make Matches = {∅}
        For each j, 1 ≤j≤l, where l=length(list_lhs), do:
            If lhs_j is a constant node, then
               Let children_lhs be the list of children of lhs_j
                   inp_g(j) be the single element in Mapping(lhs_j)
                   children_inp be the list of children of inp_g(j)
               Make Matches = {m∪m_j | m ∈ Matches and
                                      m_j ∈ Match(children_lhs, children_inp)}
            If lhs_j is a typed variable node, then
               Let children_lhs be the list of children of lhs_j
                   inp_g(j) be the single element in Mapping(lhs_j)
                   children_inp be the list of children of inp_g(j)
               Make Matches = { {(lhs_j, inp_g(j))}∪m∪m_j | m ∈ Matches and
                                      m_j ∈ Match(children_lhs, children_inp) }
            If lhs_j is a non-typed variable node, then
               Let children_lhs be the list of children of lhs_j
                   sl be the number of nodes in children_lhs
                   DESC be the set of sl-sized lists given by:
                          DESC = {[di_1, di_2, ..., di_sl] |
                                 For every 1 ≤k≤sl, di_k is a descendant
                                          of some node in Mapping(lhs_j)^1
                                 For every 1 ≤k≤sl, di_k is to the right of di_{k-1}^2 }.
                   Make matches = ∅
                   For every list Desc = [di_1, di_2, ..., di_sl] ∈ DESC do:
                       Let TreeMaterial be the list of subtrees dominated
                                 by the nodes in Mapping(lhs_j), but, with the
                                 subtrees dominated by the nodes in Desc
                                 cut off from these trees
                       Make matches = matches ∪ {{(lhs_j, TreeMaterial)}∪m_j |
                              m_j ∈ Match(children_lhs, Desc)}
                   Make Matches = { m∪m_j | m ∈ Matches and m_j ∈ matches }
        Make MATCHES = MATCHES ∪ Matches
    Return MATCHES
```

[1]It's not necessary to be a proper descendant, i.e., di_k may be a node in $Mapping(lhs_j)$

[2]Recall that a node n is to the right of a node m, if n and m are not descendant of each other, and all the leaves dominated by n are to the right of the leaves dominated by m.

Figure 27.1: Metarule matching algorithm

27.2.3 Output Generation

Although nothing has yet been said about the feature equations (see the next subsection), we assume that only matches that meet the additional constraints imposed by feature equations are considered for output. If no structural match survives feature equations checking, that matching has failed.

If the process of matching **lhs** to **inp** fails, there are two alternative behaviors according to the value of a parameter[6]. If the parameter is set to "false", which is the *default* value, no output is generated. On the other hand, if it is set to "true", then the own **inp** tree is copied to the output[7].

If the process of matching succeeds, as many trees will be generated in the output as the number of possible matches obtained in the process. For a given match, the output tree is generated by substituting the occurrences of variables in the **rhs** tree of the metarule by the material to which they have been instantiated in the match. In the case of typed-variables, the name of the variable is just substituted by the name of the node to which it has been instantiated from **inp**. An important detail is how the markers (foot, substitution, head, NA) are set in the output tree node. The rule is that, if the occurrence of the variable at the **rhs** tree has any marker, then the generated node will inherit ALL markers from the **rhs** node. If it has no marker, then all the markers at the generated node will come from **inp**. The exception to the rule is that if the generated node has any children, then it will not be marked substitution, foot, or anchor, despite what the previous rule say.

The case of non-typed variables, not surpringly, is not so simple. In the output tree, this node will be substituted by the subtree list that was associated to this variable, in the same order, attaching to the parent of this non-typed variable node. But remember that some subtrees may have been removed from some of the trees in this list (maybe entire elements of this list) due to the effect of the children of the metavariable in **lhs**. It is a requirement that any occurence of a non-typed variable node at the **rhs** tree has exactly the same number of children as the unique occurence of this non-typed variable node in **lhs**. Hence, when generating the output tree, the subtrees at **rhs** will be inserted exactly at the points where subtrees were removed during matching, in a positional, one to one correspondance. If one wants any of these cutting points to be left empty, as if the subtree has been removed at the substitution points, the corresponding children at the **rhs** tree have to be labeled **EMPTY**.

27.2.4 Feature Matching

In the previous subsections we have considered only the aspects of a metarule involving the structural part of the XTAG trees. In a feature based grammar such as XTAG, accounting for features is essential. A metarule must account for the proper change of feature equations[8] from the input to the output tree. The aspects that have to be considered here are:

- Which feature equations should be required to be present in **inp** in order for the match to succeed.

[6] The parameter is accessible from the Lisp interface by the name *XTAG::*metarules-copy-unmatched-trees**. At the end of section 27.4 it is shown how to change the value of this parameter through the XTAG interface.

[7] As will be seen in section 27.4 the cumulative mode of application is not affected by this parameter.

[8] Notice that what is really important is not the features themselves, but the feature equations that relate the feature values of nodes of the same tree.

- Which feature equations should be generated in the output tree as a function of the feature equations in the input tree.

Based on the combinations of these aspects the user may specify for a feature equation to be considered in the following ways:

- *Require & Retain:* The feature equation is required to be in **inp** in order for matching to succeed. Upon matching, the equation will be copied to the output tree. To achieve this behaviour, the equation must be placed in the **lhs** tree of the metarule preceded by a plus character (e.g. $+V.t :< trans >= +$).[9]

- *Require & Don't Copy:* The equation is required to be in **inp** for matching, but should not be copied to the output tree. To have this behavior, the equation must be in **lhs** preceded by the minus character (e.g. $-NP_1 :< case >= acc$).

- *Negative Requirement:* In order for matching to succeed the equation should not be found in **inp**. This is achieved by including the equation in **lhs** preceded by a caret (e.g., $\hat{}V.b :< passive >= -$).

- *Optional & Don't Copy:* The equation is not required for matching, but we have to make sure not to copy it to the output tree set of equations, regardless of it being present or not in **inp**. Those equations must be in **lhs** in raw form, i.e., neither preceded by a plus nor minus character (e.g. $S_r.b :< perfect >= VP.t :< perfect >$).

- *Optional & Retain:* The equation is not required for matching, but in case it is found in **inp** it must be copied to the output tree. This is the *default* case, and hence these equations should not be present in the metarule specification.

- *Add:* The equation is not required for matching but we want it to be put in the output tree anyway. These equations are placed in raw form in the **rhs** (notice in this case it is the right hand side tree).

There are also some directives available to handle feature equations. For convenience and conformance with the pre-existing interface functions, these directives were defined to be included in the **lhs** tree as feature equations among the equations for requirement/copy defined above. The equations/directives will be interpreted and executed in the order they appear in the feature equations list definition of the **lhs** tree. The general format of a directive is:

$$\mathrm{D}\textit{name1} : < \textit{dummy1} > = \textit{name2}$$

where "D" is a single character that specifies the directive, as listed below; *name1* and *name2* are parameters of the directive; and *dummy1* (which may literally be the word "dummy") is required only to conform to the pre-existing format for feature equations in XTAG. There must be no blank space between the directive symbol and *name1*. The currently implemented directives are:

$\backslash$: replace *name1* with *name2* in all equations from the input tree.

[9] Commutativity of equations is accounted for in the system. Hence an equation $x = y$ can also be specified as $y = x$. Associativity is not accounted for and its need by an user is viewed as indicating misspecification at the input trees.

| : exchange *name1* and *name2* in all equations from the input tree.

! : remove all equations where *name1* appears (here, *name2* is also dummy).

Notice that the equations are executed in order. Hence, for instance, if there is a directive to remove equation E before an equation that requires E to be present for matching, matching will always fail. Similarly replacing and exchanging feature names will affect and be affected by the behavior of surrounding equations.

In the rest of this subsection we define the role of the meta-variables in the feature equations. Importantly, we will overload the notation used up til now in this chapter and use the terms "left" and "right hand side" (**lhs** and **rhs** for short) to refer to the sides of a feature equation from a tree, and not only to the sides of the metarule.

Non-typed variables don't play any role in the manipulation of feature equations by the metarules. Typed variables can be used in feature equations in both **lhs** and **rhs** side of the equation. They are intended to represent the nodes of the input tree with which the variables have been instantiated. For each resulting match from the structural matching process the following is done:

- The (typed) variables in the equations at **lhs** and **rhs** are substituted by the names of the nodes they have been instantiated to.
- The requirements concerning feature equations are checked, according to the above rules.
- If the match survives feature equation checking, the proper output tree is generated, according to Section 27.2.3 and to the rules described above for the feature equations.

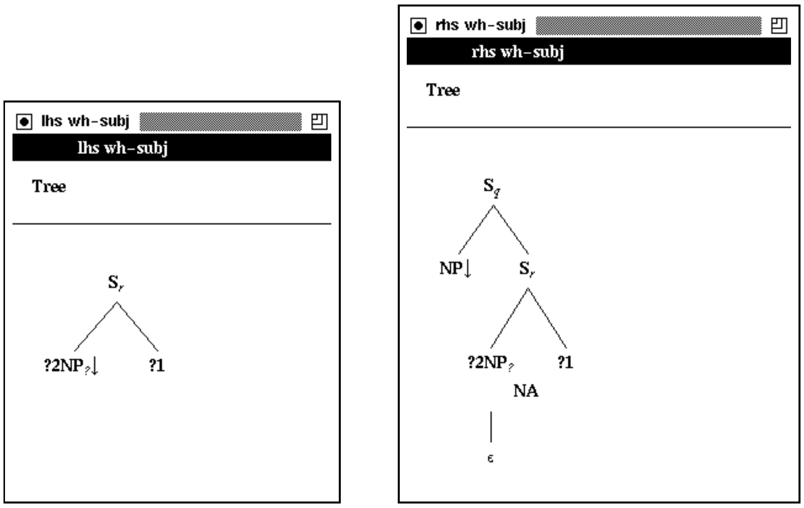
Finally, a new kind of metavariable, which is not used at the nodes, can be introduced in the feature equations part. It has the same form of the non-typed variables, i.e., a question mark followed by a number, and is used in place of feature values and feature names. Hence, if the equation $NP_r.b :<?2>=?3$ appears in **lhs**, then all feature equations of **inp** that equate a certain bottom attribute(?2) of node NP_r to a certain feature value (?3) (but not to a feature path) will not be copied to the output. Notice that before the first time the variables ?2 and ?3 match an equation from **inp**, they are free and can match any feature value/name. But after a match occurs, they become bound to the matched values, and henceforth they will only match these values.

27.3 Examples

Figure 27.2 shows a metarule for wh-movement of the subject. Among the trees to which it have been applied are the basic trees of intransitive, transitive and ditransitive families (including prepositional complements), passive trees of the same families, and ergative.

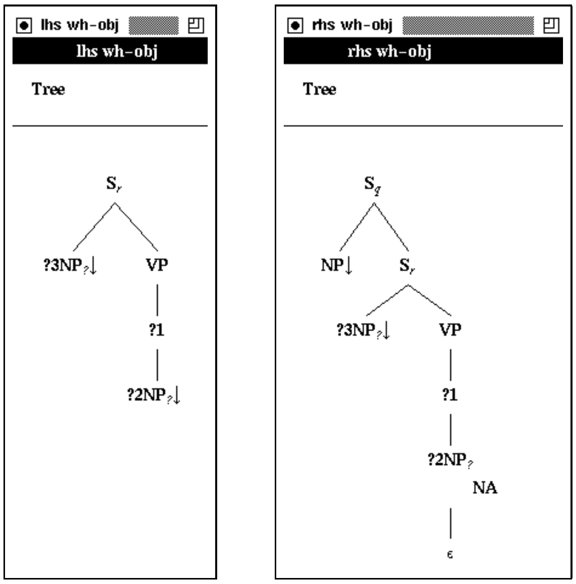
Figure 27.3 shows a metarule for wh-movement of an NP in object position. Among the trees to which it have been applied are the basic and passive trees of transitive and ditransitive families.

Figure 27.4 shows a metarule for general wh-movement of an NP. It can be applied to generate trees with either subject or object NP moved. Figure 27.5 shows the basic tree for the family Tnx0Vnx1Pnx2 and the three wh-trees generated by the application of the rule.



lhs rhs

Figure 27.2: Metarule for wh-movement of subject

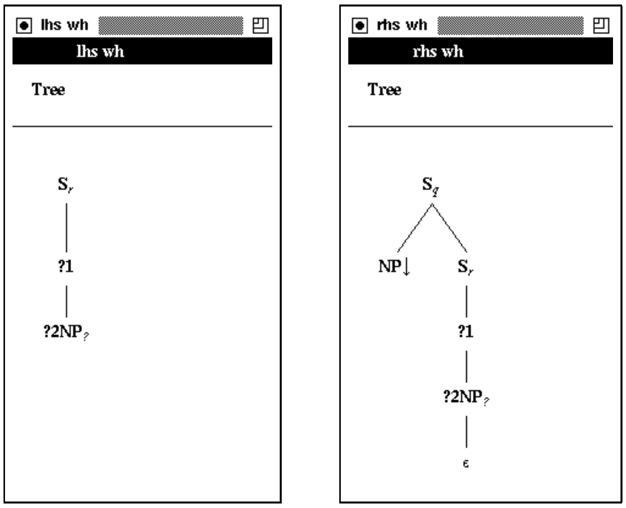


lhs rhs

Figure 27.3: Metarule for wh-movement of object

27.4 The Access to the Metarules through the XTAG Interface

We first describe the access to the metarules subsystem using buffers with single metarule applications. Then we proceed by describing the application of multiple metarules in what we



lhs rhs

Figure 27.4: Metarule for general wh movement of an NP

call the parallel, sequential, and cumulative modes to input tree files.

We have defined a metarule conceptually as an ordered pair of trees. In the implementation of the metarule subsystem it works the same: a metarule is a buffer with two trees. The name of the metarule is the name of the buffer. The first tree that appears in the main window under the metarule buffer is the *left hand side*, the next appearing below is the *right hand side*[10]. The positional approach allows us to have naming freedom: the tree names are irrelevant[11]. Since we can save buffers into text files, we can also talk about metarule files.

The available options for applying a metarule which is in a buffer are:

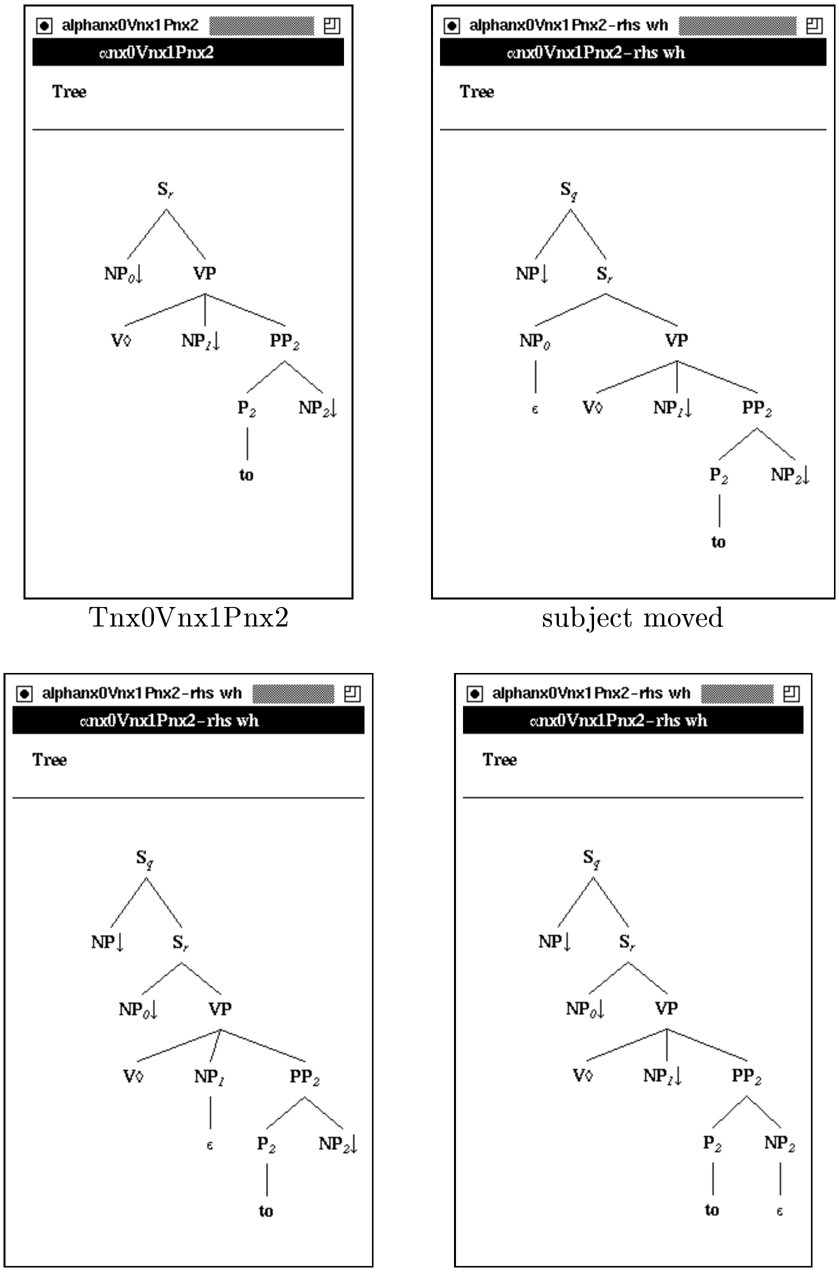
- For applying a metarule to a single input tree, click on the name of the tree in the main window, and choose the option *apply metarule to tree*. You will be prompted for the name of the metarule to apply to the tree which should be, as we mentioned before, the name of the buffer that contains the metarule trees. The output trees will be generated at the end of the buffer that contains the input tree. The names of the trees depend on a LISP parameter: **metarules-change-name** . If the value of the parameter is **false**, (the *default* value), then the new trees will have the same name as the input, otherwise, the name of the input tree followed by a dash ('-') and the name of the right hand side of the tree[12].

  The value of the parameter can be changed by choosing *Tools* at the menu bar and then

[10] Although a buffer is intended to implement the concept of a set (not a sequence) of trees we take advantage of the actual organization of the system to realize the concept of (ordered) tree pair in the implementation.

[11] So that even if we want to have mnemonic names resembling their distinct character - left or right hand side, - we have some flexibility in naming them, e.g. *lhs23* or *lhs-passive*.

[12] The reason we do not use the name of the metarule, that is, the name of the buffer, is because in some forms of application the metarules do not carry individual names, which, as we will see, is the case when a set of metarules from a file is applied.



Tnx0Vnx1Pnx2 | subject moved

NP object moved | NP object moved from PP

Figure 27.5: Application of wh-movement rule to Tnx0Vnx1Pnx2

either *name mr output trees = input* or *append rhs name to mr output trees.*

- For applying a metarule to all the trees of a buffer, click on the name of the buffer that contains the trees and proceed as above. The output will be a new buffer with all the output trees. The name of the new buffer will be the same as the input buffer prefixed by "MR-". The names of the trees follow the conventions above.

The other options concern application to files (instead of buffers). We will first define the concepts of parallel, sequential and cumulative application of metarules. One metarule file can contain more than one metarule. The first two trees, i.e., the first tree pair, form one metarule – call it mr_0. Subsequent pairs in the sequence of trees define additional metarules — mr_1, mr_2, ..., mr_n.

- We say that a metarule file is applied in parallel to a tree (see Figure 27.6) if each of the metarules is applied independently to the input generating its particular output trees[13]. We generalize the concept to the application in parallel of a metarule file to a tree file (with possibly more than one tree), generating all the trees as if each metarule in the metarule file was applied to each tree in the input file.

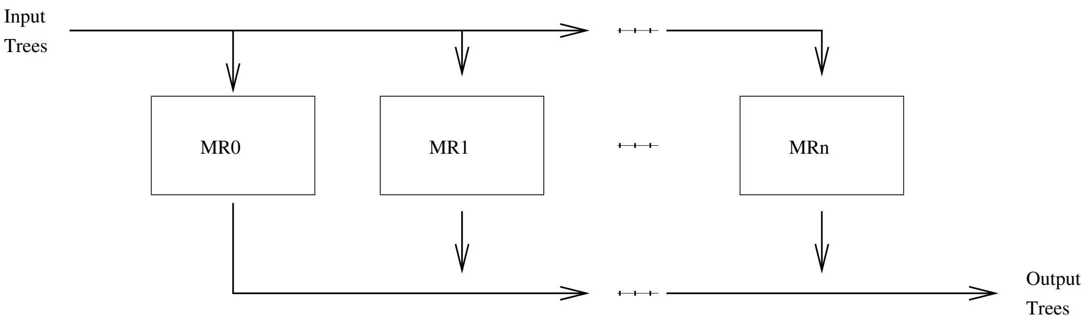


Figure 27.6: Parallel application of metarules

- We say that a metarule file $mr_0, mr_1, mr_2, ..., mr_n$ is applied in sequence to an input tree file (see Figure 27.7) if we apply mr_0 to the trees of the input file, and for each $0 < i \leq n$ apply metarule mr_i to the trees generated as a result of the application of mr_{i-1}.

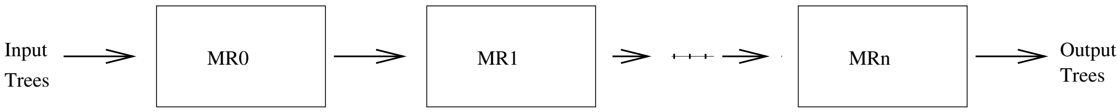


Figure 27.7: Sequential application of metarules

- Finally, the cumulative application is similar to the sequential, except that the input trees at each stage are passed to the output together with the newly generated ones (see Figure 27.8).

Remember that in case of matching failure the output result is decided (as explained in subsection 27.2.3) either to be empty or to be the input tree. The reflex here of having the parameter set for copying the input is that for the parallel application the output will have as many copies of the input as matching failures. For the sequential case the decision is applied at

[13]Remember that a metarule application generates as many output trees as the number of matches.

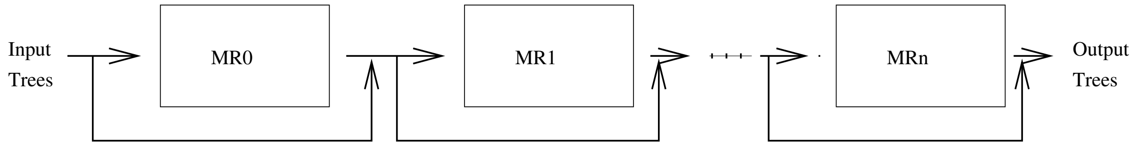


Figure 27.8: Cumulative application of metarules

each level, and setting the parameter for copying, in a certain sense, guarantees that subsequent metarule applications will not break. Due to its nature, and unlike the two other modes, the cumulative application is not affected by this parameter.

The options for application of metarules to files are available by clicking on the menu item *Tools* and then choosing the appropriate function from the following:

- *Apply metarule to files:* You will be prompted for the metarule file name which should contain one metarule[14], and for input file names. Each input file name **inpfile** will be independently submitted to the application of the metarule generating an output file with the name **MR-inpfile**.
- *Apply metarules in parallel to files:* You will be prompted for the metarules file name with one or more metarules and for input file names. Each input file name **inpfile** will be independently submitted to the application of the metarules in parallel. For each parallel application to a file **inpfile** an output file with the name **MRP-inpfile** will be generated.
- *Apply metarules in sequence to files:* The interaction is as described for the application in parallel, except that the application of the metarules are in sequence and that the output files are prefixed by **MRS-** instead of **MRP-**.
- *Apply metarules cumulatively to files:* The interaction is as described for the applications in parallel and in sequence, except that the mode of application is cumulative and that the output files are prefixed by **MRC-**.

The *Tools* menu also has entries to change the parameters of execution of metarules. We saw earlier in this section how to set the parameter that controls the name of the tree. Another parameter, explained in Subsection 27.2.3, is the one that controls the output result on matching failure. We can change it in the menu by selecting either *copy input on mr matching failure* or *no output on mr matching failure*. Recall that this parameter does not affect the cumulative mode. The third parameter controls comments generation. By choosing *append metarule comments* at the *Tools* menu, the subsequent metarule applications will produce output trees whose comments are the comments at the **lhs** tree of the metarule followed by the comments at **inp**. Both parts are introduced by appropriate headers, allowing the user to have a complete history of each tree. Choosing *do not append metarule comments* makes the comments at the output trees be exactly the same at the input tree. The third option, *clear comments on metarule application*, causes the comment field to be left empty at the ouput trees.

The default values for the tree parameters when XTAG is started are set: to generate the output trees with the same name as the input; not to copy the input tree in case of matching failure; and to add the metarule comments to the output tree.

[14]If the file contains more than 2 trees, the additional trees are ignored.

Chapter 28

Lexical Organization

28.1 Introduction

An important characteristic of an FB-LTAG is that it is lexicalized, i.e., each lexical item is anchored to a tree structure that encodes subcategorization information. Trees with the same canonical subcategorizations are grouped into tree families. The reuse of tree substructures, such as wh-movement, in many different trees creates redundancy, which poses a problem for grammar development and maintenance [Vijay-Shanker and Schabes, 1992]. To consistently implement a change in some general aspect of the design of the grammar, all the relevant trees currently must be inspected and edited. Vijay Shanker and Schabes suggested the use of hierarchical organization and of tree descriptions to specify substructures that would be present in several elementary trees of a grammar. Since then, in addition to ourselves, Becker, [Becker, 1994], Evans et al. [Evans *et al.*, 1995], and Candito[Candito, 1996] have developed systems for organizing trees of a TAG which could be used for developing and maintaining grammars.

Our system is based on the ideas expressed in Vijay-Shanker and Schabes, [Vijay-Shanker and Schabes, 1992], to use partial-tree descriptions in specifying a grammar by separately defining pieces of tree structures to encode independent syntactic principles. Various individual specifications are then combined to form the elementary trees of the grammar. The chapter begins with a description of our grammar development system, and its implementation. We will then show the main results of using this tool to generate the Penn English grammar as well as a Chinese TAG. We describe the significant properties of both grammars, pointing out the major differences between them, and the methods by which our system is informed about these language-specific properties. The chapter ends with the conclusion and future work.

28.2 System Overview

In our approach, three types of components – subcategorization frames, blocks and lexical redistribution rules – are used to describe lexical and syntactic information. Actual trees are generated automatically from these abstract descriptions, as shown in Figure 28.1. In maintaining the grammar only the abstract descriptions need ever be manipulated; the tree descriptions and the actual trees which they subsume are computed deterministically from these high-level descriptions.

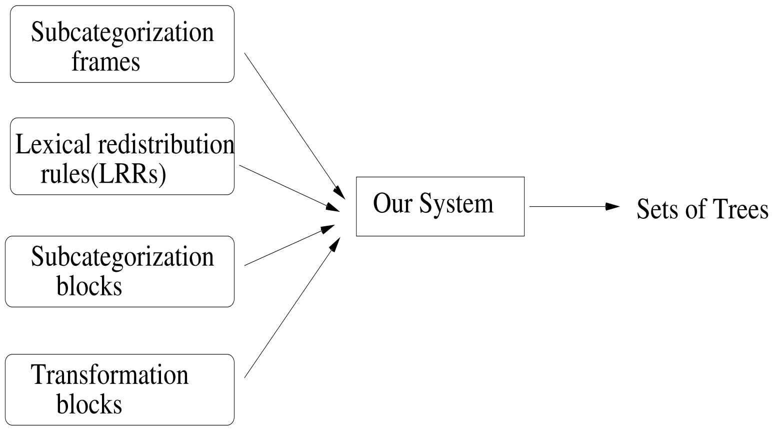


Figure 28.1: Lexical Organization: System Overview

28.2.1 Subcategorization frames

Subcategorization frames specify the category of the main anchor, the number of arguments, each argument's category and position with respect to the anchor, and other information such as feature equations or node expansions. Each tree family has one canonical subcategorization frame.

28.2.2 Blocks

Blocks are used to represent the tree substructures that are reused in different trees, i.e. blocks subsume classes of trees. Each block includes a set of nodes, dominance relation, parent relation, precedence relation between nodes, and feature equations. This follows the definition of the tree descriptions specified in a logical language patterned after Rogers and Vijay-Shanker[Rogers and Vijay-Shankar, 1994].

Blocks are divided into two types according to their functions: subcategorization blocks and transformation blocks. The former describes structural configurations incorporating the various information in a subcategorization frame. For example, some of the subcategorization blocks used in the development of the English grammar are shown in Figure 28.2.[1]

When the subcategorization frame for a verb is given by the grammar developer, the system will automatically create a new block (of code) by essentially selecting the appropriate primitive subcategorization blocks corresponding to the argument information specified in that verb frame.

The transformation blocks are used for various transformations such as wh-movement. These transformation blocks do not encode rules for modifying trees, but rather describe the properties of a particular syntactic construction. Figure 28.3 depicts our representation of phrasal extraction. This can be specialized to give the blocks for wh-movement, topicalization, relative clause formation, etc. For example, the wh-movement block is defined by further

[1]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 value specification. The parent, (immediate dominance), relationship is illustrated by a plain line and the dominance relationship by a dotted line. The arc between nodes shows the precedence order of the nodes are unspecified. The nodes' categories are enclosed in parentheses.

specifying that the ExtractionRoot is labeled S, the NewSite has a +wh feature, and so on.

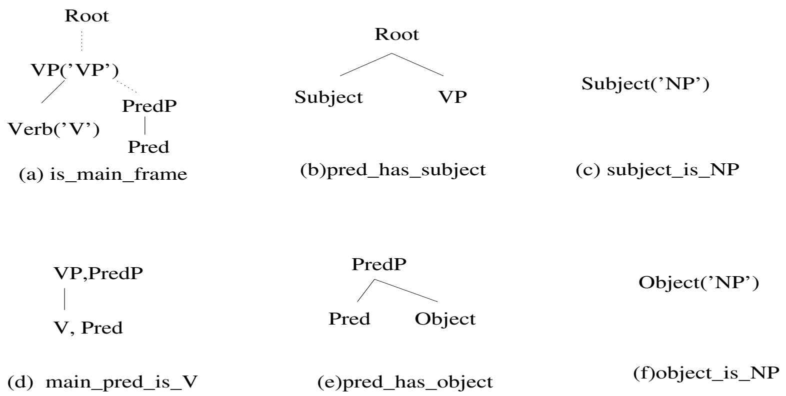


Figure 28.2: Some subcategorization blocks

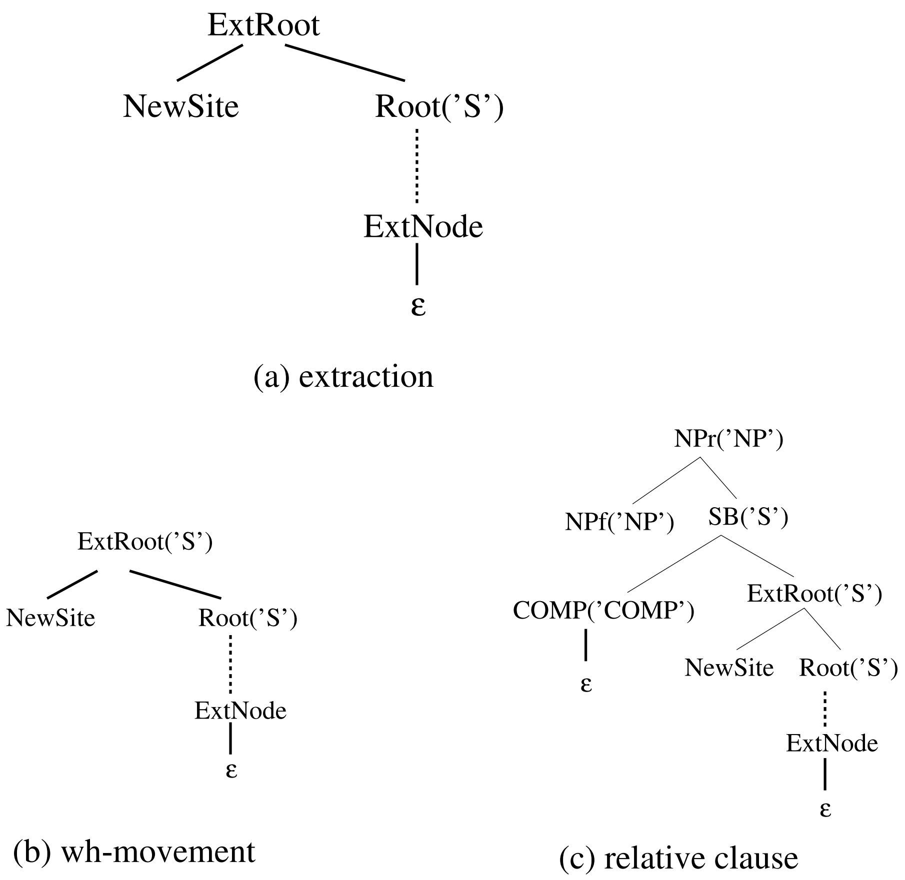


Figure 28.3: Transformation blocks for extraction

28.2.3 Lexical Redistribution Rules (LRRs)

The third type of machinery available for a grammar developer is the Lexical Redistribution Rule (LRR). An LRR is a pair (r_l, r_r) of subcategorization frames, which produces a new frame when applied to a subcategorization frame s, by first *matching*[2] the left frame r_l of r to s, then combining information in r_r and s. LRRs are introduced to incorporate the connection between subcategorization frames. For example, most transitive verbs have a frame for active(a subject and an object) and another frame for passive, where the object in the former frame becomes the subject in the latter. An LRR, denoted as passive LRR, is built to produce the passive subcategorization frame from the active one. Similarly, applying dative-shift LRR to the frame with one NP subject and two NP objects will produce a frame with an NP subject and an PP object.

Besides the distinct content, LRRs and blocks also differ in several aspects:

- They have different functionalities: Blocks represent the substructures that are reused in different trees. They are used to reduce the redundancy among trees; LRRs are introduced to incorporate the connections between the closely related subcategorization frames.
- Blocks are strictly additive and can be added in any order. LRRs, on the other hand, produce different results depending on the order they are applied in, and are allowed to be non-additive, i.e., to remove information from the subcategorization frame they are being applied to, as in the procedure of passive from active.

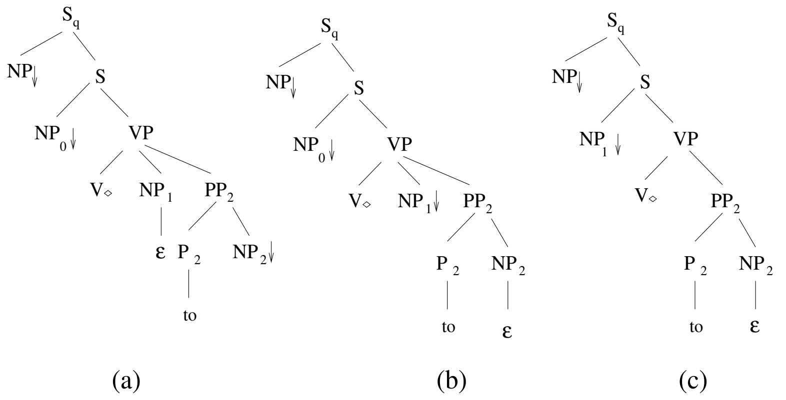


Figure 28.4: Elementary trees generated from combining blocks

28.2.4 Tree generation

To generate elementary trees, we begin with a canonical subcategorization frame. The system will first generate related subcategorization frames by applying LRRs, then select subcategorization blocks corresponding to the information in the subcategorization frames, next the combinations of these blocks are further combined with the blocks corresponding to various

[2]Matching occurs successfully when frame s is compatible with r_l in the type of anchors, the number of arguments, their positions, categories and features. In other words, incompatible features etc. will block certain LRRs from being applied.

transformations, finally, a set of trees are generated from those combined blocks, and they are the tree family for this subcategorization frame. Figure 28.4 shows some of the trees produced in this way. For instance, the last tree is obtained by incorporating information from the ditransitive verb subcategorization frame, applying the dative-shift and passive LRRs, and then combining them with the wh-non-subject extraction block. Besides, in our system the hierarchy for subcategorization frames is implicit as shown in Figure 28.5.

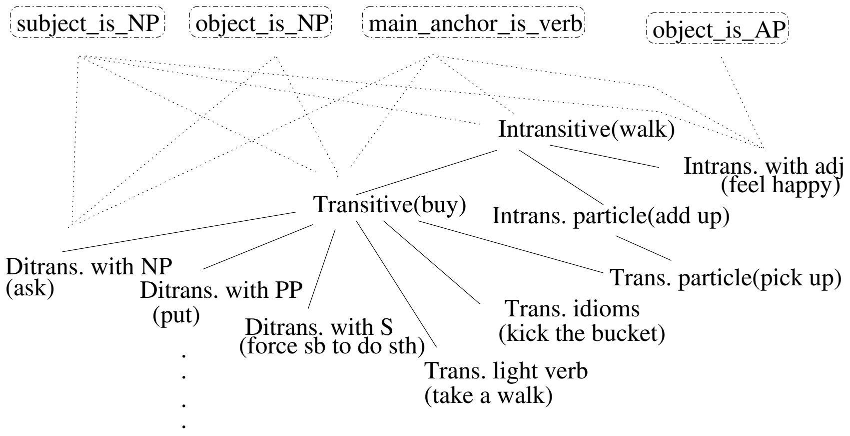


Figure 28.5: Partial inheritance lattice in English

28.3 Implementation

The input of our system is the description of the language, which includes the subcategorization frame list, LRR list, subcategorization block list and transformation lists. The output is a list of trees generated automatically by the system, as shown in Figure 28.6. The tree generation module is written in Prolog, and the rest part is in C. We also have a graphic interface to input the language description. Figure 28.7 and 28.8 are two snapshots of the interface.

28.4 Generating grammars

We have used our tool to specify a grammar for English in order to produce the trees used in the current English XTAG grammar. We have also used our tool to generate a large grammar for Chinese. In designing these grammars, we have tried to specify the grammars to reflect the similarities and the differences between the languages. The major features of our specification of these two grammars[3] are summarized in Table 28.1 and 28.2.

[3]Both grammars are still under development, so the contents of these two tables might change a lot in the future according to the analyses we choose for certain phenomenon. For example, the majority of work on Chinese grammar treat ba-construction as some kind of object-fronting where the character *ba* is either an object marker or a preposition. According to this analysis, an LRR rule for ba-construction is used in our grammar to generate the preverbal-object frame from the postverbal frame. However, there has been some argument for treating *ba* as a verb. If we later choose that analysis, the main verbs in the patterns "NP0 VP" and "NP0 ba NP1 VP" will be different, therefore no LRR will be needed for it. As a result, the numbers of LRRs, subcat frames and tree generated will change accordingly.

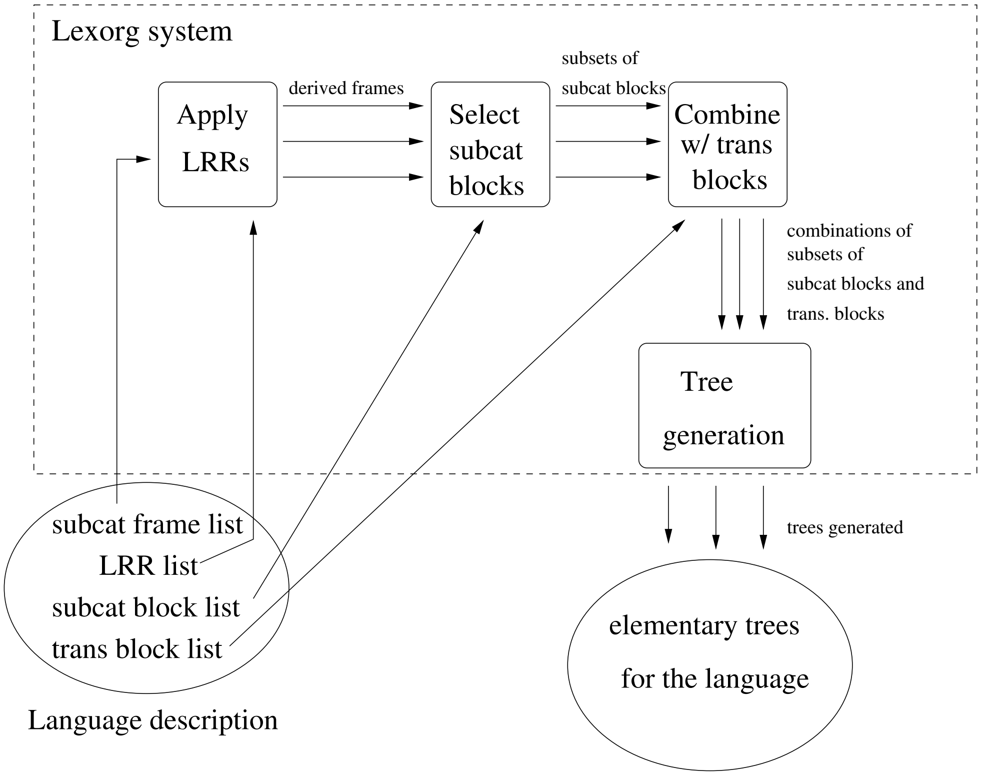


Figure 28.6: Implementation of the system

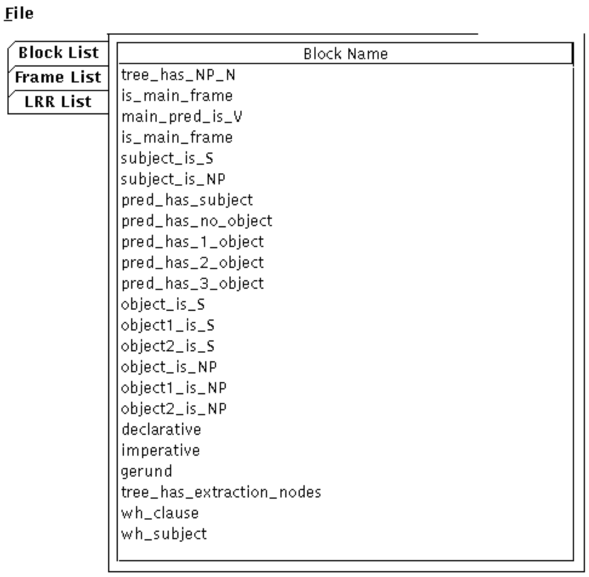


Figure 28.7: Interface for creating a grammar

By focusing on the specification of individual grammatical information, we have been able to generate nearly all of the trees from the tree families used in the current English grammar

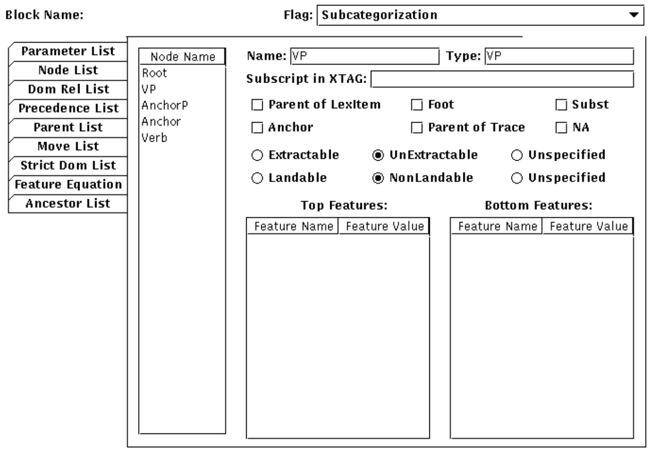


Figure 28.8: Part of the Interface for creating blocks

	English	Chinese
examples of LRRs	passive dative-shift ergative	bei-construction object fronting ba-construction
examples of transformation blocks	wh-question relativization declarative	topicalization relativization argument-drop
# LRRs	6	12
# subcat blocks	34	24
# trans blocks	8	15
# subcat frames	43	23
# trees generated	638	280

Table 28.1: Major features of English and Chinese grammars

developed at Penn[4]. Our approach, has also exposed certain gaps in the Penn grammar. We are encouraged with the utility of our tool and the ease with which this large-scale grammar was developed.

We are currently working on expanding the contents of subcategorization frame to include trees for other categories of words. For example, a frame which has no specifier and one NP complement and whose predicate is a preposition will correspond to PP $\rightarrow$ P NP tree. We'll also introduce a modifier field and semantic features, so that the head features will propagate

[4]We have not yet attempted to extend our coverage to include punctuation, it-clefts, and a few idiosyncratic analyses.

	both grammars	English	Chinese
LRRs	causative short passive	long passive ergative dative-shift	VO-inversion ba-const
trans blocks	topicalization relativization declarative	gerund	argument-drop
subcat blocks	NP/S subject S/NP/PP object V predicate	PL object prep predicate	zero-subject preverbal object

Table 28.2: Comparison of the two grammars

from modifiee to modified node, while non-head features from the predicate as the head of the modifier will be passed to the modified node.

28.5 Summary

We have described a tool for grammar development in which tree descriptions are used to provide an abstract specification of the linguistic phenomena relevant to a particular language. In grammar development and maintenance, only the abstract specifications need to be edited, and any changes or corrections will automatically be proliferated throughout the grammar. In addition to lightening the more tedious aspects of grammar maintenance, this approach also allows a unique perspective on the general characteristics of a language. Defining hierarchical 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 an overview of the theory that is being implemented and exposing gaps that remain unmotivated and need to be investigated. The type of gaps that can be exposed could include a missing subcategorization frame that might arise from the automatic combination of blocks and which would correspond to an entire tree family, a missing tree which would represent a particular type of transformation for a subcategorization frame, or inconsistent feature equations. 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.

Chapter 29

Tree Naming conventions

The various trees within the XTAG grammar are named more or less according to the following tree naming conventions. Although these naming conventions are generally followed, there are occasional trees that do not strictly follow these conventions.

29.1 Tree Nodes

In the description of a tree, nodes are named as a pair (l, s) (also represented as l_s in the graphic representation of its structure and l_s in the feature equations description), where l is the label or grammar symbol assigned to the node and s is a subscript whose primary purpose is to make a node name unique for any given tree. Typical examples are: S_r, VP, NP_0, NP_1. Notice that the subscript may be empty as in VP. There are several conventions generally followed for the use of subscripts, as naming S_r the node immediately dominating the subject position in a verb tree. We will not exhaustively describe them here, except a few, which are used for naming the trees. They should give the user an idea of what the tree is about.

Anchors are generally assigned a null subscript, unless the tree has more than on anchor with the same label, in which case each receive numeric subscripts 1, 2, etc. The main consistency condition here is that these subscripts have to match the multi-anchor entries in the syntactic lexicon that select the tree.

Arguments in verbal trees are assigned subscripts according to their thematic roles. The main idea is that the subscript for a certain argument should be preserved across the trees in the grammar, whenever it is seen that these trees are transformationally related (the arguments should be at the same position in the logical form). For instance, the subscript for the NP subject of a passive tree should be the same as for the NP in the corresponding declarative tree that has been passivized. Additionally, the general convention is that the subscript 0 is assigned to the underlying subject of a tree, 1 to the first object, and so on. Notable exceptions arise when a certain family is also used as a part of the subcategorization description of a class of verbs. For instance, dative verbs take two families: one having NP_1 and PP_2 as arguments (actually a multi-anchor tree where P is an anchor); the other being the double object family, for the dative shift. In order to maintain the relation that the leftmost object in the dative shift tree is logically related to the PP_2, it also receives the subscript 2. Another example is that subjects in the ergative family have the same subscript, 1, as the object in the base transitive tree.

When a verb argument is also the anchor of a tree, as in the predicative families, light verbs and some multi-anchor idioms, the projection of the argument as well as its own arguments (e.g. PP and NP for a P anchor,) should generally also take the subscript corresponding to its position w.r.t. to the verb, which will be generally diferent from the anchor that carries no subscript as a role. Trees for wh-moved objects have no subscript in the extracted site. The original subscript is used for the landing site. Relative clause trees on the other hand preserve the subscript in the original position and use the subscript w for the landing site.

29.2 Tree Families

Tree families are named according to the basic declarative tree structure in the tree family (see section 29.3), but with a T as the first character instead of an α or β.

29.3 Trees within tree families

Each tree begins with either an α (alpha) or a β (beta) symbol, indicating whether it is an initial or auxiliary tree, respectively. Following an α or a β the name may additionally contain one of the following depending on the family:

E	trees in the Ergative family
R	trees in a Resultative family
RE	trees in a Resultative family for Ergatives
X	ECM trees (eXceptional case marking)

Next, the name may contain one of the following, the digit corresponding to the subscript of the node moved in the tree. In Nc, Npx the absence of digit means relativized adjunct.

I	imperative
W0,W1,W2	wh-NP extraction
pW0,pW1,pW2	wh-PP extraction
N0,N1,N2	relative clause, NP argument relativized, wh-word
Nc,Nc0,Nc1,Nc2	relative clause, NP argument relativized, no wh-word
Npx,Npx1,Npx2	relative clause, PP relativized
Nby	relative clause, by-clause relativized in passive constructions
G	NP gerund
D	Determiner gerund
Inv	Inverted arguments (for equative BE and It-clefts)

The rest of the name consists of a string where each component correspond to one leaf of the tree from the left to right. The formation of a component is as follows: start with one of the elements in the table below that corresponds to the leaf being translated: in lower case if the node is a substitution or foot node; or upper case if it is an anchor. Then add "x" if the node is a projection (or "X" if an anchor and a projection). Finally add the subscript at the node if any. Notice that empty elements (ϵ) are generally ignored and their dominating node is used instead, except in the case of PRO, which by the way is capitalized.

s	sentence
a	adjective
arb	adverb
be	*be*
d	determiner
v	verb
lv	light verb
conj	conjunction
comp	complementizer
it	*it*
n	noun
p	preposition
PRO	a PRO subject
pl	particle
by	*by*
neg	negation

As an example, the transitive declarative tree consists of a subject NP, followed by a verb (which is the anchor), followed by the object NP. This translates into αnx0Vnx1. If the subject NP had been extracted, then the tree would be αW0nx0Vnx1. A passive tree with the *by* phrase in the same tree family would be αnx1Vbynx0. Note that even though the object NP has moved to the subject position, it retains the object encoding (nx1).

29.4 Assorted Initial Trees

Trees that are not part of the tree families are generally gathered into several files for convenience. The various initial trees are located in `lex.trees`. All the trees in this file should begin with an α, indicating that they are initial trees. This is followed by the root category which follows the naming conventions in the previous section (e.g. n for noun, x for phrasal category). The root category is in all capital letters. After the root category, the node leaves are named, beginning from the left, with the anchor of the tree also being capitalized. As an example, the αNXN tree is rooted by an NP node (NX) and anchored by a noun (N).

29.5 Assorted Auxiliary Trees

The auxiliary trees are mostly located in the buffers `prepositions.trees`, `conjunctions.trees`, `determiners.trees`, `advs-adjs.trees`, and `modifiers.trees`, although a couple of other files also contain auxiliary trees. The auxiliary trees follow a slightly different naming convention from the initial trees. Since the root and foot nodes must be the same for the auxiliary trees, the root nodes are not explicitly mentioned in the names of auxiliary trees. The trees are named according to the leaf nodes, starting from the left, and capitalizing the anchor node. All auxiliary trees begin with a β, of course. For example, βARBs, indicates a tree anchored by an adverb (ARB), that adjoins onto the left of an S node (Note that S must be the foot node, and therefore also the root node).

Chapter 30

Features

This appendix consists of short 'biographical' sketches of the various features currently in use in the XTAG English grammar. Table 30.1 contains a comprehensive list of the features in the XTAG grammar and their possible values.

30.1 Agreement

⟨**agr**⟩ is a complex feature. It can have as its subfeatures:
⟨**agr 3rdsing**⟩, possible values: $+/-$
⟨**agr num**⟩, possible values: $plur, sing$
⟨**agr pers**⟩, possible values: $1, 2, 3$
⟨**agr gen**⟩, possible values: $masc, fem, neut$

These features are used to ensure agreement between a verb and its subject.

Where does it occur:
Nouns comes specified from the lexicon with their ⟨**agr**⟩ features. e.g. *books* is ⟨**agr 3rdsing**⟩**:** $-$, ⟨**agr num**⟩**: plur**, and ⟨**agr pers**⟩**: 3**. Only pronouns use the <**gen**> (gender) feature.

The ⟨**agr**⟩ features of a noun are transmitted up the NP tree by the following equation:
NP.b:⟨**agr**⟩ = **N.t:**⟨**agr**⟩

Agreement between a verb and its subject is mediated by the following feature equations:

(629) $\mathbf{NP}_{subj}$**:**⟨**agr**⟩ = **VP.t:**⟨**agr**⟩

(630) **VP.b:**⟨**agr**⟩ = **V.t:**⟨**agr**⟩

Agreement has to be done as a two step process because whether the verb agrees with the subject or not depends upon whether some auxiliary verb adjoins in and upon what the ⟨**agr**⟩ specification of the verb is.

Verbs also come specified from the lexicon with their ⟨**agr**⟩ features, e.g. the ⟨**agr**⟩ features of the verb *sings* are ⟨**agr 3rdsing**⟩**:** $+$, ⟨**agr num**⟩**: sing**, and ⟨**agr pers**⟩**: 3**; Non-finite forms of the verb *sing* e.g. *singing* do not come with an ⟨**agr**⟩ feature specification.

30.1.1 Agreement and Movement

The ⟨**agr**⟩ features of a moved NP and its trace are co-indexed. This captures the fact that movement does not disrupt a pre-existing agreement relationship between an NP and a verb.

Feature	Value
<agr 3rdsing>	+, −
<agr num>	plur,sing
<agr pers>	1,2,3
<agr gen>	fem,masc,neuter
<assign-case>	nom,acc,none
<assign-comp>	that,whether,if,for,ecm,inf_nil,ind_nil,ppart_nil,none
<card>	+, −
<case>	nom,acc,gen,none
<comp>	that,whether,if,for,rel,inf_nil,ind_nil,nil
<compar>	+, −
<compl>	+, −
<conditional>	+, −
<conj>	and,or,but,comma,scolon,to,disc,nil
<const>	+, −
<contr>	+, −
<control>	no value, indexing only
<decreas>	+, −
<definite>	+, −
<equiv>	+, −
<extracted>	+, −
<gen>	+, −
<gerund>	+, −
<inv>	+, −
<invlink>	no value, indexing only
<irrealis>	+, −
<mainv>	+, −
<mode>	base,ger,ind,inf,imp,nom,ppart,prep,sbjunt
<neg>	+, −
<nocomp-mode>	inf,ger,ppart
<passive>	+, −
<perfect>	+, −
<pred>	+, −
<progressive>	+, −
<pron>	+, −
<punct bal>	dquote,squote,paren,nil
<punct contains colon>	+, −
<punct contains dash>	+, −
<punct contains dquote>	+, −
<punct contains scolon>	+, −
<punct contains squote>	+, −
<punct struct>	comma,dash,colon,scolon,nil
<punct term>	per,qmark,excl,nil
<quan>	+, −
<refl>	+, −
<rel-clause>	+, −
<super>	+, −
<tense>	pres,past
<trace>	no value, indexing only
<weak>	+, −
<wh>	+, −

Table 30.1: List of features and their possible values

(631) [Which boys]$_i$ does John think [t$_i$ are/*is intelligent]?

30.2 Case

30.2.1 Approaches to Case

30.2.1.1 Case in GB theory

GB (Government and Binding) theory proposes the following 'case filter' as a requirement on S-structure.[1]

CASE FILTER: Every overt NP must be assigned abstract case. [Haegeman, 1991]

Abstract case is taken to be universal. Languages with rich morphological case marking, such as Latin, and languages with very limited morphological case marking, like English, are all presumed to have full systems of abstract case that differ only in the extent of morphological realization.

In GB, abstract case is argued to be assigned to NP's by various case assigners, namely verbs, prepositions, and INFL. Verbs and prepositions are said to assign accusative case to NP's that they govern, and INFL assigns nominative case to NP's that it governs. These governing categories are constrained as to where they can assign case by means of 'barriers' based on 'minimality conditions', although these are relaxed in 'exceptional case marking' situations. The details of the GB analysis are beyond the scope of this technical report, but see [Chomsky, 1986] for the original analysis or [Haegeman, 1991] for an overview. Let it suffice for us to say that the notion of abstract case and the case filter are useful in accounting for a number of phenomena including the distribution of nominative and accusative case, and the distribution of overt NP's and empty categories (such as PRO).

30.2.1.2 Minimalism and Case

A major conceptual difference between GB theories and Minimalism is that in Minimalism, lexical items carry their features with them rather than being assigned their features in the course of the derivation. For example, nouns have a case feature when it comes into the derivation. This feature is 'checked' in a particular configuration (in SPEC position of AGR$_s$ or AGR$_o$) and then deleted. If the feature is not checked (deleted), the derivation fails.[Chomsky, 1992]

30.2.2 Case in XTAG

The English XTAG grammar adopts the notion of case and the case filter for many of the same reasons argued in the GB literature. However, in some respects the English XTAG grammar's implementation of case more closely resembles the treatment in Chomsky's Minimalism framework [Chomsky, 1992] than the system outlined in the GB literature [Chomsky, 1986]. As in Minimalism, nouns in the XTAG grammar carry case with them, which is eventually 'checked'.

[1] There are certain problems with applying the case filter as a requirement at the level of S-structure. These issues are not crucial to the discussion of the English XTAG implementation of case and so will not be discussed here. Interested readers are referred to [Lasnik and Uriagereka, 1988].

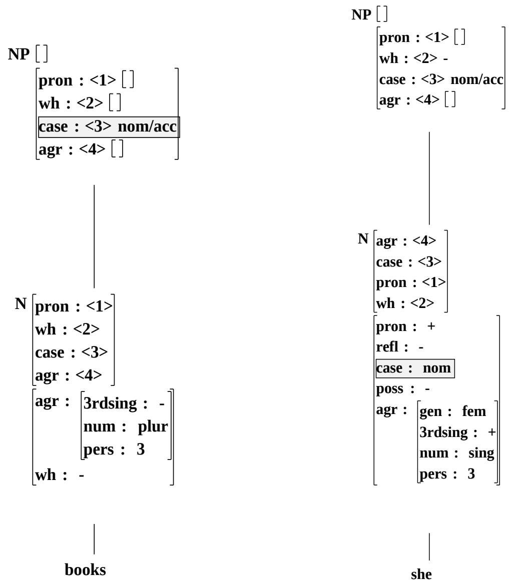
However, in the XTAG grammar, noun cases are checked against the case values assigned by the verb during the unification of the feature structures. Unlike Chomsky's Minimalism, there are no separate AGR nodes; the case checking comes from the verbs directly. Case assignment from the verb is more like the GB approach than the requirement of a SPEC-head relationship in Minimalism.

There are two features responsible for case-assignment:
⟨**case**⟩, possible values: **nom, acc, gen, none**
⟨**assign-case**⟩, possible values: **nom, acc, none**

Case assigners (prepositions and verbs) as well as the VP, S and PP nodes that dominate them have an ⟨**assign-case**⟩ case feature. Phrases and lexical items that have case i.e. Ns and NPs have a ⟨**case**⟩ feature.

30.2.2.1 Lexical Noun Phrases

Most nouns in English do not have separate forms for nominative and accusative case, and so they are ambiguous between the two. Pronouns, of course, are morphologically marked for case, and each carries the appropriate case in its feature. Figures 30.1(a) and 30.1(b) show the NP tree anchored by a noun and a pronoun, respectively, along with the feature values associated with each word. Note that *books* simply gets the default case **nom/acc**, while *she* restricts the case to be **nom**.



(a) (b)

Figure 30.1: Lexicalized NP trees with case markings

30.2.2.2 Case Assigners

Case is assigned in the XTAG English grammar by two lexical categories - verbs and prepositions.[2]

- **Prepositions**

Prepositions assign accusative case (**acc**) through their <**assign-case**> feature, which is specified in the lexicon.

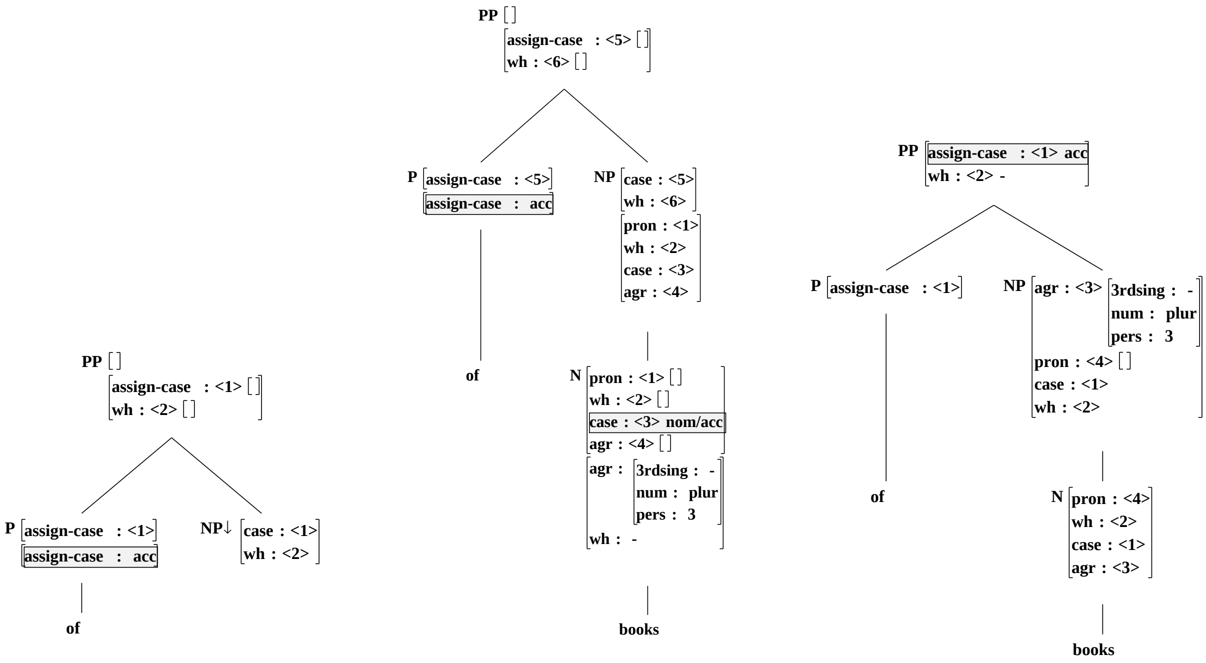
(632) **P.b:**$\langle$**assign-case**$\rangle$ = **acc**

This feature is linked directly to the <**case**> feature of their objects, which is accomplished with the following feature equations:

(633) **PP.b:**$\langle$**assign-case**$\rangle$ = **P.t:**$\langle$**assign-case**$\rangle$

(634) **NP.t:**$\langle$**case**$\rangle$ = **P.t:**$\langle$**assign-case**$\rangle$

Figure 30.2(a) shows a lexicalized preposition tree, while Figure 30.2(b) shows the same tree with the NP tree from Figure 30.1(a) substituted into the NP position. Figure 30.2(c) is the tree in Figure 30.2(b) after unification has taken place. Note that the case ambiguity of *books* has been resolved to accusative case.



(a) (b) (c)

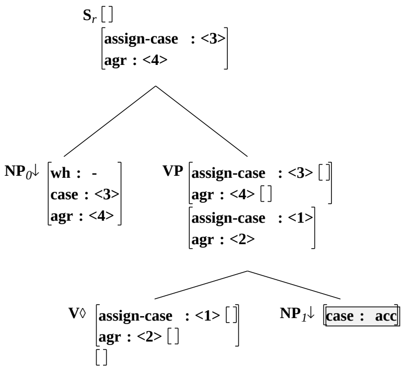
Figure 30.2: Assigning case in prepositional phrases

[2] *For* also assigns case as a complementizer. See section 8.5 for more details.

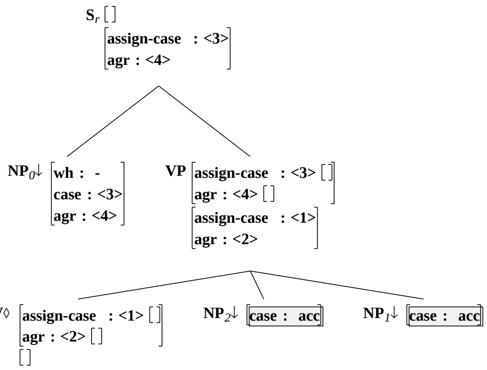
- **Verbs**

Verbs are the other part of speech in the XTAG grammar that can assign case. Because XTAG does not distinguish INFL and VP nodes, verbs must provide case assignment on the subject position in addition to the case assigned to their NP complements.

Assigning case to NP complements is handled by building the case values of the complements directly into the tree that the case assigner (the verb) anchors. Figures 30.3(a) and 30.3(b) show an S tree[3] that would be anchored[4] by a transitive and ditransitive verb, respectively. Note that the case assignments for the NP complements are already in the tree, even though there is not yet a lexical item anchoring the tree. Since every verb that selects these trees (and other trees in each respective subcategorization frame) assigns the same case to the complements, building case features into the tree has exactly the same result as putting the case feature value in each verb's lexical entry. 635 shows the equation used for specifying accusative case on the object NP (where *object* is replaced by an underscore and the number of the argument – either 1 or 2 – which is getting accusative case).



(a)



(b)

Figure 30.3: Case assignment to NP arguments

(635) $\mathbf{NP}_{object}\mathbf{.t{:}}\langle\mathbf{case}\rangle = \mathbf{acc}$

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 trees in Figure 30.4 show the path of linkages that joins the **<assign-case>** feature of the V to the **<case>** feature of the subject NP. The morphological form of the verb determines the value of the **<assign-case>** feature. Figures 30.4(a) and 30.4(b) show the same tree[5] anchored by different morphological forms of the verb *sing*, which give different values for the **<assign-case>** feature. Tensed verbs assign case **nom**, whereas

[3]Features not pertaining to this discussion have been taken out to improve readability and to make the trees easier to fit onto the page.

[4]The diamond marker (◊) indicates the anchor(s) of a structure if the tree has not yet been lexicalized.

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

non-tensed verbs assign no case at all; that is, a verb such as *singing* has no <**assign-case**> feature.

Assignment of case to the subject involves the following two equations.

(636) $\mathbf{NP}_{subj}$**:⟨case⟩ = VP.t:⟨assign-case⟩**

(637) **VP.b:⟨assign-case⟩ = V.t:⟨assign-case⟩**

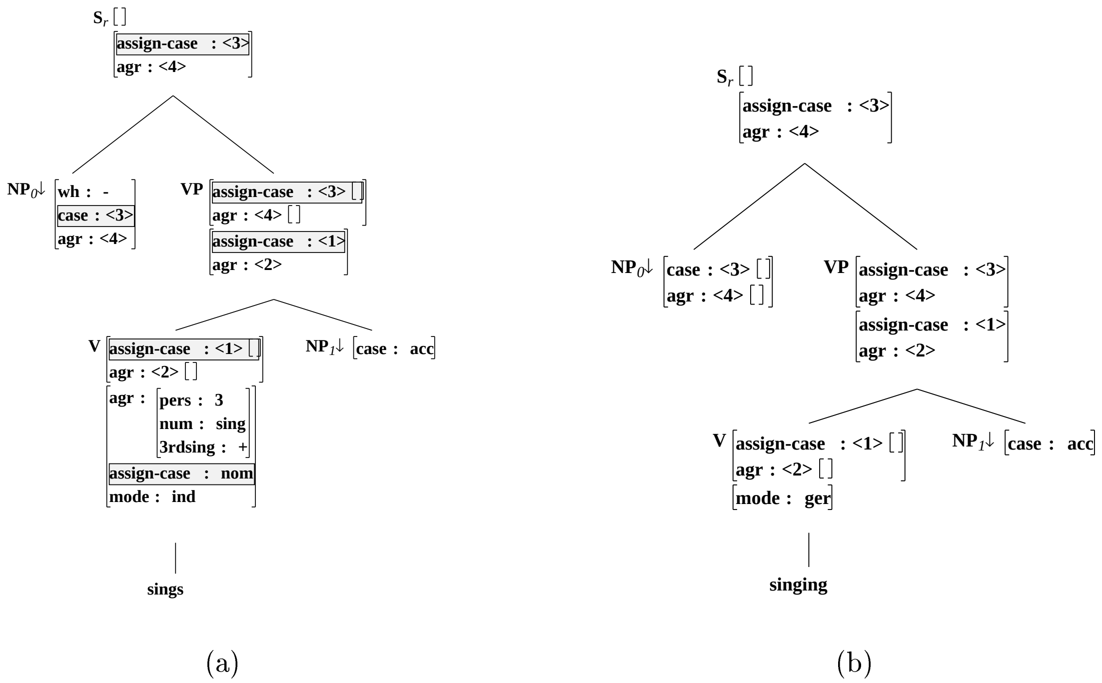


Figure 30.4: Assigning case according to verb form

In the case of a tenseless verb, a tensed auxiliary will adjoin in and assign case to the subject, or the subject will be assigned accusative from a higher ECM verb.[6] Figure 30.5(a) shows the auxiliary tree for *is*, and Figure 30.5(b) shows the result of adjoining *is* into the transitive tree anchored by *singing*. The case feature on the subject NP has the value <**nom**>, which comes from the auxiliary verb *is*.

30.2.2.3 PRO

In GB theory, the null NP element PRO can only appear in positions where it does not receive case (and in fact, where it is ungoverned). PRO only appears in subject position of tenseless (non-matrix) clauses, where the clause is either an adjunct, as in 638, or a CP infinitival complement clause, as in 639. Since only tensed verbs can assign nominative case, a PRO in the subject position of a tenseless clause will not receive nominative case. PRO also cannot

[6]Currently, there is nothing in the grammar which acts as a Case Filter to rule out sentences with nouns that are not receiving case. Thus, sentences such as **I hope Carlos to be happy* are accepted, even though the NP *Carlos* does not receive case. We are exploring ways of handling this for future releases.

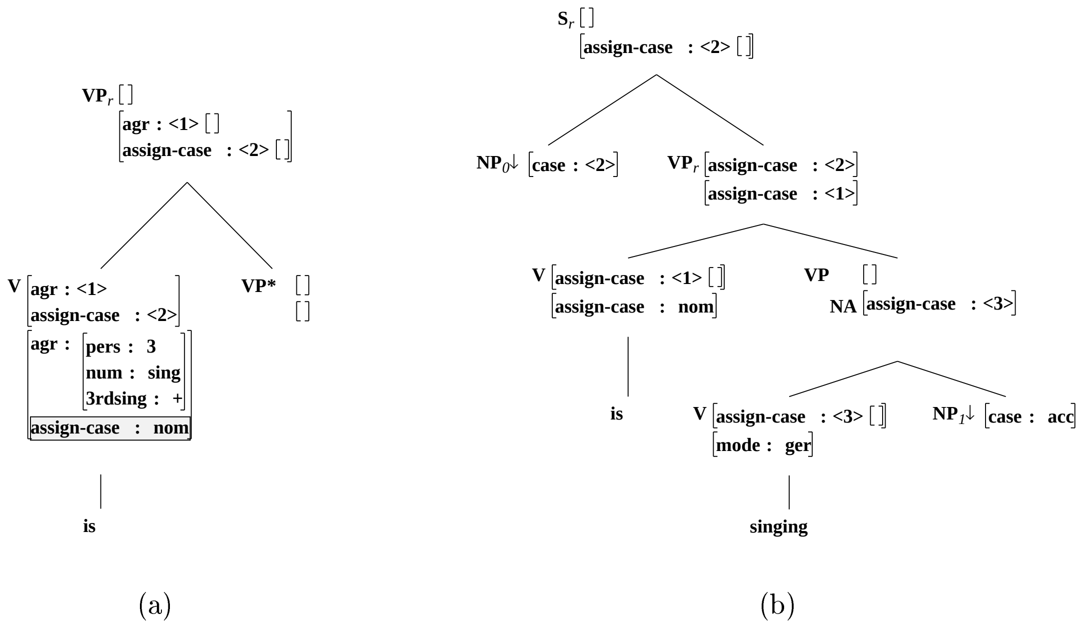


Figure 30.5: Proper case assignment with auxiliary verbs

appear in the complement of an ECM verb, since ECM verbs assign accusative case to the subject of their complement clause.

(638) While PRO talking to Anoop, Carlos drank a coke.

(639) Tonia wants PRO to eat lunch.

In previous versions of the grammar, the distribution of PRO was controlled by case features, similar to its treatment GB theory. PRO anchored an NP initial tree and had the case feature **none**. This tree could substitute into NP positions, but only unify in a position where either case **none** or no case at all was assigned, but not in positions with case **nom** or **acc**.

Currently, however, we have dispensed with PRO as an anchoring element, and tenseless verbs no longer have the feature **<assign-case>=none**. Rather than substituting in, PRO is built into the subject position of the relevant sentential (and gerund) trees.[7] [8]

The distribution of PRO is still controlled by case features to some degree, however. The trees with PRO built-in have an **<assign-case>=none** feature in the root S node (rather than being assigned by the verb). By having this feature, we can make sure that the PRO tree cannot appear as the complement of an ECM verb, since it will clash with the ECM verb's **<assign-case>=acc** feature. The equations in the PRO trees are given in 640 and 641.

(640) $\mathbf{NP}_{subj}$**:<case> = none**

(641) $\mathbf{S}_r$**.b:<case> =** $\mathbf{NP}_{subj}$**:<case>**

30.2.2.4 ECM

Certain verbs, such as *want, believe, consider*, and one complementizer (*for*) are able to assign case to the subject of their complement clause.

The complementizer *for*, like the preposition *for*, has the ⟨**assign-case**⟩ feature of its complement set to **acc**. Since the ⟨**assign-case**⟩ feature of the root S_r of the complement tree and the ⟨**case**⟩ feature of its NP subject are co-indexed, this leads to the subject being assigned accusative case.

ECM verbs have the ⟨**assign-case**⟩ feature of their foot S node set to **acc**. The co-indexation between the ⟨**assign-case**⟩ feature of the root S_r and the ⟨**case**⟩ feature of the NP subject leads to the subject being assigned accusative case.

30.2.2.5 The Case of Extracted NPs

The ⟨**case**⟩ features of a moved NP and its trace are co-indexed. This captures the fact that movement does not disrupt a pre-existing relationship of case-assignment between a verb and an NP.

(642) Her_i/*She_i, I think that Odo likes t_i.

[7] In this, we follow the treatment of PRO in the French XTAG project [Abeillé *et al.*, 2000].

[8] Note that, as of the Spring 2001 release, not all the PRO trees had been included in the grammar. Thus, some necessary trees are missing.

30.3 Extraction and Inversion

30.3.1 Extraction

$\langle$**extracted**$\rangle$, possible vales are $+/-$

All sentential trees with extracted components, with the exception of relative clauses are marked **S.b**$\langle$**extracted**$\rangle = +$ at their top S node. The extracted element may be a *wh*-NP or a topicalized NP. The $\langle$**extracted**$\rangle$ feature is currently used to block embedded topicalizations as exemplified by the following example.

(643) * John wants [Bill$_i$ [PRO to leave t$_i$]]

$\langle$**trace**$\rangle$: this feature is not assigned any value and is used to co-index moved NPs and their traces which are marked by ϵ.
$\langle$**wh**$\rangle$: possible values are $+/-$
NPs like *who*, *what* etc. come marked from the lexicon with a value of $+$ for the feature $\langle$**wh**$\rangle$. Non *wh*-NPs have $-$ as the value of their $\langle$**wh**$\rangle$ feature. Note that $\langle$**wh**$\rangle = +$ NPs are not restricted to occurring in extracted positions, to allow for the correct treatment of echo questions.

The $\langle$**wh**$\rangle$ feature is propagated up by possessives – e.g. the $+$ $\langle$**wh**$\rangle$ feature of the determiner *which* in *which boy* is propagated up to the level of the NP so that the value of the $\langle$**wh**$\rangle$ feature of the entire NP is $+\langle$**wh**$\rangle$. This process is recursive e.g. *which boy's mother*, *which boy's mother's sister*.

The $\langle$**wh**$\rangle$ feature is also propagated up PPs. Thus the PP *to whom* has $+$ as the value of its $\langle$**wh**$\rangle$ feature.

In trees with extracted NPs, the $\langle$**wh**$\rangle$ feature of the root node S node is equated with the $\langle$**wh**$\rangle$ feature of the extracted NPs.

The $\langle$**wh**$\rangle$ feature is used to impose subcategorizational constraints. Certain verbs like *wonder* can only take interrogative complements, other verbs such as *know* can take both interrogative and non-interrogative complements, and yet other verbs like *think* can only take non-interrogative complements (cf. the $\langle$**extracted**$\rangle$ and $\langle$**mode**$\rangle$ features also play a role in imposing subcategorizational constraints).

The $\langle$**wh**$\rangle$ feature is also used to get the correct inversion patterns.

30.3.2 Inversion

The following three features are used to ensure the correct pattern of inversion:
$\langle$**wh**$\rangle$: possible values are $+/-$
$\langle$**inv**$\rangle$: possible values are $+/-$
$\langle$**invlink**$\rangle$: possible values are $+/-$

Facts to be captured:

1. No inversion with topicalization
2. No inversion with matrix extracted subject *wh*-questions
3. Inversion with matrix extracted object *wh*-questions
4. Inversion with all matrix *wh*-questions involving extraction from an embedded clause
5. No inversion in embedded questions
6. No matrix subject topicalizations.

Consider a tree with object extraction, where NP is extracted. The following feature equations are used:

(644) $\mathbf{S}_q$**.b:**$\langle$**wh**$\rangle$ = **NP.t:**$\langle$**wh**$\rangle$

(645) $\mathbf{S}_q$**.b:**$\langle$**invlink**$\rangle$ = $\mathbf{S}_q$**.b:**$\langle$**inv**$\rangle$

(646) $\mathbf{S}_q$**.b:**$\langle$**inv**$\rangle$ = $\mathbf{S}_r$**.t:**$\langle$**inv**$\rangle$

(647) $\mathbf{S}_r$**.b:**$\langle$**inv**$\rangle$ = $-$

Root restriction: A restriction is imposed on the final root node of any XTAG derivation of a tensed sentence which equates the $\langle$**wh**$\rangle$ feature and the $\langle$**invlink**$\rangle$ feature of the final root node.

If the extracted NP is not a *wh*-word i.e. its $\langle$**wh**$\rangle$ feature has the value $-$, at the end of the derivation, $\mathbf{S}_q$**.b:**$\langle$**wh**$\rangle$ will also have the value $-$. Because of the root constraint $\mathbf{S}_q$**.b:**$\langle$**wh**$\rangle$ will be equated to $\mathbf{S}_q$**.b:**$\langle$**invlink**$\rangle$ which will also come to have the value $-$. Then, by (646), $\mathbf{S}_r$**.t:**$\langle$**inv**$\rangle$ will acquire the value $-$. This will unify with $\mathbf{S}_r$**.b:**$\langle$**inv**$\rangle$ which has the value $-$ (cf. 647). Consequently, no auxiliary verb adjunction will be forced. Hence, there will never be inversion in topicalization.

If the extracted NP is a *wh*-word i.e. its $\langle$**wh**$\rangle$ feature has the value $+$, at the end of the derivation, $\mathbf{S}_q$**.b:**$\langle$**wh**$\rangle$ will also have the value $+$. Because of the root constraint $\mathbf{S}_q$**.b:**$\langle$**wh**$\rangle$ will be equated to $\mathbf{S}_q$**.b:**$\langle$**invlink**$\rangle$ which will also come to have the value $+$. Then, by (646), $\mathbf{S}_r$**.t:**$\langle$**inv**$\rangle$ will acquire the value $+$. This will not unify with $\mathbf{S}_r$**.b:**$\langle$**inv**$\rangle$ which has the value $+$ (cf. 647). Consequently, the adjunction of an inverted auxiliary verb is required for the derivation to succeed.

Inversion will still take place even if the extraction is from an embedded clause.

(648) Who$_i$ does Loida think [Miguel likes t$_i$]

This is because the adjoined tree's root node will also have its $\mathbf{S}_r$**.b:**$\langle$**inv**$\rangle$ set to $-$.

Note that inversion is only forced upon us because S_q is the final root node and the **Root restriction** applies. In embedded environments, the root restriction would not apply and the feature clash that forces adjunction would not take place.

The $\langle$**invlink**$\rangle$ feature is not present in subject extractions. Consequently there is no inversion in subject questions.

Subject topicalizations are blocked by setting the $\langle$**wh**$\rangle$ feature of the extracted NP to $+$ i.e. only *wh*-phrases can go in this location.

30.4 Multi-component Adjoining

$\langle$**displ-const**$\rangle$:
Possible values: **[set1: +]**, **[set1: $-$]**
This feature can be used to simulate multi-component adjoining[9] and was previously used this

[9] The $\langle$**displ-const**$\rangle$ feature is also used in the ECM analysis.

way in the analysis of inversion. In the previous analysis, an empty verb trace tree adjoined in at VP whenever an auxiliary verb tree adjoined at S. However, we have dispensed with the empty verb tree and thus do not require a multi-component analysis to account for inversion.

We leave a description of the feature here for possible future analyses of phenomena which might require a multi-component analysis, such as extraposition, for example.

The ⟨**displ-const**⟩ feature is used to ensure adjunction of two trees. The following equations are used:

(649) $\mathbf{S_r.b{:}\langle displ\text{-}const\ set1\rangle} = -$

(650) $\mathbf{S.t{:}\langle displ\text{-}const\ set1\rangle} = +$

(651) $\mathbf{VP.b{:}\langle displ\text{-}const\ set1\rangle} = \mathbf{V.t{:}\langle displ\text{-}const\ set1\rangle}$

(652) $\mathbf{V.b{:}\langle displ\text{-}const\ set1\rangle} = +$

(653) $\mathbf{S_r.b{:}\langle displ\text{-}const\ set1\rangle} = \mathbf{VP.t{:}\langle displ\text{-}const\ set1\rangle}$

30.5 Clause Type

There are several features that mark clause type.[10] They are:
⟨**mode**⟩
⟨**passive**⟩: possible values are +/−
⟨**mode**⟩: possible values are **base, ger, ind, inf, imp, nom, ppart, prep, sbjnct**
The ⟨**mode**⟩ feature of a verb in its root form is **base**. The ⟨**mode**⟩ feature of a verb in its past participial form is **ppart**, the ⟨**mode**⟩ feature of a verb in its progressive/gerundive form is **ger**, the ⟨**mode**⟩ feature of a tensed verb is **ind**, and the ⟨**mode**⟩ feature of a verb in the imperative is **imp**.
nom is the ⟨**mode**⟩ value of AP/NP predicative trees headed by a null copula. **prep** is the ⟨**mode**⟩ value of PP predicative trees headed by a null copula. Only the copula auxiliary tree, some sentential complement verbs (such as *consider* and raising verb auxiliary trees have **nom/prep** as the ⟨**mode**⟩ feature specification of their foot node. This allows them, and only them, to adjoin onto AP/NP/PP predicative trees with null copulas.

30.5.1 Auxiliary Selection

The ⟨**mode**⟩ feature is also used to state the subcategorizational constraints between an auxiliary verb and its complement. We model the following constraints:
have takes past participial complements
passive *be* takes past participial complements
active *be* takes progressive complements
modal verbs, *do*, and *to* take VPs headed by verbs in their base form as their complements.

An auxiliary verb transmits its own mode to its root and imposes its subcategorizational restrictions on its complement i.e. on its foot node. e.g. the auxiliary *have* in its infinitival form involves the following equations:

[10] We have already seen one instance of a feature that marks clause-type: ⟨**extracted**⟩, which marks whether a certain S involves extraction or not.

(654) $\mathbf{VP}_r\mathbf{.b{:}\langle mode\rangle = V.t{:}\langle mode\rangle}$

(655) $\mathbf{V.t{:}\langle mode\rangle = base}$

(656) $\mathbf{VP.b{:}\langle mode\rangle = ppart}$

⟨**passive**⟩: This feature is used to ensure that passives only have *be* as their auxiliary. Passive trees start out with their ⟨**passive**⟩ feature as +. This feature starts out at the level of the verb and is percolated up to the level of the VP. This ensures that only auxiliary verbs whose foot node has + as their ⟨**passive**⟩ feature can adjoin on a passive. Passive trees have **ppart** as the value of their ⟨**mode**⟩ feature. So the only auxiliary trees that we really have to worry about blocking are trees whose foot nodes have **ppart** as the value of their ⟨**mode**⟩ feature. There are two such trees – the *be* tree and the *have* tree. The *be* tree is fine because its foot node has + as its ⟨**passive**⟩ feature, so both the ⟨**passive**⟩ and ⟨**mode**⟩ values unify; the *have* tree is blocked because its foot node has – as its ⟨**passive**⟩ feature.

30.6 Complementizer Selection

The following features are used to ensure the appropriate distribution of complementizers:
⟨**comp**⟩, possible values: **that, if, whether, for, inf_nil, ind_nil, nil**
⟨**assign-comp**⟩, possible values: **that, if, whether, for, ecm, ind_nil, inf_nil, none**
⟨**mode**⟩, possible values: **ind, inf, sbjnct, ger, base, ppart, nom, prep**
⟨**wh**⟩, possible values: **+, –**

The value of the ⟨**comp**⟩ feature tells us what complementizer we are dealing with. The trees which introduce complementizers come specified from the lexicon with their ⟨**comp**⟩ feature and ⟨**assign-comp**⟩ feature. The ⟨**comp**⟩ of the Comp tree regulates what kind of tree goes above the Comp tree, while the ⟨**assign-comp**⟩ feature regulates what kind of tree goes below. e.g. the following equations are used for *that*:

(657) $\mathbf{S}_c\mathbf{.b{:}\langle comp\rangle = Comp.t{:}\langle comp\rangle}$

(658) $\mathbf{S}_c\mathbf{.b{:}\langle wh\rangle = Comp.t{:}\langle wh\rangle}$

(659) $\mathbf{S}_c\mathbf{.b{:}\langle mode\rangle = ind/sbjnct}$

(660) $\mathbf{S}_r\mathbf{.t{:}\langle assign\text{-}comp\rangle = Comp.t{:}\langle comp\rangle}$

(661) $\mathbf{S}_r\mathbf{.b{:}\langle comp\rangle = nil}$

By specifying $\mathbf{S}_r\mathbf{.b{:}\langle comp\rangle = nil}$, we ensure that complementizers do not adjoin onto other complementizers. The root node of a complementizer tree always has its ⟨**comp**⟩ feature set to a value other than **nil**.

Trees that take clausal complements specify with the ⟨**comp**⟩ feature on their foot node what kind of complementizer(s) they can take. The ⟨**assign-comp**⟩ feature of an S node is determined by the highest VP below the S node and the syntactic configuration the S node is in.

30.6.1 Verbs with object sentential complements

Finite sentential complements:

(662) $\mathbf{S_1.t{:}\langle comp\rangle = that/whether/if/nil}$

(663) $\mathbf{S_1.t{:}\langle mode\rangle = ind/sbjnct}$ or $\mathbf{S_1.t{:}\langle mode\rangle = ind}$

(664) $\mathbf{S_1.t{:}\langle assign\text{-}comp\rangle = ind\_nil/inf\_nil}$

The presence of an overt complementizer is optional.
Non-finite sentential complements, do not permit *for*:

(665) $\mathbf{S_1.t{:}\langle comp\rangle = nil}$

(666) $\mathbf{S_1.t{:}\langle mode\rangle = inf}$

(667) $\mathbf{S_1.t{:}\langle assign\text{-}comp\rangle = ind\_nil/inf\_nil}$

Non-finite sentential complements, permit *for*:

(668) $\mathbf{S_1.t{:}\langle comp\rangle = for/nil}$

(669) $\mathbf{S_1.t{:}\langle mode\rangle = inf}$

(670) $\mathbf{S_1.t{:}\langle assign\text{-}comp\rangle = ind\_nil/inf\_nil}$

Cases like '*I want for to win' are independently ruled out due to a case feature clash between the **acc** assigned by *for* and the intrinsic case feature **none** on the PRO.
Non-finite sentential complements, ECM:

(671) $\mathbf{S_1.t{:}\langle comp\rangle = nil}$

(672) $\mathbf{S_1.t{:}\langle mode\rangle = inf}$

(673) $\mathbf{S_1.t{:}\langle assign\text{-}comp\rangle = ecm}$

30.6.2 Verbs with sentential subjects

The following contrast involving complementizers surfaces with sentential subjects:

(674) *(That) John is crazy is likely.

Indicative sentential subjects obligatorily have complementizers while infinitival sentential subjects may or may not have a complementizer. Also *if* is possible as the complementizer of an object clause but not as the complementizer of a sentential subject.

(675) $\mathbf{S_0.t{:}\langle comp\rangle = that/whether/for/nil}$

(676) $\mathbf{S_0.t{:}\langle mode\rangle = inf/ind}$

(677) **S_0.t:⟨assign-comp⟩** = **inf_nil**

If the sentential subject is finite and a complementizer does not adjoin in, the ⟨**assign-comp**⟩ feature of the S_0 node of the embedding clause and the root node of the embedded clause will fail to unify. If a complementizer adjoins in, there will be no feature-mismatch because the root of the complementizer tree is not specified for the ⟨**assign-comp**⟩ feature.

The ⟨**comp**⟩ feature **nil** is split into two ⟨**assign-comp**⟩ features **ind_nil** and **inf_nil** to capture the fact that there are certain configurations in which it is acceptable for an infinitival clause to lack a complementizer but not acceptable for an indicative clause to lack a complementizer.

30.6.3 *That*-trace and *for*-trace effects

(678) Who$_i$ do you think (*that) t$_i$ ate the apple?

That trace violations are blocked by the presence of the following equation:

(679) **S_r.b:⟨assign-comp⟩** = **inf_nil/ind_nil/ecm**

on the bottom of the S_r nodes of trees with extracted subjects (W0). The **ind_nil** feature specification permits the above example while the **inf_nil/ecm** feature specification allows the following examples to be derived:

(680) Who$_i$ do you want [t$_i$ to win the World Cup]?

(681) Who$_i$ do you consider [t$_i$ intelligent]?

The feature equation that ruled out the *that*-trace filter violations will also serve to rule out the *for*-trace violations above.

30.7 Relative Clauses

Features that are peculiar to the relative clause system are:
<nocomp-mode>, possible values are **inf/ger/ppart**
⟨**rel-clause**⟩, possible values are +/−

- **<nocomp-mode>**:

  Relative clauses which have subject extraction cannot have a null COMP, unless there is an intervening clause or the relative clause is a reduced relative (**mode = inf/ppart/ger**). In order to ensure this, the top feature set of the root node of the relative clause is specified for **<nocomp-mode>=inf/ger/ppart** in conjunction with the following equations:

  (682) **S_r.t:⟨nocomp-mode⟩** = **inf/ger/ppart** (in relative clause trees with subject extraction)

  (683) **S_r.b:⟨nocomp-mode⟩** = **S_r.b:⟨mode⟩**

The full set of the equations above is only present in Comp substitution trees involving subject extraction. So the following will not be ruled out.

(684) the toy [ϵ_w [ϵ_C [Dafna likes t_i]]]

The feature mismatch induced by the above equations is not remedied by adjunction of just any S-adjunct because all other S-adjuncts are transparent to the $\langle$**nocomp-mode**$\rangle$ feature because of the following equation:

(685) $\mathbf{S}_m$**.b:**$\langle$**nocomp-mode**$\rangle$ =$\mathbf{S}_f$**.t:**$\langle$**nocomp-mode**$\rangle$

- $\langle$**rel-clause**$\rangle$:

  The XTAG analysis forces the adjunction of the determiner below the relative clause. This is done by using the $\langle$**rel-clause**$\rangle$ feature. The relevant equations are:

  (686) On the root of the Relative clause: $\mathbf{NP}_r$**.b:**$\langle$**rel-clause**$\rangle$ = +

  (687) On the foot node of the Determiner tree: $\mathbf{NP}_f$**.t:**$\langle$**rel-clause**$\rangle$ = −

  Relative clauses also use the <**mode**> and the <**comp**> features in a manner largely parallel to the way they are used in sentential complementation. Some of the feature specifications and equations that are specific to relative clause formation are discussed below:

- Complementizers (which appear in relative clauses via adjunction) can never cooccur with the overt extracted + <**wh**> NP (cf. **I saw the man who$_i$ that Muriel saw ϵ_i*). Consequently, the auxiliary βCOMPs trees are prevented from adjoining at the $\mathbf{S}_r$ node in the relative clause trees by the equation

  (688) $\mathbf{S}_r$**.t:**$\langle$**comp**$\rangle$ = **nil**

  which will always fail to unify with the (non-**nil**) values of the <**comp**> feature in the βCOMPs trees.

- Relative clause trees that have $\mathbf{NP}_w$ as a substitution node have the feature equation given below.

  (689) $\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ =**ind**

- Trees with a $\mathbf{PP}_w$ substitution node have the feature equation:

  (690) $\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ = **ind/inf**

- In relative clauses with covert extracted + <**wh**> **NP**, the mode of relative clause varies depending on which argument has been extracted. The full set of mode restrictions on the different relative clause trees is as follows:

  (691) For all non-passive cases of subject extraction, $\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ = **ind/ger/inf**

  (692) For all passive cases of subject extraction, $\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ = **ind/ger/ppart/inf**

  (693) For all cases of non-subject extraction, $\mathbf{S}_r$**.t:**$\langle$**mode**$\rangle$ = **ind/inf**

30.8 Determiner ordering

⟨**card**⟩, possible values are +, −
⟨**compl**⟩, possible values are +, −
⟨**const**⟩, possible values are +, −
⟨**decreas**⟩, possible values are +, −
⟨**definite**⟩, possible values are +, −
⟨**gen**⟩, possible values are +, −
⟨**quan**⟩, possible values are +, −
For detailed discussion see Chapter 20.

30.9 Punctuation

⟨**punct**⟩ is a complex feature. It has the following as its subfeatures:
⟨**punct bal**⟩, possible values are **dquote, squote, paren, nil**
⟨**punct contains colon**⟩, possible values are +, −
⟨**punct contains dash**⟩, possible values are +, −
⟨**punct contains dquote**⟩, possible values are +, −
⟨**punct contains scolon**⟩, possible values are +, −
⟨**punct contains squote**⟩, possible values are +, −
⟨**punct struct**⟩, possible values are **comma, dash, colon, scolon, none, nil**
⟨**punct term**⟩, possible values are **per, qmark, excl, none, nil**
For detailed discussion see Chapter 25.

30.10 Conjunction

⟨**conj**⟩, possible values are **but, and, or, comma, scolon, to, disc, nil**
The ⟨**conj**⟩ feature is specified in the lexicon for each conjunction and is passed up to the root node of the conjunction tree. If the conjunction is *and*, the root ⟨**agr num**⟩ is ⟨**plural**⟩, no matter what the number of the two conjuncts. With *or*, the the root ⟨**agr num**⟩ is equated to the ⟨**agr num**⟩ feature of the right conjunct.

The ⟨**conj**⟩**=disc** feature is only used at the root of the βCONJs tree. It blocks the adjunction of one βCONJs tree on another. The following equations are used, where S_r is the substitution node and S_c is the root node:

(694) S_r.t:⟨conj⟩ = disc

(695) S_c.b:⟨conj⟩ = and/or/but/nil

30.11 Comparatives

⟨**compar**⟩, possible values are +, −
⟨**equiv**⟩, possible values are +, −
⟨**super**⟩, possible values are +, −
For detailed discussion see Chapter 24.

30.12 Control

$\langle$**control**$\rangle$ has no value and is used only for indexing purposes. The root node of every clausal tree has its $\langle$**control**$\rangle$ feature coindexed with the control feature of its subject. This allows adjunct control to take place. In addition, clauses that take infinitival clausal complements have the control feature of their subject/object coindexed with the control feature of their complement clause S, depending upon whether they are subject control verbs or object control verbs respectively.

30.13 Other Features

$\langle$**neg**$\rangle$, possible values are **+, –**
Used for controlling the interaction of negation and auxiliary verbs.
$\langle$**pred**$\rangle$, possible values are **+, –**
The $\langle$**pred**$\rangle$ feature is used in the following tree families: Tnx0N1.trees and Tnx0nx1ARB.trees. In the Tnx0N1.trees family, the following equations are used:
for αW1nx0N1:

(696) NP_1.t:$\langle$pred$\rangle$ = +

(697) NP_1.b:$\langle$pred$\rangle$ = +

(698) NP.t:$\langle$pred$\rangle$ = +

(699) N.t:$\langle$pred$\rangle$ = NP.b:$\langle$pred$\rangle$

This is the only tree in this tree family to use the $\langle$**pred**$\rangle$ feature.

The other tree family where the $\langle$**pred**$\rangle$ feature is used is Tnx0nx1ARB.trees. Within this family, this feature (and the following equations) are used only in the αW1nx0nx1ARB tree.

(700) AdvP_1.t:$\langle$pred$\rangle$ = +

(701) AdvP_1.b:$\langle$pred$\rangle$ = +

(702) NP.t:$\langle$pred$\rangle$ = +

(703) AdvP.b:$\langle$pred$\rangle$ = NP.t:$\langle$pred$\rangle$

$\langle$**pron**$\rangle$, possible values are **+, –**
This feature indicates whether a particular NP is a pronoun or not. Certain constructions which do not permit pronouns use this feature to block pronouns.
$\langle$**tense**$\rangle$, possible values are **pres, past**
It does not seem to be the case that the $\langle$**tense**$\rangle$ feature interacts with other features/syntactic processes. It comes from the lexicon with the verb and is transmitted up the tree in such a way that the root S node ends up with the tense feature of the highest verb in the tree. The equations used for this purpose are:

(704) $\mathbf{S}_r$**.b:**$\langle$**tense**$\rangle$ **= VP.t:**$\langle$**tense**$\rangle$

(705) **VP.b:**$\langle$**tense**$\rangle$ **= V.t:**$\langle$**tense**$\rangle$

Chapter 31

Evaluation and Results

In this appendix we describe various evaluations done of the XTAG grammar. Some of these evaluations were done on an earlier version of the XTAG grammar (the 1995 release), while others were done more recently. We will try to indicate in each section which version was used.

31.1 Parsing Corpora

In the XTAG project, we have used corpus analysis in two main ways: (1) to identify gaps in the grammar and (2) to measure the performance of the English grammar on a given genre.

The first type of evaluation involves performing detailed error analysis on the sentences rejected by the parser, and we have done this several times on WSJ and Brown data. Based on the results of such analysis, we prioritize upcoming grammar development efforts. The results of an error analysis done in 1994 [Doran *et al.*, 1994] are shown in Table 31.1.

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 31.1: Results of Corpus Based Error Analysis with XTAG-95

The table does not show errors in parsing due to mistakes made by the POS tagger which contributed the largest number of errors: 32. Consequent to this error analysis, 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). A more recent evaluation was done on a corpus of weather reports [Doran *et al.*, 1997]. The weather reports were provided to us by CoGenTex.[1] The sentences in this kind of corpus tend to be quite long (an average of 20 tokens/sentence) and complex. The examples given below are illustrative of the type of sentences and terminology in this domain.

(706) [The warm air mass] affecting [southwestern Quebec] for [the past few days] is still moving slowly eastwards and will make room for [a much colder air mass] overnight and Wednesday.

(707) Behind [this area] [a moderate flow] will cause [an inflow] of [milder air] in [southwestern Quebec] producing temperatures slightly above normal on Sunday.

The performance evaluation showed that the weather reports contained several relative clauses that caused problems for the grammar and also that the parser was unable to handle some of the more complex longer sentences. The problematic cases involved two kinds of relative clauses (roughly 40%) which were not accounted for by the grammar developer at the time. The first kind contained examples like *A frontal system approaching from the west* which included an *-ing* form in the relative clause predicate, and the other contained examples like *The disturbance south of Nova Scotia early this morning* which had a directional noun phrase as the predicate. As a result of these shortcomings and also due to the long and complex sentences in this test set, we could parse only about 20% of the test set (10 out of 48 sentences).

The current version of XTAG contains an overhaul of the relative clause analysis, which has been updated to extend the coverage of the grammar. An evaluation of the current grammar was done in [Prasad and Sarkar, 2000], which shows a much better performance of the XTAG parser on the sentences in the weather domain. Of the 48 sentences in the corpus, we were able to parse 43 sentences (89.6%). A comprehensive breakdown of the errors is described in Table 31.2.[2] [3]

We find that this corpus-based method of error analysis is very useful in focusing grammar development in a productive direction. Furthermore, to ensure that we are not losing coverage of certain phenomena as we extend the grammar, we have a benchmark set of grammatical and ungrammatical sentences from this technical report. We parse these sentences periodically to ensure that in adding new features and constructions to the grammar, we are not blocking previous analyses. There are approximately 590 example sentences in this set.

[1] Thanks to the Contrastive Syntax Project, Linguistics Department of the University of Montreal, for the use of their weather synopsis corpus.

[2] A description of the error types is as follows:

R1: Lexical item does not get the right part of speech and therefore does not select the correct trees.

R2: Lexical item does not select the necessary tree or family.

R3: Missing analysis for VP coordination in the current grammar.

[3] We are in the process of extending the parser to handle VP coordination (#9) (See Chapter 23 on recent work to handle VP and other predicative coordination).

Error Class	No.	%
POS Tag (R1)	1	2.08%
Missing Tree (R2)	1	2.08%
No VP coordination (R3)	3	6.25%
Total	**5**	**10.4%**

Table 31.2: Results of Corpus-Based Error Analysis with XTAG-98

31.2 Parsing Test-Suites

In addition to corpus-based evaluation, we have also run the English Grammar on test-suites. Test-suites are intended to be a systematic collection of different types of grammatical phenomena and are used to track improvements and verify consistency between successive generations of grammar development in a system.

31.2.1 TSNLP

In an earlier evaluation of the grammar on test-suites, we used the TSNLP (Test Suites Natural Language Processing) English corpus [Lehmann *et al.*, 1996]. This test set contains construction types which include 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.

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 31.3: Breakdown of TSNLP Errors

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 31.3. As with the error analysis described above, we used this information to help direct our grammar development efforts. It also highlighted the fact that our grammar is heavily slanted toward American English—our grammar did not handle *dare* or *need* as auxiliary verbs[4], and there were a number of very British particle constructions, e.g. *She misses him out.*

31.2.2 CSLI LKB

In a more recent test-suite based evaluation [Prasad and Sarkar, 2000], we used the CSLI LKB (Linguistic Knowledge Building) test suite [Copestake, 1999]. Of the 966 grammatical sentences

[4]The current version of the grammar, however, has been extended to handle these auxiliaries

we were unable to parse 26 (2.7%). A breakdown of the errors is given in Table 31.4):

Error Class	No.	%
Missing Lexical Entry	4	0.4%
Missing Lexicalized Tree	4	0.4%
No analysis for Inverse Predications	2	0.2%
No analysis for Ellipsis	18	1.8%
Default entry error	1	0.1%
Total	**26**	**2.7%**

Table 31.4: Results of CSLI LKB Test-Suite based Evaluation

The analysis for ellipsis and inverse predication has not been added yet in our grammar. Of the 387 ungrammatical sentences, 189 did not get any valid parses after feature unification. We did not examine the parses obtained for the remaining sentences to see if they contained a legitimate ambiguity or an error in the grammar.

There are several problems with the test-suites as a stand alone metric for evaluating wide-coverage grammars like XTAG. Firstly, they are constructed by hand to allow for systematic variation only within a particular range of grammatical phenomena and, in contrast to the sentences found in naturally-occurring corpora, are unlikely to point us towards an increasing set of novel constructions. Secondly, in a test-suite, the same lexical items are used very often, causing a distribution of words that is unlike naturally occurring text (see [Doran *et al.*, 1997] and [Prasad and Sarkar, 2000] for further discussions about the use of test-suites for evaluation) and, in addition, can also cause a disproportionate amount of grief. For instance, in the TSNLP test-suite, *used to* appears 33 times and we got all 33 wrong. Thirdly, each sentence in a test-suite usually handles a single grammatical phenomena and, so, interactions between different phenomena are seldom explored. Finally, the XTAG grammar has analyses for syntactic phenomena that were not represented in the test suites, such as sentential subjects and subordinating clauses, among others. The test-suite evaluations, therefore, were useful in highlighting some deficiencies in our grammar, but did not provide the same sort of general evaluation as parsing corpus data.

31.3 Chunking and Dependencies in XTAG Derivations

We evaluated the XTAG parser for the text chunking task [Abney, 1991]. In particular, we compared NP chunks and verb group (VG) chunks[5] produced by the XTAG parser with the NP and VG chunks from the Penn Treebank [Marcus *et al.*, 1993]. The test involved 940 sentences of length 15 words or less from sections 17 to 23 of the Penn Treebank, parsed using the XTAG English grammar. The results are given in Table 31.5.

As described earlier, the results cannot be directly compared with other results in chunking such as in [Ramshaw and Marcus, 1995] since we do not train from the Treebank before testing. However, in earlier work, text chunking was done using a technique called supertagging [Srinivas,

[5]We treat a sequence of verbs and verbal modifiers, including auxiliaries, adverbs, modals as constituting a verb group.

	NP Chunking	VG Chunking
Recall	82.15%	74.51%
Precision	83.94%	76.43%

Table 31.5: Text Chunking performance of the XTAG parser

System	Training Size	Recall	Precision
Ramshaw & Marcus	Baseline	81.9%	78.2%
Ramshaw & Marcus (without lexical information)	200,000	90.7%	90.5%
Ramshaw & Marcus (with lexical information)	200,000	92.3%	91.8%
Supertags	Baseline	74.0%	58.4%
Supertags	200,000	93.0%	91.8%
Supertags	1,000,000	93.8%	92.5%

Table 31.6: Performance comparison of the transformation based noun chunker and the supertag based noun chunker

1997b] (which uses the XTAG English grammar) which can be used to train from the Treebank. The comparative results of text chunking between supertagging and other methods of chunking is shown in Figure 31.6.[6]

We also performed experiments to determine the accuracy of the derivation structures produced by XTAG on WSJ text, where the derivation tree produced after parsing XTAG is interpreted as a dependency parse. We took sentences that were 15 words or less from the Penn Treebank [Marcus *et al.*, 1993]. The sentences were collected from sections 17–23 of the Treebank. 9891 of these sentences were given at least one parse by the XTAG system. Since XTAG typically produces several derivations for each sentence we simply picked a single derivation from the list for this evaluation. Better results might be achieved by ranking the output of the parser using the sort of approach described in [Srinivas *et al.*, 1995].

There were some striking differences in the dependencies implicit in the Treebank and those given by XTAG derivations. For instance, often a subject NP in the Treebank is linked with the first auxiliary verb in the tree, either a modal or a copular verb, whereas in the XTAG derivation, the same NP will be linked to the main verb. Also XTAG produces some dependencies within an NP, while a large number of words in NPs in the Treebank are directly dependent on the verb. To normalize for these facts, we took the output of the NP and VG chunker described above and accepted as correct any dependencies that were completely contained within a single chunk.

For example, for the sentence *Borrowed shares on the Amex rose to another record*, the XTAG and Treebank chunks are shown below.

```
XTAG chunks:
 [Borrowed shares] [on the Amex] [rose]
```

[6] It is important to note in this comparison that the supertagger uses lexical information on a per word basis only to pick an initial set of supertags for a given word.

```
    [to another record]
Treebank chunks:
 [Borrowed shares on the Amex] [rose]
    [to another record]
```

Using these chunks, we can normalize for the fact that in the dependencies produced by XTAG *borrowed* is dependent on *shares* (i.e. in the same chunk) while in the Treebank *borrowed* is directly dependent on the verb *rose*. That is to say, we are looking at links between chunks, not between words. The dependencies for the sentence are given below.

```
XTAG dependency     Treebank dependency

Borrowed::shares    Borrowed::rose
shares::rose        shares::rose
on::shares          on::shares
the::Amex           the::Amex
Amex::on            Amex::on
rose::NIL           rose::NIL
to::rose            to::rose
another::record     another::record
record::to          record::to
```

After this normalization, testing simply consisted of counting how many of the dependency links produced by XTAG matched the Treebank dependency links. Due to some tokenization and subsequent alignment problems we could only test on 835 of the original 9891 parsed sentences. There were a total of 6135 dependency links extracted from the Treebank. The XTAG parses also produced 6135 dependency links for the same sentences. Of the dependencies produced by the XTAG parser, 5165 were correct giving us an accuracy of 84.2%.

31.4 Comparison with IBM

The evaluation in this section was done with the earlier 1995 release of the grammar. This section describes an experiment to measure the crossing bracket accuracy of the XTAG-parsed IBM-manual sentences. In this experiment, XTAG parses of 1100 IBM-manual sentences have been ranked using certain heuristics. The ranked parses have been compared[7] against the bracketing given in the Lancaster Treebank of IBM-manual sentences[8]. Table 31.7 shows the results of XTAG obtained in this experiment, which used the highest ranked parse for each system. It also shows the results of the latest IBM statistical grammar ([Jelinek *et al.*, 1994]) on the same genre of sentences. Only the highest-ranked parse of both systems was used for this evaluation. Crossing Brackets is the percentage of sentences with no pairs of brackets crossing the Treebank bracketing (i.e. ((a b) c) has a crossing bracket measure of one if compared to (a (b c))). Recall is the ratio of the number of constituents in the XTAG parse to the number of constituents in the corresponding Treebank sentence. Precision is the ratio of the number of correct constituents to the total number of constituents in the XTAG parse.

[7] We used the parseval program written by Phil Harison (phil@atc.boeing.com).

[8] The Treebank was obtained through Salim Roukos (roukos@watson.ibm.com) at IBM.

System	# of sentences	Crossing Bracket Accuracy	Recall	Precision
XTAG	1100	81.29%	82.34%	55.37%
IBM Statistical grammar	1100	86.20%	86.00%	85.00%

Table 31.7: Performance of XTAG on IBM-manual sentences

As can be seen from Table 31.7, the precision figure for the XTAG system is considerably lower than that for IBM. For the purposes of comparative evaluation against other systems, we had to use the same crossing-brackets metric though we believe that the crossing-brackets measure is inadequate for evaluating a grammar like XTAG. There are two reasons for the inadequacy. First, the parse generated by XTAG is much richer in its representation of the internal structure of certain phrases than those present in manually created treebanks (e.g. IBM: [$_N$ your personal computer], XTAG: [$_{NP}$ [$_G$ your] [$_N$ [$_N$ personal] [$_N$ computer]]]). This is reflected in the number of constituents per sentence, shown in the last column of Table 31.8.[9]

System	Sent. Length	# of sent	Av. # of words/sent	Av. # of Constituents/sent
XTAG	1-10	654	7.45	22.03
	1-15	978	9.13	30.56
IBM Stat. Grammar	1-10	447	7.50	4.60
	1-15	883	10.30	6.40

Table 31.8: Constituents in XTAG parse and IBM parse

A second reason for considering the crossing bracket measure inadequate for evaluating XTAG is that the primary structure in XTAG is the derivation tree from which the bracketed tree is derived. Two identical bracketings for a sentence can have completely different derivation trees (e.g. *kick the bucket* as an idiom vs. a compositional use). A more direct measure of the performance of XTAG would evaluate the derivation structure, which captures the dependencies between words.

31.5 Comparison with Alvey

The evaluation in this section was done with the earlier 1995 release of the grammar. This section compares XTAG to the Alvey Natural Language Tools (ANLT) Grammar. We parsed the set of LDOCE Noun Phrases presented in Appendix B of the technical report ([Carroll, 1993]) using XTAG. Table 31.9 summarizes the results of this experiment. A total of 143 noun phrases were parsed. The NPs which did not have a correct parse in the top three derivations were considered failures for either system. The maximum and average number of derivations columns show the highest and the average number of derivations produced for the NPs that have a correct derivation in the top three. We show the performance of XTAG both with and

[9] We are aware of the fact that increasing the number of constituents also increases the recall percentage. However we believe that this a legitimate gain.

without the tagger since the performance of the POS tagger is significantly degraded on the NPs because the NPs are usually shorter than the sentences on which it was trained. It would be interesting to see if the two systems performed similarly on a wider range of data.

System	# of NPs	# parsed	% parsed	Maximum derivations	Average derivations
ANLT Parser	143	127	88.81%	32	4.57
XTAG Parser with POS tagger	143	93	65.03%	28	3.45
XTAG Parser without POS tagger	143	120	83.91%	28	4.14

Table 31.9: Comparison of XTAG and ANLT Parser

31.6 Comparison with CLARE

The evaluation in this section was done with the earlier 1995 release of the grammar. This section compares the performance of XTAG against that of the CLARE-2 system ([Alshawi *et al.*, 1992]) on the ATIS corpus. Table 31.10 shows the performance results. The percentage parsed column for both systems represents the percentage of sentences that produced any parse. It must be noted that the performance result shown for CLARE-2 is without any tuning of the grammar for the ATIS domain. The performance of CLARE-3, a later version of the CLARE system, is estimated to be 10% higher than that of the CLARE-2 system.[10]

System	Mean length	% parsed
CLARE-2	6.53	68.50%
XTAG	7.62	88.35%

Table 31.10: Performance of CLARE-2 and XTAG on the ATIS corpus

System	Corpus	Mean length	% parsed
CLARE-2	LOB	5.95	53.40%
XTAG	WSJ	6.00	55.58%

Table 31.11: Performance of CLARE-2 and XTAG on LOB and WSJ corpus respectively

In an attempt to compare the performance of the two systems on a wider range of sentences (from similar genres), we provide in Table 31.11 the performance of CLARE-2 on LOB corpus and the performance of XTAG on the WSJ corpus. The performance was measured on sentences of up to 10 words for both systems.

[10]When CLARE-3 is tuned to the ATIS domain, performance increases to 90%. However XTAG has not been tuned to the ATIS domain.

Bibliography

[Abeillé and Schabes, 1989] Anne Abeillé and Yves Schabes. Parsing Idioms in Lexicalized TAGs. In *Proceedings of EACL '89*, pages 161–65, 1989.

[Abeillé *et al.*, 2000] Anne Abeillé, Marie-Hélène Candito, and Alexandra Kinyon. The Current Status of FTAG. In *Fifth International Workshop on Tree Adjoining Grammars and Related Formalisms*, 2000.

[Abeillé, 1990] Anne Abeillé. French and english determiners: Interaction of morphology, syntax, and semantics in lexicalized tree adjoining grammars. In *Tree Adjoining Grammar,First International Workshop on TAGs: Formal Theory and Applications (abstracts)*, Shloss Dagstuhl, Sarrbrücken, 1990.

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

[Abney, 1991] Steven Abney. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-based parsing*. Kluwer Academic Publishers, 1991.

[Akmajian, 1970] A. Akmajian. On deriving cleft sentences from pseudo-cleft sentences. *Linguistic Inquiry*, 1:149–168, 1970.

[Akmajian, 1984] Adrian Akmajian. Sentence types and the form-function fit. *Natural Language and Linguistic Theory*, 2:1–23, 1984.

[Alshawi *et al.*, 1992] Hiyan Alshawi, David Carter, Richard Crouch, Steve Pullman, Manny Rayner, and Arnold Smith. *CLARE – A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine*. SRI International, Cambridge, England, 1992.

[Ball, 1991] Catherine N. Ball. *The historical development of the it-cleft*. PhD thesis, University of Pennsylvania, 1991.

[Baltin, 1989] Mark Baltin. Heads and projections. In Mark Baltin and Anthony S. Kroch, editors, *Alternative conceptions of phrase structure*, pages 1–16. University of Chicago Press, Chicago, Illinois, 1989.

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

[Becker, 1993] Tilman Becker. *HyTAG: A new Type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Word Order Languages.* PhD thesis, Universität des Saarlandes, 1993.

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

[Bhatt, 1994] Rajesh Bhatt. Pro-control in tags. Unpublished paper, University of Pennsylvania, 1994.

[Browning, 1987] Marguerite Browning. *Null Operator Constructions.* PhD thesis, MIT, 1987.

[Burzio, 1986] Luigi Burzio. *Italian syntax. A Government-Binding approach.* Studies in natural language and linguistic theory. Reidel, Dordrecht, 1986.

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

[Carrier and Randall, 1992] Jill Carrier and Janet H. Randall. The argument structure and syntactic structure of resultatives. *Linguistic Inquiry*, 23:173–234, 1992.

[Carroll, 1993] John Carroll. *Practical Unification-based Parsing of Natural Language.* University of Cambridge, Computer Laboratory, Cambridge, England, 1993.

[Chomsky, 1965] Noam Chomsky. *Aspects of the Theory of Syntax.* MIT Press, Cambridge, Massachusetts, 1965.

[Chomsky, 1970] Noam Chomsky. Remarks on Nominalization. In *Readings in English Transformational Grammar*, pages 184–221. Ginn and Company, Waltham, Massachusetts, 1970.

[Chomsky, 1986] Noam Chomsky. *Barriers.* MIT Press, Cambridge, Massachusetts, 1986.

[Chomsky, 1992] Noam Chomsky. A Minimalist Approach to Linguistic Theory. *MIT Working Papers in Linguistics*, Occasional Papers in Linguistics No. 1, 1992.

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

[Cinque, 1990] G. Cinque. *Types of A-bar-Dependencies.* MIT Press, Cambridge, Massachusetts; London, England, 1990.

[Copestake, 1999] A. Copestake. *The (new) LKB System.* CSLI, Stanford University, 1999.

[Cowie and Mackin, 1975] A. P. Cowie and R. Mackin, editors. *Oxford Dictionary of Current Idiomatic English*, volume 1. Oxford University Press, London, England, 1975.

[Delahunty, 1984] Gerald P. Delahunty. The Analysis of English Cleft Sentences. *Linguistic Analysis*, 13(2):63–113, 1984.

[Delin, 1989] Judy L. Delin. *Cleft Constructions in Discourse.* PhD thesis, University of Edinburgh, 1989.

[Doran *et al.*, 1994] Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. XTAG System - A Wide Coverage Grammar for English. In *Proceedings of the* 17^{th} *International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan, August 1994.

[Doran *et al.*, 1997] C. Doran, B. Hockey, P. Hopely, J. Rosenzweig, A. Sarkar, B. Srinivas, F. Xia, A. Nasr, and O. Rambow. Maintaining the Forest and Burning out the Underbrush in XTAG. In *Workshop on Computational Environments for Grammar Development and Language Engineering (ENVGRAM)*, Madrid, Spain, July 1997.

[Doran, 1998] Christine Doran. *Incorporating Punctuation into the Sentence Grammar: A Lexicalized Tree Adjoining Grammar Perspective.* PhD thesis, University of Pennsylvania, 1998.

[Egedi and Martin, 1994] 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.

[Emonds, 1970] J. Emonds. *Root and structure-preserving transformations.* PhD thesis, Massachusetts Institute of Technology, 1970.

[Ernst, 1983] T. Ernst. More on Adverbs and Stressed Auxiliaries. *Linguistic Inquiry 13*, pages 542–48, 1983.

[Evans *et al.*, 1995] 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.

[Frank, 2000] Robert Frank. *Phrase Structure Composition and Syntactic Dependencies.* MIT Press, Cambridge, MA, 2000.

[Gazdar *et al.*, 1985] G. Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge, Massachusetts, 1985.

[Grimshaw, 1990] Jane Grimshaw. *Argument Structure.* MIT Press, Cambridge, Massachusetts, 1990.

[Haegeman, 1991] Liliane Haegeman. *Introduction to Government and Binding Theory.* Blackwell Publishers, Cambridge, Massachusetts, Oxford, U.K., 1991.

[Hale and Keyser, 1986] Ken Hale and Samuel Jay Keyser. Some transitivity alternations in English. Technical Report #7, MITCCS, July 1986.

[Hale and Keyser, 1987] Ken Hale and Samuel Jay Keyser. A view from the middle. Technical Report #10, MITCCS, January 1987.

[Han, 1999] Chung-Hye Han. *The Structure and interpretation of imperatives: mood and force in universal grammar.* PhD thesis, University of Pennsylvania, 1999.

[Hanks, 1979] Patrick Hanks, editor. *Collins Dictionary of the English Language.* Collins, London, England, 1979.

[Heycock and Kroch, 1993] Caroline Heycock and Anthony S. Kroch. Verb movement and the status of subjects: implications for the theory of licensing. *GAGL*, 36:75–102, 1993.

[Heycock, 1991] Caroline Heycock. Progress Report: The English Copula in a LTAG. Internal XTAG project report, University of Pennsylvania, Summer 1991.

[Hockey and Mateyak, 1998] Beth Ann Hockey and Heather Mateyak. Determining determiner sequencing: A syntactic analysis for english. Technical Report 98-10, Institute for Research in Cognitive Science, University of Pennsylvania, 1998.

[Hockey, 1994] Beth Ann Hockey. Echo questions, intonation and focus. In *Proceedings of the Interdisciplinary Conference on Focus and Natural Language Processing in Celebration of the 10^{th} anniversary of the Journal of Semantics*, Eschwege, Germany, 1994.

[Hoekstra, 1988] Teun Hoekstra. Small clause results. *Lingua*, 74:101–139, 1988.

[Hornby, 1974] A. S. Hornby, editor. *Oxford Advanced Learner's Dictionary of Current English.* Oxford University Press, London, England, third edition, 1974.

[Iatridou, 1991] Sabine Iatridou. *Topics in Conditionals.* PhD thesis, MIT, 1991.

[Jackendoff, 1972] R. Jackendoff. *Semantic Interpretation in Generative Grammar.* MIT Press, Cambridge, Massachusetts, 1972.

[Jackendoff, 1990] R. Jackendoff. On Larson's Analysis of the Double Object Construction. *Linguistic Inquiry*, 21:427–456, 1990.

[Jelinek *et al.*, 1994] Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, Adwait Ratnaparkhi, and Salim Roukos. Decision Tree Parsing using a Hidden Derivation Model. In *Proceedings from the ARPA Workshop on Human Language Technology Workshop*, March 1994.

[Joshi and Kulick, 1997] Aravind Joshi and Seth Kulick. Partial proof trees as building blocks for a categorial grammar. *Linguistics and Philosophy*, 20:637–667, 1997. To appear.

[Joshi and Schabes, 1996] Aravind Joshi and Yves Schabes. *Handbook of Formal Lanaguages and Automata*, chapter Tree-Adjoining Grammars. Springer-Verlag, Berlin, 1996.

[Joshi and Vijay-Shanker, 1999] A. K. Joshi and K. Vijay-Shanker. Compositional semantics with lexicalized tree-adjoining grammar (ltag): How much underspecification is necessary? In *Proceedings of the Third International Workshop on Computational Semantics (IWCS-3)*, pages 131–145, Tilburg, 1999.

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

[Joshi *et al.*, 1999] Aravind Joshi, Seth Kulick, and Natasha Kurtonina. Semantic composition for partial proof trees. In *Proc. of the Twelfth Amsterdam Colloquium*, pages 169–174, University of Amsterdam, Amsterdam, 1999.

[Joshi, 1985] Aravind K. Joshi. Tree Adjoining Grammars: How much context Sensitivity is required to provide a reasonable structural description. In D. Dowty, I. Karttunen, and A. Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge, U.K., 1985.

[Joshi, 1990] Aravind K Joshi. Processing crossed and nested dependencies: an automaton perspective on psycholinguisitc results. *Language and Cognitive Processes*, 5(1), 1990.

[Karp *et al.*, 1992] 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.

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

[Knowles, 1986] John Knowles. The Cleft Sentence: A Base-Generated Perspective. *Lingua: International Review of General Linguistics*, 69(4):295–317, August 1986.

[Kroch and Joshi, 1985] Anthony S. Kroch and Aravind K. Joshi. The Linguistic Relevance of Tree Adjoining Grammars. Technical Report MS-CIS-85-16, Department of Computer and Information Science, University of Pennsylvania, 1985.

[Kroch and Joshi, 1987] Anthony S. Kroch and Aravind K. Joshi. Analyzing Extraposition in a Tree Adjoining Grammar. In G. Huck and A. Ojeda, editors, *Discontinuous Constituents, Syntax and Semantics*, volume 20. Academic Press, 1987.

[Lapointe, 1980] S. Lapointe. A lexical analysis of the English auxiliary verb system. In T. Hoekstra et al, editor, *Lexical Grammar*, pages 215–254. Foris, Dordrecht, 1980.

[Larson, 1988] R. Larson. On the Double Object construction. *Linguistic Inquiry*, 19:335–391, 1988.

[Larson, 1990] R. Larson. Double Objects Revisited: Reply to Jackendoff. *Linguistic Inquiry*, 21:589–612, 1990.

[Lasnik and Saito, 1984] H. Lasnik and M. Saito. On the Nature of Proper Government. *Linguistic Inquiry*, 15:235–289, 1984.

[Lasnik and Uriagereka, 1988] H. Lasnik and J. Uriagereka. *A Course in GB Syntax*. MIT Press, Cambridge, Massachusetts, 1988.

[Lees, 1960] Robert B. Lees. *The Grammar of English Nominalizations*. Indiana University Research Center in Anthropology, Folklore, and Linguistics, Indiana University, Bloomington, Indiana, 1960.

[Lehmann *et al.*, 1996] 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.

[Liberman, 1989] Mark Liberman. Tex on tap: the ACL Data Collection Initiative. In *Proceedings of the DARPA Workshop on Speech and Natural Language Processing*. Morgan Kaufman, 1989.

[Marcus *et al.*, 1993] Mitchell M. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19.2:313–330, June 1993.

[Mateyak, 1997] Heather Mateyak. Negation of noun phrases with *not*. Technical Report 97-18, UPENN-IRCS, October 1997.

[McCawley, 1988] James D. McCawley. *The Syntactic Phenomena of English*. The University of Chicago Press, Chicago, Illinois, 1988.

[Moro, 1990] A. Moro. *There* as a Raised Predicate. Paper presented at GLOW, 1990.

[Napoli, 1988] Donna Jo Napoli. Ergative Verbs in English. *Journal of the Linguistic Society of America (Language)*, 64:1(1):130–142, March 1988.

[Obernauer, 1984] H. Obernauer. On the Identification of Empty Categories. *Linguistic Review 4*, 2:153–202, 1984.

[Paroubek *et al.*, 1992] 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.

[Partee *et al.*, 1990] Barbara Partee, Alice ter Meulen, and Robert E. Wall. *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, 1990.

[Perlmutter, 1978] David Perlmutter. Impersonal passives and the unaccusative hypothesis. In *Proceedings of the Fourth Annual Meeting of the Berkeley Linguistics Society*, 1978.

[Pollard and Sag, 1994] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. CSLI, 1994.

[Pollock, 1989] J-Y. Pollock. Verb Movement, UG, and the Structure of IP. *Linguistic Inquiry*, 20.3:365–424, 1989.

[Potsdam, 1997] Eric Potsdam. *Syntactic issues in the English imperative*. PhD thesis, University of California at Santa Cruz, 1997.

[Prasad and Sarkar, 2000] Rashmi Prasad and Anoop Sarkar. Comparing test-suite based evaluation and corpus-based evalation of a wide-coverage grammar for english. In *Proceedings, Workshop on Using Evaluation within HLT Programs: Results and Trends LREC 2000, Athens, Greece*, 2000.

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

[Ramshaw and Marcus, 1995] Lance Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*, MIT, Cambridge, Boston, 1995.

[Rizzi, 1990] Luigi Rizzi. *Relativized Minimality.* MIT Press, Cambridge, Massachusetts; London, England, 1990.

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

[Rosen, 1981] Carol Rosen. *The Relational Structure of Reflexive Clauses: Evidence from Italian.* PhD thesis, Harvard, 1981.

[Rosenbaum, 1967] Peter S. Rosenbaum. *The grammar of English predicate complement constructions.* MIT press, Cambridge, Massachusetts, 1967.

[Sag *et al.*, 1985] I. Sag, G. Gazdar, T. Wasow, and S. Weisler. Coordination and How to distinguish categories. *Natural Language and Linguistic Theory*, 3:117–171, 1985.

[Sarkar and Joshi, 1996] 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.

[Schabes and Joshi, 1988] 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.

[Schabes *et al.*, 1988] Yves Schabes, Anne Abeillé, and Aravind K. Joshi. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August 1988.

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

[Seuss, 1971] Dr. Seuss. *The Lorax.* Random House, New York, New York, 1971.

[Soong and Huang, 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.

[Sornicola, 1988] R. Sornicola. IT-clefts and WH-clefts: two awkward sentence types. *Journal of Linguistics*, 24:343–379, 1988.

[Srinivas *et al.*, 1994] B. Srinivas, D. Egedi, C. Doran, and T. Becker. Lexicalization and grammar development. In *Proceedings of KONVENS '94*, pages 310–9, Vienna, Austria, 1994.

[Srinivas *et al.*, 1995] B. Srinivas, Christine Doran, and Seth Kulick. Heuristics and parse ranking. In *Proceedings of the* 4^{th} *Annual International Workshop on Parsing Technologies*, Prague, September 1995.

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

[Srinivas, 1997b] B. Srinivas. Performance Evaluation of Supertagging for Partial Parsing. In *Proceedings of Fifth International Workshop on Parsing Technology*, Boston, USA, September 1997.

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

[Vijay-Shanker and Schabes, 1992] 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.

[Vijay-Shanker, 1987] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1987.

[Webber *et al.*, 1999] Bonnie Webber, Aravind K. Joshi, Alistair Knott, and Matthew Stone. What are little texts made of? a structural and presuppositional account using lexicalized tag. In *Proceedings of the International Workshop on Levels of Representation in Discourse (LORID '99)*, pages 151–156, Edinburgh, 1999.

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