Technical Report

UCAM-CL-TR-162
ISSN 1476-2986

Number 162

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

The Alvey natural language tools grammar (2nd Release)

Claire Grover, Ted Briscoe, John Carroll, Bran Boguraev

April 1989

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

© 1989 Claire Grover, Ted Briscoe, John Carroll,
Bran Boguraev

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

The Alvey Natural Language Tools Grammar (2nd Release)

Claire Grover

Department of Linguistics, University of Lancaster
Bailrigg, Lancaster, LA1 4YT, UK.

Ted Briscoe, John Carroll, Bran Boguraev

Computer Laboratory, University of Cambridge
Pembroke Street, Cambridge, CB2 3QG, UK.

April 1989

The ANLT grammar is a wide-coverage syntactic description of English expressed in a computationally-tractable unification-based formalism. We describe the formalism and give a detailed account of the analyses adopted for different English syntactic constructions in the current version of the grammar. Appendices provide a complete listing of the grammar, sample lexical entries, and a corpus of parsable sentences. The grammar is fully compatible with the Grammar Development Environment (Technical Report 127) which provides an integrated software environment, supporting automated parsing, generation, and modification of grammars expressed in the formalism described here.[1]

Natural Language Tools Distribution Arrangements

The complete ANLT system, consisting of a morphological analyser and lexicon system, a parser, grammar development software, word and sentence grammars for English, and an English lexicon containing approximately 35000 entries, is distributed at nominal cost. For further details write to:

Ms. N. Honeyman
Edinburgh Artificial Intelligence Applications Institute
80 Southbridge, Edinburgh, EH1 1HN, UK.

[1] This work was supported by research grant GR/D/05554 from the U.K. Science and Engineering Research Council under the Alvey Information Technology initiative.

Contents

1. Overview 1

2. The Grammar Formalism 1
- 2.1 Rule Types 2
  - 2.1.1 Feature Declarations 2
  - 2.1.2 Feature Set Declarations 3
  - 2.1.3 Alias Declarations 3
  - 2.1.4 Category Declarations 3
  - 2.1.5 Extension Declaration 3
  - 2.1.6 Immediate Dominance Rules 4
  - 2.1.7 Linear Precedence Rules 4
  - 2.1.8 Phrase-Structure Rules 4
  - 2.1.9 Feature Propagation Rules 5
  - 2.1.10 Feature Default Rules 5
  - 2.1.11 Metarules 5
- 2.2 Metagrammar Interpretation 6

3. The Feature Make-up of Categories 7
- 3.1 Major Categories 7
  - 3.1.1 Verbal Categories 8
  - 3.1.2 Nominal Categories 11
  - 3.1.3 Prepositional Categories 14
  - 3.1.4 Adjectival Categories 15
- 3.2 Minor Categories 17
- 3.3 The Values of SLASH and AGR 18
- 3.4 Extension Features 19
- 3.5 Other Features 19

4. Feature Propagation 20
- 4.1 Capturing the HFC 20
- 4.2 Capturing the FFP 20
- 4.3 Capturing the CAP 21
- 4.4 Other Propagations 21
- 4.5 Feature Defaults 22

5. Category Structure 23
- 5.1 The Structure of NP 23
  - 5.1.1 The Top Level 23
  - 5.1.2 Modifiers 23
  - 5.1.3 Complements 24
  - 5.1.4 Possessive NP 24
  - 5.1.5 Partitives 25
  - 5.1.6 Other NP Types 25
  - 5.1.7 Compounds 26
- 5.2 The Structure of PP 26
- 5.3 The Structure of AP 27
- 5.4 The Structure of VP 27
  - 5.4.1 Modifiers 27
  - 5.4.2 Complements 28
  - 5.4.3 Auxiliaries 28
  - 5.4.4 Phrasal & Prepositional Verbs 28
- 5.5 The Structure of S 29

6. Constructions 30
6.1 Passive 30
6.2 Subject-Auxiliary Inversion 31
6.3 Extraposition 31
6.4 Agreement and Control 32
6.5 Unbounded Dependencies 34
6.5.1 Topicalisation 35
6.5.2 Wh-questions 35
6.5.3 Relative Clauses 37
6.5.4 Tough Movement 39
6.5.5 It-Clefts 39
6.6 Comparatives 39
6.7 Coordination 40
6.8 Negation 41

7. Limitations of the Grammar 42

8. References 45

Appendix 1. Grammar Listing i

Appendix 2. Sample Lexical Entries xli

Appendix 3. Sample Test Sentences xlvi

1. Overview

This report describes one component of the Alvey Natural Language Tools (ANLT) initiative to develop a general-purpose wide-coverage morphological and syntactic analyser for English. The ANLT system comprises a morphology and lexicon system (Pulman et al. 1988), a chart parser for unification-based formalisms (Phillips & Thompson 1987), word and sentence grammars of English, and a substantial English lexicon (Carroll & Grover 1989). The ANLT system has been integrated with a software environment designed to facilitate rapid development of substantial, wide-coverage grammars (hereafter the Grammar Development Environment (GDE), Carroll et al. 1988). This report describes the latest version of the ANLT sentence grammar produced using the GDE. A previous version of the grammar is described in Grover et al. (1988) which is superseded by this document. The ANLT sentence grammar is written in a formalism for grammar rules which supports a notation similar to that of Generalized Phrase Structure Grammar (GPSG, Gazdar, Klein, Pullum and Sag 1985, henceforth GPSG85). The analyses adopted in the ANLT grammar are based on those proposed in GPSG85. This document presupposes some familiarity with GPSG and users/readers not familiar with the theory are recommended to consult Sells (1986) and GPSG85.

The body of this report provides a general description of the grammar. Detailed explanations of specific rules may be found in the comments associated with each rule in the grammar listing in Appendix 1. Throughout the paper reference will be made to specific rules via their rule names and these are identical to those used in the grammar listing. Sometimes, blocks of rules will be referred to; for example, the block of ID rules 'S/THAT1 – S/AS' and these correspond to blocks in the grammar listing.

Section 2 provides an introduction to the different rule types employed in the grammar. Section 3 describes the feature system, explaining which features appear on which categories and what purpose they serve. Section 4 outlines the way in which we achieve the kinds of feature propagation which, in GPSG, are dealt with by means of feature propagation principles such as the Head Feature Convention. Section 5 describes the basic constituent structure of the major category types and Section 6 describes how we deal with various constructions such as unbounded dependency constructions, coordination, comparatives etc. Section 7 discusses the limitations in coverage of the grammar. There are three appendices: Appendix 1 contains the grammar listing, Appendix 2 gives some sample lexical entries and Appendix 3 contains some of the sentences which we have used to test the grammar.

2. The Grammar Formalism

The grammatical formalism employed is a metagrammatical notation which induces an 'object' grammar. This notation is based on GPSG85, but has been modified and extended to be more flexible and expressive and is interpreted somewhat differently. The motivation for these changes is to provide a formalism which provides a specialised programming language for specifying grammatical theories and grammars for particular languages, rather than defining a (restrictive) theory directly (see Briscoe et al. 1987a).

The object grammar consists of a set of phrase structure rules whose categories are feature complexes, and which form the input to the parser. Rules are input to the GDE in a form such as in (1) and these rules are expanded out into object grammar rules as in (2).

```
(1) IDRULE VP/NP : VP --> H[SUBCAT NP], N2[-PRD].

(2) [N -, V +, BAR 2, SUBJ -, VFORM @12, FIN @13, PAST @14,
       PRD @15, AUX -, AGR [N +, V -, BAR 2, NFORM NORM,
       PER @23, PLU @24, COUNT @25, CASE @26], NEG @20,
       COORD @49] -->
    [N -, V +, BAR 0, H +, VFORM @12, FIN @13, PAST @14,
       PRD @15, AUX -, AGR [N +, V -, BAR 2, NFORM NORM,
       PER @23, PLU @24, COUNT @25, CASE @26], INV @18,
       PSVE -, NEG @20, SUBCAT NP, SUBTYPE @69],
    [N +, V -, BAR 2, PRD -, NEG -, NFORM NORM, PER @82, PLU @74,
       COUNT @80, CASE ACC, PN @27, PRO @28, POSS @30, DEF @31,
       SPEC @32, AFORM @36, NUM @39, COORD @86, REFL @50].
```

Features consist of feature name/feature value pairs. Feature values can be variables (introduced by @, as in (2)), which can be bound within the rule. Parsing with the object grammar involves matching categories by unifying their feature sets. Unlike GPSG, our metagrammar defines a set of partially instantiated phrase structure rules and not a set of fully instantiated local trees. Our definition of what it is for two categories to 'match' or unify differs from the GPSG definition. In GPSG categories match by 'extension': that is, a category which is less specified in terms of the number of features it contains will match a category which is more specified providing that the feature values of their common features do not conflict. In our grammar we have chosen to define matching in terms of fixed-arity term unification whereby for two categories to match we require that they each have exactly the same features with non-conflicting values for those features. To give an example, the unification indicated in (3) is legitimate in GPSG, but not in our formalism. For us, the second category has to have the same number of features as the first category, so a legitimate unification must be as in (4).

```
(3)   [N +, V -, PLU +, PER 3]    ∪      [N +, V -]

                        =   [N +, V -, PLU +, PER 3]

(4)   [N +, V -, PLU +, PER 3]    ∪      [N +, V -, PLU @1, PER @2]

                        =   [N +, V -, PLU +, PER 3]
```

Thus, for each different category type, we define the full set of features that it must have. The Category rules described in Section 2.1.4 are designed to 'flesh out,' in the course of expanding the grammar, a category mentioned in a rule with the full set of features that are appropriate to it. (See Briscoe et al. 1987b for further discussion of the use of fixed-arity unification.)

2.1 Rule Types

The metagrammar is designed to capture linguistic generalisations and so simplify and abbreviate the statement of 'object' rules, such as (2), and contains rules of the following eleven kinds:

2.1.1 Feature Declarations

Feature Declarations define the feature system used by the grammar. They encode the possible values of a given feature. The feature system is very similar to that used in GPSG, but a feature can additionally take a variable value which ranges over the set of actual values as declared. (5a) and (5b) are atomic-valued features. (5c) is category-valued.

```
(5) a. FEATURE PLU{+, -}
    b. FEATURE VFORM{BSE, ING, EN, TO, NOT}
    c. FEATURE SLASH CAT
```

2.1.2 Feature Set Declarations

Feature Set Declarations define sets of features which propagate in the same manner and which will appear together on particular categories. For example, features which propagate between NPs and their head daughters (PLU, PER, etc.) may be grouped together in a set called NOMINALHEAD, as in (6). The name of this set may then be used to refer to this collection of features in the rules which perform this type of propagation.

```
(6) SET NOMINALHEAD = {PLU, POSS, CASE, PRD, PN, PRO,
                        COUNT, NFORM, PER, REFL, NUM}
```

2.1.3 Alias Declarations

Aliases are a convenient abbreviatory device for naming categories and feature complexes in rules. They do not affect the expressive power of the formalism.

```
(7) a. ALIAS N2 = [N +, V -, BAR 2].
    b. ALIAS +NOM = [CASE NOM].
```

2.1.4 Category Declarations

Category Declarations define a particular category as consisting of a given set of features. These declarations are used to expand out the partiallly specified categories which typically occur in the specification of immediate dominance or phrase structure rules. They make the formalism more explicit by obliging the grammarian to state which features will appear on a given category. Category Declarations replace part of the function of Feature Cooccurrence Restrictions in GPSG. The category declaration in (8a) causes all nominal categories in immediate dominance and phrase structure rules to be expanded with whichever features in the set NOMINALHEAD they are not already specified with. These features will be given variable values.

```
(8) a. CATEGORY NOUN : [N +, V -] => NOMINALHEAD
```

LCategory declarations function in the same way but they apply only to lexical categories and are utilised by the morphology system as well. For example, the lcategory rule in (8b) causes the features POSS, DEF and AGR to be added to all determiner categories in rules and, when used by the morphology system, causes these features to be added to all lexical entries for determiners.

```
(8) b. LCATEGORY W_DETN : [QUA +, SUBCAT DETN] => {DEF, POSS, AGR}.
```

2.1.5 Extension Declaration

Some features, such as SLASH, are not part of the 'basic' make up of a category and in this system are added to categories in rules by the application of metarules. Features which appear on categories only by virtue of metarule application are defined as extension features. Extension declarations serve the same function as category declarations in terms of making the feature and category structure of the grammar more explicit. However, their interpretation at 'compile time' when the 'object' grammar is produced is different, since categories are not automatically 'fleshed out' with extension features.

```
(9) EXTENSION {WH, UB, EVER, SLASH, NULL}
```

2.1.6 Immediate Dominance Rules

Immediate dominance (ID) rules encode permissible dominance relations in phrase structure trees. The immediate dominance properties of the 'object' rule in (2) above can be expressed by the ID rule in (1) (reproduced as (10a) below); however, other properties of the object rule (e.g. the ordering of the categories in it) are determined by other types of rules in the grammar. (10a) states that a transitive VP will contain a lexical head, which must be subcategorised appropriately, and a NP sister. The ID rule in (10b) states that a finite, non-inverted S is made up of a nominative NP, a VP head which agrees with an NP, and an optional adverbial phrase modifier.

```
(10) a. IDRULE VP/NP : VP --> H[SUBCAT NP], N2[-PRD].

     b. IDRULE S1 : S[COMP NORM, -INV, +FIN] -->
                        N2[+NOM], ( A2[+ADV] ), H2[-SUBJ, AGR N2].
```

2.1.7 Linear Precedence Rules

Linear Precedence (LP) rules encode permissible precedence relations in ID rules. The rule (11a) states that categories with both a SUBCAT feature and a BAR feature (major category lexical heads) precede their sister categories which do not have a SUBCAT feature. (11b) states that +N categories (NP or AP) always precede PPs and non-head V2s (VP or S) and that PPs always precede non-head V2s in ID rules which contain any of these categories on the right-hand side. The two LP rules in (11) are slightly more complex versions of two of the three LP rules in GPSG85.

```
(11) a. LPRULE LP1 : [SUBCAT, BAR] < [~SUBCAT].
     b. LPRULE LP9 : [N +] < P2 < V2[~H].
```

2.1.8 Phrase-Structure Rules

Phrase-structure (PS) rules encode permissible dominance and precedence relations. They are included in the formalism so that the expressive power of the system is not restricted by the Exhaustive Constant Partial Ordering (ECPO) property. (A grammar has the ECPO property if the set of expansions, defined by ID rules, of any one category observes a partial ordering that is also observed by the expansions of all the other categories, see GPSG85). The grammarian may choose not to use PS rules, or may choose to abandon the ID/LP format entirely in favour of PS rules, or may choose to use a mixed system. Our grammar uses ID/LP format predominantly but it does contain a few 'marked' PS rules, for example, some rules for cardinal numbers (see PS rules NUM1 – NUM4) and some rules for names (PS rules N/NAME1 – N/NAME2B), one of which is reproduced in (12):

```
(12) PSRULE N/NAME2B : N[-PLU, PN -, -POSS] -->
               N[+PN, SUBCAT NULL] N[ADDRESS +, SUBCAT NULL].
```

2.1.9 Feature Propagation Rules

Feature Propagation Rules define how features propagate between mother and daughter categories in ID or PS rules. The effect of propagation rules is to bind variables or instantiate values of features in rules of the 'object' grammar. Propagation rules can be used to encode particular feature propagation principles, such as the various versions of the Head Feature Convention proposed for GPSG. However, such principles are not 'hard-wired' into the formalism, so that maximum flexibility and expressiveness is maintained (see Briscoe et al. 1987a,b for further discussion). The propagation rule in (13) propagates nominal head features between a nominal mother and its head daughter. It is stated in terms of an ID rule pattern which prospective input ID rules must match. The pattern includes a variable over categories (U in this case, but W is used where the input must be a lexical ID rule (i.e. one with a lexical head)) and thereby specifies that the input must be any ID rule which has a nominal mother, contains a head and optionally contains other daughters of any type. Following the pattern is a statement of feature bindings. The terms 0 and 1 refer to categories in the input rule where 0 is the mother, 1 is the first daughter named by the pattern, and 2 would be the second daughter named by the pattern etc. The rule, then, is stating that the values of the features defined by the set NOMINALHEAD on the mother must be bound to the values of the same features on the head daughter.

```
(13) PROPRULE HFC_NOMINAL :
          [N +, V -] --> [H +], U. F(0) = F(1), F in NOMINALHEAD.
```

Propagation rules can also be used to bind feature values on categories which are the values of category-valued features, as in the agreement rule below, where 'F(2[AGR])' is to be read as 'features on the category which is the value of 2's AGR feature'.

```
(14) PROPRULE AGR/NP_VP : S --> N2, H2[-SUBJ, AGR N2], U.
                              F(1) = F(2[AGR]), F in AGRFEATS.
```

2.1.10 Feature Default Rules

Feature Default Rules default specified values onto features with no value in categories in a particular environment in an ID or PS rule. These default rules replace Feature Specification Defaults in GPSG; because the rules assign values to features in the context of an ID or PS rule, their application can be accurately controlled, and thus the need for Feature Cooccurrence Restrictions to prevent the construction of 'illegal' categories is diminished (see Briscoe et al. 1987b). Both Feature Default rules and Feature Propagation rules have a similar syntax to Kilbury's (1986) independently motivated Category Cooccurrence Restrictions, although their function is somewhat different. Rule (15a) states that a NP daughter of P1 will be accusative (if unspecified for any other CASE value). (15b) states that a VP mother will have the category N2[NFORM NORM] as the value for its AGR feature unless it has already been otherwise specified.

```
(15) a. DEFRULE RHS_N2_CASE : P1 --> N2, U. CASE(1) = ACC.
     b. DEFRULE VP/AGR : VP --> W. AGR(0) = N2[NFORM NORM].
```

2.1.11 Metarules

Metarules encode systematic relationships between sets of ID or PS rules and will automatically add further rules to the basic set produced by the grammar writer and the derived set produced by the application of other metarules. Metarules can be written to apply to ID rules or PS rules (including PS rules which result from linearisation of ID rules). Metarules can be restricted to apply only to ID rules containing a lexical head through use of the W category variable (as opposed to the U 'unrestricted' category variable) in the input pattern. For example, the metarule in (16a) (a simplified version of the actual passive metarule) adds a new VP rule where the VP is marked as [+PRD, VFORM EN] (= passive) and where the NP direct object is replaced by an optional 'by' phrase, for every basic VP rule with a lexical head, a non-predicative NP daughter and normal agreement properties. The metarule in (16b) creates new versions of input lexical ID rules where a SLASH feature appears on a [V +] non-head daughter with a variable value which is bound to the SLASH feature of the mother.

```
(16)
a. METARULE PASSIVE :
   VP[AGR N2[NFORM NORM]] -->
                H[AGR N2[NFORM NORM]], N2[-PRD, NFORM NORM], W.
                            ==>
   VP[VFORM EN, +PRD, AGR N2[NFORM NORM]] -->
                H[AGR N2[NFORM NORM]], ( P2[PFORM BY] ), W.

b. METARULE SLASHBAR0A_1 :
      X[~WH, ~INV, ~SLASH] --> H, X2[V +, ~UB, ~SLASH, AGR N2], W.
         ==>
      X[SLASH @x] -->  H, X2[SLASH @x], W.
```

2.2 Metagrammar Interpretation

In GPSG, the set of legal local trees is defined declaratively by simultaneous application of the various rule types during the 'projection' from ID rules to fully instatiated local trees. In this system, although the metagrammatical notation shares many similarities with GPSG, there are crucial differences in the interpretation of the notation. Metagrammatical rules jointly specify a set of partially instantiated phrase structure rules, rather than fully instantiated local trees. That is, rules are allowed to contain variable values for features and these variables are instantiated at parse time by unification. Propagation rules specify restrictions on the instantiation of these variables; for example, the expanded rule for NPs introducing determiners and nominal phrases will force agreement between the determiner and nominal phrase by binding the PLU variables on all the categories as illustrated in (17) (irrelevant features excluded).

```
(17) N2[SPEC +, PLU @34] -->
          DetN[AGR N2[PLU @34]], N2[SPEC -, PLU @34, H +].
```

If we assume that *a* is specified as PLU – in its AGR value whilst *the* is unspecified for a PLU value, then the result of matching *the* to the Det category in (17) will not instantiate @34. However, matching *boys* to N1 will instantiate all the instances of @34 to +. Therefore, * *a boys* will not be accepted by (17) because @34 cannot both be instantiated as + and –.

The object grammar is produced by applying category, propagation and default rules, in that order, to ID (and PS) rules and then applying the non-linear metarules one-by-one in a predefined order to the set of fleshed out ID rules. After each metarule has applied, propagation and default rules are applied to any new ID rules and these are added to the original set before the application of the next metarule. Next, the resulting expanded set of rules is linearised according to the LP rules and any linear metarules are applied to the complete pool of PS rules. This procedure is summarised in Figure 1.

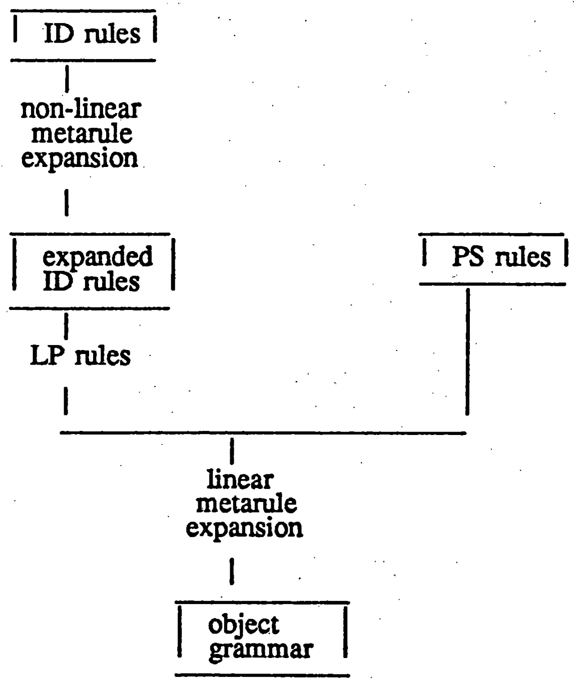


Figure 1: Rule Application

3. The Feature Make-up of Categories

3.1 Major Categories

Major categories are ones which are defined in terms of the features N, V and BAR. If a category has the feature BAR, it must also have the features N and V and similarly, if it has the features N and V, it must also have BAR. N and V combine to define the four major category types: verbal categories are [V +, N –], nominal categories are [N +, V –], adjectival categories are [V +, N +] and prepositional categories are [V –, N –]. The feature BAR distinguishes lexical categories ([BAR 0]) from phrasal categories ([BAR 2]). [BAR 1] allows us to utilise one distinct category which is intermediate between lexical and phrasal. For each major category type there is a set of head features which must appear on all instances of that category type, regardless of their BAR feature value. In addition there are other features which must appear on only some instances of a category type depending on their BAR feature value or some other feature value. The tables below indicate the distribution of features in categories and some explanation of the purpose of the features is given.

3.1.1 Verbal Categories

```
         S                         VP                          V

[V +, N -, BAR 2, SUBJ +] [V +, N -, BAR 2, SUBJ -] [V +, N -, BAR 0]

{VERBALHEAD}              {VERBALHEAD}              {VERBALHEAD}
 COMP                      NEG                      NEG
 INV                       COORD                    INV
 COORD                                              PSVE
 (TAG)                                              SUBCAT
                                                    SUBTYPE
                                                    (PRT)
                                                    (PFORM)
                                                    (PREP)

where set VERBALHEAD = {PRD, FIN, AUX, VFORM, PAST, AGR}
```

VFORM

The feature VFORM (verb form) distinguishes between a number of possible untensed forms as follows:

VFORM BSE : the base or uninflected form of the verb as in:

(18) a. John wants to *sing*
  b. John may *sing*
  c. John wants to *be* an astronaut
  d. They require that John *dance*

VFORM ING : appears on all verbs with the 'ing' suffix, whether present participle (19a) or gerund (19b).

(19) a. John is *reading* the book
  b. John *reading* the book surprised me

VFORM EN : appears on both the past participle and the passive form of the verb, for example:

(20) a. John has *finished* his tea
  b. The mouse was *eaten* by the cat

VFORM TO : this is used to identify infinitive VPs. Following GPSG85, we treat the word *to* occuring in infinitives (*John wants to go*) as a main verb albeit of a rather marked type since it doesn't inflect in any way: *to* is the only verb in the dictionary specified as [VFORM TO] and this feature name-value pair propagates up to the VP dominating it, thus ensuring that the VP immediately containing it is also marked as [VFORM TO].

VFORM NOT : this value appears on all finite, non-imperative verb forms and is to be interpreted as meaning that the verb has no meaningful value for VFORM – the purpose of the VFORM feature is to distinguish between different types of non-finite forms and as such it has no real contribution to make to the specification of finite forms.

FIN

The feature FIN (finite) distinguishes tensed forms of a verb (+FIN) from untensed forms (–FIN). FIN interacts with VFORM in the following way: in general +FIN verbs will be VFORM NOT whilst –FIN verbs will have one of BSE, ING, EN, TO as a VFORM value. The only exception to this are imperatives which are [+FIN, VFORM BSE], following the analysis of Warner (1984).

PRD

The feature PRD (predicative) appears on major categories. When it appears on verbs, it interacts with the VFORM feature to distinguish between passives and past participles, and present participles and gerunds: passive verbs are VFORM EN, +PRD; past participles are VFORM EN, –PRD; present participles are VFORM ING, +PRD; gerunds are VFORM ING, –PRD. The linguistic motivation for this analysis stems from the observation that the verb *be* subcategorises for predicative complements. Passive verbs and present participles are both complements of *be*, therefore they are +PRD. Past participles and gerunds are not complements of *be*, therefore they are –PRD.

AUX

The feature AUX distinguishes auxiliaries from non-auxiliaries: the auxiliaries *be, have, do*, the modals *can, may, ought* etc, and the infinitive verb *to* are all +AUX. All other verbs are –AUX.

PAST

The feature PAST encodes past tense (+PAST) and present tense (–PAST). All finite verbs are either +PAST or –PAST but since tense is not relevant to non-finite forms these are all marked as [PAST NOT] where this is interpreted as meaning that the verb has no meaningful value for PAST.

AGR

AGR is a category-valued feature which encodes information about the category that verbs, adjectives or determiners agree with. The value of AGR is restricted such that only a limited set of features appear as part of the category value and this means that AGR in our grammar is really only 'pseudo' category-valued, unlike GPSG85 where the value of AGR is an exact copy of the category agreed with.

Verbs will be either [AGR N2] (agrees with an N2 subject) or [AGR V2] (agrees with an S or VP subject). These AGR values will be further instantiated:
The N2 in [AGR N2] has additional features defined by the set AGRFEATS (NFORM, PER, PLU, CASE, COUNT) since these are the ones relevant to agreement. The values for these features will be bound to the values for the same features on the subject N2. The NFORM feature is the one which defines whether a verb can agree with a dummy subject (*it, there*) or not.
The V2 in [AGR V2] has the additional features FIN, SUBJ, VFORM and UB. The values for these features will be bound to the values for the same features on the V2 subject. These features define the type of V2 subject (finite S, infinitival S, infinitival VP, base S and question S) that a verb can cooccur with (see ID rules S/V2_SUBJ1 – S/V2_SUBJ6 and propagation rule AGR/V2_VP).

INV

The feature INV distinguishes inverted sentences from uninverted ones. The S rules which are output from the subject-auxiliary inversion (SAI) metarule will be +INV. S and V are specified for INV, whilst VP is not. Non-auxiliary verbs will be specified as –INV since they cannot invert. Auxiliaries have a variable value for INV to allow them to appear in both inverted and non-inverted structures. Some, however, are always +INV e.g. the 1st person singular entry for *aren't* ([+INV, AGR N2[PER 1, PLU –]]) as in *aren't I clever*.

SUBJ

The feature SUBJ occurs on S and VP and serves to distinguish one from the other: both are [V +, N –, BAR 2] but S is +SUBJ and VP is –SUBJ. This analysis is the same as the GPSG85 one and originates with Borsley (1983).

NEG

The feature NEG (negative) occurs on V and VP but not on S. Most lexical entries for verbs are –NEG but verbs with the 'n't' suffix are +NEG. VPs will be +NEG either by virtue of containing a +NEG verb or because they are modified by *not*. Only –FIN VPs can be modified by *not*.

COMP

The COMP feature occurs on S and encodes the type of complementiser an S contains. For example, an S[COMP THAT] contains a *that* complementiser, an S[COMP WHETHER] contains a *whether* complementiser etc. An S with no complementiser is [COMP NORM].

PSVE

All verbs have an extra feature PSVE (passive). This feature is necessary since there is nothing in the grammar to prevent passive verbs from matching active VP rules. (Passive verbs are distinguished by the feature specification [VFORM EN, PRD +] but since active VP rules are left unspecified for VFORM and PRD, these features cannot be used to prevent passive verbs from matching.) All passive verbs will therefore be specified as [PSVE +] and a default rule (V/PASS) will ensure that the lexical head of all passive VP rules is similarly specified. All active verbs will be marked as [PSVE –] and this feature specification will again be defaulted onto the heads of all active VP rules (default rule V/UNPASS). Without this feature **fido is attacked the dog* would get a successful parse.

COORD

The feature COORD (coordinated) occurs on all [BAR 2] categories and therefore appears on S and VP. The top node of a coordination of S or VP will be +COORD whilst any occurrence of a non-coordinated S or VP will be –COORD.

TAG

The feature TAG is used to mark the tag in a tag-question (i.e. *isn't he* in *he is clever, isn't he*). We analyse the tag as a rather marked type of S (see ID rule S/TAG) and distinguish it as [TAG +]. No other S has the feature TAG.

SUBTYPE

All verbs are are marked both for the feature SUBCAT and the feature SUBTYPE. SUBCAT describes the type of complement that a verb may subcategorise for and SUBTYPE provides further information about that particular subcategorisation. For example, the verbs *tell* and *bother* can both take a sentential complement (*he tells me that it will rain* and *it bothers me that it will rain*) and both have the SUBCAT value NP_SFIN. They are, however, very different from one another – *tell* has normal agreement properties whilst *bother* agrees with the dummy NP *it* and, furthermore, it is an 'extraposition' verb which means it can also occur with a sentential subject as in *that it rains bothers me*. To distinguish these two verbs, we use the feature SUBTYPE with values NONE and EXTRAP respectively and this feature is picked up by the appropriate ID rules (see VP/NP_SFIN1, VP/NP_SFIN2A and VP/NP_SFIN2B). Often SUBTYPE doesn't really have a role to play since not all SUBCAT types have variants – in this case the default SUBTYPE NONE is assumed.

PRT

The feature PRT is used to describe particles but it also occurs on verbs which need to cooccur with a particle ('phrasal verbs'). When it does occur on a verb, it describes which particle the verb is to occur with. For example, the verb *blow up* will have an entry under the head word *blow* and this entry will contain the feature specification [PRT UP], to ensure (a) that it is recognised as a phrasal verb and (b) that it only occurs with the particle *up*. In rules which introduce phrasal verbs (VP/*PHR) the value for PRT on the verb is bound to the value for PRT on the particle. Only phrasal verbs have a specification for PRT.

PFORM

PFORM is a feature which describes prepositions but it also occurs on verbs which subcategorise for a PP complement of a particular type. Its function on verbs is to ensure that they occur with the right preposition, for example, one subcategorisation for the verb *agree* is for a PP[PFORM WITH] (*I agree with you*), so the entry for this use of *agree* will have the feature specification [PFORM WITH]. The rules which introduce PP complements bind the value of PFORM on the verb to the value of PFORM on the PP to ensure correct pairings. Only verbs which subcategorise for a PP have a specification for PFORM.

PREP

Some verbs occur with a free-floating preposition (one that is not part of a PP) such as *on* in *I was banking on seeing him* . The feature PREP encodes the nature of this preposition and functions in exactly the same way as the feature PFORM on verbs. Only verbs which subcategorise for this kind of preposition have a specification for PREP.

3.1.2 Nominal Categories

<u>N2</u>	<u>N1</u>	<u>N</u>
[N +, V -, BAR 2]	[N +, V -, BAR 1]	[N +, V -, BAR 0]
{NOMINALHEAD}	{NOMINALHEAD}	{NOMINALHEAD}
SPEC	MOD	DEF (only on +PRO)
DEF	AFORM	SUBCAT
NEG		(PART)
AFORM		(ADDRESS)
COORD		(PRT)
(PART)		

```
where set NOMINALHEAD = {PLU, POSS, CASE, PRD, PN, PRO, COUNT,
                         NFORM, PER, REFL, NUM}
```

NFORM

This feature distinguishes 'dummy' forms (*it, there*) from normal nouns; pleonastic *it* is specified as [NFORM IT], existential *there* is specified as [NFORM THERE], whilst all other nouns, pronouns and proper names are specified as [NFORM NORM]. NFORM is relevant to agreement since not all verbs and adjectives can have N2 subjects of all types.

PER

This feature indicates whether a noun, pronoun or proper name is 1st, 2nd or 3rd person.

PLU

This feature indicates whether a noun is singular (–PLU) or plural (+PLU).

CASE

CASE indicates case marking and has two values, NOM (nominative) and ACC (accusative). Most nouns are unspecified for CASE since they don't inflect for case. Pronouns, however, are case-marked and so these do have a specification for CASE.

PN

The feature PN (proper name) distinguishes proper names (+PN) from other nouns (–PN).

PRO

The feature PRO distinguishes pronouns (+PRO) from other nouns (–PRO).

COUNT

The feature COUNT distinguishes mass nouns (–COUNT) from countable nouns (+COUNT).

POSS

The feature POSS encodes possessiveness. All noun entries are –POSS except for the entry for the possessive morpheme *s* which is +POSS. The grammar contains a special rule for building a possessive N2 out of a non-possessive N2 and the possessive morpheme. Currently the morphological analyser will not separate the possessive morpheme from the word it attaches to, so a possessive N2 has to be entered to the parser with a space between the N2 and the possessive morpheme (i.e. instead of *the cat's* one must type *the cat 's*).

PRD

All nouns currently have variable values for PRD (predicative), although during the parse N2s in which they appear may be instantiated to +PRD by virtue of rules that licence them, for example, the rule for copula *be* followed by an N2.

REFL

REFL indicates whether a noun has reflexive morphology ([REFL +] – *myself, himself*) or not ([REFL –]). The default is [REFL –].

NUM

The feature NUM (values CARD, ORD and –) is used to encode whether a noun is a cardinal or ordinal number or neither.

SPEC

The feature SPEC (specifier) appears on N2 but not on N1 or N, thereby creating two distinct types of N2 (N2[+SPEC] and N2[–SPEC]). The distinction between the two types of N2 was introduced because of the need to recognise more levels of structure while keeping N2 as the top node of an NP. Either N2[+SPEC] or N2[–SPEC] can appear in N2 positions which are unspecified for SPEC.

DEF

This feature encodes definiteness. DEF appears on N2 but not on N1. An N2 acquires its DEF value from the DEF value of the determiner which appears in it. Pronouns can be either definite or indefinite (e.g. *some* is indefinite whilst *all* is definite) and since we treat pronouns as nouns (and not N2s) we have to introduce the feature DEF onto pronoun entries (N[+PRO] entries). It does not occur on other N entries.

NEG

NEG appears on N2 but not on N1 or N. A class of negative N2s (ones modified by *not*) is recognised in the grammar but their distribution is restricted such that they only appear in coordinations of N2 (e.g. *a pencil but not a pen*). The *not* occurring with copula 'be' and an N2 (e.g. *fido is not a cat*) is not treated as being in constituency with the N2.

AFORM

The feature AFORM (adjective form) encodes information about whether an adjective is comparative, equative, superlative or none of these. The feature has to appear on N1 and N2 as well as on adjectival categories since the rules for comparatives need to know whether an N2 contains a comparative adjective or not. For example, in *Lee ate more chocolates than Kim ate*, whilst it is the adjective *more* which is marked as comparative, it is the N2 it is contained in (*more chocolates*) which actually participates in the comparative structure.

COORD

The feature COORD (coordinated) occurs on all [BAR 2] categories and therefore appears on N2. The top node of a coordination of N2s will be +COORD whilst any occurrence of a non-coordinated N2 will be –COORD. This feature helps to prevent overgeneration in cases of coordination of N2 after copula 'be' : there are rules to allow any +PRD category to be coordinated and for these coordinations to appear after *be*. Since N2s can be +PRD categories an example such as *fido is either a dog or a cat* would receive two parses, one containing a coordination of N2s and one containing a coordination of +PRD categories both of which happened to be N2. To prevent this happening the N2 following *be* is marked as –COORD thus ensuring that any coordination of N2s after *be* is recognised only as a coordination of +PRD categories.

PART

The grammar contains an analysis of the partive construction (*some/all/many of the books*etc.) which allows for a +PRO NP to appear in the position preceding *of*. PART occurs on pronouns which head these constructions (e.g. *all, some, any, many*etc.) and indicates whether that pronoun requires the *of* to occur ([PART OF] and [PART OF2] – *some of the books, * some the books*) or not ([PART NO_OF] – *all of the books, all the books*). The value OF appears on those pronouns which both need to take *of* and need to agree with the following NP (*some of the book is .., some of the books are ..*) while the value OF2 appears on those which need an *of* but which mustn't agree with the following NP (*each of the books is/ *are*).

MOD

The feature MOD (modified) has values POST, PRE and NONE and appears only on N1s. It is used to control the order in which modifiers attach to N1 (pre-modifiers attach lower than post-modifiers). It is simply a device to prevent multiple parses reflecting syntactically spurious attachment ambiguities in NPs which contain both pre- and post-modifiers.

ADDRESS

[ADDRESS +] ooccurs on nouns which are titles such as *Mr, Mrs, Sir, esq* etc. This feature does not occur on any other noun.

PRT

PRT occurs on some nouns which occur with particles, for example, *a rummage about*. It functions in exactly the same way as it functions when it appears on verbs.

3.1.3 Prepositional Categories

```
     P2                    P1                     P

[N -, V -, BAR 2]     [N -, V -, BAR 1]     [N -, V -, BAR 0]

{PREPHEAD}            {PREPHEAD}            {PREPHEAD}
 GERUND                GERUND                NEG
 POSS                  POSS                  SUBCAT
 NEG
 COORD

where set PREPHEAD = {PFORM, LOC, PRD, PRO}
```

PFORM

The feature PFORM (prepositional form) encodes the type of preposition appearing in a P2. For example, the preposition *to* will be [PFORM TO] and this information will pass from the preposition to the P2 mother. This will enable particular prepositions to be subcategorised for: for example, *give* subcategorises for a P2[PFORM TO]. There is one value for PFORM which is not the name of a preposition – NORM. This value is used for PPs which occur in modifying positions where the particular type of preposition is irrelevant. For this reason there are at least two lexical entries for most prepositions, one which reflects which preposition it is (e.g. [PFORM TO]) and one which characterises it as [PFORM NORM]. In this way we capture the distinction between case-marking prepositions and contentful ones.

LOC

The feature LOC distinguishes between locative and non-locative Ps and PPs. It is necessary because some lexical heads subcategorise for +LOC PPs, for example *put*, which can be introduced by a range of distinct prepositions (*on/under/in*, etc.).

PRD

As with nouns above, PRD (predicative) appears on all prepositional categories, but all prepositions currently have only a variable value for PRD. **PRO**

The feature PRO identifies prepositional pro-forms such as *where, there, now, then, when* etc. These are marked as +PRO and dealt with by a special rule to build a pro-PP, whilst all other prepositional categories are marked as –PRO.

POSS

The feature POSS appears on P2 and P1, but not on P. Here it is used to identify possessive PPs such as *of John's* in *a book of John's*. Thus, a P2 or P1 will be +POSS if it contains a possessive N2 and –POSS otherwise.

GERUND

The feature GERUND appears on P2 and P1 but not on P and is used to identify a PP containing a gerundive complement; for example, *about fido's attacking the cat.* A P2 or P1 will be +GERUND if it contains a gerund and –GERUND otherwise.

NEG

NEG appears on P2 and on P but not on P1. A class of negative P2s and Ps (ones modified by *not*) is recognised in the grammar but their distribution is restricted such that they only appear in coordinations of P2 (*on the table but not under it*) and P (*on but not under the table*). The *not* occurring with copula *be* and a P2 (e.g. *fido is not in his kennel*) is not treated as being in constituency with the P2.

COORD

The feature COORD (coordinated) occurs on all [BAR 2] categories and therefore appears on P2. The top node of a coordination of P2s will be +COORD whilst any occurrence of a non-coordinated P2 will be –COORD. This feature helps to prevent overgeneration in cases of coordination of P2s following copula *be* in exactly the same way as described with reference to COORD on NPs above. In addition COORD prevents overgeneration in cases of coordination of P2s in modifier position: there are rules to allow adverbials (either A2[+ADV] or P2) to be coordinated and for these coordinations to appear in positions modifying VP and S. This would mean that an example such as *lee went to the cinema both on Monday and on Wednesday* would receive two parses, one containing a coordination of P2s and one containing a coordination of +ADV categories both of which happened to be P2. To prevent this happening, a P2 in positions modifying VP and S is marked as –COORD thus ensuring that any coordination of P2s in these positions is recognised only as a coordination of +ADV categories.

3.1.4 Adjectival Categories

```
    A2                       A1                         A

[N +, V +, BAR 2]     [N +, V +, BAR 1]         [N +, V +, BAR 0]

{ADJHEAD}              {ADJHEAD}                {ADJHEAD}
 AFORM                  AFORM                    AFORM
 COORD                                           SUBCAT
                                                 SUBTYPE

where set ADJHEAD = {PRD, QUA, ADV, NUM, NEG, AGR, DEF, DISTR}
```

QUA

The feature QUA distinguishes quantifying adjectives (+QUA – *all, many, some, both* etc) from non-quantifying ones (–QUA – *big, stupid* etc. – i.e. most adjectives). Quantifying adjectives are seen as a special case since they occur in positions not available to all adjectives – as specifiers in NPs:

(21) a. all/*big three books
     b. all/*big the books

PRD

The feature PRD (predicative), in combination with the feature DISTR, controls the distribution of adjectives and adjective phrases (APs). In the lexicon, some adjectives must be marked as [PRD –] because they can only head APs which occur in attributive position, for example, *main* (*the main reason, *the reason is main*), and others must be [PRD +], for example, *asleep* (*the man is asleep, *an asleep man*). Quantifying (+QUA) adjectives fall into two classes depending on whether they are + or – PRD: *all* and *some* are A[–PRD, +QUA] whilst *many* and *few* are

A[+PRD, +QUA]. Each type attaches at a different level inside an NP and the –PRD type cannot occur in predicative position.

DISTR

DISTR (distribution), in combination with PRD, controls the distribution of adjective phrases. There are three positions that APs can occur in, prenominal modifer position, postnominal modifier position and predicative positions (as a complement to a verb). It is not the case that all APs can occur in all three positions. APs containing complements or a 'than' clause of comparison cannot occur in prenominal position (**the happy to see you man*), most APs not containing complements or 'than' clauses cannot occur in postnominal position (**the man happy*)), but the presence or absence of complements does not affect the ability of APs to occur in predicative position (*the man is happy to see you, the man is happy*). The grammar reflects these facts by marking the three positions as follows: prenominal – [DISTR ATT, PRD –], postnominal – [DISTR PRD, PRD +] and predicative – [DISTR @, PRD +]. The rules that build APs, with or without complements, are then suitably marked for the features PRD and DISTR so that the resulting APs will only occur in the right places.

ADV

The feature ADV distinguishes adverbs from adjectives. For example, *clever* is an A[–ADV] whilst *cleverly* is an A[+ADV].

NUM

Cardinal numbers can occur as quantifying adjectives (*three books*) and are marked as [NUM CARD] (see ID rule A/NUMBER). Ordinals are treated as ordinary adjectives (*the third book*) except that they are marked as [NUM ORD]. Other adjectives are [NUM –].

NEG

With adjectives, NEG is a head feature since some quantifying adjectives are inherently negative, for example, *no* and *neither* etc. Adjective phrases modified by *not* (*not clever, not easy to please*) are +NEG.

AGR

All adjectives have an AGR specification and most behave exactly like verbs with respect to the AGR feature – they are either [AGR N2] or [AGR V2]. The exception is quantifying adjectives, such as *many* or *few*, which can occur in either predicative or attributive (prenominal) position. When they occur in predicative position they must agree with an N2 but when they occur attributively they must agree with an N1. For this reason the BAR feature on the [N +, V –] category value of AGR on the lexical entries for such adjectives has been left unspecified.

DEF

When quantifying adjectives occur as NP specifiers the DEF value of the NP is derived from their DEF value. For example, *many* in *many books* is indefinite and so is the containing NP, and *both* in *both books* is definite and so is the containing NP.

AFORM

The feature AFORM (adjective form) encodes information about whether an AP is comparative (AFORM ER), equative (AFORM AS), superlative (AFORM EST), or none of these (AFORM NONE). It appears on A2, A1 and A but has not been treated as a head feature, since sometimes the AFORM value of an AP derives from the lexical head (*bigger, biggest*) and sometimes it derives from the DetA (*more stupid, as stupid, most stupid*).

COORD

The feature COORD (coordinated) on all [BAR 2] categories and therefore appears on A2. The top node of a coordination of A2s will be +COORD whilst any occurrence of a non-coordinated A2 will be –COORD. This feature helps to prevent overgeneration in cases of coordination of A2s following copula *be* in exactly the same way as described with reference to COORD on NPs above. In addition COORD prevents overgeneration in cases of coordination of A2s in modifier position: there are rules to allow adverbials (either A2[+ADV] or P2) to be coordinated and for these coordinations to appear in positions modifying VP and S. This would mean that an example such as *lee read the book quickly but carefully* would receive two parses, one containing a coordination of A2s and one containing a coordination of +ADV categories both of which happened to be A2. To prevent this happening, an A2 in positions modifying VP and S is marked as –COORD thus ensuring that any coordination of A2s in these positions is recognised only as a coordination of +ADV categories.

SUBTYPE

This feature operates on adjectives in exactly the same way as it operates on verbs in that it provides extra distinctions between adjectives that have the same SUBCAT value. For example, many adjectives take an infinitival VP complement and are marked as [SUBCAT VPINF], but there are differences between these adjectives – *easy* is marked as [SUBTYPE TOUGH] because it is a 'tough-movement' adjective (its VP complement contains an object gap: *he is easy to beat e*), while *normal* is marked as [SUBTYPE EXTRAP] because it allows extraposition of its VP complement (*to stare wildy at people is not normal, it is not normal to stare wildly at people*).

3.2 Minor Categories

DetN

DetNs (noun determiners) are characterised by the feature set:

[QUA +, SUBCAT DETN].

In addition they have the features DEF, POSS and AGR. The feature DEF propagates from the DetN to its NP mother. For example, *the* is +DEF and *a* is –DEF. The feature POSS identifies possessive determiners, for example, *my*, *her* are +POSS, whilst other DetNs are –POSS. The feature AGR ensures that determiners agree with their nominal sisters. For example, *a* is [AGR N2[PLU –, COUNT +]] and *those* is [AGR N2[PLU +, COUNT +]].

DetA

DetAs (adjective determiners) are characterised by the feature set:

[SUBCAT DETA]

DetAs have the additional feature AFORM which contributes to the AFORM value of the entire AP. For example, *more* is [AFORM ER], *as* is [AFORM AS], *most* is [AFORM EST] and *so* and *too* are [AFORM NONE].

Complementisers

Complementisers are characterised simply by a SUBCAT feature:

that = [SUBCAT THAT]
for = [SUBCAT FOR]
whether = [SUBCAT WHETHER]

if = [SUBCAT IF]

Conjunctions

Conjunctions are characterised by the features:

[CONJN +, SUBCAT (AND, OR, NOR, BOTH, BUT, EITHER, NEITHER)]

For example, *and* is [CONJN +, SUBCAT AND], *both* is [CONJN +, SUBCAT BOTH], etc.

Particles

Particles which appear with phrasal verbs (*turn off, pick up*, etc.) are simply characterised by the feature PRT:

[PRT UP] = the particle in *pick up*
[PRT DOWN] = the particle in *put down*

The *of* which appears in partitives is described by the feature PFORM

[PFORM OF] = partitive *of*

3.3 The Values of SLASH and AGR.

The features AGR and SLASH take categories as values. It seemed unnecessary to require that these categories should have all the features normally associated with them – instead we require them to have a subset of the usual set of features where this subset comprises those features that are relevant to agreement either between subject and verb/adjective in the case of AGR or between the preposed constituent and the gap in the case of SLASH.

AGR

AGR takes either N2 or V2 as a value. When N2 is the value the additional features defined by the set AGRFEATS are required:

SET AGRFEATS = {NFORM, PLU, COUNT, PER, CASE}

When V2 is the value then the additional features SUBJ, FIN, VFORM and UB are required.

SLASH

SLASH takes N2, P2 or A2 as a value. When N2 is the value the additional features defined by the set AGRFEATS are required. When P2 is the value the additional features {PFORM, LOC and GERUND} are required. When A2 is the value the additional feature ADV is required. These values are bound between the preposed category and the first instance of SLASH at the top of the tree (see propagation rules SLASH/AGR_NP – SLASH/AGR_AP). They are also bound again at the point where the SLASH terminates (see propagation rules BIND_SLASH1 – BIND_SLASH4A) thus ensuring that the preposed constituent bears the same feature specification as the 'gap'.

3.4 Extension Features

Extension features are ones which are not associated with any particular category type but which may appear as additional features on any category. Their distribution and propagation is generally controlled by means of metarule, so they seldom appear in basic ID rules.

The features WH, UB and EVER do the job of the GPSG85 WH and WHMOR features. In general, a category will either have all three of WH, UB and EVER or none of them (the exception is embedded S or VP questions occurring as complements of verbs such as *wonder* (e.g. *wonder whether lee is asleep, wonder whether to wake him*) which are [UB Q] but unmarked for WH and EVER). WH expresses 'wh-ness' – the relative pronoun *who* is +WH whilst the relative pronoun *that* is –WH. UB (unbounded) distinguishes questions (Q) and relatives (R). EVER distinguishes free relatives (*whatever you want*) where *whatever* is +EVER. The three features appear on nominal, adjectival and prepositional categories but not on verbal ones (except for S).

The category-valued feature SLASH carries information about missing or empty nodes in trees involving unbounded dependencies. Like WH, UB and EVER, SLASH does not appear on nodes as a matter of course. Instead, 'slashed' versions of basic ID rules are created by means of metarule. The topmost occurrence of SLASH in a tree will be licensed by a basic ID rule such as the S_UDC rules, which expand as a preposed constituent and an S slashed for that constituent, or such as the A1 rule for tough-movement, which introduces a slashed VP. Below the initial SLASH introducing node, a slashed category will be expanded by a slashed version of a basic ID rule. SLASH is terminated at the bottom of a tree by rules output from the slash termination metarules which add the feature specification +NULL. SLASH never appears on a lexical (BAR 0, SUBCAT) node.

3.5 Other Features

SUBCAT

SUBCAT appears on all lexical nodes (except for particles). In the case of major categories this feature serves to ensure that lexical heads appear with appropriate complements.

H

The feature H appears on all heads in ID and PS rules. Its function is to identify heads for the purposes of (head) feature propagation.

T

The parser will find all possible parses of an input string, whether these parses are root sentences or not. The feature T defines a root node for the parser such that [T +] will dominate all sentences which are root sentences but not non-root sentences (see ID rules T and T2). The GDE has a flag which, when on, will cause the parser only to return [T +] parses. The grammar also contains a rule which marks an NP as [T +] (ID rule T/NP) since this was found useful for testing purposes.

CONJ

CONJ is a feature which appears on all conjuncts and which encodes the type of conjunction that occurs with a conjunct. For example, in *the dog and the cat* the first conjunct (*the dog*) is [CONJ NULL] (contains no conjunction) whilst the second conjunct (*and the cat*) is [CONJ AND] (contains the conjunction *and*).

CN1, CN2, AND

The grammar contains a small set of rules for generating cardinal numbers, such as *three, thirty three, three hundred, three hundred and three, three hundred thousand.* These three features are sufficient to define cardinals since it was decided to describe them independently of the main feature system. Numbers only start to participate in the main grammar by virtue of ID rules N/NUMBER and A/NUMBER.

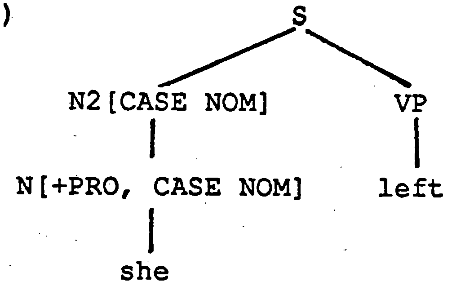
4. Feature Propagation

Our grammar has three methods of encoding information about the way in which features propagate in trees – propagation rules, metarules and explicit statements in ID rules. These methods are used to express various feature propagation principles similar to those of GPSG, such as the Head Feature Convention (HFC), the Foot Feature Principle (FFP) and the Control Agreement Principle (CAP), and to encode more idiosyncratic propagation regimes.

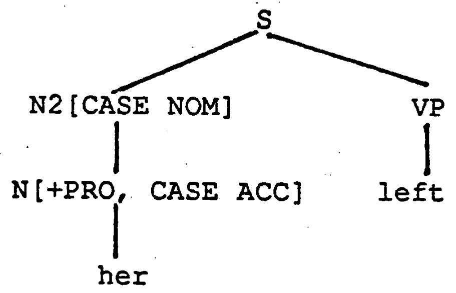
4.1 Capturing the HFC

The HFC is designed to capture the intuition that mothers and heads both have the same feature value pairs, for example a passive VP contains a passive verbal head. In our grammar, we state this relationship by means of propagation rules which bind the values of head features on the mother category to the values of those features on its head daughter. Firstly we have propagation rules which bind values for the features N, V and BAR between mothers and heads (propagation rules PROP_HEAD_V, PROP_HEAD_N and PROP_BAR). Thereafter, we define a set of head features for each major category type (the sets NOMINALHEAD, VERBALHEAD, ADJHEAD and PREPHEAD) and propagation rules which bind values for these features between mother and head (propagation rules HFC_VERBAL, HFC_NOMINAL, HFC_ADJ and HFC_PREP). The result of such propagation rules is to block analyses in which mother and head have conflicting values for some head feature; for example, the tree in (22) is licensed by the grammar, however the tree in (23) is not, since the N2 subject is required by the S rule to be nominative but the lexical item *her* is accusative.

(22)



(23)



4.2 Capturing the FFP

The FFP controls the propagation of a set of features (known as 'foot' features) which propagate in a less predictable way than head features do. Foot features may propagate either between a head and a mother or between a non-head and a mother. In our grammar, the features which resemble GPSG's foot features are the extension features WH, UB, EVER and SLASH. Unlike other features, extension features are not considered to be part of the essential make-up of any particular category. Instead, under certain conditions they may appear as 'extra' features on a number of different category types. In general, ID rules which contain categories which have these extra features are generated by means of metarule (see the FOOT and SLASH metarules). So corresponding to a basic ID rule there might exist a metarule output which is, for example, the 'slashed' version of that ID rule, for example, (24) is a basic ID rule and (25) would be a slashed version of it:

```
(24) IDRULE VP/NP : VP --> H[SUBCAT NP], N2[-PRD].

(25) VP[SLASH @x] --> H[SUBCAT NP], N2[-PRD, SLASH @x].
```

The metarule responsible for the output rule (25) would be as in (26) (n.b. the statement of metarule SLASHBAR0A_2 is actually more complex than this but details have been omitted here, for reasons of clarity).

```
(26) METARULE SLASHBAR0A_2 :
       X --> H, X2, W.
       ==> X[SLASH @x] --> H, X2[SLASH @x], W.
```

Notice that the metarule not only adds SLASH to a mother and daughter category, it also binds the value of SLASH (with the variable @x), so if S is instantiated to [SLASH N2] then VP must be [SLASH N2] too, and not [SLASH P2] for example. So these metarules control both the positions in which these features may appear and the appropriate bindings of their values.

There are a number of metarules in the grammar which control the propagation of SLASH and a number of other ones which control the propagation of the feature WH, UB and EVER. For details see the section on unbounded dependencies below.

4.3 Capturing the CAP

The CAP is responsible for ensuring that categories which need to agree with other categories do, in fact, do so. In our grammar we state explicit bindings of agreement features by means of propagation rules. For example, (27) is an example of a propagation rule which forces agreement between an NP subject and a VP:

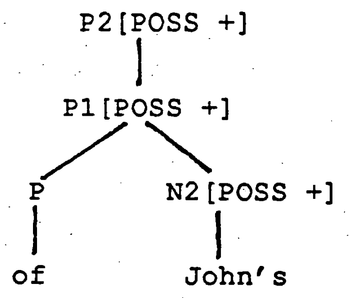
```
(27)  PROPRULE AGR/NP_VP : S --> N2, H2[-SUBJ, AGR N2], U.
                   F(1) = F(2[AGR]), F in AGRFEATS.
```

The effect of this is to make the values of the features defined by the set AGRFEATS on the subject N2 (i.e. F(1)) be equal to the values of the same features on the N2 which is the category value of the VP (i.e. F(2[AGR])). More details about agreement and the places where agreement takes place are provided in the section on agreement and control below.

4.4 Other Propagations

A few feature propagations fall outside of the scope of the GPSG feature principles. For these, a miscellaneous set of propagation rules is used. For example, some features are very like head features in that they propagate between mother and head, but are not included in the set of head features because they do not appear on all instances of a category type. For example, the feature POSS appears on P2 and P1 but not on P, as in (28):

(28)



In such cases, these features are bound by rather more specific propagation rules. (29) is the propagation rule responsible for POSS propagation between P1 and P2 in (28). (It is also responsible for propagating the feature GERUND, which also appears on P2 and P1 but not P.)

```
(29)   PROPRULE PP  : P2 --> [H +, BAR (1, 2)], U.
                      F(0) = F(1), F in {POSS, GERUND}.
```

Other features may propagate from non-heads to mothers. For example the definiteness of an N2 is generally determined by the definiteness of the specifier, that is, DEF propagates from a specifier to its N2 mother as expressed by the following propagation rule:

```
(30) PROPRULE DEF3 : N2[+SPEC] --> [+QUA], U. DEF(0) = DEF(1).
```

One further way exists to state bindings between feature values. This is to state them explicitly on ID or PS rules. For example, the rules which introduce particles for phrasal verbs and adjectives explicitly bind PRT on the head to PRT on the particle as in (31) to ensure the grammaticality patterns in (32):

```
(31) IDRULE VP/NP_PHRA :
        VP --> H[SUBCAT NP, PRT @p], N2[-PRD, -PRO], [PRT @p].
```

(32) a. he rang up his sister
     b. * he rang away his sister
     c. * he rang over sister

4.5 Feature Defaults

Some comment is in order about the default rules in the grammar. These rules state default values for features on particular categories in particular environments. As such, they prevent the need for restating regular feature value specifications on each individual rule. An explanation for each particular default rule can be found in the grammar. Default rules enrich the featural specification of categories in ID or PS rules to ensure, for example, that all N2 complements will be CASE ACC unless they are already marked as CASE NOM, or that most N2 complements will be NFORM NORM, or that all VP mothers will be marked as –AUX unless they are already marked as +AUX, and so forth. (33) is the rule that achieves the AUX default:

```
(33) DEFRULE AUX : VP --> W. AUX(0) = -.
```

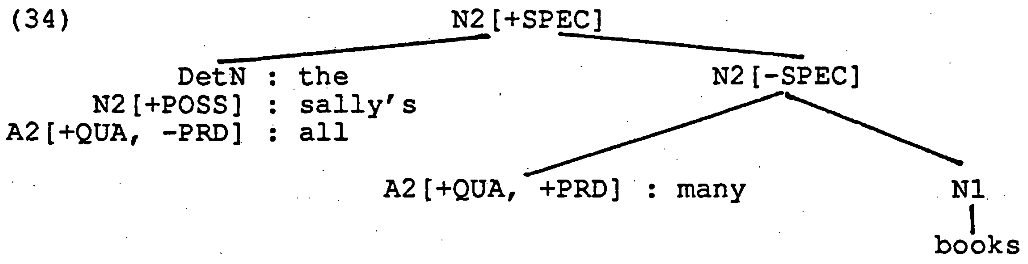
5. Category Structure

5.1 *The Structure of NP*

5.1.1 *The Top Level*

The top levels of NP structure can be schematised as in (34).

(34)

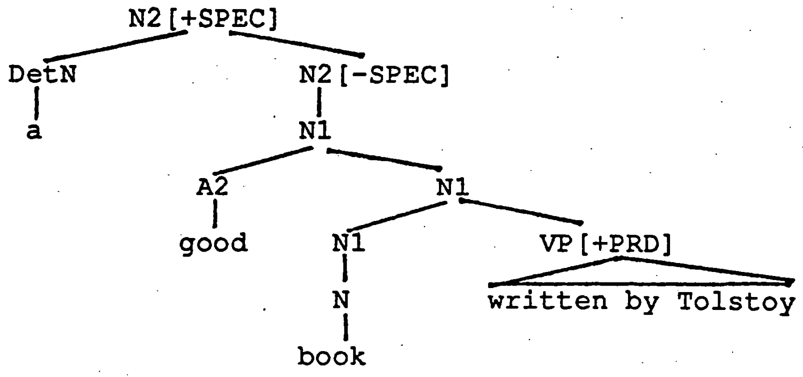


In (34), there are two N2 nodes distinguished by the feature SPEC. This allows us to recognise an extra level of structure inside NP without the need to recognise an extra BAR level. In specifier position (under N2[+SPEC]), determiners, possessive N2s or –PRD quantifying adjectives (*all, both,* etc.) can appear. N2[–SPEC] may contain a +PRD quantifying adjective (*many, few,* etc., cardinals and ordinals) and N1. The reason for identifying two types of quantifying adjectives occurring in two different positions inside NP stems from the observation that the +PRD quantifying adjectives can cooccur with specifiers – *the many books, the three books, John's many virtues*. This analysis achieves a fairly general account of noun specifiers but it does involve some overgeneration in that **all many books* and other sequences of two quantifiers will be parsed.

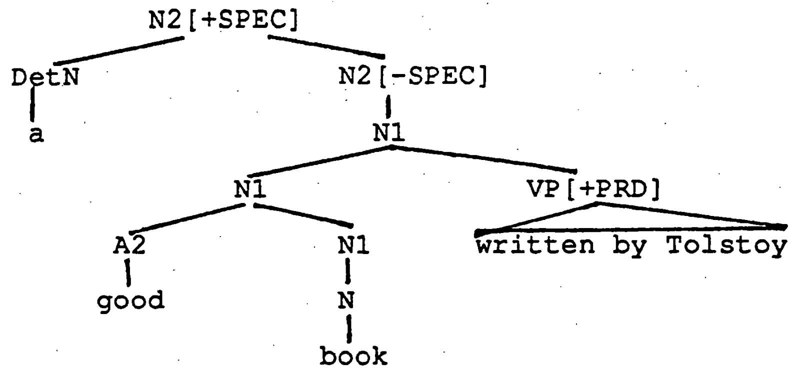
5.1.2 *Modifiers*

Modifiers are introduced by means of recursive rules as daughter to N1 and sister to N1. The set of NP modifiers defined by the grammar is as follows: prenominal AP (*good book*), prenominal possessive NP (*the women's javelin*), postnominal AP (*a man taller than lée*), reduced relatives (described as VP[+PRD] – *a book written by Tolstoy, a man writing a book*), PP (*a man with an umbrella*), possessive PP (*a book of lee's*), infinitival, slashed VP (*the man to buy books from*), relative clauses (*the man who sold me a book, the man who I bought a book from*) and 'that'-less relatives (*the man I bought a book from*). Free relatives are also catered for – see the section on relative clauses below. A problem arises when both a pre-modifier and a post-modifier occur inside an NP, in that a spurious attachment ambiguity is created. For example, *a good book written by Tolstoy* would be analysed as ambiguous between (35) and (36).

(35)

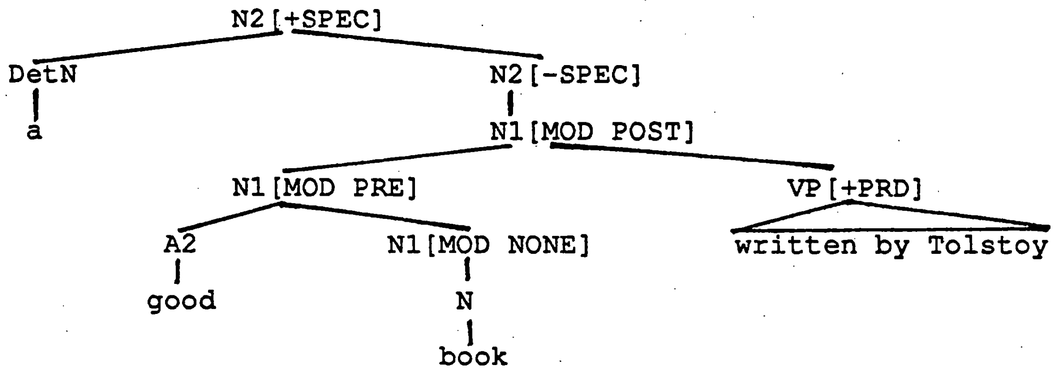


(36)



To solve this problem we have introduced a feature MOD with values PRE, POST and NONE which is used in such a way as to ensure that pre-modifiers attach lower in an N1 than post-modifiers. Thus (36) is the only tree which will be produced. (37) demonstrates the distribution of the feature MOD.

(37)



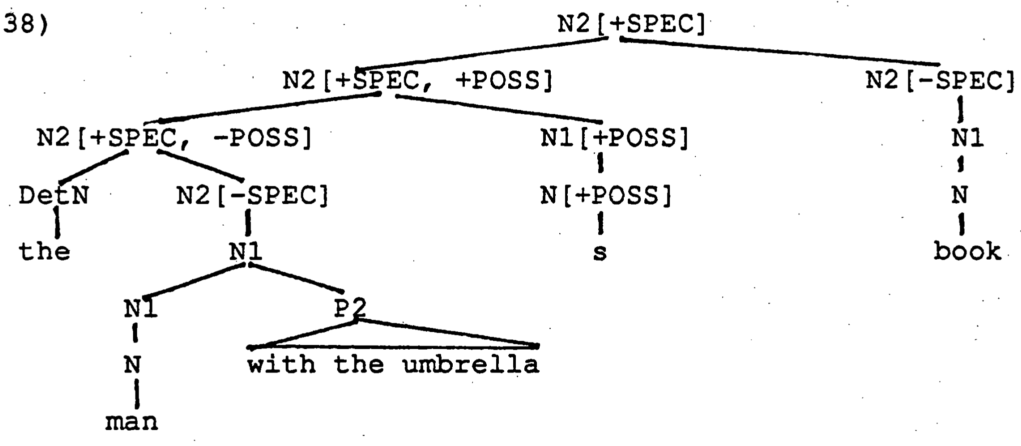
5.1.3 Complements

Complements to nouns are introduced as daughters of N1 and sisters of N (the lexical head). The lexical head carries a SUBCAT feature to ensure that nouns cooccur with appropriate complements. The full set of N complements recognised by the grammar can be found in the grammar – the rules N1/N – N1/VPINF are ones which introduce nouns with their complements.

5.1.4 Possessive NP

Special rules define possessive NPs and their specifier position (see ID rules N2+/POSSNP, POSSNP, and N1/POSS). The structure assigned to an NP such as *the man with the umbrella's book* is shown in (38).

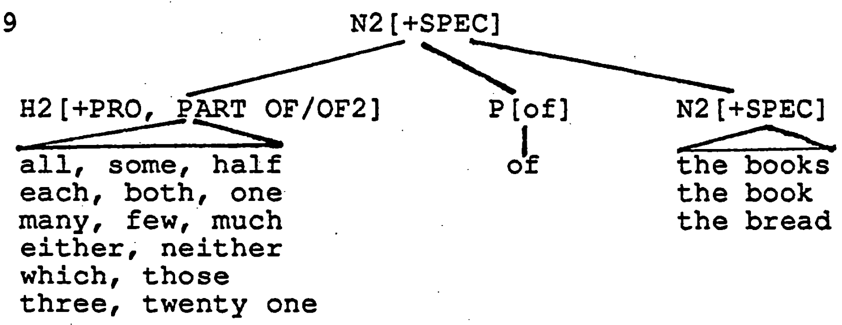
(38)



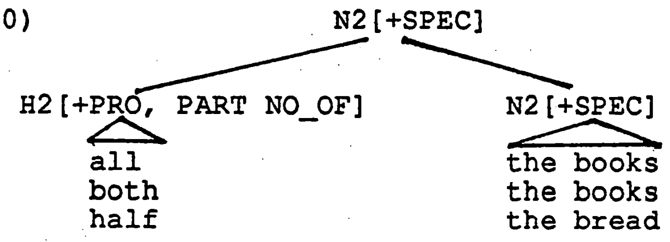
5.1.5 Partitives

An additional type of N2 is the partitive N2. Partitives will be assigned one of the following two structures (see ID rules N2+/PART1 – N2+/PART3):

(39



(40)

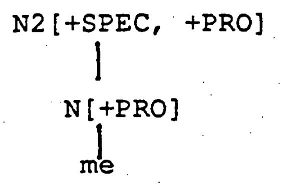


In both trees the head is the left-most pronominal N2. The presence or absence of *of* is determined by the value for the feature PART which appears on the pronouns. Those pronouns which must agree with the following NP (e.g. *all, some, any, both,* etc.) match ID rule N2/PART2, and those which must not (e.g. *each, either, neither*) match ID rule N2+/PART3. Cardinal numbers (as defined by PS rules NUM1 – NUM3) can head partitives, so ID rule N/NUMBER1 introduces them with an appropriate value for PART.

5.1.6 Other NP Types

Pronouns are simply assigned the structure in (41) (see ID rules N2+/PRO and N2+/PRO2).

(41)



Proper names are treated like ordinary nouns, except they are marked as [PN +]. This allows us to handle examples such as *Kim, the Kim that I know, which Kim*etc. Complex names are defined by means of the PS rules N/NAME1 – N/NAME2B.

NPs which are apparently headed by adjectives (*the poor*) are treated as such, rather than being treated by some lexical rule which converts adjectives to nouns. The reason for this is that the adjectives still get modified in the same way as other adjectives rather than in the same way as nouns (*the extremely poor vs *the extremely poverty). ID rules N2+/ADJ1 – N2+/ADJ3 generate this kind of NP.*

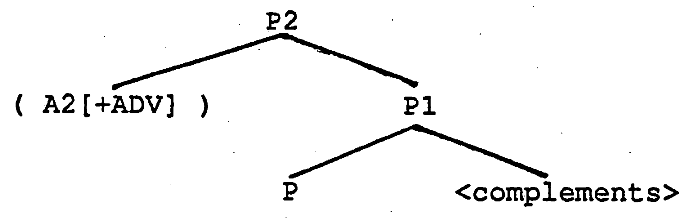
5.1.7 Compounds

Our system distinguishes between morphological compounding and syntactic compounding, in that compounds which orthographically contain no blank spaces (blackboard, spin-dryer) will be recognised by the word grammar while compounds with blank spaces (*shoe lace, spin dryer*) must be recognised by the sentence grammar. Accordingly, the grammar contains some ID rules (N/COMPOUND1 – N/COMPOUND3) for noun-noun compounds (*shoe lace*), adjective-noun compounds (*cold drink*) and preposition-noun compounds (*after care*), respectively. This approach to compounding is neither principled nor adequate (see the section on limitations of the grammar below).

5.2 The Structure of PP

The overall structure of PP is sketched in (42).

(42)

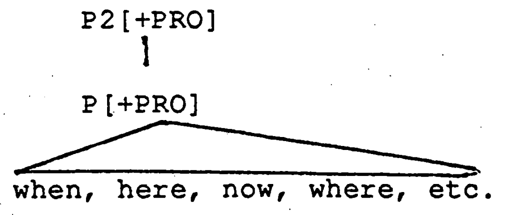


Under P2, the optional A2[+ADV] node allows for adverbial modifiers, as in *right under your nose* and *immediately after Christmas*. The set of complements of P recognised by the grammar is given in (43).

(43)	a. on the table	ID rule P1/NP
	b. up on the roof	ID rule P1/PP
	c. of John's	ID rule P1/POSS
	d. because he ate too much	ID rule P1/SFIN

Pro-PPs are assigned the following structure:

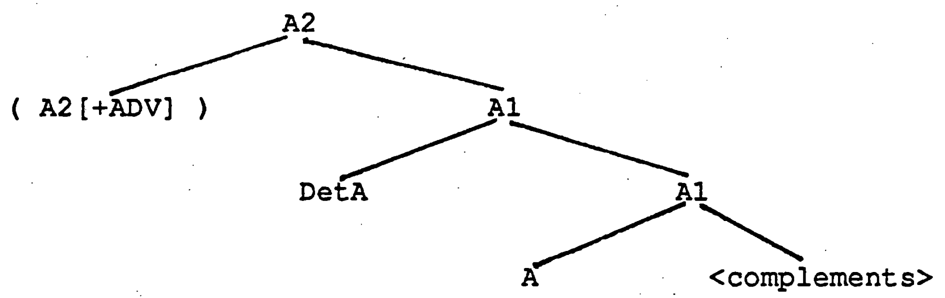
(44)



5.3 The Structure of AP

AP structure is outlined in (45).

(45)

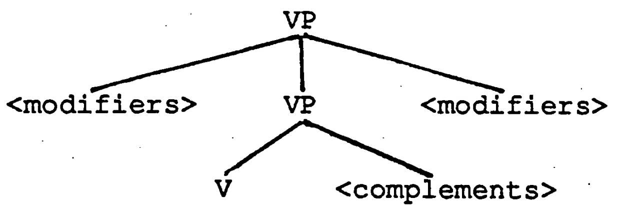


The grammar provides for a number of different complements of A – see rules A1/A – A1/PP_WHSC. The types of adjective recognized include ones taking a variety of PP and S complements, control adjectives (*eager to see the film, likely to see the film*), tough-movement adjectives (*the book is easy to read*), adjectives which take V2 subjects (*that you haven't read the book is obvious, for you to read the book would be easy, to stay at home would be possible*) and adjectives that take pleonastic *it* as subject (*it would be possible (for us) to stay at home*). For more details see the sections on control and agreement, unbounded dependencies, extraposition and comparatives.

5.4 The Structure of VP

The general structure of VP is:

(46)



5.4.1 Modifiers

A VP may be modified by an AP[+ADV] or a PP or a coordination of +ADV categories. Because the structure assigned to auxiliary sequences is a deep right-branching structure, care was taken to restrict the positions in which modifiers can attach. In general, they can only attach to –AUX VPs thus ensuring that in an example like *lee may have been reading the book without understanding it*, the modifier *without understanding it* can only attach to the VP containing the main verb (*reading the book*). An exception to this occurs in examples such as *lee has obviously/?carefully been reading the book* where the modifier has to attach to a +AUX VP. To deal with these cases, a rule allowing adverbs to attach to a +AUX VP has been introduced but this rule linearises in such a way that the adverb must precede the VP. Adverbs attaching to –AUX VPs may either precede or follow the VP (*lee is happily reading the book, lee is reading the book happily*). PP modifiers may only follow the VP (*lee read the book with care, *lee with care read the book*).

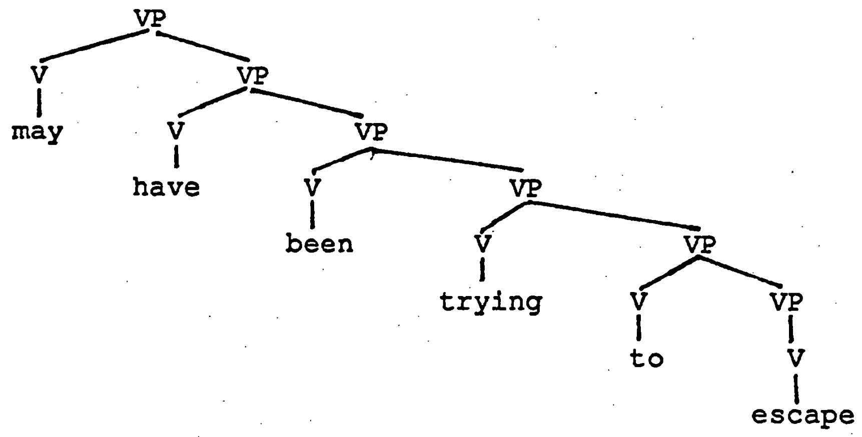
5.4.2 Complements

A large variety of verb complements are included in the grammar. The set of VP rules includes ones for intransitives, transitives, ditransitives, verbs taking PPs, verbs taking NPs and PPs, verbs taking APs, verbs taking S complements of various types, auxiliary verbs, control verbs (subject raising, subject equi, object raising, object equi), verbs taking embedded question complements, copula *be*, verbs taking sentential or pleonastic *it* subjects etc. For details see the grammar and the sections below.

5.4.3 Auxiliaries

Sequences of auxiliaries receive a recursive structure as in (47) where the auxiliary is treated as the head of its VP and where its complement is another VP. The required VFORM value of the VP complement is encoded in the auxiliary rules and this ensures that only grammatical sequences of auxiliaries are recognised.

(47)



5.4.4 Phrasal and Prepositional Verbs

Phrasal verbs occur with a number of different subcategorisation possibilities, for example, there are intransitive phrasal verbs (*fall over*), transitive ones (*blow it up*), ditransitive ones (*give him back the book*), ones taking sentential complements (*point out that he is wrong*) etc. For this reason phrasal verbs are treated in exactly the same way as their non-phrasal counterparts, except that they have an additional feature PRT to encode the form of the particle that they require. ID rules with 'PHR' in their names are rules specifically for phrasal verbs and look much like the the equivalent non-phrasal rules except for the feature PRT both on the head and the particle. The presence of particles in so many rules leads to a certain degree of complication in the LP rules.

Prepositional verbs are ones which subcategorise for a preposition which cannot be interpreted as part of a PP, for example ones which precede AP (*condemn it as useless*) or VP (*bank on being promoted*). These verbs have an extra feature PREP to control the form of the preposition and there are ID rules which deal specifically with them (ones with 'PREP' in their names).

5.5 The Structure of S

ID rules S1 – S/V2_SUBJ6 expand S as a subject and a VP. The following possibilities are recognised:

(48)	a. kim reads the book	(nominative NP subject, finite verb)	rule S1
	b. Fido be a good dog	(nominative NP subject, BSE verb)	rule S2
	c. Him to read the book	(accusative subject, infinitive verb)	rule S3
	d. Him reading the book	(accusative subject, ING verb)	rule S4
	e. That kim reads bothers lee	(finite S subject, finite verb)	rule S/V2_SUBJ1
	f. For us to go would be possible	(non-finite S subject, finite verb)	rule S/V2_SUBJ2
	g. To go would be possible	(non-finite VP subject, finite verb)	rule S/V2_SUBJ3
	h. That he leave is necessary	(bse S subject, finite verb)	rule S/V2_SUBJ4
	i. Whether he won is not clear	(+Q S subject, finite verb)	rule S/V2_SUBJ5
	j. What we should do is not clear	(WH +Q S subject, finite verb)	rule S/V2_SUBJ6

ID rules S/THAT1 - S/AS expand S as a complementiser followed by an appropriate type of S. The following possibilities are recognised:

(49)	a. That kim reads	(*that* + finite S)	rule S/THAT1
	b. That fido be a good dog	(*that* + BSE S)	rule S/THAT2
	c. For him to read the book	(*for* + non-finite S)	rule S/FOR
	d. Whether he reads the book	(*whether* + finite S)	rule S/Q1
	e. If he reads the book	(*if* + finite S)	rule S/Q2
	f. Than kim reads books	(*than* + finite S)	rule S/THAN
	g. As kim reads books	(*as* + finite S)	rule S/AS

ID rules S/NP_UDC – S/AP_UDC expand S as a preposed constituent (NP, PP or AP) followed by a slashed S whose value for SLASH is of the same category as the preposed constituent.

ID rule S/IMPER expands S just as [+FIN, VFORM BSE] (imperative) VP.

ID rules S/TAGQUESTION and S/TAG define complete tag questions and the tag part of tag questions, respectively.

S can occur with an A2[+ADV] modifier, a P2 modifier or a coordinated adverbial modifier. These are introduced by ID rules S/ADVMOD, S/PPMOD and S/ADV/CONJ. Each of these can occur either sentence-initially or sentence-finally. In addition, an adverb may optionally occur between the NP subject and the VP in ID rule S1 as, for example, in *lee clearly likes reading books*. Notice that this sentence will receive two parses, one where *clearly* is identified as an S modifier and one where it is identified as a VP modifier. The latter would be the structure appropriate to *lee happily reads books.*

Inverted sentences are produced by means of the SAI metarule (see section 6.2. below).

6. Constructions

6.1 Passive

Included in the grammar is a passive metarule (50) which takes as input active VP rules and produces as output passive versions of these. This treatment closely parallels the GPSG85 analysis of passive.

```
(50) METARULE PASSIVE :
       VP[AGR N2[NFORM NORM]] -->
       H[SUBCAT (NP, NP_@NP, NP_PP, NP_PP_PP, NP_LOC, NP_ADVP,
                 NP_SFIN, NP_NP_SFIN, NP_SBSE, NP_WHS, NP_WHVP,
                 OC_NP, OC_AP, OC_INF),
         AGR N2[NFORM NORM], PSVE -],
       N2[NFORM NORM, PRD -, PRO (@, -)],
       W.
    ==> VP[EN, +PRD, AGR N2] --> H[PSVE +, AGR N2], (P2[PFORM BY]), W.
```

The metarule is restricted in a number of ways: firstly, only [AGR N2[NFORM NORM]] VP rules can be inputs. There are passive versions of VPs with other AGR values, but these are not dealt with by metarule (see below). Secondly, the NP object of the input rule must be [–PRD]. By this means, we block the analysis of passive examples such as **two pounds is cost by the book* (see ID rule VP/NOPASS), **a fool is been by lee* (see ID rule VP/BE_COP1) and **a fool is considered kim by lee* (see ID rule VP/TAKES_2NP – the last example will receive a parse, but only on the interpretation equivalent to the active *lee considers a fool kim*). Thirdly, the list of SUBCAT values in the input pattern restrict the set of VP rules that the metarule can apply to. This prevents rules such as VP/OR_BSE from matching, thereby preventing a parse for **lee is made wash up*.

There are a further set of passive VP rules which exist as basic ID rules instead of being generated by the passive metarule. These ID rules account for various 'odd' passives: 'prepositional passives' (51), ones with sentential subjects (52), ones which in the active are object raising but in the passive are subject raising (these can take any kind of N2 or V2 subject – (53) and (54)), and ones which are passive versions of extraposed actives (55) and some phrasal versions of these (see ID rules VP/PP_PHR/PASS, VP/OR_INF_PHR/PASS_A and VP/OR_INF_PHR/PASS_B).

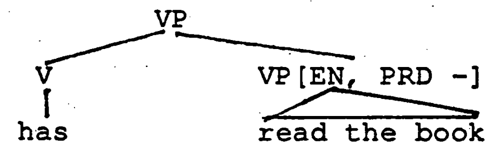
(51)	a. We are being looked at (by everyone)	VP/PP/PASS
(52)	That fido is a dog is believed (by lee)	VP/SFIN/PASS
(53)	a. Fido is believed to be a dog (by lee)	VP/OR_INF/PASS_A
	b. It is believed to bother lee that fido is a dog	"
	c. There are believed to be dogs in the garden	"
	d. That he fell is believed to bother lee	VP/OR_INF/PASS_B
	e. (For us) to go is believed to be possible	"
(54)	a. Fido is considered likely to be a dog (by lee)	VP/OR_AP/PASS_A
	b. It is considered likely to bother lee that fido is a dog	"
	c. There are considered likely to be dogs in the garden	"
	d. That he fell is considered likely to bother lee	VP/OR_AP/PASS_B
	e. (For us) to go is considered likely to be possible	"
(55)	Kim is bothered that lee likes reading books	VP/NP_SFIN2B/PASS

6.2 Subject-Auxiliary Inversion

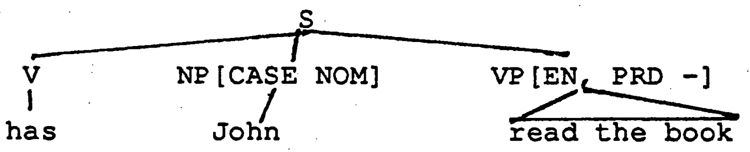
Following GPSG85 we handle subject-auxiliary inversion by means of a metarule (SAI). This takes as input +AUX VP rules such as (56) and produces as output inverted S rules. The tree generated by the input rule (56) is shown in (57) and the tree generated by the SAI generated version of (56) is shown in (58).

```
(56) IDRULE VP/HAVE1 : VP[+AUX, AGR N2] -->
                         H[SUBCAT HAVE], VP[EN, PRD -, AGR N2].
```

(57)



(58)



The SAI metarule is restricted such that only NP subjects can be inverted and not S subjects (**has that lee reads the book bothered lee*).

Inverted sentences occurring on their own will be recognised as root sentences. These are yes-no questions. They can also occur as the S which is sister to a preposed +WH constituent (wh-questions, e.g. *which book did kim read*).

6.3 Extraposition

GPSG85 treats extraposition by means of metarule, but we have chosen to deal with it by means of directly written basic ID rules, partly because of the small number of potential input and output rules and partly because there are a number of odd cases of extraposition not covered by the GPSG85 metarule. Extraposition verbs and adjectives occur with a number of different SUBCAT values but they are all specified as [SUBTYTPE EXTRAP]. The grammar can parse the following kinds of example:

(59) a. That he's disappeared matters	ID rule VP/SFIN3A
It matters that he's disappeared here	ID rule VP/SFIN3B
b. That he's disappeared bothers lee	ID rule VP/NP_SFIN2A
It bothers lee that he's disappeared	ID rule VP/NP_SFIN2B
c. That he's disappeared matters to me	ID rule VP/PP_SFIN2A
It matters to me that he's disappeared here	ID rule VP/PP_SFIN2B
d. To play cards amuses him	ID rule VP/OE_INF2A
It amuses him to play cards	ID rule VP/OE_INF2B
e. To see them hurts	ID rule VP/INFA
It hurts to see them	ID rule VP/INFB

(60) a. That he's disappeared is convenient	ID rule A1/SFIN2A
It is convenient that he's disappeared	ID rule A1/SFIN2B
b. That he's disappeared is clear to me	ID rule IDRULE A1/PP_SFINA
It is clear to me that he's disappeared	ID rule IDRULE A1/PP_SFINB
c. That he disappear was necessary	ID rule IDRULE A1/SBSE2A
It was necessary that he disappear	ID rule IDRULE A1/SBSE2B
d. For him to disappear was essential	ID rule IDRULE A1/SINF2A

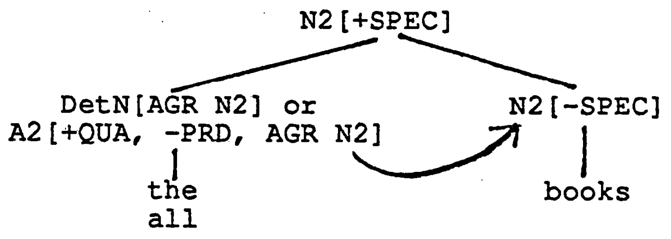
	It was essential for him to disappear	ID rule IDRULE A1/SINF2B
e.	To be nervous is normal	ID rule IDRULE A1/VPINF1A
	It is normal to be nervous	ID rule IDRULE A1/VPINF1B
f.	Whether he's ill is not clear	ID rule IDRULE A1/WHSA
	It is not clear whether he's ill	ID rule IDRULE A1/WHSC
f.	Whether he's ill is not clear to us	ID rule IDRULE A1/PP_WHSA
	It is not clear to us whether he's ill	ID rule IDRULE A1/PP_WHSC

6.4 Agreement and Control

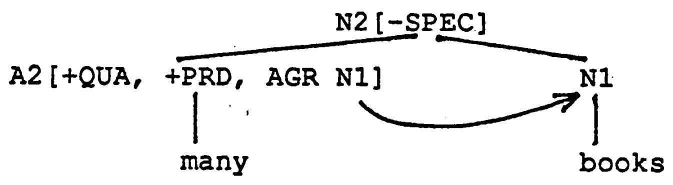
In GPSG85, the category valued AGR feature is used both to ensure subject-verb concord and to ensure an appropriate selection of subject for control predicates. The positions where AGR must be bound are defined by reference to the semantic type of a category. Once these positions have been identified, the Control Agreement Principle ensures appropriate bindings. In our grammar, since we assume no specific theory of semantics, agreement positions must be identified by the grammar writer. Once identified, propagation rules ensure binding of features on the category value of AGR on some node to features on a sister of that node. The following agreement positions have been identified:

(61)

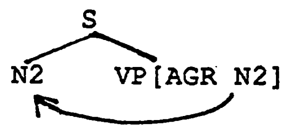
a.



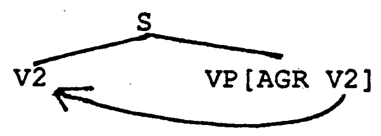
b.



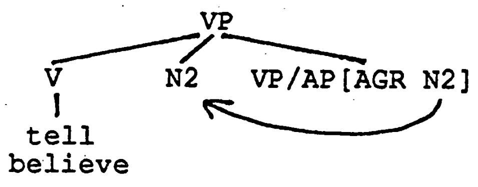
c.



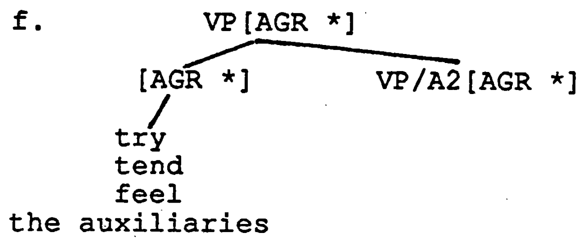
d.



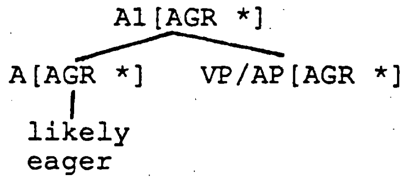
e.



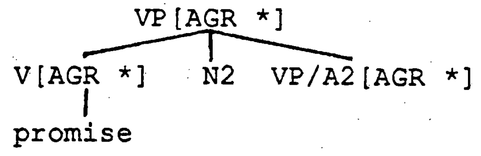
f.



g.



h.



In (61a) and (61b), determiners and quantifying adjectives must agree with their nominal sisters. Propagation rule AGR/SPEC_N ensures that the values for the features PLU and COUNT on the the nominal category which is the value of AGR on the specifier are bound to the values for these features on the nominal head, thus blocking examples such as **a books, *both book, *many book* and **much books.*

In (61c) and (61d) there must be agreement between a VP and its N2 or V2 subject. In the case of the N2 subject, propagation rule AGR/NP_VP binds the features PER, PLU, CASE and NFORM (and COUNT somewhat redundantly) between the AGR N2 on the VP and the N2 subject. The binding of the PER and PLU features ensures agreement whilst the binding of the NFORM feature ensures that a verb will only occur with the type of NP subject it requires. For example, one entry for the verb *bother* is specified as [AGR N2[NFORM IT]] so the binding of AGR ensures that it will only occur with a pleonastic *it* subject. Note that in inverted sentences agreement occurs between the subject N2 and the inverted verb. Propagation rule AGR/INV achieves the appropriate binding of values. In the case of the V2 subject, the values for the features SUBJ, FIN and VFORM are bound by propagation rule AGR/V2_VP. In this way, verbs or adjectives which require one particular type of V2 subject (finite S, infinitival S or infinitival VP) will only be able to appear with that type of subject.

The tree in (61e) exemplifies the configuration in which object control verbs appear. Here the N2 object can in some sense be identified as the subject of the VP complement. Accordingly, propagation rule OBJ_CONTROL binds the values for the features PER, PLU, NFORM and COUNT on the [AGR N2] on the VP to values for these features on the N2 object. The only feature that is of significance here is NFORM since this restricts the type of N2 which can occur as object. In object equi cases (see ID rules VP/OE*) the N2 must be [NFORM NORM] so the NFORM value on the AGR N2 on the VP must also be NORM. This ensures that examples such as *persuade lee to read* will be parsed whilst *persuade it to bother lee that kim reads* will not. In the case of object raising (see ID rules VP/OR*), the NFORM value on the N2 is unspecified with the result that all of *believe lee to be a fool, believe it to bother lee that kim reads, believe there to be elephants in the garden* will be parsed.

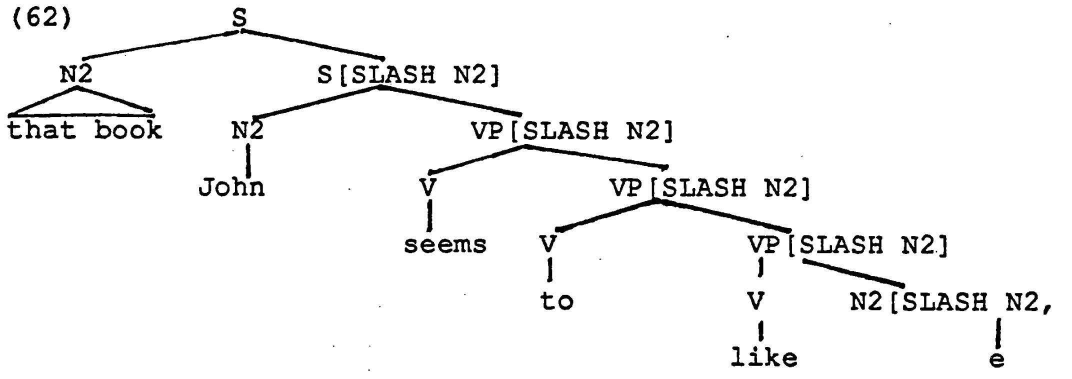
The trees in (61f) and (61g) are subject control environments. Here there is no N2 object to be interpreted as the subject of the VP complement (or A2 complement (*feel intelligent* (61f)). Instead, the subject of the mother VP or A1 must be interpreted as the subject of the complement – the * in the trees indicates that they share the same AGR value and hence the same subject. The SUBJ_CONTROL propagation rules ensure this identity of values. In the case of subject equi, the control predicate (such as *try, feel* or *eager*) is specified as [AGR N2[NFORM NORM]] and this means that the only kind of VP or A2 complement that can occur here will also have to be [AGR N2[NFORM NORM]]. Examples such as **it tries to bother lee that kim reads, *there feel likely to be elephants in the garden, *that kim reads is eager to bother lee* will therefore be blocked. In the case of subject raising predicates (such as *tend, likely* and the auxiliaries and modals) there is no restriction on the type of subject that these predicates can occur with and hence no restriction on the type of their VP or A2 complements. Subjects can be V2s of any type or N2s of any type – in a sense these predicates aquire the agreement properties of their complements. For each type of subject raising predicate in the grammar, there are two ID rules, one AGR N2 and one AGR V2. This was necessary for propagation rules to work properly since values for features on these AGR categories need to be bound and the set of features on AGR V2 is different from the set of features on AGR N2. The AGR N2 versions are bound by the SUBJ_CONTROL propagation rules whilst the AGR V2 versions are bound by the

AGR/V2_CONTROL propagation rule. The result is that sets of sentences such as *kim seems to read, it seems to bother kim that lee reads, there seem to be elephants in the garden, that kim reads seems to bother lee, for us to go seems to be possible, to go seems to be possible* are all accepted by the grammar.

The tree in (61h) is the one in which the verbs *promise* and *strike as* appear. These verbs are peculiar in that although they occur in an environment typical of object control verbs, they are in fact subject control verbs. The NP object is not to be construed as the subject of the VP complement. The ID rules introducing *promise* (VP/SE_NP_INF) and *strike as* (VP/SR_NP_AP) are correctly bound by the SUBJ_CONTROL propagation rules SUBJ_CONTROL6 and not by the OBJ_CONTROL one.

6.5 Unbounded Dependencies

Our treatment of unbounded dependencies is much the same as the GPSG85 treatment and involves the use of the category-valued feature SLASH which encodes information about 'gaps' in trees. In GPSG85, SLASH is both a head and a foot feature and its distribution is therefore controlled both by the HFC and the FFP. In our grammar, SLASH is an extension feature. Its distribution is defined by means of metarules. This means it does not generally appear in basic ID rules except for the ones which introduce the 'top' of the unbounded dependency. Thereafter, a slashed category is expanded not by a basic ID rule but by a metarule-generated slashed version of a basic ID rule. For example, an ordinary VP category might be expanded by the ID rule VP/NP but a slashed VP category would be expanded by a metarule-generated slashed version of VP/NP. The tree in (62) demonstrates some typical examples of SLASH distribution in a topicalised sentence.

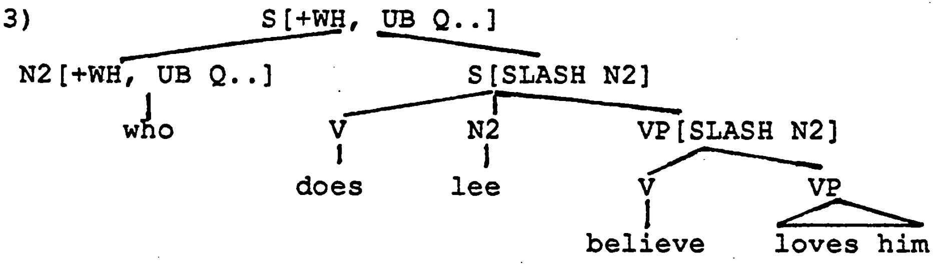
(62)


Here, the topmost instance of SLASH has been introduced by a basic ID rule (in this case S/NP_UDC. The S[SLASH N2] is then expanded, not by a basic S rule, but a slashed version of an S rule. Likewise, the VP rules used are slashed versions of VP rules. At the bottom of the tree, the rule used to build the VP *like e* is generated via two metarule applications. The metarule SLASHBAR0A_2 produces a slashed VP rule where the SLASH is passed to the NP object. This slashed VP rule forms the input to a slash termination metarule (STM1_N2) which marks this NP as +NULL.

A somewhat simplified description of the way in which SLASH propagates in GPSG85 is as follows: in all non-lexical ID rules SLASH passes from a mother to a head. In lexical ID rules the lexical head cannot be specified with SLASH so it passes to a non-head. This has two consequences: firstly, the only categories which can introduce 'gaps' (+NULL categories) will be ones in argument positions (sisters to a lexical head). Secondly, it makes it possible to achieve the effects of some island constraints – for example, the Complex Noun Phrase Constraint which, amongst other things, prevents extractions out of relative clauses. For a detailed discussion of this aspect of GPSG85 see Harlow (1986). There are further complexities involved in the GPSG85 account such as a treatment of double extractions in parasitic gap constructions but these are largely irrelevant here, although it should be noted that our grammar does not extend to an analysis of parasitic gaps.

In our grammar, SLASH propagation follows the same general principle outlined above. Metarules SLASHBAR2A – SLASHBAR2E pass SLASH from a mother to a [BAR 2] head, metarule SLASHBAR1 passes it from a mother to a [BAR 1] head and metarules SLASHBAR0A – SLASHBAR0E pass it from the mother in a lexical ID rule to some complement daughter. The STM1 metarules terminate SLASH on N2, A2 and P2 sisters to a lexical head, in much the same as the STM1 metarule in GPSG85 does. The two STM2 metarules are for terminating SLASH in missing subject constructions and operate in much the same way as the STM2 metarule in GPSG85 does. The case of empty subjects is worthy of some discussion: in sentences such as *who does lee believe loves him* GPSG85 assumes that although the subject of *loves* is missing, there is no empty category in this position. Instead, the embedded S complement of *believe* is replaced by a VP. Our grammar also induces this analysis, so the structure for this sentence will look like (63).

(63)



Further metarules are required for propagating SLASH to deal with cases not accounted for by the metarules already mentioned, as follows: odd propagations of SLASH in comparatives (SLASH/COMPAR and STM/COMPAR – for details see below), propagation of SLASH onto modifiers of VP (as in *what did lee read the book for e, why did lee read the book e* – MOD/SLASH, STM/MOD1 and STM/MOD2) and various propagations of SLASH in coordination rules where it must pass onto each conjunct (CONJ/SLASHA - CONJ/SLASHD).

In the rest of this section we discuss briefly the analysis of various unbounded dependency constructions.

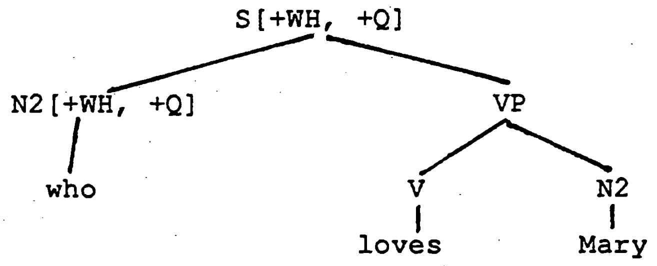
6.5.1 Topicalisation

Topicalised sentences are the simplest type of unbounded dependency. The topicalised constituent is introduced at the top of a tree by the S/UDC rules and the SLASH is passed down the tree until it is terminated. The tree in (62) is an example of a topicalisation structure.

6.5.2 Wh-questions

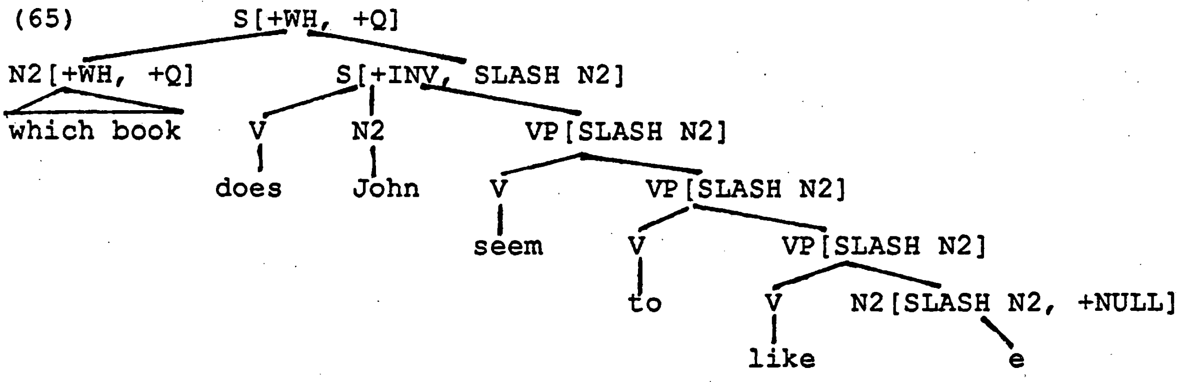
Simple wh-questions such as *who loves Mary* merely involve a FOOT metarule generated version of the S1 rule and receive a structure as in (64).

(64)



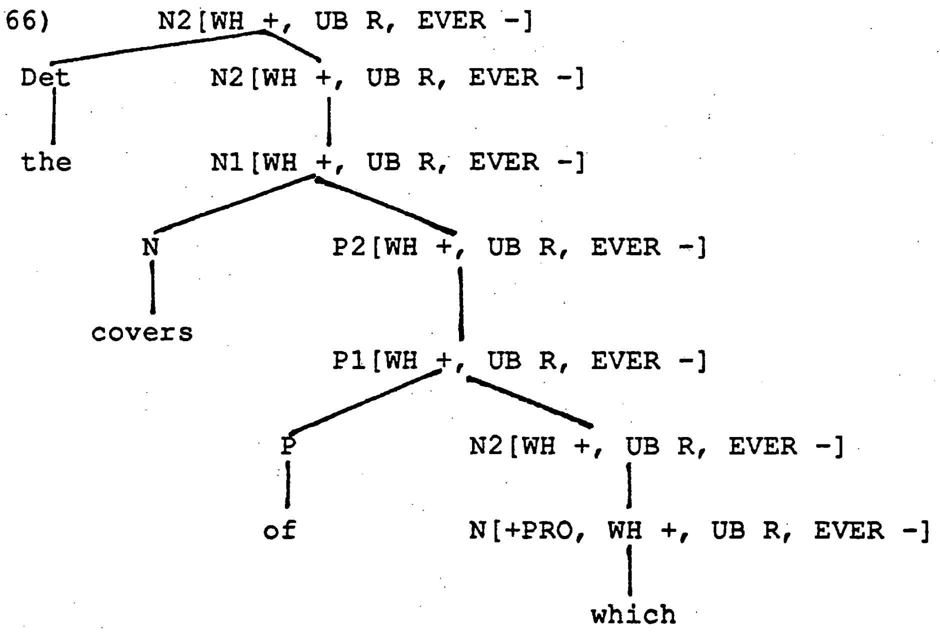
Wh-questions are very similar to topicalisations in that they too involve a preposed constituent and a SLASH passed down the tree. The difference is that the preposed constituent has the features WH, UB and EVER which pass up to the mother node (so FOOT metarule generated versions of the S/UDC rules are the ones used here). In addition, the slashed S which is the sister of the preposed constituent is inverted. The tree in (63) is an example of a WH question, as is the tree in (65).

(65)



Some comment is in order about the FOOT metarules which are responsible for the propagation of the features WH, UB and EVER. These metarules are similar to the SLASH metarules in that they generate new versions of basic ID rules which contain the features WH, UB and EVER. These features propagate in a rather unpredictable way and for this reason there is an odd selection of FOOT metarules picking out each case of propagation. The tree in (66) shows an example of the propagation of these features inside a noun phrase. (66) is the kind of relativising NP which might occur in a relative clause such as *the books the covers of which are torn.*

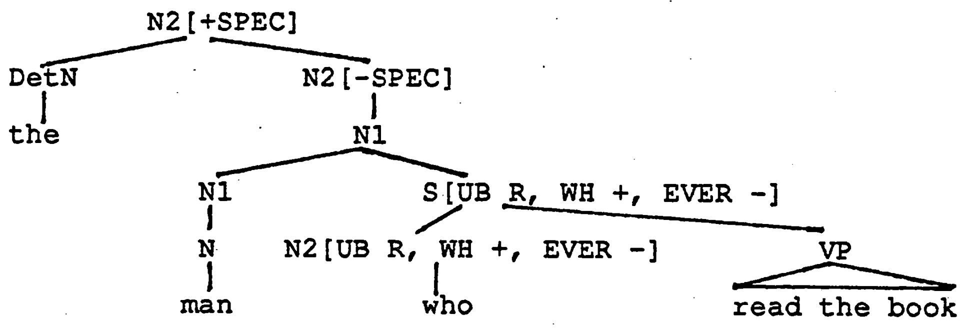
(66)



6.5.3 Relative Clauses

For ordinary relatives, our account is the same as the GPSG85 account. Ordinary subject relatives will receive structures like the following:

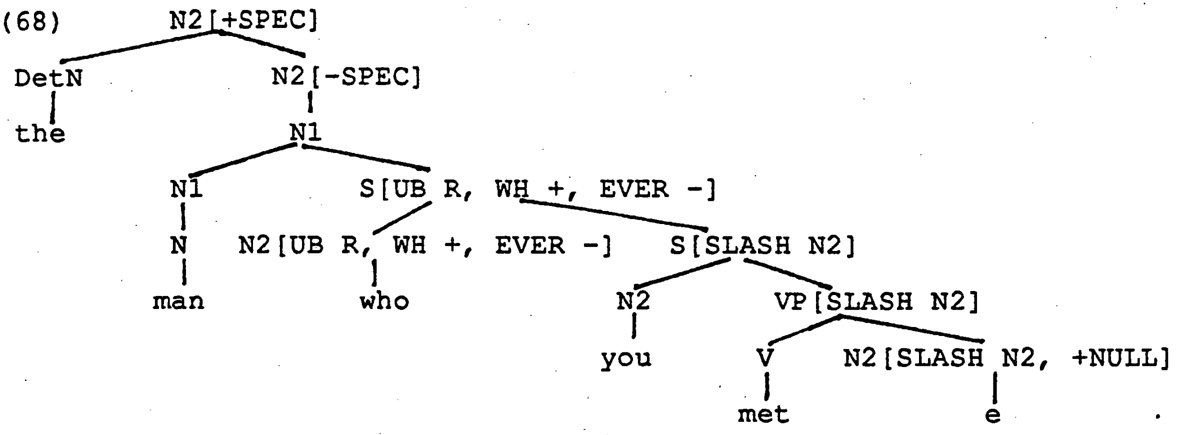
(67)



Notice that there are no gaps involved in subject relatives. In accounts that do propose gaps in subject relatives, the relative pronoun is said to be preposed ('string-vacuously') and the subject of the relative is a 'gap'. In this account, on the other hand, the relative pronoun is not preposed but simply occupies the position of the subject. A similar structure is assigned to examples with a *that* relative pronoun such as *the tree that was blown down*. The only difference is that the WH feature on the relative pronoun and relative clause has the value –. The rule which expands the S[UB R..] node is simply a FOOT metarule generated version of ID rule S1. Likewise, the rule expanding the N2[UB R..] is a FOOT metarule generated version of ID rule N2+/PRO.

Object relatives receive analyses like (68):

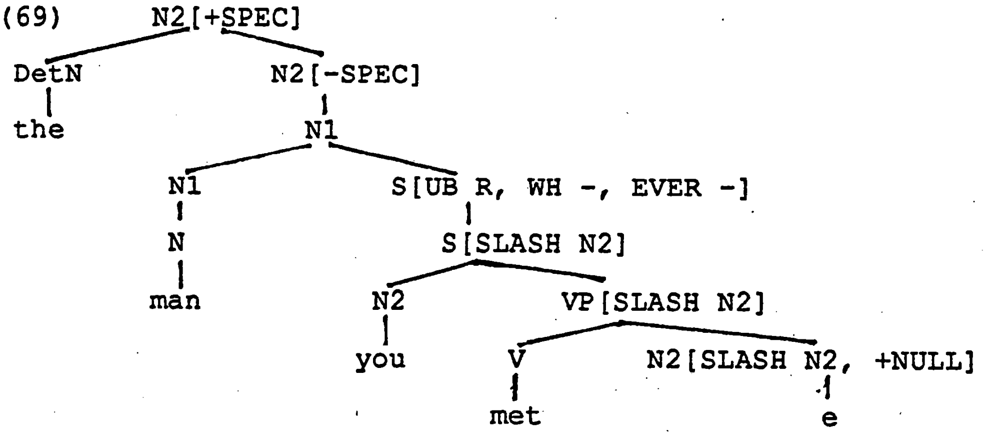
(68)



Here the relative pronoun is preposed and its sister is an S[SLASH N2]. The rule responsible for this expansion of S[UB R..] is a FOOT metarule generated version of ID S/NP_UDC. Object relatives with *that* (*the window that you broke*) are analysed in the same way except that the WH feature has a negative value again. The grammar also analyses relatives where the relativiser is a PP and not an NP as in *the table on which you put the book.* Here a FOOT metarule generated version of the S/PP_UDC ID rule is used to expand the S[UB R ..]. The grammar is restricted so as to only allow +WH PP relativisers, so examples such as **the table on that you put the book* will be blocked.

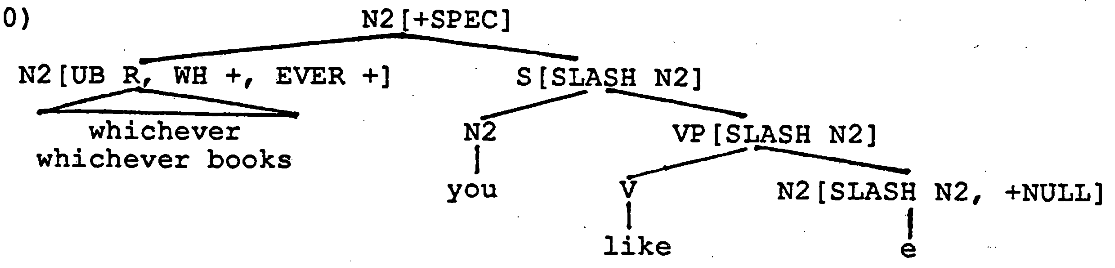
'That'-less relatives receive the analysis in (69):

(69)

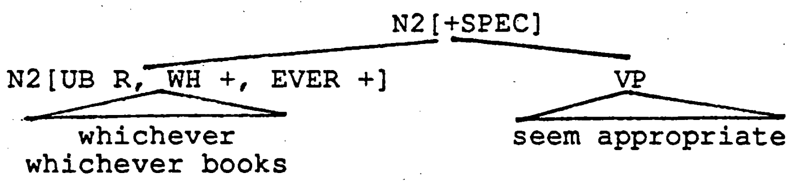


Free relatives receive the analyses in (70) and (71) (see ID rules FREE/REL and FREE/REL2).

(70)



(71)



6.5.4 Tough Movement

The rules A1/TOUGH1 and A1/TOUGH2 introduce tough-movement adjectives which appear with an S[SLASH N2[CASE ACC]] or a VP[SLASH N2[CASE ACC]] complement. These will generate examples such as *lee is easy to upset, lee is easy for us to upset.* The [CASE ACC] restriction blocks examples like *lee is easy to believe likes cats, lee is easy for us to believe likes cats.*

6.5.5 It-Clefts

The rules VP/BE_CLEFT1 and VP/BE_CLEFT2 allow it-clefts to be generated. VP/CLEFT1 is responsible for examples such as *it is lee who/that relies on kim, it is lee who/that kim relies on.* Here, the who/that.... clauses are analysed as relative clauses. VP/CLEFT2 is responsible for *it is on lee (that) kim relies.*

6.6 Comparatives

Our analysis of comparatives is based on an account in Gazdar (1980). It is not exhaustive, but it recognises all of the following:

Comparatives inside A2:

(72) a. Kim is *taller/more stupid than lee*
b. Kim is *as tall as lee*

(73) a. Kim is taller than lee is
b. Kim is *as tall as lee is*

(74) a. Kim is *taller than lee is short*
b. Kim is *as tall as lee is short*

Comparatives inside N2:

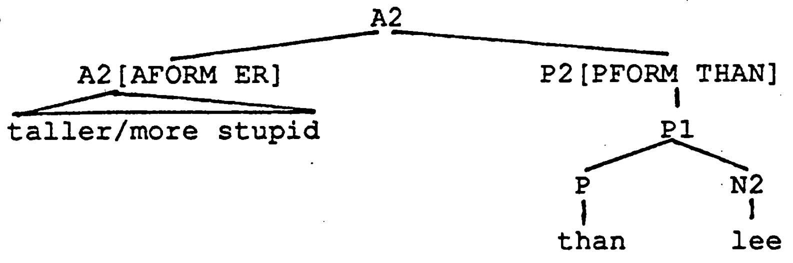
(75) a. Kim reads *more books than lee*
b. Kim reads *as many books as lee*

(76) a. Kim reads *more books than lee reads*
b. Kim reads *as many books as lee reads*

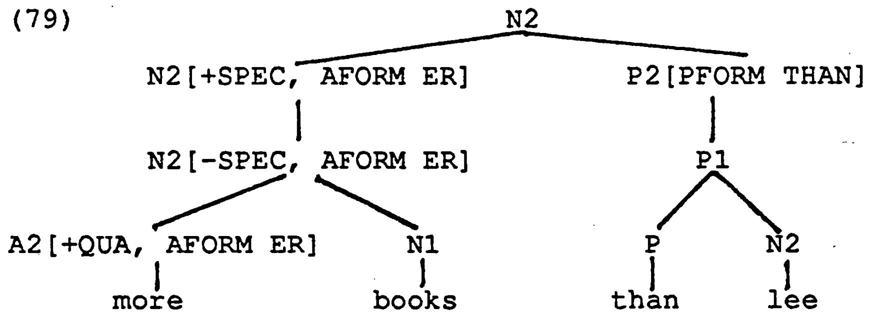
(77) a. Kim reads *more books than lee reads magazines*
b. Kim reads *as many books as lee reads magazines*

The a. sentences are all comparative sentences whilst the b. ones are equative sentences. Since the latter receive the same structure as the former the following discussion will refer mostly to comparatives. There are two lexical entries for the word *than* (and the word *as*), one as a preposition, and one as a complementiser. It is the preposition that occurs in (72) and (75) but the complementiser that appears in (73), (74), (76) and (77). The analyses for (72a) and (75a) (see ID rules A2/COMPAR1, A2/COMPAR2, N2/COMPAR1 and N2/COMPAR2) are as follows:

(78)



(79)

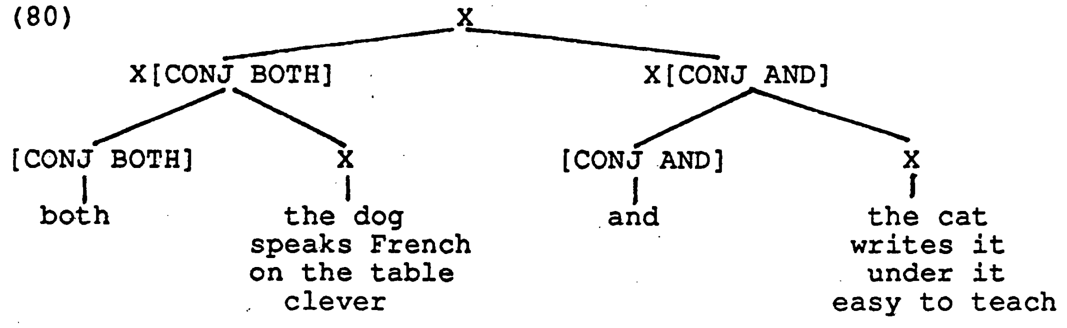


The trees for (73a) and (76a) look much the same except that instead of the P2[PFORM THAN] there is an S[COMP THAN, SLASH A2] in the case of (73a) and an S[COMP THAN, SLASH N2] in the case of (76a). The trees for (74a) and (77a), again, look much the same except that instead of the P2[PFORM THAN] there is an S[COMP THAN, SLASH DetA] in the case of (74a) and an S[COMP THAN, SLASH DetN] in the case of (77a). In these cases the [SLASH DetA] or [SLASH DetN] propagates in the normal way but is terminated, not by metarule, but by ID rules A2/COMPAR7 and N2/COMPAR7.

6.7 Coordination

Because our system uses fixed-arity term unification (instead of graph-theoretic unification with extension), no simple elegant treatment, such as the GPSG coordination schemata, is available to us. Instead, coordination rules have to be stated for each of the major category types. All of the coordination rules induce the same broad analysis, however, and a schematic example is given in (80).

(80)



Conjunctions are treated as being in constituency with the conjuncts that follow them. Conjuncts with no conjunction, as with the first conjunct in *the dog and the cat*, have the feature specification [CONJ NULL] and expand simply as the conjunct. The order in which conjuncts can occur is controlled by LP rule LP25 which allows the following possibilities: NULL AND, NULL OR, NULL BUT, BOTH AND, NEITHER NOR, EITHER OR.

Rules have been provided for coordinations of N2 (*the dog and the cat*), N1 (*the clever dogs and stupid cats*), N (*the clever dogs and cats*), S (*lee sings and kim dances*), VP (*reads books and understands them*), V (*reads and understands books*), P2 (*on the table and under it*), P (*on and under the table), A2 (exceptionally clever and clearly capable of learning*), A1 (*so clever but so arrogant*) and A (*too clever and arrogant*).

Because GPSG uses a more complex category matching/unification technique, it is able to treat all conjuncts as heads. This is not possible in our grammar, however, since it would entail that all conjuncts had the same values for their head features, and counterexamples to this are numerous. As a result, apart from coordinations of V and A, conjuncts are not treated as heads. Our analysis in effect, treats them as partial heads – for each type of coordination a subset of head features have been identified which must be shared between conjuncts. Various propagation rules ensure correct bindings for these features (see the propagation rules named CONJ/... and sets CONJ_NOMHEAD, CONJ_ADJHEAD, CONJ_VERBHEAD and CONJ_S_HEAD).

The rules for coordinations of nominal categories are more complex than the other coordination rules since an attempt has been made to describe the propagation of number (the feature PLU) in these cases. For example, the grammar reflects the fact that a coordination of N2s with *and* will always be [PLU +], irrespective of the PLU values of the conjuncts, whilst the PLU value of a coordination with *or* is dependent on the PLU values of the conjuncts. A correct statement of person (PER) propagation, on the other hand, has completely defeated the grammar writer.

Whilst it is generally the case that only like categories can be coordinated, there are well known exceptions to this rule. There are particular positions where unlike categories may be coordinated. These positions are, firstly, after copula *be* where any +PRD category may be coordinated with any other +PRD category (*lee is in the bath and singing, lee is a fool and unable to learn* etc) and secondly, in modifier to V2 positions where any 'adverbial' (A2[+ADV] or P2) categories may be coordinated (*lee sings tunelessly but with great gusto, lee works at a fast rate but carelessly*). ID rules named CONJ/PRD.. and PRD/COORD.. allow for coordinations of +PRD categories, whilst ID rules CONJ/MOD.. and MOD/COORD.. allow for coordinations of adverbial categories. The distribution of these coordinations is restricted such that they only appear in the positions outlined above (see ID rules VP/PRD/CONJ, VP/PRD/CONJ2, S/ADV/CONJ and VP/MOD/CONJ).

Extractions out of coordinations may only occur in an 'across-the-board' fashion. That is, if one conjunct contains a gap, then so too must all the other conjuncts (*who is it that John likes e but Bill hates e* vs **who is it that John likes e but Bill hates Mary*). Our grammar contains metarules which propagate SLASH in coordinations. Metarule CONJ/SLASHA passes a SLASH from the top node of a coordination onto all of the conjuncts, thus ensuring the across-the-board phenomenon. The features WH, UB and EVER distribute in a similar way in that if one conjunct is marked for them, all others must be too (*which books and which records do you want* vs **which books and the records do you want*). The FOOT metarules CONJ/FOOTA and CONJ/FOOTD ensure this distribution. All other CONJ/.. metarules are there simply to handle idiosyncratic patterns of propagation not dealt with by the other metarules.

6.8 Negation

An attempt has been made to describe the distribution of the negative word *not*. In non-coordinated sentences, *not* appears to have a relatively limited distribution. It can occur with VPs (*lee is not singing, lee may not have been singing* – see ID rule VP/NEG) and with some quantifying APs (*not many people, not all the people* – see ID rule A2/NOT). In these cases we treat *not* as being in constituency with the category it modifies.

Not also occurs before the +PRD complement of copula *be* (*fido is not a cat, fido is not in the kennel, fido is not barking, fido is not clever*), but here, we assume that it does not form a constituent with the +PRD category; firstly, because such a constituent cannot appear elsewhere (**not a man wants to see you, *I met not a man*) and, secondly, because the *not* never gets preposed with the +PRD constituent (**not a cat is what fido is*). Instead we treat this occurrence of *not* as a daughter of the VP (see ID rules VP/BE*/NEG).

In coordinations, a different pattern emerges; *not* appears to form a constituent with any of the major category types – *I have a dog and not a cat, lee is stupid but not that stupid, I put the butter in the fridge and not in the cupboard*). For this reason there are rules which build a constituent out of *not* followed by an NP, AP, PP or P (see ID rules N2/NEG, P2/NEG, P/NEG and A2/NEG) but these constituents will only be able to appear in coordinate structures.

7. Limitations of the Grammar

Although the grammar has relatively wide-coverage compared to other explicit generative (computational) grammars of which we are aware, there are certain aspects of English to which it does not extend. The following sections provide an indication of what the grammar does not cover.

Punctuation

The grammar does not incorporate a treatment of punctuation; indeed punctuation characters in the input will cause an error in the ANLT parser. (The exception to this is the GDE 'fparse' command which reads sentences from a file for batch processing – here a full stop, question mark or exclamation mark is used to identify the end of a sentence.) Because punctuation is not handled, direct quotations have not been covered, even if punctuation is omitted. (It is controversial whether the linguistic information should be considered 'syntactic', in any case.)

Titles etc.

Titles, dates, addresses, digits, formulae, etc. are also not covered.

Parentheticals

Some types of parenthetical insertions of the kind usually delimited by commas will be parsed, but some will not. For instance, *John, clearly, will head the committee* will receive a parse so long as the commas are omitted, although this parse will be indistinguishable from one where *clearly* is interpreted as a sentence adverb. Other parenthetical insertions such as *John, I think, will head the committee* will not be parsed.

Stylistic Variations

Stylistic inversions, such as 'Heavy NP Shift' (*John handed to the committee a report which he had had specially prepared*) are not covered.

Ellipsis

Ellipsis is not catered for in any coherent way. Some cases of VP ellipsis can be handled because there is a rule (and lexical items) for 'pro-VPs' (e.g. *John hasn't read the report but I have _*). Other cases of ellipsis, such as Gapping (as in *John speaks French and Bill _ German*) are not catered for.

Non-Standard Questions

There are some interrogative structures not covered. Specifically, the grammar does not handle echo-questions (*John read what/which report?*) or multiple wh-questions (*Who has read what/which report?*). Whilst it will recognise tag questions (*John has read the report, hasn't he?*), there is no mechanism for ensuring that the same auxiliary occurs in the main sentence and the tag, so examples such as **he didn't do it, must he?*are parsed.

Anaphor Binding

There has been no attempt to identify antecedents of anaphors or to state agreement between them. The grammar will therefore parse both of the following examples: *John sent a letter to himself, *John sent a letter to herself.*

Tense Sequencing

There is no device in the grammar to ensuring proper sequencing of tenses. Therefore, all of the following will receive parses: *It was John who did it, *it was John who does it, John went to the shops and bought a paper, *John went to the shops and buys a paper.*

Unbounded Dependencies

Some types of unbounded dependency are not covered: VP preposing (*Head the committee, John did*), parasitic gaps (*which committee did John head _ without attending _*) and wh-clefts (*what John did was head the committee*). There is no mechanism for dealing with rightward movement rules, whether they can be said to involve unbounded dependencies or not. So, for example, Right Node Raising constructions are not handled (*John wrote _ and Bill printed _ the report*).

Modification

Whilst the grammar includes adverbials in various positions, it does not distinguish between them according to which position they can occur in. That is, it does not distinguish sentence adverbs from VP adverbs (e.g. manner adverbs), nor does it distinguish those adverbs that can modify adjectives fron those that cannot.

Conditionals

No analysis has been developed for conditional sentences (*If John was in the supermarket then he wasn't in the newsagents, Should you see John, ask him whether he's coming*).

Compounds

Syntactic rules have been included to cover some types of compounds usually written with spaces between their constituents. Compounds written as single words or with hyphenation can be entered in the lexicon and analysed with the morphology system. However, compounds putatively involving N1 categories of either type are not catered for (*menswear, the geologists' association excursion*).

Comparatives

There are examples of more idiomatic comparatives involving more idiosyncratic comparative-introducing lexical items than *more... than,* such as *such... that, as little as... of,* as well as wholely idiomatic comparatives, such as *as much God's handiwork as a man,* which cannot be parsed, presently.

Finally, no doubt there are many gaps in our existing coverage of which we are unaware. We would be grateful for information about such gaps.

8. References

Borsley, R. 1983. A Welsh Agreement Process and the Status of VP and S. In Gazdar, G., Klein, E. & Pullum, G. (eds.) *Order, Concord & Constituency*. Foris, Dordrecht.

Briscoe, E., C. Grover, B. Boguraev & J. Carroll 1987a. A Formalism and Environment for the Development of a Large Grammar of English. *Proc. of Int. Joint Conf. on Artificial Intelligence*, Milan, Italy, pp. 703-8.

Briscoe, E., C. Grover, B. Boguraev & J. Carroll 1987b. Feature Defaults, Propagation and Reentrancy. In Klein, E. & van Bentham, J. (eds.) *Categories, Polymorphism and Unification*. Centre for Cognitive Science, University of Edinburgh, pp. 19-34.

Carroll, J., B. Boguraev, C. Grover & E. Briscoe 1988. The Grammar Development Environment User Manual. University of Cambridge, Computer Laboratory Technical Report No. 127.

Carroll, J. & C. Grover 1989. The Derivation of a Large Computational Lexicon for English from LDOCE. In Boguraev, B. & Briscoe, E. (eds.) *Computational Lexicography for Natural Language Processing*. Longman, London.

Gazdar, G. 1980. A Phrase Structure Syntax for Comparative Clauses. In Hoekstra, T., van der Hulst, H. & Moortgat, M. (eds.) *Lexical Grammar*. Foris, Dordrecht.

Gazdar, G., E. Klein, G. Pullum & I. Sag 1985. *Generalized Phrase Structure Grammar*. Blackwell, Oxford.

Grover, C., E. Briscoe, J. Carroll & B. Boguraev 1988. The Alvey Natural Language Tools Project Grammar – A large compuational grammar of English. *Lancaster Working Papers in Linguistics*, 47, Department of Linguistics, University of Lancaster.

Harlow, S. 1986. Gaps in Generalized Phrase Structure Grammar. *York Papers in Linguistics 12*. University of York.

Kilbury, J. 1986. Category Cooccurrence Restrictions and the Elimination of Metarules. *Proc. of 11th Int. Conf. on Computational Linguistics*, Bonn, pp. 50-55.

Phillips, J. & H. Thompson 1987. A Parser and an Appropriate Computational Representation for GPSG. In Klein, E. & N. Haddock (eds.) *Cognitive Science Working Papers*, 1, Centre for Cognitive Science, University of Edinburgh.

Pulman, S., G. Russell, G. Ritchie & A. Black 1988. Computational Morphology of English. *Linguistics*, 26, 545-560.

Sells, P. 1985. *Lectures on Contemporary Syntactic Theories*. CSLI Lecture Notes 3, Stanford University, Stanford and University of Chicago Press.

Warner, A. 1984. The Structuring of English Auxiliaries: A Phrase Structure Grammar. *York Papers in Linguistics*. Dept. of Linguistics, University of York.

Appendix 1. Grammar Listing

; feature declarations

```
FEATURE N{+, -}
FEATURE V{+, -}
FEATURE BAR{-1, 0, 1, 2}
FEATURE SUBJ{+, -}
FEATURE H{+, -}
FEATURE T{+, -}
FEATURE VFORM{BSE, ING, EN, TO, NOT}
FEATURE FIN{+, -}
FEATURE PAST{+, -, NOT}
FEATURE PRD{+, -}
FEATURE AUX{+, -}
FEATURE AGR CAT
FEATURE INV{+, -}
FEATURE PSVE{+, -}
FEATURE NEG{+, -}
FEATURE COMP{THAT, FOR, IF, WHETHER, NORM, THAN, AS}
FEATURE NFORM{NORM, THERE, IT}
FEATURE PER{1, 2, 3}
FEATURE PLU{+, -}
FEATURE COUNT{+, -}
FEATURE CASE{NOM, ACC}
FEATURE PN{+, -}
FEATURE PRO{+, -}
FEATURE PART{OF, OF2, NO_OF}
FEATURE POSS{+, -}
FEATURE DEF{+, -}
FEATURE SPEC{+, -}
FEATURE PFORM{ABOUT, ABOUT_TO, ABOUT_WITH, ABOVE, ACROSS, AFTER,
   AGAINST, ALL, ALONG, AMONG, AMONGST, AROUND, AS, AS_TO, AT,
   AT_ABOUT, AT_FOR, AWAY, BEFORE, BEHIND, BELOW, BESIDE, BETWEEN, BY,
   BY_FOR, DOWN, DOWN_TO, DURING, FOR, FOR_TO, FOR_WITH, FROM,
   FROM_FOR, FROM_INTO, FROM_TO, IN, INTO, INTO_WITH, IN_AT,
   IN_FAVOUR_OF, IN_FRONT_OF, IN_WITH, LIKE, OF, OFF, OF_AS, OF_FOR,
   ON, ONTO, ON_AS, ON_BOARD, ON_FOR, ON_TO, ON_WITH, OUT, OUT_OF,
   OVER, OVER_WITH, ROUND, THAN, THROUGH, THROUGHOUT, TILL, TO,
   TOGETHER, TOWARD, TOWARDS, TO_AGAINST, TO_AS, TO_FOR, TO_FROM,
   UNDER, UNTIL, UP, UPON, UP_TO, WITH, WITHOUT, WITH_ABOUT,
   WITH_AGAINST, WITH_AT, WITH_FOR, WITH_ON, WITH_OVER, NORM}
FEATURE LOC{+, -}
FEATURE GERUND{+, -}
FEATURE AFORM{ER, EST, NONE, AS}
FEATURE QUA{+, -}
FEATURE ADV{+, -}
FEATURE NUM{CARD, ORD, -}
FEATURE SUBCAT{ADL, ADVP, ADVP_PP, IT_WHS, LOC, MP, NP, NP_ADL,
   NP_ADVP, NP_LOC, NP_MP, NP_NP, NP_NP_SFIN, NP_PP, NP_PP_PP,
   NP_SBSE, NP_SFIN, NP_WHS, NP_WHVP, NULL, OC_AP, OC_BSE, OC_INF,
   OC_ING, OC_NP, OC_PASS, OC_PP_BSE, OC_PP_INF, OC_PP_ING, PP, PPING,
   PPSING, PP_PP, PP_SBSE, PP_SFIN, PP_SINF, PP_VPINF, PP_WHS,
   PP_WHVP, SBSE, SC_AP, SC_BSE, SC_INF, SC_ING, SC_NP, SC_NP_AP,
   SC_NP_INF, SC_NP_NP, SC_PASS, SC_PP_INF, SFIN, SINF, VPINF, WHS,
   WHVP, AND, BOTH, BUT, EITHER, NEITHER, NOR, OR, DETA, DETN, AS,
   FOR, IF, THAN, THAT, WHETHER, NOT, BE, DO, HAVE, OUGHT, MODAL, TO}
```

```
FEATURE WH{+, -}
FEATURE UB{R, Q, -}
FEATURE EVER{+, -}
FEATURE SLASH CAT
FEATURE NULL{+, -}
FEATURE MOD{PRE, POST, NONE}
FEATURE CONJ{NULL, BOTH, NEITHER, EITHER, AND, OR, NOR, BUT}
FEATURE CONJN{+, -}
FEATURE COORD{+, -}
FEATURE REFL{+, -}
FEATURE PRT{ABACK, ABOUT, ABOVE, ABROAD, ACROSS, AGAIN, AHEAD, ALONG,
   ALOUD, APART, AROUND, ASIDE, ASTRAY, AWAY, BACK, BEHIND, BY, DOWN,
   FAR, FLAT, FOR, FORTH, FORWARD, FORWARDS, HOME, IN, INTO, LOOSE,
   LOW, OFF, ON, OPEN, OUT, OVER, OVER_WITH, ROUND, THROUGH, TO,
   TOGETHER, UNDER, UP, UPON, WASTE, WITH, WITHOUT}
FEATURE ADDRESS{+, -}
FEATURE DISTR{PRD, PST, ATT, PREPP}
FEATURE PREP{ABOUT, AGAINST, AS, AT, BY, FOR, FROM, IN, OF, ON,
   OUT_OF, TO, UPON, WITH}
FEATURE SUBTYPE{ASIF, IF, EQUI, EQU_EXTRAP, EXTRAP, PVERB, PVERB_OE,
   PVERB_OR, RAIS, READY, SILLY, TOUGH, NONE}
FEATURE CN1{A, ZERO, TEN, TEEN, TY, HUN, THOU}
FEATURE CN2{BIG, SMALL}
FEATURE AND{+, -}
FEATURE TAG{+, -}
```

; features relevant to the morphology system but not to the grammar

```
FEATURE AT{+, -}
FEATURE LAT{-, +}
FEATURE FIX{PRE, SUF, NOT}
FEATURE INFL{-, +}
FEATURE STEM CAT
FEATURE COMPOUND{N, V, A, NOT}
FEATURE CAT{A, N, P, V, NONE, MINOR}
FEATURE MAJ{+, -}
FEATURE REG{+, -}
FEATURE ARITY{0, 1, 2, 3, 4}
FEATURE COMPAR{INFL, YES, NO}
FEATURE GROUP{+, -}
FEATURE ORDER{FREE, PRENP, POSTNP}
```

; set declarations

```
SET VERBALHEAD = {PRD, FIN, AUX, VFORM, PAST, AGR}
SET NOMINALHEAD = {PLU, POSS, CASE, PRD, PN, PRO, COUNT, NFORM, PER,
   REFL, NUM}
SET PREPHEAD = {PRO, PFORM, LOC, PRD}
SET ADJHEAD = {PRD, QUA, ADV, NUM, NEG, AGR, DEF, DISTR}
SET FOOT = {WH, UB, EVER}
SET AGRFEATS = {NFORM, PLU, COUNT, PER, CASE}
SET CONJ_NOMHEAD = {POSS, CASE, PRD, NFORM}
SET CONJ_ADJHEAD = {ADV, AGR, QUA}
SET CONJ_VERBHEAD = {PRD, FIN, VFORM, AGR}
SET CONJ_S_HEAD = {PRD, FIN, VFORM, AGR, COMP, INV}
SET MORPHOLOGYONLY = {AT, LAT, COMPOUND, FIX, INFL, STEM, CAT, REG,
   MAJ, ARITY, COMPAR, ORDER}
SET WHEAD = {N, V, INFL, PAST, AFORM, VFORM, ADV, PLU, AT, NUM, REG,
   LAT, PRD, FIN, COUNT, QUA, PART, PRO, PN, PER, NFORM, POSS, PSVE}
SET WDAUGHTER = {SUBCAT, AUX, PFORM, PRT, PREP, AGR, SUBTYPE, INV, NEG}
```

; alias declarations

```
ALIAS +N = [N +].
ALIAS +V = [V +].
ALIAS V = [V +, N -, BAR 0].
ALIAS N = [N +, V -, BAR 0].
ALIAS A = [V +, N +, BAR 0].
ALIAS P = [V -, N -, BAR 0].
ALIAS P1 = [N -, V -, BAR 1].
ALIAS N1 = [N +, V -, BAR 1].
ALIAS VP = [N -, V +, BAR 2, SUBJ -].
ALIAS A1 = [N +, V +, BAR 1].
ALIAS N2 = [N +, V -, BAR 2].
ALIAS A2 = [N +, V +, BAR 2].
ALIAS S = [N -, V +, BAR 2, SUBJ +].
ALIAS P2 = [N -, V -, BAR 2].
ALIAS V2 = [V +, N -, BAR 2].
ALIAS ADVP = [BAR 2, ADV +].
ALIAS DetN = [SUBCAT DETN, QUA +].
ALIAS DetA = [SUBCAT DETA].
ALIAS H = [H +, BAR 0].
ALIAS H1 = [BAR 1, H +].
ALIAS H2 = [BAR 2, H +].
ALIAS X = [].
ALIAS X0 = [BAR 0].
ALIAS X1 = [BAR 1].
ALIAS X2 = [BAR 2].
ALIAS T = [T +].
ALIAS BSE = [VFORM BSE].
ALIAS EN = [VFORM EN].
ALIAS TO = [VFORM TO].
ALIAS ING = [VFORM ING].
ALIAS PSP = [VFORM EN, PRD -].
ALIAS PAS = [VFORM EN, PRD +].
ALIAS PRP = [VFORM ING, PRD +].
ALIAS GER = ; GER is alias for V categories ie V, VP, S. GERUND as in
            ; the next two aliases is a feature for prepositions -
            ; didn't want to clutter up PPs with VFORM and PRD
   [VFORM ING, PRD -].
ALIAS IMP = [FIN +, VFORM BSE].
ALIAS +PRD = [PRD +].
ALIAS -PRD = [PRD -].
ALIAS +GER = [GERUND +].
ALIAS -GER = [GERUND -].
ALIAS +POSS = [POSS +].
ALIAS -POSS = [POSS -].
ALIAS +ADV = [ADV +].
ALIAS -ADV = [ADV -].
ALIAS +INV = [INV +].
ALIAS -INV = [INV -].
ALIAS +NOM = [CASE NOM].
ALIAS +ACC = [CASE ACC].
ALIAS +NULL = [NULL +].
ALIAS +NEG = [NEG +].
ALIAS -NEG = [NEG -].
ALIAS +FIN = [FIN +].
ALIAS -FIN = [FIN -].
ALIAS +SUBJ = [SUBJ +].
ALIAS -SUBJ = [SUBJ -].
ALIAS +LOC = [LOC +].
ALIAS -AUX = [AUX -].
ALIAS +AUX = [AUX +].
ALIAS +DEF = [DEF +].
ALIAS -DEF = [DEF -].
ALIAS +SPEC = [SPEC +].
ALIAS -SPEC = [SPEC -].
ALIAS +PLU = [PLU +].
ALIAS -PLU = [PLU -].
ALIAS +PRO = [PRO +].
ALIAS -PRO = [PRO -].
ALIAS +QUA = [QUA +].
ALIAS -QUA = [QUA -].
ALIAS +COUNT = [COUNT +].
ALIAS -COUNT = [COUNT -].
ALIAS CARD = [NUM CARD].
ALIAS ORD = [NUM ORD].
ALIAS -NUM = [NUM -].
ALIAS +R = [UB R].
ALIAS +Q = [UB Q].
ALIAS +WH = [WH +].
ALIAS -WH = [WH -].
ALIAS +EVER = [EVER +].
ALIAS -EVER = [EVER -].
ALIAS by = [PFORM BY].
ALIAS that = [COMP THAT].
ALIAS er = [AFORM ER].
ALIAS est = [AFORM EST].
ALIAS as = [AFORM AS].
ALIAS +PN = [PN +].
```

; category declarations

```
CATEGORY S : S => {COMP, INV}.
CATEGORY S_WH : S[UB R] => {WH, EVER}.
CATEGORY VERB : [N -, V +] => VERBALHEAD.
CATEGORY VP : VP => {NEG}.
CATEGORY NOUN : [N +, V -] => NOMINALHEAD.
CATEGORY N1 : N1 => {MOD}.
CATEGORY N2 : N2 => {SPEC, DEF, NEG}.
CATEGORY PREP : [N -, V -] => PREPHEAD.
CATEGORY P : P => {SUBCAT}.
CATEGORY P1 : P1 => {POSS, GERUND}.
CATEGORY P2 : P2 => {POSS, GERUND, NEG}.
CATEGORY ADJ : [N +, V +] => ADJHEAD.
CATEGORY A : A => {DISTR}.
CATEGORY A2 : A2 => {DISTR}.
CATEGORY AGR : (AGR) N2 => {PLU, PER, COUNT, NFORM, CASE}.
CATEGORY AGR2 : (AGR) V2 => {FIN, SUBJ, VFORM, UB}.
CATEGORY AGR3 : (AGR) [N +, V -]
    => {BAR, PLU, PER, COUNT, NFORM, CASE}.
CATEGORY COORD : [BAR 2] => {COORD}.
CATEGORY AFORM_1 : [N +, V +] => {AFORM}.
CATEGORY AFORM_2 : DetA => {AFORM}.
CATEGORY AFORM_3 : N1 => {AFORM}.
CATEGORY AFORM_4 : N2 => {AFORM}.
CATEGORY SUBTYPE : [V +, BAR 0] => {SUBTYPE}.
CATEGORY PRD/CONJ : [BAR 2, PRD +, ~N] => {NEG, AGR}.
CATEGORY NUM : [CN2 (BIG, SMALL, @)] => {AND}.
```

; lcategory declarations

```
LCATEGORY W_VERB : [N -, V +] => VERBALHEAD.
LCATEGORY W_V : V => {NEG, INV, PSVE, SUBTYPE}.
LCATEGORY W_NOUN : [N +, V -] => NOMINALHEAD.
LCATEGORY W_PRONOUN : N[+PRO] => {DEF, AFORM}.
LCATEGORY W_PREP : [N -, V -] => PREPHEAD.
LCATEGORY W_P : P => {NEG}.
LCATEGORY W_ADJ : [N +, V +] => ADJHEAD.
LCATEGORY W_A : [N +, V +] => {AFORM}.
LCATEGORY W_A2 : A => {DISTR}.
LCATEGORY W_DETN : [QUA +, SUBCAT DETN] => {DEF, POSS, AGR}.
LCATEGORY W_DETA : DetA => {AFORM}.
LCATEGORY W_AGR_V2 : (AGR) V2 => {FIN, SUBJ, VFORM, UB}.
LCATEGORY W_AGR_N : (AGR) [N +, V -]
    => {BAR, PLU, PER, COUNT, NFORM, CASE}.
LCATEGORY W_SLASH_N2 : (SLASH) N2 => {PLU, COUNT, PER, NFORM, CASE}.
LCATEGORY W_SLASH_P2 : (SLASH) P2 => {PFORM, LOC, GERUND}.
LCATEGORY W_SLASH_A2 : (SLASH) A2 => {ADV}.
LCATEGORY W_NUM : [CN2 (BIG, SMALL, @)] => {AND}.
```

; extension declaration

```
EXTENSION {WH, UB, EVER, SLASH, NULL}
```

; propagation rules

```
PROPRULE PROP_HEAD_V : ; copy value of V from mother to head daughter
   [V (+, -)] -> [H +], U. V(1) = V(0).
PROPRULE PROP_HEAD_N : ; copy value of N from mother to head daughter
   [N (+, -)] -> [H +], U. N(1) = N(0).
PROPRULE PROP_BAR : ; non-lexical heads have same BAR level as mother
   [] -> [H +, ~SUBCAT, ~BAR], U. BAR(0) = BAR(1).
PROPRULE HFC_VERBAL : ; head feature propagation for verbal
                      ; categories
   [N -, V +] -> [H +], U. F(0) = F(1), F in VERBALHEAD.
PROPRULE HFC_NOMINAL : ; head feature propagation for nominal
                       ; categories
   [N +, V -] -> [H +], U. F(0) = F(1), F in NOMINALHEAD.
PROPRULE HFC_ADJ : ; head feature propagation for adjectival
                   ; categories
   [N +, V +] -> [H +], U. F(0) = F(1), F in ADJHEAD.
PROPRULE HFC_PREP : ; head feature propagation for prepositional
                    ; categories
   [N -, V -] -> [H +], U. F(0) = F(1), F in PREPHEAD.
PROPRULE PP : ; POSS and GERUND not head features (only appear on P2
              ; and P1) since the values come from the NP not from
              ; P0. This passes them from a P2 mother to either a P1
              ; or a P2 head daughter.
   P2 -> [H +, BAR (1, 2)], U. F(0) = F(1), F in {POSS, GERUND}.
PROPRULE VP/NEG : ; passes neg (not a head feature) from a V0 to its
                  ; mother
   VP -> H, U. NEG(0) = NEG(1).
PROPRULE S : ; to pass nonhead features between S mother and S head
   S -> [H +, +SUBJ], U. F(0) = F(1), F in {COMP, INV}.
PROPRULE DEF1 : ; binds DEF and AFORM on N[+PRO] to mother N2[+SPEC]
   N2[+SPEC] -> H[PRO +]. F(0) = F(1), F in {DEF, AFORM}.
PROPRULE DEF2 : ; binds DEF on A2[+QUA, +PRD] to N2[-SPEC] mother
   N2[-SPEC] -> A2[+QUA, +PRD], H1. DEF(0) = DEF(1).
PROPRULE DEF3 : ; binds DEF on [+QUA] (DetN or AP) to N2[+SPEC]
                ; mother
   N2[+SPEC] -> [+QUA], U. DEF(0) = DEF(1).
PROPRULE DEF4 : ; binds DEF on the N2 heads of the partitive rules to
                ; their mothers
   N2[+SPEC] -> N2[H +, SPEC +, PART], U. DEF(0) = DEF(1).
PROPRULE AGR/NP_VP : ; binds feature values on subject N2 to feature
                     ; values on the category that is the value of
                     ; VP[AGR]
   S -> N2, H2[-SUBJ, AGR N2], U. F(1) = F(2[AGR]), F in AGRFEATS.
PROPRULE AGR/V2_VP : ; five kinds of V2 subject are possible -
                     ; S[that, FIN +], S[that, VFORM BSE], S[for, FIN
                     ; -], S[+Q] and VP[VFORM TO, FIN -]. This
                     ; proprule binds the appropriate features
                     ; between V2 and AGR V2.
   S -> V2, H2[-SUBJ, AGR V2].
   F(1) = F(2[AGR]), F in {FIN, SUBJ, VFORM}.
PROPRULE AGR/INV : ; to bind subject NP to AGR value on verb in
                   ; inverted sentences
   S[+INV] -> H, N2[+NOM], W. F(1[AGR]) = F(2), F in AGRFEATS.
PROPRULE AGR/SPEC_N : ; binds AGR features on +QUA categories to
                      ; features on their nominal head sisters.
   N2 -> [+QUA, AGR +N], [H +]. F(1[AGR]) = F(2), F in {PLU, COUNT}.
PROPRULE SUBJ_CONTROL1 : ; establishes subject control agreement
                         ; pattern for all subject raising and
                         ; subject equi predicates by binding AGR on
                         ; the VP/AP complement to AGR on the VP/AP
                         ; mother.
   [V +, AGR N2] -> H[SUBTYPE (RAIS, EQUI)], [V +, BAR 2, AGR N2].
   F(0[AGR]) = F(2[AGR]), F in AGRFEATS.
PROPRULE SUBJ_CONTROL2 : ; subject control agreement pattern for
                         ; auxiliary rules.
   V2[+AUX, AGR N2] -> [V +, BAR 2, AGR N2], W.
   F(0[AGR]) = F(1[AGR]), F in AGRFEATS.
PROPRULE SUBJ_CONTROL3 : ; subject control for various passive VPs -
                         ; VP/OR*/PASS_A and VP/OE*(PASSIVE). The
                         ; ~SLASH on the mother prevents it applying
                         ; to the outputs of STM2.
   VP[PAS, AGR N2, ~SLASH] -> [V +, BAR 2, AGR N2], W.
   F(0[AGR]) = F(1[AGR]), F in AGRFEATS.
PROPRULE SUBJ_CONTROL4 : ; agreement binding for phrasal subject
                         ; control verbs.
   [V +, AGR N2] -> H[SUBTYPE (RAIS, EQUI)], [V +, BAR 2, AGR N2],
   [PRT @]. F(0[AGR]) = F(2[AGR]), F in AGRFEATS.
PROPRULE SUBJ_CONTROL5 : ; agreement binding for prepositional
                         ; subject control verbs.
   [V +, AGR N2] -> H[SUBTYPE (RAIS, EQUI)], [V +, BAR 2, AGR N2],
   [PFORM @]. F(0[AGR]) = F(2[AGR]), F in AGRFEATS.
PROPRULE SUBJ_CONTROL6 : ; for verbs like 'promise' and 'strike as'
                         ; which appear with an object but are
                         ; actually subject control.
   VP[AGR N2] -> H[SUBCAT (SC_NP_INF, SC_NP_AP)],
   [V +, BAR 2, AGR N2], W. F(0[AGR]) = F(2[AGR]), F in AGRFEATS.
PROPRULE OBJ_CONTROL : ; establishes object control agreement pattern
                       ; by binding AGR on the VP complement to the
                       ; N2 object. The SUBTYPE list excludes
                       ; application to VP/SE_NP_INF (promise).
   VP -> H[SUBTYPE (RAIS, EQUI, PVERB_OE, PVERB_OR, EQU_EXTRAP)], N2,
   [V +, BAR 2, AGR N2], U.
   F(2) = F(3[AGR]), F in {PLU, PER, COUNT, NFORM}.
PROPRULE AGRV2_CONTROL : ; subject control for AGR V2 versions of
                         ; subject raising predicates
   [V +, AGR V2] -> H, [V +, BAR 2, AGR V2], W.
   F(0[AGR]) = F(2[AGR]), F in {FIN, SUBJ, VFORM, UB}.
PROPRULE SLASH/AGR_NP : ; agreement between preposed N2 and the
                        ; category value of SLASH
   V2 -> N2, V2[H +, SLASH N2]. F(1) = F(2[SLASH]), F in AGRFEATS.
PROPRULE SLASH/AGR_PP1 : ; agreement between preposed P2 and the
                         ; category value of SLASH
   S -> P2, S[H +, SLASH P2].
   F(1) = F(2[SLASH]), F in {LOC, PFORM, GERUND}.
PROPRULE SLASH/AGR_PP2 : ; agreement between P2 and value of SLASH in
                         ; VP/BE_CLEFT2
   V2 -> P2, S[+FIN, SLASH P2], W.
   F(1) = F(2[SLASH]), F in {LOC, PFORM, GERUND}.
PROPRULE SLASH/AGR_AP : ; agreement between preposed A2 and the
                        ; category value of SLASH
   S -> A2, S[H +, SLASH A2]. ADV(1) = ADV(2[SLASH]).
PROPRULE BIND_SLASH1 : ; operates on output of STM1_N2. Binds
                       ; AGRFEATS on null N2 to the N2 that's the
                       ; value of its SLASH
   [] -> N2[+NULL, SLASH N2], W. F(1) = F(1[SLASH]), F in AGRFEATS.
PROPRULE BIND_SLASH1A : ; operates on output of STM1_N2. Binds the N2
                        ; value of SLASH on the mother to N2 value of
                        ; SLASH on the daughter
   X[SLASH N2] -> N2[+NULL, SLASH N2], W.
   F(0[SLASH]) = F(1[SLASH]), F in AGRFEATS.
PROPRULE BIND_SLASH2 : ; operates on output of STM1_P2 and
                       ; STM/COMPAR. Binds LOC and PFORM on null P2
                       ; to the P2 that's the value of its SLASH
   [] -> P2[+NULL, SLASH P2], U.
   F(1) = F(1[SLASH]), F in {LOC, PFORM, GERUND}.
PROPRULE BIND_SLASH2A : ; operates on output of STM1_P2 and
                        ; STM/COMPAR. Binds the P2 value of SLASH on
                        ; the mother to P2 value of SLASH on the
                        ; daughter
   X[SLASH P2] -> P2[+NULL, SLASH P2], U.
   F(0[SLASH]) = F(1[SLASH]), F in {PFORM, LOC, GERUND}.
PROPRULE BIND_SLASH3 : ; operates on output of STM1_A2. Binds ADV on
                       ; null A2 to the A2 that's the value of its
                       ; SLASH
   [] -> A2[+NULL, SLASH A2], W. ADV(1) = ADV(1[SLASH]).
PROPRULE BIND_SLASH3A : ; operates on output of STM1_A2. Binds the A2
                        ; value of SLASH on the mother to A2 value of
                        ; SLASH on the daughter
   X[SLASH A2] -> A2[+NULL, SLASH A2], W.
   ADV(0[SLASH]) = ADV(1[SLASH]).
PROPRULE BIND_SLASH4 : ; operates on the output of STM2 binding the
                       ; values of the missing subject to the AGR
```

```
                         ; value of the verb.
   VP[SLASH N2] -> VP[~SLASH, +FIN], W.
   F(0[SLASH]) = F(1[AGR]), F in AGRFEATS.
PROPRULE BIND_SLASH4A : ; operates on the output of STM2_B binding
                         ; the values of the missing subject to the
                         ; AGR value of the adjective.
   A1[SLASH N2] -> VP[~SLASH, +FIN], W.
   F(0[SLASH]) = F(1[AGR]), F in AGRFEATS.
PROPRULE CONJ/SUBCAT : ; propagates SUBCAT in lexical conjuncts
   [BAR 0, CONJ (NULL, @)] -> [BAR 0], U. SUBCAT(0) = SUBCAT(1).
PROPRULE CONJ/NOMHEAD : ; propagates a subset of nominalhead features
                         ; between an N2 mother and its conjunct
                         ; daughters - in effect the conjuncts are
                         ; partial heads.
   N2 -> N2[CONJ], U. F(0) = F(1), F in CONJ_NOMHEAD.
PROPRULE CONJ/N2_A : ; passes SPEC between an N2 mother and its
                      ; conjunct daughters.
   N2 -> N2[CONJ], U. SPEC(0) = SPEC(1).
PROPRULE CONJ/N2_B : ; to pass nonhead features in rules which expand
                      ; an N2 conjunct as a coordinator and an N2
                      ; head.
   N2 -> [CONJN +], H2. F(0) = F(2), F in {SPEC, NEG}.
PROPRULE CONJ/N2_C : ; to pass nonhead features where an N2 conjunct
                      ; has a null coordinator
   N2[CONJ NULL] -> H2. F(0) = F(1), F in {SPEC, NEG}.
PROPRULE CONJ/ADJHEAD : ; passes CONJ_ADJHEAD features between a
                         ; mother and its adjectival conjunct
                         ; daughters
   [N +, V +] -> [CONJ], U. F(0) = F(1), F in CONJ_ADJHEAD.
PROPRULE CONJ/VERBHEAD : ; passes CONJ_VERBHEAD features between a
                          ; verbal (VP or V) mother and its conjunct
                          ; daughters
   [V +, N -] -> [V +, N -, CONJ], U. F(0) = F(1), F in CONJ_VERBHEAD.
PROPRULE CONJ/S_HEAD : ; to pass CONJ_S_HEAD features between an S
                        ; mother and its conjunct daughters
   S -> S[CONJ], U. F(0) = F(1), F in CONJ_S_HEAD.
PROPRULE CONJ/S_A : ; to pass nonhead features in rules which expand
                     ; an S conjunct as a coordinator and an S head.
   S -> S[H +], [CONJN +]. F(0) = F(1), F in {INV, COMP}.
PROPRULE CONJ/S_B : ; to pass nonhead features where an S conjunct
                     ; has a null coordinator
   S[CONJ NULL] -> S[H +]. F(0) = F(1), F in {INV, COMP}.
PROPRULE CONJ/P2 : ; ensures that P2 conjuncts share POSS and GERUND
                    ; with their mother.
   P2 -> P2[CONJ], U. F(0) = F(1), F in {POSS, GERUND}.
PROPRULE CONJ/PREP_A : ; to pass NEG (a nonhead feature) in rules
                        ; which expand a prepositional conjunct as a
                        ; coordinator and a prepositional head.
   [V -, N -] -> [CONJN +], [H +]. NEG(0) = NEG(2).
PROPRULE CONJ/PREP_B : ; to pass NEG where an prepositional conjunct
                        ; has a null coordinator
   [V -, N -, CONJ NULL] -> [H +]. NEG(0) = NEG(1).
PROPRULE CONJ/PREP_C : ; propagates PFORM (but not the other prephead
                        ; features) between a prepositional mother
                        ; (P2, P) and its conjunct daughters - in
                        ; effect the conjuncts are partial heads
   [V -, N -] -> [V -, N -, CONJ], U. PFORM(0) = PFORM(1).
PROPRULE A2/AFORM : ; passes AFORM (er, as etc) from A1 to A2.
```

```
   A2 -> [H +], U. AFORM(0) = AFORM(1).
PROPRULE A1/AFORM_1 : ; the AFORM value of an A1 mother in a lexical
                      ; id rule is passed from the head daughter
   A1 -> H, W. AFORM(0) = AFORM(1).
PROPRULE A1/AFORM_2 : ; the AFORM value of an A1 mother in a
                      ; non-lexical id rule comes from the DetA (more
                      ; stupid, as stupid)
   A1 -> DetA, H1. AFORM(0) = AFORM(1).
PROPRULE N1/AFORM : ; N1 and N2 also have AFORM since they
                    ; participate in comparatives. Here the AFORM
                    ; comes from the prenominal adjective (smaller
                    ; dogs)
   N1 -> A2[-PRD, -QUA], H1. AFORM(0) = AFORM(1).
PROPRULE N2/AFORM_1 : ; The AFORM value of an N2 containing a DetN
                      ; passes from the head (a smaller dog).
   N2[+SPEC] -> DetN, H2. AFORM(0) = AFORM(2).
PROPRULE N2/AFORM_2 : ; The AFORM value of an N2 expanding just as an
                      ; N1 head passes from that head (smaller dogs)
   N2 -> H1. AFORM(0) = AFORM(1).
PROPRULE N2/AFORM_3 : ; When N2 expands as a quantifier plus head N1,
                      ; AFORM passes from the quantifier to the
                      ; mother (as many dogs)
   N2[-SPEC] -> A2[+QUA], H1. AFORM(0) = AFORM(1).
PROPRULE PART : ; to make the head and N2 in partitives agree (except
                ; for [PART OF2] see idrule N2+/PART3 where they
                ; mustn't agree).
   N2 -> H2[PART (OF, NO_OF)], N2, U. F(1) = F(2), F in {PLU, COUNT}.
PROPRULE CONJ/PRD/NEG : ; to percolate NEG in +PRD conjuncts
   X2[PRD +, CONJ (NULL, @)] -> [BAR 2, +PRD], U. NEG(0) = NEG(1).
PROPRULE CONJ/PRD/AGR : ; to percolate AGR in +V, +PRD conjuncts
   X2[PRD +, CONJ (NULL, @)] -> [BAR 2, V +, +PRD], U.
   AGR(0) = AGR(1).
PROPRULE PRD/COORD : ; to percolate AGR in +PRD coordinations.
   X2[+PRD] -> X2[+PRD, CONJ], U. AGR(0) = AGR(1).
PROPRULE PRD/AGR : ; to achieve subject control when a +PRD
                   ; coordination follows 'be'.
   VP -> H[SUBCAT BE], X2[COORD +], W. AGR(0) = AGR(2).
PROPRULE V+/CONJ : ; to percolate SUBTYPE in coordinations of Vs or
                   ; As.
   [V +, BAR 0] -> [V +, BAR 0, H +], U. SUBTYPE(0) = SUBTYPE(1).
PROPRULE CONJ/NA : ; ensures that conjoined Ns have the same PN and
                   ; PRO values
   N -> N[CONJ], U. F(0) = F(1), F in {PN, PRO}.
PROPRULE CONJ/NB : ; ensures that conjoined Ns have the same PN and
                   ; PRO values
   N[CONJ (NULL, @)] -> N, U. F(0) = F(1), F in {PN, PRO}.
PROPRULE CONJ/SUBCAT2 : ; ensures that conjoined lexical categories
                        ; have the same SUBCAT value
   [BAR 0] -> [BAR 0, CONJ], U. SUBCAT(0) = SUBCAT(1).
PROPRULE COMPOUND/SUBCAT : ; propagates SUBCAT from the head in
                           ; compounds
   [BAR 0, ~CONJ] -> H[~CONJ], U. SUBCAT(0) = SUBCAT(1).
PROPRULE POST_PASSIVE : ; the AGR feature on mother and head in
                        ; PASSIVE expanded VP rules ends up not being
                        ; bound so this does it again
   VP[PAS] -> H, W. F(0[AGR]) = F(1[AGR]), F in AGRFEATS.
```

; default rules

```
DEFRULE DEF_BAR0 : ; lexical heads are BAR 0
   [] -> [H +, SUBCAT], U. BAR(1) = 0.
DEFRULE VP_VFORM : ; defaults -FIN onto rhs VPs which have a VFORM
                   ; value. Only imperatives are VFORM BSE, FIN + and
                   ; the imperative rule is marked as such.
   [] -> VP[VFORM], U. FIN(1) = -.
DEFRULE VFORM_NOT : ; defaults all finite VPs and S's to VFORM NOT
                    ; except the S dominated by T (since this could
                    ; be imperative).
   [~T] -> V2[FIN +], U. VFORM(1) = NOT.
DEFRULE V/PASS : ; defaults all passive verbs to PSVE +.
   VP -> H[VFORM EN, PRD +], W. PSVE(1) = +.
DEFRULE V/UNPASS : ; defaults all other verbs to PSVE -. (To ensure
                   ; that passive verbs won't match active rules.)
   VP -> H, W. PSVE(1) = -.
DEFRULE S_VFORM : ; defaults -FIN onto rhs S with VFORM value
   [] -> S[VFORM], U. FIN(1) = -.
DEFRULE RHS_N2_CASE1 : ; defaults N2s inside VP to CASE ACC
   VP -> N2, U. CASE(1) = ACC.
DEFRULE RHS_N2_CASE : ; defaults N2s inside PP to CASE ACC
   P1 -> N2, U. CASE(1) = ACC.
DEFRULE RHS_N1 : ; defaults -PRO onto N1 to prevent it from
                 ; dominating pronouns
   N1 -> [H +], U. PRO(1) = -.
DEFRULE SLASH_N2A : ; for S/N2 which isn't introduced by the S/UDC
                    ; rule and which therefore doesn't have features
                    ; propagated onto it by proprule SLASH/AGR_NP
   S -> S[H +, SLASH N2], U. F(1[SLASH]) = @, F in AGRFEATS.
DEFRULE VP/AGR : ; VPs default to AGR N2[NFORM NORM]
   VP -> W. AGR(0) = N2[NFORM NORM].
DEFRULE VP_PRD : ; defaults rhs VP to -PRD so the only time passives
                 ; or present participles will turn up is after 'be'
                 ; (and as an N modifier)
   [~CONJ] -> VP[~CONJ], W. PRD(1) = -.
DEFRULE NP/NEG1 : ; lhs NPs can only be negative if they appear in a
                  ; coordinate structure
   N2[~CONJ] -> U. NEG(0) = -.
DEFRULE NP/NEG2 : ; rhs NPs can only be negative if they appear in a
                  ; coordinate structure
   [~CONJ] -> N2[~CONJ], U. NEG(1) = -.
DEFRULE LHS_N1 : ; N1 has the feature MOD to control the order in
                 ; which modifiers attach. This defaults the MOD
                 ; value to NONE
   N1 -> U. MOD(0) = NONE.
DEFRULE PP/NEG1 : ; lhs PPs can only be negative if they appear in a
                  ; coordinate structure
   P2[~CONJ] -> U. NEG(0) = -.
DEFRULE PP/NEG2 : ; rhs PPs can only be negative if they appear in a
                  ; coordinate structure
   [~CONJ] -> P2[~CONJ], U. NEG(1) = -.
DEFRULE P/NEG1 : ; lhs Ps can only be negative if they appear in a
                 ; coordinate structure
   P[~CONJ] -> U. NEG(0) = -.
DEFRULE P/NEG2 : ; rhs Ps can only be negative if they appear in a
                 ; coordinate structure
   [~CONJ] -> P[~CONJ], U. NEG(1) = -.
DEFRULE A1/AGR : ; the AGR value of an A1 defaults to AGR N2[NFORM
```

```
                    ; NORM].
   A1 -> W. AGR(0) = N2[NFORM NORM].
DEFRULE NFORM1 : ; all N2s in non-agreement positions are NFORM NORM.
                 ; This and the next few pick these out
   [N +, V -] -> N2, U. NFORM(1) = NORM.
DEFRULE NFORM2 : [N -] -> H, N2. NFORM(2) = NORM.
DEFRULE NFORM3 : [N -] -> H, N2, S. NFORM(2) = NORM.
DEFRULE NFORM4 : [N -] -> H, N2, N2, W. NFORM(2) = NORM.
DEFRULE NFORM5 : [N -] -> H, N2, P2, W. NFORM(2) = NORM.
DEFRULE NFORM6 : [N -] -> H, N2, [~BAR]. NFORM(2) = NORM.
DEFRULE NFORM7 : [N -] -> H, N2, [ADV +], W. NFORM(2) = NORM.
DEFRULE NFORM8 : [N -] -> H, N2, VP[+Q], W. NFORM(2) = NORM.
DEFRULE A2/ADV : ; A2 defaults to [ADV -] when it's not a daughter of
                 ; an adjectival category (~ADV) or S (~COMP).
   [~ADV, ~COMP] -> A2, U. ADV(1) = -.
DEFRULE AUX : ; VP defaults to [AUX -]
   VP -> W. AUX(0) = -.
DEFRULE RHS_P2 : ; rhs P2's default to -POSS and -GERUND
   [] -> P2[~CONJ], U. F(1) = -, F in {POSS, GERUND}.
DEFRULE COORD : ; marks the mother of a coordinate structure as
                ; +COORD.
   X2 -> X2[CONJ], U. COORD(0) = +.
DEFRULE COORD2 : ; forces the various predicates following 'be' to be
                 ; -COORD. Coordination of these predicates is done
                 ; by VP/PRD/CONJ so this stops multiple parses.
   VP -> H[SUBCAT BE], [BAR 2, +PRD], W. COORD(2) = -.
DEFRULE A1/PRD_1 : ; defaults all complement taking A1s to +PRD to
                   ; prevent them appearing in prenominal position.
   A1 -> H, X2, W. PRD(0) = +.
DEFRULE A1/PRD_2 : ; defaults all AGR V2 A1s to +PRD to prevent them
                   ; appearing in prenominal position.
   A1[AGR V2] -> H, W. PRD(0) = +.
DEFRULE N1/AFORM_A : ; N1s dominating lexical heads must be AFORM
                     ; NONE
   N1 -> H, W. AFORM(0) = NONE.
DEFRULE N1/AFORM_B : ; N1s dominating VP or PP modifiers must be
                     ; AFORM NONE
   N1 -> H1, [BAR 2, N -]. AFORM(0) = NONE.
DEFRULE N2/AFORM_B : ; +PN N2s are AFORM NONE
   N2[PN +] -> U. AFORM(0) = NONE.
DEFRULE N2/AFORM_C : ; N2s containing -PRD quantifers (all dogs) are
                     ; AFORM NONE
   N2[+SPEC] -> A2, H2[-SPEC]. AFORM(0) = NONE.
DEFRULE N2/AFORM_D : ; the mother node of a comparative is AFORM NONE
   N2[+SPEC] -> H2[AFORM], U. AFORM(0) = NONE.
DEFRULE N2/AFORM_E : ; the mother node of a N2/ADJ* is AFORM NONE
   N2[+SPEC] -> DetN, A2. AFORM(0) = NONE.
DEFRULE P1/PRO : ; there are some +PRO PPs (now, when, where etc)
                 ; dealt with by idrule PP/PRO. This makes all P1s
                 ; -PRO.
   P1 -> U. PRO(0) = -.
DEFRULE P/PRO : ; defaults P to -PRO.
   P -> U. PRO(0) = -.
DEFRULE A1/DISTR1 : ; all complement taking rhs As are DISTR PRD.
   A1 -> H, X2, W. DISTR(1) = PRD.
DEFRULE A1/DISTR2 : ; all AGR V2 As are DISTR PRD.
   A1[AGR V2] -> H, W. DISTR(1) = PRD.
DEFRULE S/SSUBJ : ; S with V2 subject is finite.
```

```
   S -> V2, H2. FIN(2) = +.
DEFRULE AGRV2 : ; [AGR V2] A1 and VP defaults to [AGR V2 [UB -]] - ie
                ; they don't usually agree with WH V2s
   [V +, AGR V2[SUBJ]] -> W. UB(0[AGR]) = -.
DEFRULE AGRV2_2 : ; VPs which take V2 subjects don't usually agree
                  ; with WH V2s.
   S -> V2, H2[-SUBJ]. UB(2[AGR]) = -.
DEFRULE A2/PRD : ; A2 daughters of A1 or VP default to PRD +.
   [V +] -> A2, W. PRD(1) = +.
DEFRULE PRO+ : ; heads of partitives are pronouns
   N2[+SPEC] -> H2[PART], U. PRO(1) = +.
DEFRULE -POSS : ; defaults lhs nominal categories to -POSS (except
                ; mothers of pronouns)
   [N +, V -, PRO (-, @)] -> U. POSS(0) = -.
DEFRULE CONJ/NUM : [N +, V -, CONJ (NULL, @)] -> U. NUM(0) = -.
DEFRULE NAME : ; a fix to stop recursion in psrule N/NAME1
   [PN +] -> N[PN +], U. COUNT(1) = -.
DEFRULE NFEATS : ; defaults for various features in N rules (eg
                 ; compounds, names etc).
   N -> N, U. F(0) = -, F in {PRO, PN, REFL, POSS, NUM}.
DEFRULE NFEATS2 : ; default for NFORM in N rules
   N -> N, U. NFORM(0) = NORM.
DEFRULE NFEATS3 : ; default for PER in N rules
   N -> N, U. PER(0) = 3.
DEFRULE NFEATS4 : ; defaults for various features on nonhead noun in
                  ; compounds.
   [] -> N[H -], [H +]. F(1) = -, F in {PRO, PN, REFL, POSS, NUM}.
DEFRULE AFEATS : ; default for various features on A in compounds
   N -> A[H -], N[H +]. F(1) = -, F in {QUA, ADV, NEG, NUM}.
DEFRULE LOC/PFORM : ; PFORM on +LOC subcategorised P2 is NORM
   [] -> H, P2[+LOC], W. PFORM(2) = NORM.
DEFRULE P/SUBCAT : ; for the odd prepositions in some of the VP rules
   [] -> P, V2[VFORM (@, TO, BSE, EN)], U. SUBCAT(1) = NP.
DEFRULE P/SUBCAT2 : ; for more odd prepositions in some of the VP
                    ; rules (eg VP/WHS1_PREP)
   VP -> H[SUBCAT (PP, WHVP, WHS)], P, U. SUBCAT(2) = NP.
DEFRULE N2/ADJ1 : ; for N2 mother in idrules N2/ADJ*
   N2 -> DetN, A2. F(0) = -, F in {PN, PRO, POSS, NUM, COORD, REFL}.
DEFRULE N2/ADJ2 : ; for N2 mother in idrules N2/ADJ*
   N2 -> DetN, A2. PER(0) = 3.
DEFRULE N2/ADJ3 : ; for N2 mother in idrules N2/ADJ*
   N2 -> DetN, A2. NFORM(0) = NORM.
DEFRULE N2/ADJ4 : ; for A2 daughter in idrules N2/ADJ*
   N2 -> DetN, A2. F(2) = -, F in {PRD, NEG, QUA, NUM}.
DEFRULE N2/ADJ5 : ; for A2 daughter in idrules N2/ADJ*
   N2 -> DetN, A2. DISTR(2) = ATT.
DEFRULE N2/ADJ6 : ; for A2 daughter in idrules N2/ADJ*
   N2 -> DetN, A2. AGR(2) = N2[NFORM NORM].
```

; linear precedence rules

```
LPRULE LP1 : ; orders categories with both SUBCAT and BAR
             ; specifications before non-subcat categories: ie
             ; lexical heads precede their complements. Had to
             ; complicate this to get the distribution of [SUBCAT
             ; NOT] right.
   [SUBCAT, BAR] < [~SUBCAT].
LPRULE LP2 : ; orders various non-major categories with SUBCAT values
```

```
               ; before their sisters.
   [SUBCAT (DETN, DETA, THAT, FOR, IF, WHETHER, THAN, AS)] < [~SUBCAT].
LPRULE LP3 : ; 'not' appears in various VP/BE_*/NEG rules (and SAI
               ; versions of them). This ensures that the verb precedes
               ; 'not'.
   [SUBCAT BE] < [SUBCAT NOT].
LPRULE LP4 : ; 'not' precedes everything else except for nominative
               ; NPs in SAI versions of the VP/BE*/NEG rules. This
               ; makes it precede everything that's unspecified for
               ; CASE (ie everything that's not nominal).
   [SUBCAT NOT] < [BAR, ~CASE, ~SUBCAT].
LPRULE LP5 : ; 'not' precedes P (in and not under the desk)
   [SUBCAT NOT] < P.
LPRULE LP6 : ; the nominative NP in SAI versions of the VP/BE*/NEG
               ; rules precedes 'not'.
   N2[+NOM] < [SUBCAT NOT].
LPRULE LP7 : ; 'not' precedes accusative NPs (as in eg VP/BE_NP/NEG
               ; and VP/BE_NP/NEG(SAI).
   [SUBCAT NOT] < N2[+ACC].
LPRULE LP8 : ; 'not' precedes an N2 head (for idrule N2/NEG)
   [SUBCAT NOT] < N2[H +].
LPRULE LP9 : ; as in the GPSG book except the V2 is [~H] (to allow an
               ; AP modifier to either precede or follow a V2). Because
               ; of this change we need LP10 and LP11.
   [N +] < P2 < V2[~H].
LPRULE LP10 : ; Subject NP precedes VP (see LP9).
   N2 < VP[H +].
LPRULE LP11 : ; preposed categories precede slashed S (see LP9)
   [BAR 2] < S[H +, SLASH].
LPRULE LP12 : ; for sentential subjects and S[GER] subjects
   S < VP.
LPRULE LP13 : ; for VP[GER] subjects
   VP[~H] < VP[H +].
LPRULE LP14 : ; to order the NPs after verbs like 'consider'
                ; (VP/OR_NP)
   N2[-PRD] < N2[+PRD].
LPRULE LP15 : ; to order NP and AP after verbs like 'consider'
                ; (VP/OR_AP)
   N2[-PRD] < A2[-ADV].
LPRULE LP16 : ; to order the NPs in VP/BE_NP(SAI)
   N2[+NOM] < N2[+PRD].
LPRULE LP17 : ; to order NP and AP in SAI version of NP/BE_AP
   N2[+NOM] < A2.
LPRULE LP18 : ; -PRD A2 precedes its nominal or adjectival head
                ; sister (small dog, all dogs).
   A2[~H, -PRD] < [N +, H +].
LPRULE LP19 : ; +PRD, +QUA A2 precedes its nominal or adjectival head
                ; sister (many dogs)
   A2[~H, +PRD, +QUA] < [N +, H +].
LPRULE LP20 : ; +PRD, -QUA A2 follows its nominal or adjectival head
                ; sister (a man taller than John, a man eager to
                ; please).
   [N +, H +] < A2[~H, +PRD, -QUA].
LPRULE LP21 : ; +ADV A2 precedes its adjectival head sister
                ; (obviously stupid)
   A2[~H, +ADV] < [N +, V +, H +].
LPRULE LP22 : ; orders partitive rules
   N2[H +, SPEC +] < [PFORM OF] < N2[~H].
```

```
LPRULE LP23 : ; Possessive NP precedes its N2 head
   N2[+POSS] < N2[H +].
LPRULE LP24 : ; possessive morpheme 's' follows the NP it attaches to
   N2[-POSS] < N1[+POSS, H +].
LPRULE LP25 : ; as in the GPSG book - orders conjuncts according to
              ; the coordinators they contain.
   [CONJ (NULL, BOTH, EITHER, NEITHER)] < [CONJ (AND, OR, NOR, BUT)].
LPRULE LP26 : ; to order coordinators before their conjuncts
   [CONJN +] < [~CONJN].
LPRULE LP27 : ; orders a nominative NP before coordinated predicate
              ; in SAI versions of VP/PRD/CONJ and VP/PRD/CONJ2.
   N2[+NOM] < X2[+PRD, COORD +].
LPRULE LP28 : ; adverbial modifiers precede +AUX VPs (see id
              ; VP/MOD3).
   A2[+ADV] < VP[+AUX, H +].
LPRULE LP29 : ; P2 modifers follow VP (see id VP/MOD2).
   VP[H +] < P2.
LPRULE LP30 : ; to order adverbial modifier before P1 ('right behind
              ; the door' - see id P2/ADVMOD).
   A2[+ADV] < P1.
LPRULE LP31 : ; stops multiple parses in sentences where a VP
              ; contains two PPs and one is a gap
   P2[+NULL] < P2[~NULL].
LPRULE LP32 : ; to order S1 (optional adverbial between subject and
              ; VP)
   A2[+ADV] < VP[FIN +].
LPRULE LP33 : ; particles precede +V, BAR 2 categories
   [PRT @, ~BAR] < [V +, BAR 2].
LPRULE LP34 : ; particles precede PPs
   [PRT @, ~BAR] < P2.
LPRULE LP35 : ; prevents multiple parses where the N2 object of a
              ; phrasal verb is a gap
   N2[+NULL] < [PRT @, ~BAR].
LPRULE LP36 : ; verbs precede free prepositions
   V < P.
LPRULE LP37 : ; NP precedes the particle & preposition in phrasal
              ; prepositional verb rules.
   N2 < [PRT @, PREP @, ~SUBCAT].
LPRULE LP38 : ; +PRO NP precedes a particle
   N2[PRO +] < [~SUBCAT, PRT @].
LPRULE LP39 : ; adjectives precedes free prepositions in eg idrule
              ; A1/PPING
   A < P.
LPRULE LP40 : ; +POSS NP modifiers precede N1 head (id rules
              ; N1/POSSMOD*)
   N2[POSS +] < N1.
LPRULE LP41 : ; orders compound rules (right headed).
   [BAR 0, SUBCAT, H -] < [BAR 0, SUBCAT, H +].
LPRULE LP42 : ; to order VP/NP_MEASP, VP/NP_MEASP_PHR
   N2[PRD @] < [PRT @p, ~SUBCAT] < N2[+PRD, ~NULL].
LPRULE LP43 : ; to order slashed versions of VP/NP_MEASP,
              ; VP/NP_MEASP_PHR
   N2[PRD @] < N2[+PRD, NULL +] < [PRT @p, ~SUBCAT].
LPRULE LP44 : ; to order VP/O*PREP
   P < VP.
LPRULE LP45 : ; to order VP/SR_NP_AP, VP/OE_AP2, VP/SR_NP_NP
   N2[-PRD] < P[SUBCAT @] < [N +, BAR 2, PRD +].
LPRULE LP46 : ; to order tag questions
```

```
   S[~TAG] < S[TAG +].
LPRULE LP47 : ; to order the tag bit of tag questions
   [AUX +, SUBCAT @] < N2.
```

; metarules

```
METARULE PASSIVE : ; passive metarule. The -PRD specification on the
                   ; input N2 prevents various non-passivisable VP
                   ; rules from matching. The [AGR N2[NFORM NORM]]
                   ; restriction prevents various VP rules with odd
                   ; agreement properties from matching. Passives for
                   ; these are done directly with idrules - see
                   ; idrules VP/*/PASS*. The PRO (@, -) restriction
                   ; stops VP/NP_PHRB from matching (only one of this
                   ; pair needs to match). The SUBCAT list stops it
                   ; from applying to various VP rules that don't
                   ; passivise even though they have an object NP,
                   ; for example, the rule VP/OE_BSE would otherwise
                   ; match to give * lee was heard wash up
   VP[AGR N2[NFORM NORM]] ->
   H[SUBCAT (NP, NP_NP, NP_PP, NP_PP_PP,
      NP_LOC, NP_ADVP, NP_SFIN,
         NP_NP_SFIN, NP_SBSE, NP_WHS, NP_WHVP, OC_NP, OC_AP, OC_INF),
      AGR N2[NFORM NORM], PSVE -], N2[NFORM NORM, -PRD, PRO (@, -)],
   W.
   ==> VP[EN, +PRD, AGR N2] -> H[PSVE +, AGR N2], ( P2[PFORM BY] ), W.

METARULE SAI : ; Subject Auxiliary Inversion metarule. Applies to all
               ; auxiliary rules except for VP/TO. The listing of
               ; possible input SUBCAT values excludes VP/TO. Only
               ; applies to AGR N2 since inversion with sentential
               ; subjects (*is that lee is a fool bothering kim) is
               ; ungrammatical.
   VP[+AUX, AGR N2, VFORM (@, NOT)] ->
   H[SUBCAT (DO, MODAL, OUGHT, HAVE, BE)], W.
   ==> S[+INV, +FIN] -> H[+INV], W, N2[+NOM].
```

; metarules for WH, UB, EVER propagation

```
METARULE FOOT1A : ; FOOT metarules create new versions of rules
                  ; adding foot features to a mother and one daughter
                  ; and binding their values. This binds S and
                  ; subject NP in id rule S.
   S -> N2, VP, U.
   ==> S[WH @x, UB @y, EVER @z] -> N2[WH @x, UB @y, EVER @z], VP, U.
METARULE FOOT1B : ; binds foot features on preposed NP and AP to foot
                  ; features on the mother S. Applies to S/UDC_NP and
                  ; S/UDC_AP. Creates a -INV version where -INV
                  ; version can be either UB Q or UB R. The ~WH on
                  ; this and other FOOT rules prevents them from
                  ; applying to each other's outputs.
   S[~WH] -> [N +, BAR 2], S[H +, SLASH].
   ==> S[-INV, EVER @z, UB @y, WH @x] -> [EVER @z, UB @y, WH @x], S.
METARULE FOOT1C : ; same as FOOT1B but creates a +INV version where
                  ; +INV version must only be UB Q (no inversion in
                  ; relatives)
   S[~WH] -> [N +, BAR 2], S[H +, SLASH].
   ==> S[+INV, EVER @z, UB Q, WH @x] -> [EVER @z, UB Q, WH @x], S.
```

```
METARULE FOOT1D : ; foot binding between S and preposed PP which must
                  ; be +WH in order to block '*the desk on that I put
                  ; the book'. -INV version.
   S[~WH] -> P2, [H +, SLASH].
   ==> S[UB @y, EVER @z, -INV, WH +] -> P2[WH +, UB @y, EVER @z], [].
METARULE FOOT1E : ; same as FOOT1D except +INV version which must be
                  ; UB Q.
   S[~WH] -> P2, [H +, SLASH].
   ==> S[UB Q, EVER @z, +INV, WH +] -> P2[WH +, UB Q, EVER @z], [].
METARULE FOOT2 : ; foot binding between N2 and Det.
   [N +, V -, ~WH] -> DetN, N2.
   ==> [EVER @z, UB @y, WH @x] -> DetN[EVER @z, UB @y, WH @x], N2.
METARULE FOOT3A : ; foot binding between a mother and its BAR 2 head.
                  ; The ~SUBJ excludes VP and S (VP can't have Wh
                  ; features). The ~T prevents it applying to id
                  ; rules T and T2.
   X[~WH, ~SUBJ, ~T] -> H2[~WH], U.
   ==> X[UB @y, EVER @z, WH @x] -> H2[UB @y, EVER @z, WH @x], U.
METARULE FOOT3B : ; foot binding between a mother and its BAR 1 head
   X[~WH, ~SUBJ] -> H1, U.
   ==> X[EVER @z, UB @y, WH @x] -> H1[EVER @z, UB @y, WH @x], U.
METARULE FOOT4 : ; foot binding between A2[+PRD, +QUA] and its
                 ; nominal mother eg 'how many (of the) books'
   [N +, V -, ~WH] -> A2[+PRD, +QUA], U.
   ==> [EVER @z, UB @y, WH @x] -> A2[EVER @z, UB @y, WH @x], U.
METARULE FOOT5 : ; foot binding between subcategorised P2 and its
                 ; nominal mother eg 'the covers of which'
   [N +, V -, ~WH] -> P2, W.
   ==> [UB @y, EVER @z, WH @x] -> P2[WH @x, UB @y, EVER @z], W.
METARULE FOOT6 : ; foot binding between non-head N2 and N2 mother
                 ; (whose book, which man s book, which of the books
   [N +, V -, BAR 2, ~WH] -> N2[PRD @], [H +], U.
   ==> [EVER @z, UB @y, WH @x] -> N2[EVER @z, UB @y, WH @x], [], U.
METARULE FOOT7 : ; foot binding between P1 and its NP daughter.
                 ; excludes possessive PP - * of which man s
   P1[-POSS] -> N2, U.
   ==> P1[EVER @z, UB @y, WH @x] -> N2[EVER @z, UB @y, WH @x], U.
METARULE FOOT8 : ; foot binding between A1 and its DetA daughter (how
                 ; many, how clever)
   A1[~WH] -> DetA, U.
   ==> A1[EVER @z, UB @y, WH @x] -> DetA[EVER @z, UB @y, WH @x], U.
METARULE FOOT9 : ; foot binding between either N2 and H0[+PRO] (who,
                 ; what etc) or P2 and H0[PRO +] (where, when).
   [V -, BAR 2, ~WH] -> H[PRO +].
   ==> [UB @y, EVER @z, WH @x] -> H[WH @x, UB @y, EVER @z].
METARULE FOOT10 : ; foot binding between A2 and adverbial modifier -
                  ; 'how obviously clever'
   A2[~WH] -> A2[+ADV], H1.
   ==> A2[EVER @z, UB @y, WH @x] -> A2[EVER @z, UB @y, WH @x], H1.
```

; metarules for SLASH propagation

```
METARULE SLASHBAR2A : ; creates new rule where SLASH appears on S and
                      ; VP and the value is bound. The ~WH here and
                      ; on other slash rules prevents SLASH and WH
                      ; features from appearing on the same category
                      ; ie the outputs of the FOOT metarules cannot
                      ; be input to the SLASH metarules.
```

```
   S[~WH] -> X2[~ADV], H2[SUBJ -], U.
   ==> S[SLASH @x] -> X2, H2[SLASH @x], U.
METARULE SLASHBAR2B : ; to pass slash from S to S head. Doesn't apply
                      ; to S/Q1, S/Q2, nor to S/TAGQUESTION
   S[~WH, ~UB] -> S[H +, ~SLASH], [~TAG].
   ==> S[SLASH @x] -> [], S[SLASH @x].
METARULE SLASHBAR2C : ; to pass SLASH from N2[+SPEC] to N2[-SPEC]
   N2[~WH] -> H2[-SPEC], U. ==> N2[SLASH @x] -> H2[SLASH @x], U.
METARULE SLASHBAR2D : ; to pass slash in partitive rules
   N2[+SPEC, ~WH] -> H2[+SPEC, PART], N2[~H], U.
   ==> N2[SLASH @x] -> H2, N2[SLASH @x], U.
METARULE SLASHBAR2E : ; to pass slash in N2/NEG, CONJ/N2*
   N2[~WH, SPEC @] -> H2[SPEC @], U.
   ==> N2[SLASH @x] -> H2[SLASH @x], U.
METARULE SLASHBAR1 : ; to pass SLASH from a mother to its BAR 1
                     ; daughter
   X[~WH, ~SLASH] -> [H +, BAR 1], U.
   ==> X[SLASH @x] -> [SLASH @x], U.
METARULE SLASHBAR0A_1 : ; to pass SLASH from a mother to a nonhead in
                        ; lexical id rules. The ~INV prevents it
                        ; applying to the outputs of SAI since these
                        ; need special treatment. This deals with
                        ; cases where the X2 is an AP or a VP - which
                        ; mustn't be AGR V2.
   X[~WH, ~INV, ~SLASH] -> H, X2[V +, ~UB, ~SLASH, AGR N2], W.
   ==> X[SLASH @x] -> H, X2[SLASH @x], W.
METARULE SLASHBAR0A_2 : ; to pass SLASH from a mother to a nonhead in
                        ; lexical id rules. This deals with cases
                        ; where the X2 is NP or PP. The rule has to
                        ; be separate from SLASHBAR0A_1 because NP
                        ; and PP don't have AGR.
   X[~WH, ~INV, ~SLASH] -> H, X2[V -, ~UB, ~SLASH], W.
   ==> X[SLASH @x] -> H, X2[SLASH @x], W.
METARULE SLASHBAR0A_3 : ; to pass SLASH from a mother to a nonhead in
                        ; lexical id rules. This deals with cases
                        ; where the X2 is S. The rule has to be
                        ; separate from SLASHBAR0A_1 because S is AGR
                        ; @
   X[~WH, ~INV, ~SLASH] -> H, S[~UB, ~SLASH], W.
   ==> X[SLASH @x] -> H, S[SLASH @x], W.
METARULE SLASHBAR0B_1 : ; to apply to the outputs of the SAI
                        ; metarule. Defined so that SLASH doesn't
                        ; pass to the subject NP. This applies where
                        ; the complement is +V (mustn't apply to
                        ; VP/BE_NP*(SAI) to prevent two parses for
                        ; 'fido is a dog'
   S[~WH, +INV] -> H, N2[+NOM], X2[V +].
   ==> S[SLASH @x] -> H, N2, X2[SLASH @x].
METARULE SLASHBAR0B_2 : ; to apply to the outputs of the SAI
                        ; metarule. Defined so that SLASH doesn't
                        ; pass to the subject NP. This applies where
                        ; the complement is a P2 (mustn't apply to
                        ; VP/BE_NP*(SAI) to prevent two parses for
                        ; 'fido is a dog'
   S[~WH, +INV] -> H, N2[+NOM], P2.
   ==> S[SLASH @x] -> H, N2, P2[SLASH @x].
METARULE SLASHBAR0C : ; to apply to VP/THERE(SAI)
   S[~WH, +INV] -> H, N2[NFORM THERE, +NOM], X2.
   ==> S[SLASH @x] -> H, N2, X2[SLASH @x].
METARULE SLASHBAR0D : ; to apply to VP/BE_CLEFT*(SAI)
   S[~WH, +INV] -> H, N2[NFORM IT, +NOM], X2, S.
   ==> S[SLASH @x] -> H, N2, X2[SLASH @x], S.
METARULE SLASHBAR0E : ; to apply to VP/COP/NEG*(SAI)
   S[~WH, +INV] -> H, N2[+NOM], X2, [NEG +].
   ==> S[SLASH @x] -> H, N2, X2[SLASH @x], [].
METARULE STM2 : ; same as STM2 in GPSG85. Provides for missing
                ; subjects in embedded S's. The AGR value on the
                ; output will be bound by a proprule. The ~SLASH
                ; restriction on the S[+FIN] is to stop it matching
                ; VP/BE_CLEFT2
   VP[~SLASH] -> W, S[+FIN, ~SLASH].
   ==> VP[SLASH N2] -> W, V2[-SUBJ, +FIN, AGR N2[CASE NOM]].
METARULE STM2_B : ; version of STM2 for S[+FIN] complements of A.
   A1[~SLASH] -> W, S[+FIN].
   ==> A1[SLASH N2] -> W, V2[-SUBJ, +FIN, AGR N2[CASE NOM]].
METARULE STM1_N2 : ; slash termination for N2. Instantiates SLASH to
                   ; N2 and puts +NULL on N2[SLASH N2]. The POSS, PRO
                   ; and NFORM restrictions prevent it applying to
                   ; various odd rules
   X[SLASH @x] -> W,
   N2[SLASH @x, POSS (@, -), PRO (@, -), NFORM (@, NORM)].
   ==> X[SLASH N2] -> W, N2[+NULL, SLASH N2].
METARULE STM1_P2 : ; slash termination for P2. Instantiates SLASH to
                   ; P2 and puts +NULL on P2[SLASH P2].
   X[SLASH @x] -> W, P2[SLASH @x].
   ==> X[SLASH P2] -> W, P2[+NULL, SLASH P2].
METARULE STM1_A2 : ; slash termination for A2. Instantiates SLASH to
                   ; A2 and puts +NULL on A2[SLASH A2].
   X[SLASH @x] -> W, A2[SLASH @x].
   ==> X[SLASH A2] -> W, A2[+NULL, SLASH A2].
METARULE SLASH/COMPAR : ; to pass slash onto P2[than] and P2[as] in
                        ; comparatives.
   [BAR 2, N +, ~WH, ~SLASH] -> H2[AFORM], P2.
   ==> [SLASH @x] -> H2, P2[SLASH @x].
METARULE STM/COMPAR : ; to terminate slash on P2[than] and P2[as] in
                      ; slashed versions of comparative rules. Notice
                      ; this violates the lexical idrule restriction
                      ; (which is why STM1_P2 won't do it).
   [BAR 2, N +, SLASH @x] -> H2, P2[SLASH @x].
   ==> [SLASH P2] -> H2, P2[+NULL, SLASH P2].
METARULE MOD/SLASH : ; to allow for extractions of/into VP modifiers.
                     ; This goes against GPSG85 which requires SLASH
                     ; to appear on both mother and head in
                     ; non-lexical id rules.
   VP[~SLASH, ~WH] -> H2[~SLASH] X2.
   ==> VP[SLASH @x] -> H2 X2[SLASH @x].
METARULE STM/MOD1 : ; terminates SLASH on an adverbial modifer of VP.
   VP[SLASH @x] -> H2 A2[+ADV, SLASH @x].
   ==> VP[SLASH A2] -> H2 A2[+NULL, SLASH A2].
METARULE STM/MOD2 : ; terminates SLASH on a P2 modifier of VP.
   VP[SLASH @x] -> H2 P2[SLASH @x].
   ==> VP[SLASH P2] -> H2 P2[+NULL, SLASH P2].
```

; metarules to propagate WH, UB, EVER in coordinations

```
METARULE CONJ/FOOTA : ; passes WH, UB and EVER from a mother to all
```

```
                          ; conjuncts in all non-verbal coordinations
                          ; (the ~SUBJ restriction because wh-features
                          ; don't appear on VP). The ~SUBCAT restriction
                          ; prevents them appearing on coordinations of
                          ; lexical categories.
   X[~SUBJ] -> ( [CONJ, ~SUBCAT] )+.
   ==> X[WH @x, UB @y, EVER @z] -> ( [WH @x, UB @y, EVER @z] )+.
METARULE CONJ/FOOTB : ; passes WH, UB, and EVER in idrule CONJ/SA
   S[CONJ] -> S[H +], U.
   ==> S[WH @x, UB @y, EVER @z] -> S[WH @x, UB @y, EVER @z], U.
METARULE CONJ/FOOTC : ; passes WH, UB, and EVER in idrule CONJ/SB
   S -> S[H +], [CONJN +].
   ==> S[WH @x, UB @y, EVER @z] -> S[WH @x, UB @y, EVER @z], [].
METARULE CONJ/FOOTD : ; passes WH, UB, EVER from an S mother of a
                      ; coordination to the S conjunct daughters.
   S -> ( S[CONJ] )+.
   ==> S[EVER @z, UB @y, WH @x] -> ( S[EVER @z, UB @y, WH @x] )+.
```

; metarules to propagate SLASH in coordinations

```
METARULE CONJ/SLASHA : ; passes SLASH from a mother to all conjunct
                       ; daughters so long as they are non-lexical
                       ; (~SUBCAT).
   X[~WH] -> ( [CONJ, ~SUBCAT] )+.
   ==> X[SLASH @x] -> ( [SLASH @x] )+.
METARULE CONJ/SLASHB : ; the normal slash metarules don't pass SLASH
                       ; in all the CONJ rules so this does it for
                       ; the rest. Also passes SLASH in some
                       ; non-coordination rules such as VP/MOD*,
                       ; VP/NEG, A2/NEG, A2/COMPAR*
   [V +, BAR 2, ~WH, ~COMP, ~UB, ~SLASH] -> H2[~SLASH], U.
   ==> [SLASH @x] -> H2[SLASH @x], U.
METARULE CONJ/SLASHC : ; passes SLASH in CONJ/P2* and also P2/NEG
   P2[~WH] -> H2, U. ==> P2[SLASH @x] -> H2[SLASH @x], U.
METARULE CONJ/SLASHD : ; passes SLASH in CONJ/SA
   S[CONJ, ~WH] -> [H +]. ==> S[SLASH @x] -> [SLASH @x].
```

; ID rules

; root sentence rules

```
IDRULE T : ; root symbol for the parser. It is here that the
           ; information that root sentences are always finite is
           ; encoded. This means that the parser gives two parses for
           ; finite sentences (one as T, one as S) whilst only one
           ; parse for non-finite sentences (as S)
   [T +, BAR 2] -> S[H +, COMP NORM, +FIN].
IDRULE T2 : ; all wh-questions to be recognised as root sentences
   [T +, BAR 2] -> S[H +, COMP NORM, +FIN, UB Q, WH +, EVER @ev].
```

; S rules

```
IDRULE S1 : ; split up ordinary S rules to allow for non-nominative
            ; subjects. This one = finite S. Optional sentence adverb
            ; interposed between subject and VP.
   S[COMP NORM, -INV, +FIN] -> N2[+NOM], ( A2[+ADV] ),
   H2[-SUBJ, AGR N2].
IDRULE S2 : ; base S
```

```
   S[COMP NORM, -INV, -FIN, BSE] -> N2[+NOM], H2[-SUBJ, AGR N2].
IDRULE S3 : ; infinitival S. Accusative subject - 'for him to go'
   S[COMP NORM, -INV, -FIN, TO] -> N2[+ACC], H2[-SUBJ, AGR N2].
IDRULE S4 : ; gerund S. Accusative subject - 'him going'
   S[COMP NORM, -INV, -FIN, ING] -> N2[+ACC], H2[-SUBJ, AGR N2].
IDRULE S/V2_SUBJ1 : ; for finite sentential subjects eg 'that fido
                    ; dances bothers lee'
   S[COMP NORM, -INV] -> S[+FIN, that], H2[-SUBJ, AGR S].
IDRULE S/V2_SUBJ2 : ; for infinitival sentential subject eg 'for us
                    ; to go would be possible
   S[COMP NORM, -INV] -> S[COMP FOR, -FIN, TO], H2[-SUBJ, AGR S].
IDRULE S/V2_SUBJ3 : ; for infinitival VP subject eg 'to go would be
                    ; possible'
   S[COMP NORM, -INV] -> VP[-FIN, TO, AGR N2[NFORM NORM]],
   H2[-SUBJ, AGR VP].
IDRULE S/V2_SUBJ4 : ; for BSE sentential subject eg 'that you answer
                    ; is necessary'
   S[COMP NORM, -INV] -> S[COMP THAT, -FIN, BSE], H2[-SUBJ, AGR S].
IDRULE S/V2_SUBJ5 : ; for S[+Q] subjects eg 'whether we should go is
                    ; not clear'
   S[COMP NORM, -INV] -> S[+Q], H2[-SUBJ, AGR S[+Q]].
IDRULE S/V2_SUBJ6 : ; for S[+Q] subjects eg 'what we should do is not
                    ; clear'
   S[COMP NORM, -INV] -> S[+Q, +WH, -EVER], H2[-SUBJ, AGR S[+Q]].
IDRULE S/THAT1 : ; introduces 'that' complementiser. +FIN version
   S[that] -> [SUBCAT THAT], S[H +, COMP NORM, -INV, +FIN].
IDRULE S/THAT2 : ; introduces 'that' complementiser. BSE version
   S[that] -> [SUBCAT THAT], S[H +, COMP NORM, -INV, BSE].
IDRULE S/FOR : ; introduces 'for' complementiser. Restricts the S to
               ; [VFORM TO]
   S[COMP FOR, TO] -> [SUBCAT FOR], S[H +, COMP NORM, -INV].
IDRULE S/Q1 : ; for embedded whether-clauses. The S is +Q but has no
              ; WH and EVER features. The fact that it doesn't have
              ; all three features stops it from being recognised as
              ; a root question.
   S[+Q, +FIN, COMP WHETHER] -> [SUBCAT WHETHER],
   S[H +, COMP NORM, -INV].
IDRULE S/Q2 : ; same as S/Q1 but for embedded if-clauses
   S[+Q, +FIN, COMP IF] -> [SUBCAT IF], S[H +, COMP NORM, -INV].
IDRULE S/THAN : ; introduces 'than' sentences.
   S[COMP THAN] -> [SUBCAT THAN], S[H +, COMP NORM, +FIN].
IDRULE S/AS : ; introduces 'as' sentences.
   S[COMP AS] -> [SUBCAT AS], S[H +, COMP NORM, +FIN].
IDRULE S/NP_UDC : ; preposed NP. The features on the [SLASH N2] get
                  ; bound to the N2 by proprule SLASH/AGR_NP
   S -> N2[NFORM NORM], S[H +, COMP NORM, SLASH N2, +FIN].
IDRULE S/PP_UDC : ; preposed PP. Features get bound by SLASH/AGR_PP1
   S -> P2, S[H +, COMP NORM, SLASH P2, +FIN].
IDRULE S/AP_UDC : ; preposed AP. Features get bound by SLASH/AGR_AP
   S -> A2, S[H +, COMP NORM, SLASH A2, +FIN].
IDRULE S/IMPER : ; imperative S,.Imperatives are distinguished by
                 ; being both VFORM BSE and +FIN. All other +FIN
                 ; forms are VFORM NOT.
   S -> H2[-SUBJ, BSE, FIN +, AGR N2[NFORM NORM]].
IDRULE S/ADVMOD : ; sentence adverb. The -COORD specification
                  ; prevents coordinated adverbs from appearing -
                  ; these are dealt with by idrule S/ADV/CONJ.
   S[COMP NORM] -> A2[+ADV, COORD -, AFORM NONE], S[H +].
```

```
IDRULE S/PPMOD : ; sentence PP modifier. The -COORD specification
                 ; prevents coordinated PPs from appearing - these
                 ; are dealt with by idrule S/ADV/CONJ.
   S[COMP NORM] -> P2[PFORM NORM, COORD -], S[H +].
IDRULE S/ADV/CONJ : ; allows for the coordination of sentence
                    ; adverbials, whether AP[+ADV] or PP (see idrules
                    ; CONJ/MOD* and MOD/COORD*).
   S -> X2[+ADV, COORD +], S[H +].
IDRULE S/TAG : ; tags 'doesn't he' 'is he' etc
   S[TAG +, INV +, AUX +, FIN +, VFORM NOT, AGR N2[NFORM NORM]] ->
   H[SUBCAT @s, AUX +], N2[+NOM].
IDRULE S/TAGQUESTION : ; tag questions - 'he's a fool, isn't he?'
   S[-INV, +FIN, COMP NORM] -> H2[+SUBJ], S[TAG +].
```

; PP rules

```
IDRULE P2/ADVMOD : ; (right) behind the house
   P2 -> ( A2[+ADV] ), H1.
IDRULE P2/NEG : ; not in the kennel. (+NEG PPs will only be able to
                ; appear in conjoined PPs - 'put the dog in the
                ; garden but not in the kennel' vs '* put the dog not
                ; in the kennel' - see defrules PP/NEG*)
   P2[NEG +] -> [SUBCAT NOT], H2[NEG -].
IDRULE P2/PRO : ; pro-PPs eg 'then, there, when, where, why'
   P2[+PRO, -POSS] -> H[SUBCAT NULL].
IDRULE P/NEG : ; not in. (+NEG Ps will only be able to appear in
               ; conjoined Ps - 'not in but under the desk' - see
               ; defrules P/NEG*.)
   P[NEG +, SUBCAT @x] -> [SUBCAT NOT], H[NEG -, SUBCAT @x].
IDRULE P1/PP : ; out of the house, down in the cellar, up on the roof
   P1 -> H[SUBCAT PP], P2[PFORM NORM].
IDRULE P1/NP : ; separate rule for possessive 'of' PP means that
               ; -POSS has to appear on P1.
   P1[-POSS, -GER] -> H[SUBCAT NP], N2[-POSS].
IDRULE P1/POSS : ; of fido s. Possessive P1. Preposition must be
                 ; 'of'.
   P1[+POSS, -GER] -> H[PFORM OF, SUBCAT NP], N2[+POSS].
IDRULE P1/SFIN : ; before he went to bed
   P1[-POSS, -GER] -> H[SUBCAT SFIN], S[+FIN, COMP NORM, -INV].
```

; VP rules

```
IDRULE VP/INTR : ; he sings, it rains (either NORM or IT subject)
   VP[AGR N2] -> H[SUBCAT NULL].
IDRULE VP/INTR_PHR : ; falls over
   VP -> H[SUBCAT NULL, PRT @p], [PRT @p].
IDRULE VP/NP : ; abandons his friends, the vet weighs fido
   VP -> H[SUBCAT NP], N2[-PRD].
IDRULE VP/NP_PHRA : ; turn off the light / turn the light off. N2
                    ; restricted to be -PRO so either linearisation
                    ; is possible. Idrule VP/NP_PHRB deals with the
                    ; +PRO case.
   VP -> H[SUBCAT NP, PRT @p], N2[-PRD, -PRO], [PRT @p].
IDRULE VP/NP_PHRB : ; turn it off (*turn off it). Linearised by LP38
   VP -> H[SUBCAT NP, PRT @p], N2[-PRD, +PRO], [PRT @p].
IDRULE VP/NP_NP : ; sends fido a book. Unlike in the case of
                  ; 'consider' both NPs are -PRD, therefore
                  ; passivisable. The multiple linearisation flag
```

```
                    ; cuts down on identical outputs from passive
   VP -> H[SUBCAT NP_NP], N2[-PRD], N2[-PRD].
IDRULE VP/NP_NP_PHR : ; bring me a book back/bring me back a book.
                      ; Not trying to block *bring back me a book -
                      ; it's too much trouble.
   VP -> H[SUBCAT NP_NP, PRT @p], N2[-PRD], N2[-PRD], [PRT @p].
IDRULE VP/PP : ; look at him
   VP -> H[SUBCAT PP, PFORM @pf], P2[PFORM @pf].
IDRULE VP/PP_PHR : ; break out of jail
   VP -> H[SUBCAT PP, PFORM @pf, PRT @p], [PRT @p], P2[PFORM @pf].
IDRULE VP/NP_PP : ; acquit him of all charges, give a book to him
   VP -> H[SUBCAT NP_PP, PFORM @pf], N2[-PRD], P2[PFORM @pf].
IDRULE VP/NP_PP_PHR : ; bring back a book for me / bring a book back
                      ; for me NB I'm ignoring ORDER for these ones
                      ; and SUBTYPE PVERB
   VP -> H[SUBCAT NP_PP, PFORM @pf, PRT @p], N2[-PRD], [PRT @p],
   P2[PFORM @pf].
IDRULE VP/PP_PP : ; account/answer for it to him, appeal to him
                  ; against it
   VP -> H[SUBCAT PP_PP, PFORM @pf], P2[PFORM @pf], P2[PFORM @pf].
IDRULE VP/PP_PP_PHR : ; come down on him for his bad behaviour
   VP -> H[SUBCAT PP_PP, PFORM @pf, PRT @p], [PRT @p], P2[PFORM @pf],
   P2[PFORM @pf].
IDRULE VP/NP_PP_PP : ; he turned it from a disaster into a victory
   VP -> H[SUBCAT NP_PP_PP, PFORM @pf], N2[-PRD], P2[PFORM @pf],
   P2[PFORM @pf].
IDRULE VP/LOC : ; he fell through the floor, he got into bed
   VP -> H[SUBCAT LOC], P2[+LOC].
IDRULE VP/LOC_PHR : ; he ended up on the floor below
   VP -> H[SUBCAT LOC, PRT @p], [PRT @p], P2[+LOC].
IDRULE VP/NP_LOC : ; puts the book on the desk
   VP -> H[SUBCAT NP_LOC], N2[-PRD], P2[+LOC].
IDRULE VP/MEASP : ; fido cost ten pounds - no PASS because of +PRD.
   VP -> H[SUBCAT MP], N2[+PRD].
IDRULE VP/NP_MEASP : ; it cost him $5
   VP -> H[SUBCAT NP_MP], N2, N2[+PRD].
IDRULE VP/NP_MEASP_PHR : ; it set him back $5
   VP -> H[SUBCAT NP_MP, PRT @p], N2, [PRT @p], N2[+PRD].
IDRULE VP/ADVP : ; augur well, act badly
   VP -> H[SUBCAT ADVP], A2[ADV +].
IDRULE VP/ADVP_PHR : ; he came off badly
   VP -> H[SUBCAT ADL, PRT @p], [PRT @p], A2[ADV +].
IDRULE VP/ADVP_PP : ; things augur well for him
   VP -> H[SUBCAT ADVP_PP, PFORM @pf], A2[ADV +], P2[PFORM @pf].
IDRULE VP/NP_ADVP : ; acquit/carry oneself well, put it well
   VP -> H[SUBCAT NP_ADVP], N2[-PRD], A2[ADV +].
IDRULE VP/SFIN1 : ; believes (that) he can do it
   VP -> H[SUBCAT SFIN, SUBTYPE NONE], S[+FIN].
IDRULE VP/SFIN1_PHR : ; let out that he can do it.
   VP -> H[SUBCAT SFIN, SUBTYPE NONE, PRT @p], [PRT @p], S[+FIN].
IDRULE VP/SFIN2 : ; it seems (that) he can do it
   VP[AGR N2[NFORM IT]] -> H[SUBCAT SFIN, SUBTYPE NONE], S[+FIN].
IDRULE VP/SFIN2_PHR : ; it turns out that he can do it
   VP[AGR N2[NFORM IT]] -> H[SUBCAT SFIN, SUBTYPE NONE, PRT @p],
   [PRT @p], S[+FIN].
IDRULE VP/SFIN3A : ; that he's not here matters
   VP[AGR S[FIN +]] -> H[SUBCAT SFIN, SUBTYPE EXTRAP].
IDRULE VP/SFIN3B : ; it matters that he wasn't there (extraposed)
```

```
   VP[AGR N2[NFORM IT]] -> H[SUBCAT SFIN, SUBTYPE EXTRAP], S[+FIN].
IDRULE VP/NP_SFIN1 : ; tells her (that) he can do it
   VP -> H[SUBCAT NP_SFIN, SUBTYPE NONE], N2[-PRD, NFORM NORM],
   S[+FIN].
IDRULE VP/NP_SFIN1_PHR : ; have her on that he can do it
   VP -> H[SUBCAT NP_SFIN, SUBTYPE NONE, PRT @p], [PRT @p],
   N2[-PRD, NFORM NORM], S[+FIN].
IDRULE VP/NP_SFIN2A : ; that she wouldn't help bothers lee
   VP[AGR S[FIN +]] -> H[SUBCAT NP_SFIN, SUBTYPE EXTRAP],
   N2[-PRD, NFORM NORM].
IDRULE VP/NP_SFIN2B : ; it bothers lee that she wouldn't help
                      ; (extrap)
   VP[AGR N2[NFORM IT]] -> H[SUBCAT NP_SFIN, SUBTYPE EXTRAP],
   N2[-PRD, NFORM NORM], S[+FIN].
IDRULE VP/PP_SFIN1 : ; she agrees with him that he should help. Will
                     ; also do non-extraposable AGR IT ones - it
                     ; appears to him that .. it dawned on him that
                     ; ..
   VP[AGR N2] -> H[SUBCAT PP_SFIN, SUBTYPE NONE, PFORM @pf],
   P2[PFORM @pf], S[+FIN].
IDRULE VP/PP_SFIN1_PHR : ; she gets through to him that he should
                         ; help
   VP -> H[SUBCAT PP_SFIN, SUBTYPE NONE, PFORM @pf, PRT @p],
   [PRT @p], P2[PFORM @pf], S[+FIN].
IDRULE VP/PP_SFIN2A : ; that lee helps matters to her
   VP[AGR S[FIN +]] -> H[SUBCAT PP_SFIN, SUBTYPE EXTRAP, PFORM @pf],
   P2[PFORM @pf].
IDRULE VP/PP_SFIN2B : ; it matters to her that lee helps (extrap)
   VP[AGR N2[NFORM IT]] ->
   H[SUBCAT PP_SFIN, SUBTYPE EXTRAP, PFORM @pf], P2[PFORM @pf],
   S[FIN +].
IDRULE VP/NP_NP_SFIN : ; he bet her $5 that he could do it
   VP -> H[SUBCAT NP_NP_SFIN], N2[-PRD], N2[+PRD], S[+FIN].
IDRULE VP/SINF : ; prefer for him to do it
   VP -> H[SUBCAT SINF], S[TO, COMP FOR].
IDRULE VP/PP_SINF : ; she arranged with him for them to help
   VP -> H[SUBCAT PP_SINF, PFORM @pf], P2[PFORM @pf], S[TO, COMP FOR].
IDRULE VP/SBSE : ; insists that he do it
   VP -> H[SUBCAT SBSE], S[BSE, that].
IDRULE VP/NP_SBSE : ; he petitioned them that he be allowed to do it
   VP -> H[SUBCAT NP_SBSE], N2[-PRD], S[BSE].
IDRULE VP/PP_SBSE : ; requires of lee that he do it
   VP -> H[SUBCAT PP_SBSE, PFORM @pf], P2[PFORM @pf], S[BSE, that].
IDRULE VP/WHS1 : ; embedded wh questions - wonders who he abandoned
                 ; (S[+Q] expanded by S/UDC), wonders who abandoned
                 ; him (S[+Q] expanded by normal S rule)
   VP -> H[SUBCAT WHS], S[+Q, -INV, +WH, -EVER].
IDRULE VP/WHS1_PHR : ; figure out where he went
   VP -> H[SUBCAT WHS, PRT @p], [PRT @p], S[+Q, -INV, +WH, -EVER].
IDRULE VP/WHS1_PREP : ; knows about what he did
   VP -> H[SUBCAT WHS, PREP @pf], P[PFORM @pf],
   S[+Q, -INV, +WH, -EVER].
IDRULE VP/WHS2 : ; asks whether/if fido is a dog. S[+Q] expanded by
                 ; idrules S/Q1 and S/Q2.
   VP -> H[SUBCAT WHS], S[+Q].
IDRULE VP/WHS2_PHR : ; figure out whether he left
   VP -> H[SUBCAT WHS, PRT @p], [PRT @p], S[+Q].
IDRULE VP/WHS2_PREP : ; knows about whether fido is a dog. S[+Q]
```

```
                    ; expanded by idrules S/Q1 and S/Q2.
   VP -> H[SUBCAT WHS, PREP @pf], P[PFORM @pf], S[+Q].
IDRULE VP/NP_WHS1 : ; advise/ask him where he might stay (embedded wh
                    ; question)
   VP -> H[SUBCAT NP_WHS], N2[-PRD], S[+Q, -INV, +WH, -EVER].
IDRULE VP/NP_WHS2 : ; advise/ask him whether he is staying
   VP -> H[SUBCAT NP_WHS], N2[-PRD], S[+Q].
IDRULE VP/IT_WHS : ; I would appreciate it if you could send me that
                   ; book
   VP -> H[SUBCAT IT_WHS, SUBTYPE IF], N2[NFORM IT], S[COMP IF, +Q].
IDRULE VP/PP_WHS1 : ; arrange with him / dictate to him where they
                    ; should meet (embedded wh question)
   VP -> H[SUBCAT PP_WHS, PFORM @pf], P2[PFORM @pf],
   S[+Q, -INV, +WH, -EVER].
IDRULE VP/PP_WHS2 : ; arrange with him / dictate to him whether they
                    ; should meet
   VP -> H[SUBCAT PP_WHS, PFORM @pf], P2[PFORM @pf], S[+Q].
IDRULE VP/PP_WHS3 : ; AGR IT - it dawned on him what should do
   VP[AGR N2[NFORM IT]] -> H[SUBCAT PP_WHS, PFORM @pf],
   P2[PFORM @pf], S[+Q, -INV, +WH, -EVER].
IDRULE VP/WHVP1 : ; wonders what to attack e (VP[+Q] expended by
                  ; idrule VP/WH2.
   VP -> H[SUBCAT WHVP], VP[+Q, -INV, +WH, -EVER].
IDRULE VP/WHVP1_PHR : ; figure out what to do
   VP -> H[SUBCAT WHVP, PRT @p], [PRT @p], VP[+Q, -INV, +WH, -EVER].
IDRULE VP/WHVP1_PREP : ; know about what to do
   VP -> H[SUBCAT WHVP, PREP @pf], P[PFORM @pf],
   VP[+Q, -INV, +WH, -EVER].
IDRULE VP/WHVP2 : ; asks whether to go (VP[+Q] expanded by idrule
                  ; VP/WHV1).
   VP -> H[SUBCAT WHVP], VP[+Q].
IDRULE VP/WHVP2_PHR : ; figure out whether to leave
   VP -> H[SUBCAT WHVP, PRT @p], [PRT @p], VP[+Q].
IDRULE VP/WHVP2_PREP : ; reflect on whether to help
   VP -> H[SUBCAT WHVP, PREP @pf], P[PFORM @pf], VP[+Q].
IDRULE VP/NP_WHVP1 : ; advise/ask him where to go
   VP -> H[SUBCAT NP_WHVP], N2[-PRD], VP[+Q, -INV, +WH, -EVER].
IDRULE VP/NP_WHVP2 : ; advise/ask him whether to go
   VP -> H[SUBCAT NP_WHVP], N2[-PRD], VP[+Q].
IDRULE VP/PP_WHVP1 : ; arrange with him / dictate to him where to
                     ; meet
   VP -> H[SUBCAT PP_WHVP, PFORM @pf], P2[PFORM @pf],
   VP[+Q, -INV, +WH, -EVER].
IDRULE VP/PP_WHVP2 : ; arrange with him / dictate to him whether to
                     ; meet
   VP -> H[SUBCAT PP_WHVP, PFORM @pf], P2[PFORM @pf], VP[+Q].
IDRULE VP/SC_NP : ; act the fool, appear a fool, look a fool. There's
                  ; no syntactic difference between raising and equi
   VP -> H[SUBCAT SC_NP], N2[+PRD].
IDRULE VP/SC_NP_PHR : ; end up / turn out a fool
   VP -> H[SUBCAT SC_NP, PRT @p], [PRT @p], N2[+PRD].
IDRULE VP/SR_APA : ; he seems/appears clever. It seems/appears
                   ; obvious that .... Subject Raising. AR N2 version
   VP[AGR N2] -> H[SUBCAT SC_AP, SUBTYPE RAIS], A2[AGR N2].
IDRULE VP/SR_APB : ; (that fido dances) seems/appears obvious etc.
                   ; AGR V2 version.
   VP[AGR V2] -> H[SUBCAT SC_AP, SUBTYPE RAIS], A2[AGR V2].
IDRULE VP/SR_APA_PHR : ; he started out poor (subject rais) AGR N2
```

```
                        ; version
   VP[AGR N2] -> H[SUBCAT SC_AP, SUBTYPE RAIS, PRT @p], [PRT @p],
   A2[AGR N2].
IDRULE VP/SR_APB_PHR : ; AGR V2 version.
   VP[AGR V2] -> H[SUBCAT SC_AP, SUBTYPE RAIS, PRT @p], [PRT @p],
   A2[AGR V2].
IDRULE VP/SE_AP : ; feel/become intelligent. (subject equi)
   VP[AGR N2] -> H[SUBCAT SC_AP, SUBTYPE EQUI],
   A2[AGR N2[NFORM NORM]].
IDRULE VP/SE_AP_PHR : ; he ended up even poorer than before (subject
                        ; equi)
   VP[AGR N2] -> H[SUBCAT SC_AP, SUBTYPE EQUI, PRT @p], [PRT @p],
   A2[AGR N2[NFORM NORM]].
IDRULE VP/SR_INFA : ; appear to be crazy (subject raising). AGR N2
                      ; version
   VP[AGR N2] -> H[SUBCAT SC_INF, SUBTYPE RAIS], VP[TO, AGR N2].
IDRULE VP/SR_INFB : ; AGR V2 version
   VP[AGR V2] -> H[SUBCAT SC_INF, SUBTYPE RAIS], VP[TO, AGR V2].
IDRULE VP/SR_INFA_PHR : ; he turned out to be a crook (subject
                          ; raising). AGR N2 version
   VP[AGR N2] -> H[SUBCAT SC_INF, SUBTYPE RAIS, PRT @p], [PRT @p],
   VP[TO, AGR N2].
IDRULE VP/SR_INFB_PHR : ; AGR V2 version
   VP[AGR V2] -> H[SUBCAT SC_INF, SUBTYPE RAIS, PRT @p], [PRT @p],
   VP[TO, AGR V2].
IDRULE VP/SE_INF : ; try to dance (subject equi)
   VP[AGR N2] -> H[SUBCAT SC_INF, SUBTYPE EQUI],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/SE_INF_PHR : ; he set out to win (subject equi)
   VP[AGR N2] -> H[SUBCAT SC_INF, SUBTYPE EQUI, PRT @p], [PRT @p],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/SR_INGA : ; start moving (subject raising). AGR N2 version
   VP[AGR N2] -> H[SUBCAT SC_ING, SUBTYPE RAIS],
   VP[ING, +PRD, AGR N2].
IDRULE VP/SR_INGB : ; AGR V2 version
   VP[AGR V2] -> H[SUBCAT SC_ING, SUBTYPE RAIS],
   VP[ING, +PRD, AGR V2].
IDRULE VP/SR_INGA_PHR : ; he carried on doing it (subject raising).
                          ; AGR N2 version
   VP[AGR N2] -> H[SUBCAT SC_ING, SUBTYPE RAIS, PRT @p], [PRT @p],
   VP[ING, +PRD, AGR N2].
IDRULE VP/SR_INGB_PHR : ; AGR V2 version
   VP[AGR V2] -> H[SUBCAT SC_ING, SUBTYPE RAIS, PRT @p], [PRT @p],
   VP[ING, +PRD, AGR V2].
IDRULE VP/SR_INGA_PREP : ; there could do with being less people
                           ; (subject raising). AGR N2 version
   VP[AGR N2] -> H[SUBCAT SC_ING, SUBTYPE RAIS, PREP @pf],
   P[PFORM @pf], VP[ING, +PRD, AGR N2].
IDRULE VP/SR_INGB_PREP : ; AGR V2 version
   VP[AGR V2] -> H[SUBCAT SC_ING, SUBTYPE RAIS, PREP @pf],
   P[PFORM @pf], VP[ING, +PRD, AGR V2].
IDRULE VP/SE_ING : ; he anticipated having to leave (subject equi)
   VP[AGR N2] -> H[SUBCAT SC_ING, SUBTYPE EQUI],
   VP[ING, +PRD, AGR N2[NFORM NORM]].
IDRULE VP/SE_ING_PHR : ; he put off leaving (subject equi)
   VP[AGR N2] -> H[SUBCAT SC_ING, SUBTYPE EQUI, PRT @p], [PRT @p],
   VP[ING, +PRD, AGR N2[NFORM NORM]].
IDRULE VP/SE_ING_PREP : ; he was banking on leaving (subject equi)
```

```
   VP[AGR N2] -> H[SUBCAT SC_ING, SUBTYPE EQUI, PREP @pf],
   P[PFORM @pf], VP[ING, +PRD, AGR N2[NFORM NORM]].
IDRULE VP/SE_ING_PREP_PHR : ; he never got around to leaving (subject
                            ; equi)
   VP[AGR N2] -> H[SUBCAT SC_ING, SUBTYPE EQUI, PREP @pf, PRT @p],
   [PRT @p, PREP @pf], VP[ING, +PRD, AGR N2[NFORM NORM]].
IDRULE VP/SR_PAS : ; get arrested (subject raising)
   VP[AGR N2] -> H[SUBCAT SC_PASS, SUBTYPE RAIS], VP[PAS, AGR N2].
IDRULE VP/SR_PP_INFA : ; seems (to fido) to be clever (subject
                       ; raising) AGR N2 version
   VP[AGR N2] -> H[SUBCAT SC_PP_INF, SUBTYPE RAIS, PFORM @pf],
   P2[PFORM @pf], VP[TO, AGR N2].
IDRULE VP/SR_PP_INFB : ; AGR V2 version
   VP[AGR V2] -> H[SUBCAT SC_PP_INF, SUBTYPE RAIS, PFORM @pf],
   P2[PFORM @pf], VP[TO, AGR V2].
IDRULE VP/SE_PP_INF : ; he conspired with them to do it
   VP[AGR N2] -> H[SUBCAT SC_PP_INF, SUBTYPE EQUI, PFORM @pf],
   P2[PFORM @pf], VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/SE_NP_INF : ; promise (fido) to go (subject equi) in spite
                      ; of the NP object.
   VP[AGR N2] -> H[SUBCAT SC_NP_INF], N2[-PRD, NFORM NORM],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/SR_NP_AP : ; he strikes me as foolish, it strikes me as
                     ; likely that .. subject control - this patterns
                     ; like 'promise' (the NP object isn't the
                     ; controller)
   VP[AGR N2] -> H[SUBCAT SC_NP_AP, PREP @pf], N2[-PRD, NFORM NORM],
   P[PFORM @pf], A2[-ADV, AGR N2, +PRD].
IDRULE VP/SR_NP_NP : ; he strikes me as a fool subject control - but
                     ; no binding as there's no point syntactically
   VP[AGR N2] -> H[SUBCAT SC_NP_AP, PREP @pf], N2[-PRD, NFORM NORM],
   P[PFORM @pf], N2[+PRD].
IDRULE VP/OC_NP : ; consider fido a fool. 1st NP = -PRD
                  ; (passivisable), second NP = +PRD (not
                  ; passivisable)
   VP -> H[SUBCAT OC_NP], N2[-PRD], N2[+PRD].
IDRULE VP/OR_AP : ; consider fido clever. Object Raising - see
                  ; proprule OBJ_CONTROL.
   VP -> H[SUBCAT OC_AP, SUBTYPE RAIS], N2[-PRD], A2[-ADV, AGR N2].
IDRULE VP/OR_AP_PHR : ; make him out crazy
   VP -> H[SUBCAT OC_AP, SUBTYPE RAIS, PRT @p], N2[-PRD], [PRT @p],
   A2[-ADV, AGR N2].
IDRULE VP/OE_AP : ; count him absent
   VP -> H[SUBCAT OC_AP, SUBTYPE EQUI], N2[-PRD, NFORM NORM],


   A2[-ADV, AGR N2].
IDRULE VP/OE_AP_PHR : ; sand it down smooth
   VP -> H[SUBCAT OC_AP, SUBTYPE EQUI, PRT @p], N2[-PRD, NFORM NORM],
   [PRT @p], A2[-ADV, AGR N2].
IDRULE VP/OE_AP_PREP : ; condemn him as stupid
   VP -> H[SUBCAT OC_AP, SUBTYPE EQUI, PREP @pf],
   N2[-PRD, NFORM NORM], P[PFORM @pf], A2[-ADV, AGR N2, +PRD].
IDRULE VP/OE_AP_PREP_PHR : ; put him down as stupid
   VP -> H[SUBCAT OC_AP, SUBTYPE EQUI, PREP @pf, PRT @p],
   N2[-PRD, NFORM NORM], [PRT @p, PREP @pf], A2[-ADV, AGR N2].
IDRULE VP/OR_INF : ; believe fido to be an idiot (object raising)
```

```
   VP -> H[SUBCAT OC_INF, SUBTYPE RAIS], N2[-PRD], VP[TO, AGR N2].
IDRULE VP/OR_INF_PHR : ; make him out to be crazy
   VP -> H[SUBCAT OC_INF, SUBTYPE RAIS, PRT @p], [PRT @p], N2[-PRD],
   VP[TO, AGR N2].
IDRULE VP/OR_INF_PREP : ; allow for there to be traffic jams
   VP -> H[SUBCAT OC_PP_INF, PFORM @pf, SUBTYPE PVERB_OR],
   P[PFORM @pf], N2[-PRD], VP[TO, AGR N2].
IDRULE VP/OE_INF1 : ; persuade fido to dance (object equi)
   VP -> H[SUBCAT OC_INF, SUBTYPE EQUI], N2[-PRD],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/OE_INF1_PHR : ; draw on her strength to help him (object
                        ; equi)
   VP -> H[SUBCAT OC_INF, SUBTYPE EQUI, PRT @p], [PRT @p], N2[-PRD],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/OE_INF1_PREP1 : ; look to him to help us
   VP -> H[SUBCAT OC_PP_INF, PFORM @pf, SUBTYPE PVERB_OE],
   P[PFORM @pf], N2[-PRD], VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/OE_INF1_PREP2 : ; AGR IT version of VP/OE_INF1_PREP1 - it
                          ; falls to you to help us)
   VP[AGR N2[NFORM IT]] ->
   H[SUBCAT OC_PP_INF, PFORM @pf, SUBTYPE PVERB_OE], P[PFORM @pf],
   N2[-PRD], VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/OE_INF1_PREP3 : ; appeal to him to help us
   VP -> H[SUBCAT OC_PP_INF, PFORM @pf, SUBTYPE EQUI], P[PFORM @pf],
   N2[-PRD], VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/OE_INF1_PREP_PHR : ; come down on him to help us
   VP -> H[SUBCAT OC_PP_INF, PFORM @pf, PRT @p, SUBTYPE PVERB_OE],
   [PRT @p], P[PFORM @pf], N2[-PRD], VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/OE_INF2A : ; to go to the opera amuses him
   VP[AGR VP[-FIN]] -> H[SUBCAT OC_INF, SUBTYPE EQU_EXTRAP], N2[-PRD].
IDRULE VP/OE_INF2B : ; it amuses him to go to the opera (extrap)
   VP[AGR N2[NFORM IT]] -> H[SUBCAT OC_INF, SUBTYPE EQU_EXTRAP],
   N2[-PRD], VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/OE_INF3 : ; it behoves you to make amends
   VP[AGR N2[NFORM IT]] -> H[SUBCAT OC_INF, SUBTYPE EQUI], N2[-PRD],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/OR_ING : ; he anticipated them leaving (object raising)
   VP -> H[SUBCAT OC_ING, SUBTYPE RAIS], N2[-PRD],
   VP[ING, +PRD, AGR N2].
IDRULE VP/OR_ING_PHR : ; he hadn't figured on them leaving (object
                     ; raising)
   VP -> H[SUBCAT OC_ING, SUBTYPE RAIS, PRT @p], [PRT @p], N2[-PRD],
   VP[ING, +PRD, AGR N2].
IDRULE VP/OR_ING_PREP : ; bank on him leaving, put him off leaving
   VP -> H[SUBCAT OC_PP_ING, PFORM @pf, SUBTYPE PVERB_OR],
   P[PFORM @pf], N2[-PRD], VP[ING, AGR N2].
IDRULE VP/OE_ING : ; their claim set him thinking (object equi)
   VP -> H[SUBCAT OC_ING, SUBTYPE EQUI], N2[-PRD],
   VP[ING, +PRD, AGR N2[NFORM NORM]].
IDRULE VP/OE_ING_PREP : ; end in him leaving, let him off washing up
   VP -> H[SUBCAT OC_PP_ING, PFORM @pf, SUBTYPE PVERB_OE],
   P[PFORM @pf], N2[-PRD], VP[ING, AGR N2[NFORM NORM]].
IDRULE VP/OE_ING_PREP_PHR : ; give oneself over to dancing
   VP -> H[SUBCAT OC_PP_ING, PFORM @pf, SUBTYPE PVERB_OE, PRT @p],
   N2[-PRD], [PRT @p, PREP @pf], VP[ING, AGR N2[NFORM NORM]].
IDRULE VP/OR_BSE : ; let/make lee wash up (object raising)
   VP -> H[SUBCAT OC_BSE, SUBTYPE RAIS], N2[-PRD], VP[BSE, AGR N2].
IDRULE VP/OE_BSE : ; hear/see/help lee leave (object equi)
```

```
   VP -> H[SUBCAT OC_BSE, SUBTYPE EQUI], N2[-PRD],
   VP[BSE, AGR N2[NFORM NORM]].
IDRULE VP/OE_BSE_PREP : ; look at him leave
   VP -> H[SUBCAT OC_PP_BSE, PFORM @pf, SUBTYPE PVERB_OE],
   P[PFORM @pf], N2[-PRD], VP[BSE, AGR N2[NFORM NORM]].
IDRULE VP/OR_PAS : ; get lee arrested (object raising)
   VP -> H[SUBCAT OC_PASS, SUBTYPE RAIS], N2[-PRD], VP[PAS, AGR N2].
IDRULE VP/OE_PAS : ; see lee arrested
   VP -> H[SUBCAT OC_PASS, SUBTYPE EQUI], N2[-PRD],
   VP[PAS, AGR N2[NFORM NORM]].
IDRULE VP/INFA : ; to see them hurts (arbitrary control)
   VP[AGR VP[-FIN]] -> H[SUBCAT VPINF, SUBTYPE EXTRAP].
IDRULE VP/INFB : ; it hurts to see them (arbitrary control) (extrap)
   VP[AGR N2[NFORM IT]] -> H[SUBCAT VPINF, SUBTYPE EXTRAP],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE VP/DO1 : ; does dance. Auxiliaries and subject raising verbs
                ; take on the AGR value of their VP complement, so
                ; can be either [AGR S] or [AGR N2]. There are two
                ; versions of each aux or subject raising rule so
                ; that the proprules can pick up AGR N2 or AGR V2 and
                ; bind their values properly. This one AGR N2.
   VP[+AUX, +FIN, AGR N2, VFORM NOT] -> H[SUBCAT DO],
   VP[AUX -, BSE, AGR N2].
IDRULE VP/DO2 : ; AGR V2 version
   VP[+AUX, +FIN, AGR V2, VFORM NOT] -> H[SUBCAT DO],
   VP[AUX -, BSE, AGR V2].
IDRULE VP/DO3 : ; for imperatives with 'do' and 'don't' - unlike the
                ; the normal case (*fido does be a nice dog), 'do'
                ; can be followed by a +AUX VP in the imperative (do
                ; be a nice dog!). There's a problem with *do be
                ; going, *do have been good, though.
   VP[+AUX, +FIN, BSE, AGR N2] -> H[SUBCAT DO], VP[BSE, AGR N2].
IDRULE VP/TO1 : ; to + base VP (AGR N2 version)
   VP[+AUX, TO, -FIN, AGR N2] -> H[SUBCAT TO], VP[BSE, AGR N2].
IDRULE VP/TO2 : ; to + base VP (AGR V2 version)
   VP[+AUX, TO, -FIN, AGR V2] -> H[SUBCAT TO], VP[BSE, AGR V2].
IDRULE VP/MODAL1 : ; can dance, may be dancing etc. AGR N2 version
   VP[+AUX, +FIN, AGR N2] -> H[SUBCAT MODAL], VP[BSE, AGR N2].
IDRULE VP/MODAL2 : ; AGR V2 version
   VP[+AUX, +FIN, AGR V2] -> H[SUBCAT MODAL], VP[BSE, AGR V2].
IDRULE VP/OUGHT1 : ; ought to dance. AGR N2 version
   VP[+AUX, +FIN, AGR N2] -> H[SUBCAT OUGHT], VP[TO, +AUX, AGR N2].
IDRULE VP/OUGHT2 : ; AGR V2 version
   VP[+AUX, +FIN, AGR V2] -> H[SUBCAT OUGHT], VP[TO, +AUX, AGR V2].
IDRULE VP/HAVE1 : ; have, has, had (gone etc) I've ignored the modal
                  ; use of 'have' ie 'have to'. AGR N2 version.
   VP[+AUX, AGR N2] -> H[SUBCAT HAVE], VP[EN, PRD -, AGR N2].
IDRULE VP/HAVE2 : ; AGR V2 version
   VP[+AUX, AGR V2] -> H[SUBCAT HAVE], VP[EN, PRD -, AGR V2].
IDRULE VP/BE_VP1 : ; is attacked by the cat, is attacking the cat.
                   ; Collapses 'be' as a passive auxiliary and 'be'
                   ; as a progressive auxiliary. Overgenerates
                   ; slightly to allow 'is being (being)* dancing' .
                   ; AGR N2 version
   VP[+AUX, AGR N2] -> H[SUBCAT BE], VP[+PRD, AGR N2].
IDRULE VP/BE_VP2 : ; AGR V2 version
   VP[+AUX, AGR V2] -> H[SUBCAT BE], VP[+PRD, AGR V2].
IDRULE VP/BE_NP : ; fido is a dog. Notice that this and the next few
```

```
                        ; rules (and the BE_VP rules) have to be separate
                        ; since with full unification its not possible to
                        ; use XP in id rules. The +PRD on the N2 prevents
                        ; the application of the passive metarule.
   VP[+AUX] -> H[SUBCAT BE], N2[+PRD].
IDRULE VP/BE_NP/NEG : ; fido is not a dog. Decided that the 'not'
                      ; isn't in constituency with the +PRD category
                      ; (what is fido not) hence the need for these
                      ; rules.
   VP[+AUX, +NEG] -> H[SUBCAT BE, +FIN, -NEG], [SUBCAT NOT], N2[+PRD].
IDRULE VP/BE_PP : ; fido is in the kennel. The PFORM NORM prevents
                  ; two parses since a particular preposition isn't
                  ; subcategorised for.
   VP[+AUX] -> H[SUBCAT BE], P2[+PRD, PFORM NORM].
IDRULE VP/BE_PP/NEG : ; fido is not in the kennel.
   VP[+AUX, +NEG] -> H[SUBCAT BE, +FIN, -NEG], [SUBCAT NOT],
   P2[+PRD, PFORM NORM].
IDRULE VP/BE_AP1 : ; fido is small. AGR N2 version. Subject raising -
                   ; see proprules SUBJ_CONTROL*
   VP[+AUX, AGR N2] -> H[SUBCAT BE], A2[+PRD, AGR N2, NEG -].
IDRULE VP/BE_AP1/NEG : ; fido is not small. AGR N2 version. Subject
                       ; raising - see proprules SUBJECT_CONTROL*
   VP[+AUX, +NEG, AGR N2] -> H[SUBCAT BE, +FIN, -NEG], [SUBCAT NOT],
   A2[+PRD, AGR N2].
IDRULE VP/BE_AP2 : ; (that fido is a dog) is obvious. AGR V2 version.
                   ; Subject raising - see proprule AGRV2_CONTROL2.
   VP[+AUX, AGR V2] -> H[SUBCAT BE], A2[+PRD, AGR V2, NEG -].
IDRULE VP/BE_AP2/NEG : ; that fido is a dog is not obvious. AGR V2
                       ; version. Subject control - see proprule
                       ; AGRV2_CONTROL2.
   VP[+AUX, AGR V2, +NEG] -> H[SUBCAT BE, +FIN, -NEG], [SUBCAT NOT],
   A2[+PRD, AGR V2].
IDRULE VP/PRD/CONJ : ; this allows for coordination of +PRD
                     ; categories after 'be', whether of the same or
                     ; differing categories - see idrules CONJ/PRD*
                     ; and PRD/COORD*. The +PRD categories in the
                     ; VP/BE_* rules will be defaulted to -COORD to
                     ; prevent multiple parses - defrule COORD2.
   VP[+AUX] -> H[SUBCAT BE], X2[+PRD, COORD +].
IDRULE VP/PRD/CONJ2 : ; allows for the coordination of various +PRD
                      ; categories following 'be' and 'not'. To avoid
                      ; overgeneration, the complements in the
                      ; VP/BE_*/NEG rules are defaulted to -COORD
                      ; (defrule COORD2).
   VP[+AUX, +NEG] -> H[SUBCAT BE, +FIN, -NEG], [SUBCAT NOT],
   X2[+PRD, COORD +].
IDRULE VP/BE_THERE : ; (there) is a mouse in the bathtub
   VP[+AUX, AGR N2[NFORM THERE, PLU @pl]] -> H[SUBCAT BE],
   N2[PLU @pl, -DEF, PRD +].
IDRULE VP/BE_CLEFT1 : ; it is lee who/that relies on kim, it is lee
                      ; who/that kim relies on. This is equivalent to
                      ; the first of the it-cleft rules in GPSG85.
                      ; The second was unnecessary.
   VP[+AUX, AGR N2[NFORM IT]] -> H[SUBCAT BE], N2[+PRD, NFORM NORM],
   S[+R].
IDRULE VP/BE_CLEFT2 : ; it is on lee (that) kim relies. Partially
                      ; equivalent to second of the it-cleft rules in
                      ; GPSG85. The P2 is bound to the SLASH P2 by
```

```
                    ; proprule SLASH/AGR_PP2.
  VP[+AUX, AGR N2[NFORM IT]] -> H[SUBCAT BE], P2, S[+FIN, SLASH P2].
IDRULE VP/PP/PASS : ; This and the next few rules to do some kinds of
                    ; passive not produced by the passive metarule.
                    ; This one does prepositional passives - he hates
                    ; being looked at - the verb has to be SUBTYPE
                    ; PVERB.
  VP[PAS] -> H[SUBCAT PP, PFORM @pf, SUBTYPE PVERB], P[PFORM @pf].
IDRULE VP/PP_PHR/PASS : ; the jail has never been broken out of
  VP[PAS] -> H[SUBCAT PP, PFORM @pf, PRT @p, SUBTYPE PVERB],
  [PRT @p], P[PFORM @pf].
IDRULE VP/SFIN/PASS : ; 'that kim drinks is believed by lee'.
  VP[PAS, AGR S[+FIN]] -> H[SUBCAT SFIN, SUBTYPE NONE], ( P2[by] ).
IDRULE VP/NP_SFIN2B/PASS : ; passive version of VP/NP_SFIN2B - 'fido
                           ; is bothered that lee attacks cats'.
  VP[PAS, AGR N2[NFORM NORM]] -> H[SUBCAT NP_SFIN, SUBTYPE EXTRAP],
  S[+FIN, that].
IDRULE VP/OR_INF/PASS_A : ; passive version of VP/OR_INF - object
                          ; raising verbs become subject raising
                          ; verbs in the passive. This one AGR N2
                          ; version (see proprules SUBJ_CONTROL*) -
                          ; 'fido is believed to be a dog' 'it is
                          ; believed to bother lee that ..' etc.
  VP[PAS, AGR N2] -> H[SUBCAT OC_INF, SUBTYPE RAIS], VP[TO, AGR N2],
  ( P2[by] ).
IDRULE VP/OR_INF/PASS_B : ; same as VP/OR_INF/PASS_A except AGR V2
                          ; version (see proprule AGRV2_CONTROL) -
                          ; 'that fido dances is believed to bother
                          ; lee' '(for us) to dance is believed to be
                          ; possible'.
  VP[PAS, AGR V2] -> H[SUBCAT OC_INF, SUBTYPE RAIS], VP[TO, AGR V2],
  ( P2[by] ).
IDRULE VP/OR_INF_PHR/PASS_A : ; passive version of VP/OR_INF_PHR -
                              ; object raising verbs become subject
                              ; raising verbs in the passive. He is
                              ; made out to be crazy (AGR N2 version)
  VP[PAS, AGR N2] -> H[SUBCAT OC_INF, SUBTYPE RAIS, PRT @p],
  [PRT @p], VP[TO, AGR N2], ( P2[by] ).
IDRULE VP/OR_INF_PHR/PASS_B : ; same as VP/OR_INF_PHR/PASS_A except
                              ; AGR V2 version (see proprule
                              ; AGRV2_CONTROL) - 'that he wouldn't
                              ; arrive was made out to be likely
  VP[PAS, AGR V2] -> H[SUBCAT OC_INF, SUBTYPE RAIS, PRT @p],
  [PRT @p], VP[TO, AGR V2], ( P2[by] ).
IDRULE VP/OR_AP/PASS_A : ; passive version of VP/TAKES_NP_AP - again
                         ; object raising becomes subject raising in
                         ; passive. This one AGR N2 version (see
                         ; proprule SUBJ_CONTROL3). 'lee is
                         ; considered stupid' 'it is considered
                         ; likely to bother lee that fido is a dog'
                         ; etc.
  VP[PAS, AGR N2] -> H[SUBCAT OC_AP, SUBTYPE RAIS], A2[AGR N2],
  ( P2[by] ).
IDRULE VP/OR_AP/PASS_B : ; same as VP/OR_AP/PASS_A except AGR V2
                         ; version (see proprule AGRV2_CONTROL) -
                         ; 'that fido dances is considered likely to
                         ; bother lee' '(for us) to dance is
                         ; considered possible'.
```

```
   VP[PAS, AGR V2] -> H[SUBCAT OC_AP, SUBTYPE RAIS], A2[AGR V2],
   ( P2[by] ).
IDRULE VP/MOD1 : ; adverbial modifier of VP. To restrict number of
                 ; parses only allowed to attach to a -AUX VP. Will
                 ; linearise either way. fido dances quickly / fido
                 ; quickly dances
   VP[-AUX] -> VP[H +], A2[+ADV, COORD -].
IDRULE VP/MOD2 : ; PP modifier of VP. Again only attaches to -AUX VP.
                 ; Lprule LP29 linearises it - fido dances on monday
                 ; / *fido on monday dances.
   VP[-AUX] -> VP[H +], P2[PFORM NORM, COORD -].
IDRULE VP/MOD3 : ; because of examples like 'fido is obviously being
                 ; eaten' adverbial modifers have to be allowed to
                 ; attach to +AUX VPS - but lprule LP28 ensures that
                 ; they can only precede the VP.
   VP[+AUX] -> A2[+ADV], VP[H +].
IDRULE VP/MOD/CONJ : ; rule to allow coordination of adverbial and PP
                     ; modifiers. Also does coordination of PP and
                     ; PP, ADV and ADV which is why rules VP/MOD1 and
                     ; VP/MOD2 contain a -COORD restriction.
   VP[-AUX] -> VP[H +], X2[+ADV, COORD +].
IDRULE VP/NEG : ; (does) not dance. -FIN prevents 'not dances'
   VP[NEG +] -> [SUBCAT NOT], H2[SUBJ -, FIN -].
IDRULE VP/WH1 : ; whether to go
   VP[+Q] -> [SUBCAT WHETHER], VP[H +, TO, AGR N2[NFORM NORM]].
IDRULE VP/WH2 : ; what to do etc
   VP[+Q, +WH, -EVER] -> N2[+Q, +WH, -EVER],
   VP[H +, TO, SLASH N2, AGR N2[NFORM NORM]].
IDRULE VP/PRO : ; pro vps 'he does' 'he wouldn't' etc
   VP[+AUX, AGR N2] -> H[SUBCAT NULL, PRO +].

PSRULE PRT/PREP : ; a fix for some phrasal prepositional verbs - it
                  ; proved too difficult to linearise them properly.
   [PRT @1, PREP @2] -> [PRT @1] P[PFORM @2].
```

; NP rules

```
IDRULE T/NP : ; this recognises NP as a TOP category (in addition to
              ; to S (see idrules T and T2)). It is useful when
              ; testing the NP rules but can be deleted if not
              ; wanted.
   [T +] -> N2[H +].
IDRULE N2+/PRO : ; pronouns - i, me, my, who, whose etc
   N2[+SPEC] -> H[SUBCAT NULL, PRO +].
IDRULE N2+/PRO2 : ; for the pronouns (all, each etc) which head
                  ; partitives and which have an extra feature PART
   N2[+SPEC, -POSS, PART @x] -> H[SUBCAT NULL, +PRO, PART @x].
IDRULE N2+/DET : ; the, a, this dog. A number of different specifiers
                 ; attach under N2[+SPEC] ie Det, POSS NP, A2[+QUA] -
                 ; in complementary distribution. This rule does Det.
   N2[+SPEC] -> DetN[AGR N2], H2[-SPEC].
IDRULE N2+/POSSNP : ; Possessive NP in determiner position - 'fido s
                    ; kennel' 'the dog s kennel'.
   N2[+SPEC, +DEF] -> N2[POSS +], H2[-SPEC].
IDRULE POSSNP : ; Possessive NP - 'fido 's' 'the man 's' etc.
   N2[SPEC @x, +POSS] -> N2[-POSS, PRO -, SPEC @x], H1[+POSS].
IDRULE N1/POSS : ; s (ie the possessive morpheme)
   N1[+POSS] -> H[SUBCAT NULL, +POSS].
```

```
IDRULE N2+/QUA : ; some, all, nearly all, both books
   N2[+SPEC] -> A2[+QUA, -PRD, AGR N2], H2[-SPEC].
IDRULE N2+/PART1 : ; partitives with 'of'
   N2[+SPEC] -> H2[+SPEC, PART OF], [PFORM OF], N2[+SPEC, CASE ACC].
IDRULE N2+/PART2 : ; partitives without 'of'
   N2[+SPEC] -> H2[+SPEC, PART NO_OF], N2[+SPEC, CASE ACC, COORD -].
IDRULE N2+/PART3 : ; partitives where the head doesn't agree with the
                   ; other N2
   N2[+SPEC] -> H2[+SPEC, PART OF2], [PFORM OF],
   N2[+SPEC, CASE ACC, +PLU].
IDRULE N2+/ADJ1 : ; the stupid, the very stupid, the extremely stupid
   N2[+SPEC, PLU +] -> DetN[DEF +], A2[AFORM NONE].
IDRULE N2+/ADJ2 : ; the most stupid, the poorest (either sing or
                  ; plural)
   N2[+SPEC] -> DetN[DEF +], A2[AFORM EST].
IDRULE N2+/ADJ3 : ; the more stupid, the cleverer (either sing or
                  ; plural)
   N2[+SPEC] -> DetN[DEF +], A2[AFORM ER].
IDRULE N2- : ; dogs, *dog, truth, bread. Doing determiner-less NPs
             ; here rather than making an N2[+SPEC] specifier
             ; optional - ensures that you get a determiner in
             ; partitives. Also gives you simple NPs like 'the book'
   N2[-SPEC] -> H1.
IDRULE N2-/QUA : ; many, too many, three, several dogs. +PRD
                 ; quantifiers attach under N2[-SPEC] which can
                 ; appear either on its own or as part of an
                 ; N2[+SPEC] - 'the many dogs'
   N2[-SPEC] -> A2[+PRD, +QUA, AGR N1], H1.
IDRULE N2/APPOS1 : ; an N2 in apposition to a name - John, my brother
   N2[+SPEC] -> H2[+SPEC, PN +], N2[+PRD, +SPEC, PN -, PRO -].
IDRULE N2/APPOS2 : ; a name in apposition to an N2 - my brother, John
   N2[+SPEC] -> H2[+SPEC, PN -], N2[+PRD, +SPEC, PN +, PRO -].
IDRULE N2/ADVP : N2[+SPEC] -> A2[+ADV], H2[+SPEC].
IDRULE N2/NEG : ; not a dog. +NEG N2s will only appear in coordinate
                ; structures - see defrules NP/NEG1 and NP/NEG2.
   N2[NEG +, SPEC @sp] -> [SUBCAT NOT], H2[NEG -, SPEC @sp].
IDRULE N1/N : ; an N with no complements
   N1 -> H[SUBCAT NULL].
IDRULE N1/N_PHR : ; a rummage around
   N1 -> H[SUBCAT NULL, PRT @p], [PRT @p].
IDRULE N1/PP : N1 -> H[SUBCAT PP, PFORM @pf], P2[PFORM @pf].
IDRULE N1/SFIN : ; fact that fido is a dog
   N1 -> H[SUBCAT SFIN], S[+FIN, that].
IDRULE N1/SBSE : ; requirement that fido dance
   N1 -> H[SUBCAT SBSE], S[that, BSE].
IDRULE N1/VPINF : ; desire to dance
   N1 -> H[SUBCAT VPINF], VP[TO, AGR N2[NFORM NORM]].
IDRULE N1/APMOD1 : ; busy man. The -PRD, DISTR ATT restriction on the
                   ; A2 prevents adjectives with complements
                   ; matching. The feature MOD on the N1 cuts down on
                   ; the number of parses by making premodifiers
                   ; attach lower than postmodifiers.
   N1[MOD PRE] -> A2[-PRD, DISTR ATT, -QUA], H1[MOD NONE].
IDRULE N1/APMOD2 : ; allows for iteration of prenominal AP
   N1[MOD PRE] -> A2[-PRD, DISTR ATT, -QUA], H1[MOD PRE].
IDRULE N1/POSSMOD1 : ; the women's javelin
   N1[MOD PRE] -> N2[POSS +, SPEC -], H1[MOD NONE].
IDRULE N1/POSSMOD2 : ; allows for iteration as with NP/APMOD1 & 2
```

```
   N1[MOD PRE] -> N2[POSS +, SPEC -], H1[MOD PRE].
IDRULE N1/POST_APMOD : ; post-nominal AP - a man taller than lee, *a
                       ; man stupid - only adjs with complements
                       ; (DISTR PRD) or comparatives can match
   N1[MOD POST] -> H1, A2[+PRD, DISTR PRD, -QUA].
IDRULE N1/PPMOD : ; man with the umbrella
   N1[MOD POST] -> H1, P2[-GER, PFORM NORM].
IDRULE N1/PP_POSS : ; a book of fidos. separate from PPMOD since P2
                    ; defaults to -POSS
   N1[MOD POST] -> H1, P2[+POSS].
IDRULE N1/INFMOD : ; the man to ask
   N1[MOD POST] -> H1, VP[TO, SLASH N2[+ACC], AGR N2[NFORM NORM]].
IDRULE N1/VPMOD : ; sheep attacked by fido, sheep attacking fido
   N1[MOD POST] -> H1, VP[PRD +, FIN -, AGR N2[NFORM NORM]].
IDRULE N1/REL : ; sheep who/that attacks fido, sheep who/that fido
                ; attacks
   N1[MOD POST] -> H1, S[+R, -EVER].
IDRULE THATLESS/REL : ; (the cat) fido attacked e. This S[+R] matches
                      ; the S[+R] in N1/REL.
   S[+R, -WH, -EVER, -INV] -> S[H +, COMP NORM, FIN +, SLASH N2].
IDRULE FREE/REL : ; whichever you like e, whatever book you like e
   N2[+SPEC] -> N2[H +, +WH, +EVER, +R], S[COMP NORM, SLASH N2, -INV].
IDRULE FREE/REL2 : ; whichever seems right
   N2[+SPEC] -> N2[H +, +WH, +EVER, +R], VP[+FIN].
IDRULE N/NUMBER1 : ; this recognises a cardinal number (as produced
                   ; by the NUM psrules) as a pronoun - this rule for
                   ; SUBCAT NULL pronouns in partitives (PART OF) eg
                   ; three of the books.
   N[-POSS, -DEF, +PLU, +PRO, +COUNT, CARD, NFORM NORM, PER 3, PN -,
      PART OF, AFORM NONE, SUBCAT NULL] -> [CN1 @x, CN2 @y, AND @z].
IDRULE N/NUMBER2 : ; numbers can be ordinary pronouns
   N[-POSS, -DEF, +PLU, +PRO, +COUNT, CARD, NFORM NORM, PER 3, PN -,
      AFORM NONE, SUBCAT NULL] -> [CN1 @x, CN2 @y, AND @z].
IDRULE N/COMPOUND1 : ; bank account
   N -> N[SUBCAT NULL, H -], H[SUBCAT NULL].
IDRULE N/COMPOUND2 : ; cold drink
   N -> A[SUBCAT NULL, H -], H[SUBCAT NULL].
IDRULE N/COMPOUND3 : ; after care
   N -> P[SUBCAT NP, H -, PFORM NORM, PRO -], H[SUBCAT NULL].
```

; AP rules

```
IDRULE A2/ADVMOD : ; exceptionally clever, nearly all. Different
                   ; adjectives co-occur with different types of
                   ; adverbial eg *nearly many, *very all so this
                   ; rule will overgenerate rather.
   A2 -> ( A2[+ADV] ), H1.
IDRULE A2/NOT : ; not all, not many. Restricted to quantifying
                ; adjectives at the moment because of *a not clever
                ; man. nb though 'a not very clever man' - here the
                ; not modifies the adverb so need to extend the rule
                ; a bit. The head must be -NEG (*not few).
   A2[+QUA] -> [SUBCAT NOT], H1[-NEG].
IDRULE A2/NEG : ; not happy. This kind of +NEG A2 will only appear in
                ; coordinate structures eg fido is happy but not
                ; clever
   A2[NEG +] -> [SUBCAT NOT], H2[NEG -].
IDRULE A1/A : ; no BAR specification on AGR because quantifying APS
```

```
                ; can be AGR N1 whilst others are AGR N2.
   A1[AGR [N +, V -, NFORM NORM], DISTR ATT] -> H[SUBCAT NULL].
IDRULE A1/A_PHR : ; full up
   A1[AGR [N +, V -, NFORM NORM]] -> H[SUBCAT NULL, PRT @p], [PRT @p].
IDRULE A1/PP : ; anxious about everything
   A1 -> H[SUBCAT PP, PFORM @pf], P2[PFORM @pf].
IDRULE A1/PP_PHR : ; made up of, shot through with
   A1 -> H[SUBCAT PP, PFORM @pf, PRT @p], [PRT @p], P2[PFORM @pf].
IDRULE A1/LOC : ; situated by the abbey
   A1 -> H[SUBCAT LOC], P2[+LOC, PFORM NORM].
IDRULE A1/PP_PP : ; accountable to us for his actions
   A1 -> H[SUBCAT PP_PP, PFORM @pf], P2[PFORM @pf], P2[PFORM @pf].
IDRULE A1/SFIN1 : ; aware (that) he might not apologize
   A1 -> H[SUBCAT SFIN, SUBTYPE NONE], S[+FIN].
IDRULE A1/SFIN2A : ; that he is able to help is convenient
   A1[AGR S[+FIN]] -> H[SUBCAT SFIN, SUBTYPE EXTRAP].
IDRULE A1/SFIN2B : ; it is convenient that he is able to help
   A1[AGR N2[NFORM IT]] -> H[SUBCAT SFIN, SUBTYPE EXTRAP], S[FIN +].
IDRULE A1/PP_SFINA : ; that he wouldn't apologize was clear to me
   A1[AGR S[+FIN]] -> H[SUBCAT PP_SFIN, SUBTYPE EXTRAP, PFORM @pf],
   P2[PFORM @pf].
IDRULE A1/PP_SFINB : ; it was clear to me that he wouldn't apologize
   A1[AGR N2[NFORM IT]] ->
   H[SUBCAT PP_SFIN, SUBTYPE EXTRAP, PFORM @pf], P2[PFORM @pf],
   S[+FIN].
IDRULE A1/SBSE1 : ; eager/fearful that you answer
   A1 -> H[SUBCAT SBSE, SUBTYPE NONE], S[COMP THAT, BSE].
IDRULE A1/SBSE2A : ; that you apologize is necessary
   A1[AGR S[-FIN, BSE]] -> H[SUBCAT SBSE, SUBTYPE EXTRAP].
IDRULE A1/SBSE2B : ; it is necessary that you apologize
   A1[AGR N2[NFORM IT]] -> H[SUBCAT SBSE, SUBTYPE EXTRAP],
   S[COMP THAT, BSE].
IDRULE A1/SINF1 : ; he is eager for us to help
   A1 -> H[SUBCAT SINF, SUBTYPE NONE], S[COMP FOR].
IDRULE A1/SINF2A : ; for him to aplogize is inessential
   A1[AGR S[-FIN, TO]] -> H[SUBCAT SINF, SUBTYPE EXTRAP].
IDRULE A1/SINF2B : ; it is inessential for him to apologize
   A1[AGR N2[NFORM IT]] -> H[SUBCAT SINF, SUBTYPE EXTRAP],
   S[TO, COMP FOR, -FIN].
IDRULE A1/VPINF1A : ; to apologize is normal
   A1[AGR VP[-FIN]] -> H[SUBCAT VPINF, SUBTYPE EXTRAP].
IDRULE A1/VPINF1B : ; it is normal to apologize
   A1[AGR N2[NFORM IT]] -> H[SUBCAT VPINF, SUBTYPE EXTRAP],
   VP[TO, -FIN].
IDRULE A1/VPINF2 : ; for 'silly' type adjectives - 'it is silly for
                   ; you to do that'. Variants of such adjs are
                   ; generated by the morphology system
   A1[AGR N2[NFORM IT]] -> H[SUBCAT VPINF, SUBTYPE SILLY],
   P2[PFORM OF], VP[TO, AGR N2[NFORM NORM]].
IDRULE A1/VPINF3A : ; for 'ready' type adjectives - 'we are ready to
                    ; eat dinner'.
   A1[AGR N2] -> H[SUBCAT VPINF, SUBTYPE READY],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE A1/VPINF3B : ; for 'ready' type adjectives - 'the dinner is
                    ; ready to eat e'. (different from tough adjs
                    ; because 'an easy food to cook e' vs '* a ready
                    ; food to cook e')
   A1[AGR N2] -> H[SUBCAT VPINF, SUBTYPE READY],
   VP[TO, SLASH N2[+ACC], AGR N2[NFORM NORM]].
IDRULE A1/VPINF3C : ; for 'ready' type adjectives - the dinner is
                    ; ready for us to eat e.
   A1 -> H[SUBCAT VPINF, SUBTYPE READY], S[COMP FOR, SLASH N2[+ACC]].
IDRULE A1/SR_INFA : ; he is certain to help
   A1[AGR N2] -> H[SUBCAT SC_INF, SUBTYPE RAIS], VP[TO, AGR N2].
IDRULE A1/SR_INFB : ; that he apologized is certain to amuse her
   A1[AGR V2] -> H[SUBCAT SC_INF, SUBTYPE RAIS], VP[TO, AGR V2].
IDRULE A1/SE_INF : ; eager to help
   A1[AGR N2] -> H[SUBCAT SC_INF, SUBTYPE EQUI],
   VP[TO, AGR N2[NFORM NORM]].
IDRULE A1/SE_ING : ; busy helping them
   A1[AGR N2] -> H[SUBCAT SC_ING, SUBTYPE EQUI],
   VP[ING, +PRD, AGR N2[NFORM NORM]].
IDRULE A1/SE_AP : ; scared silly
   A1[AGR N2] -> H[SUBCAT SC_AP, SUBTYPE EQUI],
   A2[AGR N2[NFORM NORM]].
IDRULE A1/TOUGH1 : ; easy to amuse e
   A1[AGR N2] -> H[SUBCAT VPINF, SUBTYPE TOUGH],
   VP[TO, SLASH N2[+ACC], AGR N2[NFORM NORM]].
IDRULE A1/TOUGH2 : ; easy for us to amuse e
   A1 -> H[SUBCAT VPINF, SUBTYPE TOUGH], S[COMP FOR, SLASH N2[+ACC]].
IDRULE A1/PPING : ; confident of having been appreciated
   A1[AGR N2] -> H[SUBCAT PPING, SUBTYPE EQUI, PFORM @pf],
   P[PFORM @pf], VP[PRP, AGR N2[NFORM NORM]].
IDRULE A1/PPSING : ; confident of him having being appreciated
   A1 -> H[SUBCAT PPSING, PFORM @pf], P[PFORM @pf], S[COMP NORM, PRP].
IDRULE A1/WHSA : ; it is clear what we should do
   A1[AGR N2[NFORM IT]] -> H[SUBCAT WHS, SUBTYPE EXTRAP],
   S[+Q, -INV, +WH, -EVER].
IDRULE A1/WHSB : ; it is not clear whether we should go
   A1[AGR N2[NFORM IT]] -> H[SUBCAT WHS, SUBTYPE EXTRAP], S[+Q].
IDRULE A1/WHSC : ; whether we should help/ what we should do is not
                 ; clear
   A1[AGR S[+Q]] -> H[SUBCAT WHS, SUBTYPE EXTRAP].
IDRULE A1/PP_VPINF1A : ; to apologize is typical of him
   A1[AGR VP[-FIN]] -> H[SUBCAT PP_VPINF, SUBTYPE EXTRAP, PFORM @pf],
   P2[PFORM @pf].
IDRULE A1/PP_VPINF1B : ; it is typical of him to apologize
   A1[AGR N2[NFORM IT]] ->
   H[SUBCAT PP_VPINF, SUBTYPE EXTRAP, PFORM @pf], P2[PFORM @pf],
   VP[TO, -FIN].
IDRULE A1/PP_WHSA : ; it is clear to me what we should do
   A1[AGR N2[NFORM IT]] ->
   H[SUBCAT PP_WHS, SUBTYPE EXTRAP, PFORM @pf], P2[PFORM @pf],
   S[+Q, -INV, +WH, -EVER].
IDRULE A1/PP_WHSB : ; it is not clear to me whether we should go
   A1[AGR N2[NFORM IT]] ->
   H[SUBCAT PP_WHS, SUBTYPE EXTRAP, PFORM @pf], P2[PFORM @pf], S[+Q].
IDRULE A1/PP_WHSC : ; whether we should help/ what we should do is
                    ; not clear to me
   A1[AGR S[+Q]] -> H[SUBCAT PP_WHS, SUBTYPE EXTRAP, PFORM @pf],
   P2[PFORM @pf].
IDRULE A1/DEGMOD_1 : ; so clever, so obviously, more stupid. All -QUA
                     ; adjectives and adverbs can take a degree
                     ; modifier. The head daughter is [AFORM NONE] to
                     ; prevent 'so taller' , 'more taller'.
   A1[-QUA] -> DetA, H1[AFORM NONE].
```

```
IDRULE A1/DEGMOD_2 : ; so many, too many. The only +QUA adjectives
                     ; that can take a degree modifier are the +PRD,
                     ; -NUM ones.
   A1[+QUA, +PRD, -NUM] -> DetA, H1.
IDRULE A/NUMBER : ; numbers can be quantifying adjectives (the three
                  ; books)
   A[+PRD, -ADV, -NEG, -DEF, +QUA, CARD,
      AGR +N[+PLU, +COUNT, V -, NFORM NORM], AFORM NONE, SUBCAT NULL,
      SUBTYPE NONE] -> [CN1 @x, CN2 @y, AND @z].
IDRULE A/COMPOUND : ; age aware, ice cold, sugar free
   A[ADV -] -> N[SUBCAT NULL, H -], H[SUBCAT NULL].
```

; rules for AP and NP comparatives

```
IDRULE A2/COMPAR_1 : ; taller than lee, more stupid than lee.
   A2[+PRD] -> H2[AFORM ER], P2[PFORM THAN].
IDRULE A2/COMPAR_2 : ; as tall as lee.
   A2[+PRD] -> H2[AFORM AS], P2[PFORM AS].
IDRULE A2/COMPAR_3 : ; taller than lee is, more stupid than lee is.
   A2[+PRD] -> H2[AFORM ER], S[COMP THAN, SLASH A2].
IDRULE A2/COMPAR_4 : ; as stupid as lee is.
   A2[+PRD] -> H2[AFORM AS], S[COMP AS, SLASH A2].
IDRULE A2/COMPAR_5 : ; taller than lee is short
   A2[+PRD] -> H2[AFORM ER], S[COMP THAN, SLASH DetA].
IDRULE A2/COMPAR_6 : ; as tall as lee is short
   A2[+PRD] -> H2[AFORM AS], S[COMP AS, SLASH DetA].
IDRULE A2/COMPAR_7 : ; rather than use a metarule, this terminates
                     ; the SLASH DetA introduced by idrules
                     ; A2/COMPAR5 and A2/COMPAR6. Notice that no
                     ; empty node is introduced.
   A2[SLASH DetA] -> H1[AFORM NONE].
IDRULE N2/COMPAR_1 : ; more bones than fido, more bones than biscuits
   N2[+SPEC] -> H2[AFORM ER], P2[PFORM THAN].
IDRULE N2/COMPAR_2 : ; as many bones as fido, as many bones as
                     ; biscuits.
   N2[+SPEC] -> H2[AFORM AS], P2[PFORM AS].
IDRULE N2/COMPAR_3 : ; more bones than fido eats
   N2[+SPEC] -> H2[AFORM ER], S[COMP THAN, SLASH N2].
IDRULE N2/COMPAR_4 : ; as many bones as fido eats
   N2[+SPEC] -> H2[AFORM AS], S[COMP AS, SLASH N2].
IDRULE N2/COMPAR_5 : ; more bones than fido eats biscuits
   N2[+SPEC] -> H2[AFORM ER], S[COMP THAN, SLASH DetN].
IDRULE N2/COMPAR_6 : ; as many bones as fido eats biscuits
   N2[+SPEC] -> H2[AFORM AS], S[COMP AS, SLASH DetN].
IDRULE N2/COMPAR_7 : ; rather than use a metarule, this terminates
                     ; the SLASH DetN introduced by idrules
                     ; N2/COMPAR5 and N2/COMPAR6. Notice that no
                     ; empty node is introduced.
   N2[SLASH DetN, +SPEC] -> H1[AFORM NONE].
```

; coordination rules. The CONJ rules expand a category marked with a [CONJ]
; feature as the appropriate coordinator followed by the same
; category without a CONJ feature. In the case of [CONJ NULL]
; categories, no coordinator appears. The COORD rules expand the top
; category of a coordination as conjunct daughters of the same
; category. In most cases these conjuncts are not heads since not all
; head features ought to be percolated. The rules for coordinated NPs
; are complicated in a (mostly vain) attempt to get person and number

```
; features correctly distributed. The PRD and MOD rules at the end
; allow for the coordination of unlike categories - these
; coordinations will only appear in restricted environments.

IDRULE CONJ/N2A : N2[CONJ NULL] -> H2.
IDRULE CONJ/N2B : N2[CONJ @con] -> [SUBCAT @con, CONJN +], H2.
IDRULE CONJ/N1A : N1[CONJ NULL] -> H1.
IDRULE CONJ/N1B : N1[CONJ @con] -> [SUBCAT @con, CONJN +], H1.
IDRULE CONJ/NA : N[CONJ NULL] -> H.
IDRULE CONJ/NB : N[CONJ @con] -> [SUBCAT @con, CONJN +], H.
IDRULE CONJ/P2A : P2[CONJ NULL] -> H2.
IDRULE CONJ/P2B : P2[CONJ @con] -> [SUBCAT @con, CONJN +], H2.
IDRULE CONJ/PA : P[CONJ NULL] -> H.
IDRULE CONJ/PB : P[CONJ @con] -> [SUBCAT @con, CONJN +], H.
IDRULE CONJ/A2A : A2[CONJ NULL] -> H2.
IDRULE CONJ/A2B : A2[CONJ @con] -> [SUBCAT @con, CONJN +], H2.
IDRULE CONJ/A1A : A1[CONJ NULL] -> H1.
IDRULE CONJ/A1B : A1[CONJ @con] -> [SUBCAT @con, CONJN +], H1.
IDRULE CONJ/AA : A[CONJ NULL] -> H.
IDRULE CONJ/AB : A[CONJ @con] -> [SUBCAT @con, CONJN +], H.
IDRULE CONJ/VPA : VP[CONJ NULL] -> H2[-SUBJ].
IDRULE CONJ/VPB : VP[CONJ @con] -> [SUBCAT @con, CONJN +], H2[-SUBJ].
IDRULE CONJ/VA : V[CONJ NULL] -> H.
IDRULE CONJ/VB : V[CONJ @con] -> [SUBCAT @con, CONJN +], H.
IDRULE CONJ/SA : S[CONJ NULL] -> S[H +].
IDRULE CONJ/SB : S[CONJ @con] -> [SUBCAT @con, CONJN +], S[H +].
IDRULE N2/COORD1 : N2[PLU +] -> ( N2[CONJ NULL] )+, ( N2[CONJ AND] )+.
IDRULE N2/COORD2 : N2[PLU +] -> N2[CONJ BOTH], N2[CONJ AND].
IDRULE N2/COORD3A : N2 -> ( N2[CONJ NULL, PLU -] )+,
   ( N2[CONJ OR, PLU -] )+.
IDRULE N2/COORD3B : N2[PLU +] -> N2[CONJ NULL, PLU +],
   N2[CONJ OR, PLU -].
IDRULE N2/COORD3C : N2[PLU +] -> N2[CONJ NULL, PLU -],
   N2[CONJ OR, PLU +].
IDRULE N2/COORD3D : N2[PLU +] -> ( N2[CONJ NULL, PLU +] )+,
   ( N2[CONJ OR, PLU +] )+.
IDRULE N2/COORD4A : N2[PLU @pl] -> N2[CONJ NEITHER, PLU @pl],
   ( N2[CONJ NOR, PLU @pl] )+.
IDRULE N2/COORD4B : N2[PLU +] -> N2[CONJ NEITHER, PLU +],
   N2[CONJ NOR, PLU -].
IDRULE N2/COORD4C : N2[PLU +] -> N2[CONJ NEITHER, PLU -],
   N2[CONJ NOR, PLU +].
IDRULE N2/COORD5A : N2[PLU @pl] -> N2[CONJ EITHER, PLU @pl],
   ( N2[CONJ OR, PLU @pl] )+.
IDRULE N2/COORD5B : N2[PLU +] -> N2[CONJ EITHER, PLU +],
   N2[CONJ OR, PLU -].
IDRULE N2/COORD5C : N2[PLU +] -> N2[CONJ EITHER, PLU -],
   N2[CONJ OR, PLU +].
IDRULE N2/COORD6A : N2[PLU @plur] -> N2[CONJ NULL, NEG +],
   N2[CONJ BUT, NEG -, PLU @plur].
IDRULE N2/COORD6B : N2[PLU @plur] -> N2[CONJ NULL, NEG -, PLU @plur],
   N2[CONJ BUT, NEG +].
IDRULE N1/COORD1 : N1[PLU +] -> ( N1[CONJ NULL] )+, ( N1[CONJ AND] )+.
IDRULE N1/COORD2A : N1 -> ( N1[CONJ NULL, PLU -] )+,
   ( N1[CONJ OR, PLU -] )+.
IDRULE N1/COORD2B : N1[PLU +] -> N1[CONJ NULL, PLU +],
   N1[CONJ OR, PLU -].
IDRULE N1/COORD2C : N1[PLU +] -> N1[CONJ NULL, PLU -],
   N1[CONJ OR, PLU +].
IDRULE N1/COORD2D : N1[PLU +] -> ( N1[CONJ NULL, PLU +] )+,
   ( N1[CONJ OR, PLU +] )+.
IDRULE N/COORD1 : N[PLU +, PN -, PRO -] -> ( N[CONJ NULL] )+,
   ( N[CONJ AND] )+.
IDRULE N/COORD2A : N[PLU @pl, PN -, PRO -] ->
   ( N[CONJ NULL, PLU @pl] )+, ( N[CONJ OR, PLU @pl] )+.
IDRULE N/COORD2B : N[PLU +, PN -, PRO -] -> N[CONJ NULL, PLU +],
   N[CONJ OR, PLU -].
IDRULE N/COORD2C : N[PLU +, PN -, PRO -] -> N[CONJ NULL, PLU -],
   N[CONJ OR, PLU +].
IDRULE P2/COORD1 : P2 -> ( P2[CONJ NULL] )+, ( P2[CONJ AND] )+.
IDRULE P2/COORD2 : P2 -> ( P2[CONJ NULL] )+, ( P2[CONJ OR] )+.
IDRULE P2/COORD3 : P2 -> P2[CONJ BOTH], P2[CONJ AND].
IDRULE P2/COORD4 : P2 -> P2[CONJ NEITHER], ( P2[CONJ NOR] )+.
IDRULE P2/COORD5 : P2 -> P2[CONJ EITHER], ( P2[CONJ OR] )+.
IDRULE P2/COORD6A : P2 -> P2[CONJ NULL, NEG -], P2[CONJ BUT, NEG +].
IDRULE P2/COORD6B : P2 -> P2[CONJ NULL, NEG +], P2[CONJ BUT, NEG -].
IDRULE P/COORD1 : P -> ( P[CONJ NULL] )+, ( P[CONJ AND] )+.
IDRULE P/COORD2 : P -> ( P[CONJ NULL] )+, ( P[CONJ OR] )+.
IDRULE P/COORD3 : P -> P[CONJ BOTH], P[CONJ AND].
IDRULE P/COORD4 : P -> P[CONJ NEITHER], ( P[CONJ NOR] )+.
IDRULE P/COORD5 : P -> P[CONJ EITHER], ( P[CONJ OR] )+.
IDRULE A2/COORD1 : A2 -> ( A2[CONJ NULL] )+, ( A2[CONJ AND] )+.
IDRULE A2/COORD2 : A2 -> ( A2[CONJ NULL] )+, ( A2[CONJ OR] )+.
IDRULE A2/COORD3 : A2 -> A2[CONJ BOTH], A2[CONJ AND].
IDRULE A2/COORD4 : A2 -> A2[CONJ NEITHER], ( A2[CONJ NOR] )+.
IDRULE A2/COORD5 : A2 -> A2[CONJ EITHER], ( A2[CONJ OR] )+.
IDRULE A2/COORD6 : A2 -> A2[CONJ NULL], A2[CONJ BUT].
IDRULE A1/COORD1 : A1 -> ( A1[CONJ NULL] )+, ( A1[CONJ AND] )+.
IDRULE A1/COORD2 : A1 -> ( A1[CONJ NULL] )+, ( A1[CONJ OR] )+.
IDRULE A/COORD1 : A -> ( H[CONJ NULL] )+, ( H[CONJ AND] )+.
IDRULE A/COORD2 : A -> ( H[CONJ NULL] )+, ( H[CONJ OR] )+.
IDRULE A/COORD3 : A -> H[CONJ BOTH], H[CONJ AND].
IDRULE A/COORD4 : A -> H[CONJ NEITHER], ( H[CONJ NOR] )+.
IDRULE A/COORD5 : A -> H[CONJ EITHER], ( H[CONJ OR] )+.
IDRULE VP/COORD1 : VP -> ( VP[CONJ NULL] )+, ( VP[CONJ AND] )+.
IDRULE VP/COORD2 : VP -> ( VP[CONJ NULL] )+, ( VP[CONJ OR] )+.
IDRULE VP/COORD3 : VP -> VP[CONJ BOTH], VP[CONJ AND].
IDRULE VP/COORD4 : VP -> VP[CONJ NEITHER], ( VP[CONJ NOR] )+.
IDRULE VP/COORD5 : VP -> VP[CONJ EITHER], ( VP[CONJ OR] )+.
IDRULE VP/COORD6 : VP -> VP[CONJ NULL], VP[CONJ BUT].
IDRULE V/COORD1 : V -> ( H[CONJ NULL] )+, ( H[CONJ AND] )+.
IDRULE V/COORD2 : V -> ( H[CONJ NULL] )+, ( H[CONJ OR] )+.
IDRULE V/COORD3 : V -> H[CONJ BOTH], H[CONJ AND].
IDRULE V/COORD4 : V -> H[CONJ NEITHER], ( H[CONJ NOR] )+.
IDRULE V/COORD5 : V -> H[CONJ EITHER], ( H[CONJ OR] )+.
IDRULE V/COORD6 : V -> H[CONJ NULL], H[CONJ BUT].
IDRULE S/COORD1 : S -> ( S[CONJ NULL] )+, ( S[CONJ AND] )+.
IDRULE S/COORD2 : S -> ( S[CONJ NULL] )+, ( S[CONJ OR] )+.
IDRULE S/COORD3 : S -> S[CONJ NULL], S[CONJ BUT].
IDRULE S/COORD4 : S -> S[CONJ EITHER], ( S[CONJ OR] )+.
IDRULE CONJ/PRD1 : X2[PRD +, CONJ NULL] -> N2[+PRD].
IDRULE CONJ/PRD2 : X2[PRD +, CONJ @con] -> [SUBCAT @con, CONJN +],
   N2[+PRD].
IDRULE CONJ/PRD3 : X2[PRD +, CONJ NULL] -> A2[+PRD].
IDRULE CONJ/PRD4 : X2[PRD +, CONJ @con] -> [SUBCAT @con, CONJN +],
   A2[+PRD].
```

```
IDRULE CONJ/PRD5 : X2[PRD +, CONJ NULL] -> P2[+PRD, PFORM NORM].
IDRULE CONJ/PRD6 : X2[PRD +, CONJ @con] -> [SUBCAT @con, CONJN +],
   P2[+PRD, PFORM NORM].
IDRULE CONJ/PRD7 : X2[PRD +, CONJ NULL] -> VP[+PRD].
IDRULE CONJ/PRD8 : X2[PRD +, CONJ @con] -> [SUBCAT @con, CONJN +],
   VP[+PRD].
IDRULE PRD/COORD1 : [PRD +, BAR 2] -> ( X2[PRD +, CONJ NULL] )+,
   ( X2[PRD +, CONJ AND] )+.
IDRULE PRD/COORD2 : [PRD +, BAR 2] -> ( X2[PRD +, CONJ NULL] )+,
   ( X2[PRD +, CONJ OR] )+.
IDRULE PRD/COORD3 : [PRD +, BAR 2] -> ( X2[PRD +, CONJ EITHER] )+,
   ( X2[PRD +, CONJ OR] )+.
IDRULE PRD/COORD4 : [PRD +, BAR 2] -> ( X2[PRD +, CONJ NEITHER] )+,
   ( X2[PRD +, CONJ NOR] )+.
IDRULE PRD/COORD5A : [PRD +, BAR 2] -> X2[PRD +, CONJ NULL, NEG +],
   X2[PRD +, CONJ BUT, NEG -].
IDRULE PRD/COORD5B : [PRD +, BAR 2] -> X2[PRD +, CONJ NULL, NEG -],
   X2[PRD +, CONJ BUT, NEG +].
IDRULE PRD/COORD6 : [PRD +, BAR 2] -> X2[PRD +, CONJ BOTH],
   X2[PRD +, CONJ AND].
IDRULE CONJ/MOD1 : X2[+ADV, CONJ NULL] -> A2[+ADV].
IDRULE CONJ/MOD2 : X2[+ADV, CONJ @con] -> [SUBCAT @con, CONJN +],
   A2[+ADV].
IDRULE CONJ/MOD3 : X2[+ADV, CONJ NULL] -> P2[PFORM NORM].
IDRULE CONJ/MOD4 : X2[+ADV, CONJ @con] -> [SUBCAT @con, CONJN +],
   P2[PFORM NORM].
IDRULE MOD/COORD1 : [ADV +, BAR 2] -> ( X2[+ADV, CONJ NULL] )+,
   ( X2[+ADV, CONJ AND] )+.
IDRULE MOD/COORD2 : [ADV +, BAR 2] -> ( X2[+ADV, CONJ NULL] )+,
   ( X2[+ADV, CONJ OR] )+.
IDRULE MOD/COORD3 : [ADV +, BAR 2] -> ( X2[+ADV, CONJ EITHER] )+,
   ( X2[+ADV, CONJ OR] )+.
IDRULE MOD/COORD4 : [ADV +, BAR 2] -> ( X2[+ADV, CONJ NEITHER] )+,
   ( X2[+ADV, CONJ NOR] )+.
IDRULE MOD/COORD5 : [ADV +, BAR 2] -> X2[+ADV, CONJ NULL],
   X2[+ADV, CONJ BUT].
IDRULE MOD/COORD6 : [ADV +, BAR 2] -> X2[+ADV, CONJ BOTH],
   X2[+ADV, CONJ AND].
IDRULE A/COORD6A : A -> H[CONJ NULL, NEG -], H[CONJ BUT, NEG +].
IDRULE A/COORD6B : A -> H[CONJ NULL, NEG +], H[CONJ BUT, NEG -].
IDRULE P/COORD6A : P -> P[CONJ NULL, NEG -], P[CONJ BUT, NEG +].
IDRULE P/COORD6B : P -> P[CONJ NULL, NEG +], P[CONJ BUT, NEG -].
```

; PS rules

```
PSRULE PRT/PREP : ; a fix for some phrasal prepositional verbs - it
                  ; proved too difficult to linearise them properly.
   [PRT @1, PREP @2] -> [PRT @1] P[PFORM @2].
PSRULE NUM1 : ; rules for numbers. This one - thirty three
   [CN1 @x, CN2 SMALL] -> [CN1 TY, CN2 SMALL] [CN1 TEN, CN2 SMALL].
PSRULE NUM2 : ; three hundred, thirty thousand
   [CN1 @x, CN2 BIG, AND -] -> [CN1 @y, CN2 SMALL]
   [CN1 @x, CN2 BIG, AND -].
PSRULE NUM3 : ; three hundred and thirty three
   [CN1 @x, CN2 BIG, AND +] -> [CN1 @x, CN2 BIG, AND -]
   [SUBCAT AND, CONJN +] [CN1 @y, CN2 SMALL].
PSRULE NUM4 : ; three thousand three hundred
   [CN1 THOU, CN2 BIG, AND +] -> [CN1 THOU, CN2 BIG, AND -]
   [CN1 @z, CN2 BIG].
PSRULE N/NAME1 : ; rules for complex names. This one does egs like
                 ; sandy jones, s jones, sandy lee, sandy l jones
   N[PN +, PLU -, COUNT +, NFORM NORM, SUBCAT NULL, POSS -] ->
   N[PN +, SUBCAT NULL] ( N[PN +, SUBCAT NULL] )+.
PSRULE N/NAME2A : ; mr jones, mr jones esq
   N[-PLU, PN -, -POSS] -> N[ADDRESS +, SUBCAT NULL]
   N[+PN, SUBCAT NULL] ( N[ADDRESS +, SUBCAT NULL] ).
PSRULE N/NAME2B : ; sandy jones jnr
   N[-PLU, PN -, -POSS] -> N[+PN, SUBCAT NULL]
   N[ADDRESS +, SUBCAT NULL].
```

Appendix 2. Sample Lexical Entries

```
(and) : [SUBCAT AND, CONJN +].

(abandon) :
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST -, AGR
      N2[+NOM, +PLU, NFORM NORM, PER @102642, COUNT @102641], PSVE -,
      PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST -,
      AGR N2[+NOM, -PLU, NFORM NORM, PER 1, COUNT @102643], PSVE -,
      PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST -,
      AGR N2[+NOM, -PLU, NFORM NORM, PER 2, COUNT @102644], PSVE -,
      PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[IMP, -PRD, -INV, -NEG, -AUX, PAST @102645,
      AGR N2[+NOM, NFORM
         NORM, PER @102647, PLU @102648, COUNT @102646], PSVE -,
      PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[BSE, -PRD, -INV, -NEG, -FIN, -AUX, PAST @102649,
      AGR N2[NFORM
         NORM, PER @102652, PLU @102653, COUNT @102651,
            CASE @102650], PSVE -, PFORM TO, SUBCAT NP_PP,
      SUBTYPE NONE],
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST -,
      AGR N2[+NOM, +PLU, NFORM NORM, PER @102655, COUNT @102654],
      PSVE -, SUBCAT NP, SUBTYPE NONE],
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST -,
      AGR N2[+NOM, -PLU, NFORM NORM, PER 1, COUNT @102656], PSVE -,
      SUBCAT NP, SUBTYPE NONE],
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST -,
      AGR N2[+NOM, -PLU, NFORM NORM, PER 2, COUNT @102657], PSVE -,
      SUBCAT NP, SUBTYPE NONE],
   V[IMP, -PRD, -INV, -NEG, -AUX, PAST @102658,
      AGR N2[+NOM, NFORM
         NORM, PER @102660, PLU @102661, COUNT @102659], PSVE -,
      SUBCAT NP, SUBTYPE NONE],
   V[BSE, -PRD, -INV, -NEG, -FIN, -AUX, PAST @102662,
      AGR N2[NFORM
         NORM, PER @102665, PLU @102666, COUNT @102664,
            CASE @102663], PSVE -, SUBCAT NP, SUBTYPE NONE],
   N[-POSS, -PLU, -PRO, -COUNT, -NUM, PRD @102667, NFORM NORM, PER 3,
      CASE @102668, PN -, SUBCAT NULL, REFL @102046].

(abandoned) :
   V[PAS, -INV, -NEG, -FIN, -AUX, PAST @102711, AGR
      N2[NFORM NORM,
         PER @102714, PLU @102715, COUNT @102713, CASE @102712],
      PSVE +, PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[PSP, -INV, -NEG, -FIN, -AUX, PAST @102716,
      AGR N2[NFORM NORM,
         PER @102719, PLU @102720, COUNT @102718, CASE @102717],
      PSVE -, PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST +,
      AGR N2[NFORM
         NORM, PER @102723, PLU @102724, COUNT @102722,
            CASE @102721], PSVE -, PFORM TO, SUBCAT NP_PP,
      SUBTYPE NONE],
   V[PSP, -INV, -NEG, -FIN, -AUX, PAST @102725, AGR
      N2[NFORM NORM,
         PER @102728, PLU @102729, COUNT @102727, CASE @102726],
      PSVE -, SUBCAT NP, SUBTYPE NONE],
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST +,
      AGR N2[NFORM
         NORM, PER @102732, PLU @102733, COUNT @102731,
            CASE @102730], PSVE -, SUBCAT NP, SUBTYPE NONE],
   V[PAS, -INV, -NEG, -FIN, -AUX, PAST @102734,
      AGR N2[NFORM NORM,
         PER @102737, PLU @102738, COUNT @102736, CASE @102735],
      PSVE +, SUBCAT NP, SUBTYPE NONE].

(abandoning) :
   V[GER, -INV, -NEG, -FIN, -AUX, PAST @102776, AGR
      N2[NFORM NORM,
         PER @102779, PLU @102780, COUNT @102778, CASE @102777],
      PSVE -, PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[PRP, -INV, -NEG, -FIN, -AUX, PAST @102781,
      AGR N2[NFORM NORM,
         PER @102784, PLU @102785, COUNT @102783, CASE @102782],
      PSVE -, PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[GER, -INV, -NEG, -FIN, -AUX, PAST @102786,
      AGR N2[NFORM NORM,
         PER @102789, PLU @102790, COUNT @102788, CASE @102787],
      PSVE -, SUBCAT NP, SUBTYPE NONE],
   V[PRP, -INV, -NEG, -FIN, -AUX, PAST @102791,
      AGR N2[NFORM NORM,
         PER @102794, PLU @102795, COUNT @102793, CASE @102792],
      PSVE -, SUBCAT NP, SUBTYPE NONE].

(abandons) :
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST -, AGR
      N2[+NOM, -PLU, NFORM NORM, PER 3, COUNT @102810], PSVE -,
      PFORM TO, SUBCAT NP_PP, SUBTYPE NONE],
   V[-PRD, -INV, -NEG, +FIN, -AUX, VFORM NOT, PAST -,
      AGR N2[+NOM, -PLU, NFORM NORM, PER 3, COUNT @102811], PSVE -,
      SUBCAT NP, SUBTYPE NONE].

(abbot) :
   N[-POSS, -PLU, -PRO, +COUNT, -NUM, PRD @102870, NFORM NORM, PER 3,
      CASE @102871, PN -, SUBCAT NULL, REFL @102046],
   N[-POSS, -PLU, -PRO, +COUNT, -NUM, PRD @102872, NFORM NORM, PER 3,
      CASE @102873, PN -, SUBCAT NULL, REFL @102046, ADDRESS +].

(abbots) :
   N[-POSS, +PLU, -PRO, +COUNT, -NUM, PRD @102888, NFORM NORM, PER 3,
      CASE @102889, PN -, SUBCAT NULL, REFL @102046].

(anxious) :
   A[-ADV, -NEG, -QUA, -NUM, PRD @102933, AGR N2[NFORM NORM,
      PER @102936, PLU @102937, COUNT @102935, CASE @102934],
      DEF @102932, AFORM NONE, SUBCAT SINF, DISTR PRD, SUBTYPE NONE],
   A[-ADV, -NEG, -QUA, -NUM, PRD @102939,
      AGR N2[NFORM NORM, PER
         @102942, PLU @102943, COUNT @102941, CASE @102940],
      DEF @102938, AFORM NONE, SUBCAT SC_INF, DISTR PRD,
      SUBTYPE EQUI],
   A[-ADV, -NEG, -QUA, -NUM, PRD @102945, AGR
      N2[NFORM NORM, PER
         @102948, PLU @102949, COUNT @102947, CASE @102946],
      DEF @102944, PFORM FOR, AFORM NONE, SUBCAT PP, DISTR PRD,
      SUBTYPE NONE],
   A[-ADV, -NEG, -QUA, -NUM, PRD @102951, AGR
      N2[NFORM NORM, PER
         @102954, PLU @102955, COUNT @102953, CASE @102952],
      DEF @102950, PFORM ABOUT, AFORM NONE, SUBCAT PP, DISTR PRD,
      SUBTYPE NONE],
   A[-ADV, -NEG, -QUA, -NUM, PRD @102957, AGR
      N2[NFORM NORM, PER
         @102960, PLU @102961, COUNT @102959, CASE @102958],
      DEF @102956, AFORM NONE, SUBCAT NULL, DISTR @102060,
      SUBTYPE NONE].

(anxiously) :
   A[+ADV, -NEG, -QUA, -NUM, PRD @102984, AGR N2[NFORM
      NORM, PER @102987, PLU @102988, COUNT @102986, CASE @102985],
      DEF @102983, AFORM NONE, SUBCAT NULL, DISTR @102060,
      SUBTYPE NONE].

(down) : [PRT DOWN],
   P[-PRO, PRD @103242, NEG @103241, PFORM NORM, LOC @103243,
      SUBCAT PP],
   P[-PRO, PRD @103247, NEG @103246, PFORM NORM, LOC @103248,
      SUBCAT NP],
   P[-PRO, PRD @103250, NEG @103249, PFORM DOWN, LOC @103251,
      SUBCAT NP].

(either) :
   A[-PRD, -ADV, -NEG, +DEF, +QUA, -NUM, AGR N2[-PLU, +COUNT,
      NFORM @103160, PER @103161, CASE @103159], AFORM NONE,
      SUBCAT NULL, DISTR @102060, SUBTYPE NONE],
   N[-POSS, +DEF, -PLU, +PRO, +COUNT, -NUM, PRD @103162, NFORM NORM,
      PER 3, CASE @103163, PN -, PART OF2, AFORM NONE, SUBCAT NULL,
      REFL -],
   N[-POSS, +DEF, -PLU, +PRO, +COUNT, -NUM, PRD @103164, NFORM NORM,
      PER 3, CASE @103165, PN -, AFORM NONE, SUBCAT NULL, REFL -],
   [SUBCAT EITHER, CONJN +].

(how) :
   P[+PRO, +Q, +WH, -EVER, PRD @103179, NEG @103178, PFORM NORM,
      LOC @103180, SUBCAT NULL], DetA[+Q, +WH, -EVER, AFORM NONE].

(i) :
   N[-POSS, +NOM, +DEF, -PLU, +PRO, +COUNT, -NUM, PRD @103190,
      NFORM NORM, PER 1, PN -, AFORM NONE, SUBCAT NULL, REFL -].

(many) :
   A[+PRD, -ADV, -NEG, -DEF, +QUA, -NUM, AGR +N[+PLU, +COUNT, V
      -, BAR @103341, NFORM @103339, PER @103340, CASE @103338],
      AFORM NONE, SUBCAT NULL, DISTR @102060, SUBTYPE NONE],
   N[-POSS, -DEF, +PLU, +PRO, +COUNT, -NUM, PRD @103342, NFORM NORM,
      PER 3, CASE @103343, PN -, PART OF, AFORM NONE, SUBCAT NULL,
      REFL -],
   N[-POSS, -DEF, +PLU, +PRO, +COUNT, -NUM, PRD @103344, NFORM NORM,
      PER 3, CASE @103345, PN -, AFORM NONE, SUBCAT NULL, REFL -].
```

```
(me) :
   N[-POSS, +ACC, +DEF, -PLU, +PRO, +COUNT, -NUM, PRD @103357,
      NFORM NORM, PER 1, PN -, AFORM NONE, SUBCAT NULL, REFL -].

(must) :
   V[-PRD, -NEG, +FIN, +AUX, +PRO, VFORM NOT, PAST -, AGR @103373,
      INV @103372, PSVE -, SUBCAT NULL, SUBTYPE NONE],
   V[-PRD, -NEG, +FIN, +AUX, VFORM NOT, PAST -, AGR @103375,
      INV @103374, PSVE -, SUBCAT MODAL, SUBTYPE NONE].

(sandy) :
   N[-POSS, -PLU, -PRO, -NUM, +PN, PRD @103801, NFORM NORM, PER 3,
      COUNT @103800, CASE @103802, SUBCAT NULL, REFL @102046].

(sixth) :
   A[-NEG, -QUA, ORD, PRD @103840, AGR @103839, DEF @103837,
      AFORM NONE, ADV @103838, SUBCAT NULL, DISTR @102060,
      SUBTYPE NONE],
   N[-POSS, -PRO, +COUNT, ORD, PRD @103841, NFORM NORM, PER 3,
      PLU @103843, CASE @103842, PN -, PFORM OF, SUBCAT PP,
      REFL @102046],
   N[-POSS, -PRO, +COUNT, ORD, PRD @103844, NFORM NORM, PER 3,
      PLU @103846, CASE @103845, PN -, SUBCAT NULL, REFL @102046].

(sixty) : [CN1 TY, CN2 SMALL, AND @103815].

(so) : DetA[AFORM NONE],
   P[-PRO, PRD @103865, NEG @103864, PFORM NORM, LOC -, SUBCAT SFIN].

(that) :
   N[-POSS, +PRO, +COUNT, -NUM, +R, -WH, -EVER, PRD @103884,
      NFORM NORM, PER 3, PLU @103886, CASE @103885, PN -,
      DEF @103887, AFORM NONE, SUBCAT NULL, REFL -],
   N[-POSS, +DEF, -PLU, +PRO, +COUNT, -NUM, PRD @103888, NFORM NORM,
      PER 3, CASE @103889, PN -, AFORM NONE, SUBCAT NULL, REFL -],
   [SUBCAT THAT],
   DetN[-POSS, +DEF, AGR N2[-PLU, +COUNT, NFORM
      @103891, PER @103892, CASE @103890]], DetA[AFORM NONE].

(where) :
   P[+PRO, +R, +WH, -EVER, PRD @103917, NEG @103916, PFORM NORM,
      LOC @103918, SUBCAT NULL],
   P[+PRO, +Q, +WH, -EVER, PRD @103920, NEG @103919, PFORM NORM,
      LOC @103921, SUBCAT NULL].

(which) :
   N[-POSS, +PRO, +COUNT, -NUM, +R, +WH, -EVER, PRD @103944,
      NFORM NORM, PER 3, PLU @103946, CASE @103945, PN -,
      DEF @103947, AFORM NONE, SUBCAT NULL, REFL -],
   N[-POSS, +DEF, -PLU, +PRO, +COUNT, -NUM, +Q, PRD @103948,
      NFORM NORM, PER 3, CASE @103949, PN -, PART OF2, AFORM NONE,
      SUBCAT NULL, REFL -],
   DetN[-POSS, +DEF, +Q, +WH, -EVER, AGR
      N2[NFORM @103951, PER
         @103953, PLU @103954, COUNT @103952, CASE @103950]].

(who) :
```

```
N[-POSS, +PRO, +COUNT, -NUM, +R, +WH, -EVER, PRD @103973,
   NFORM NORM, PER 3, PLU @103975, CASE @103974, PN -,
   DEF @103976, AFORM NONE, SUBCAT NULL, REFL -],
N[-POSS, +PRO, +COUNT, -NUM, +Q, +WH, -EVER, PRD @103977,
   NFORM NORM, PER 3, PLU @103979, CASE @103978, PN -,
   DEF @103980, AFORM NONE, SUBCAT NULL, REFL -].
```

Appendix 3. Sample Test Sentences

1. S rules

he doesn't help.
he certainly doesn't help.
he confidently accepted their conditions.
don't help him!
do be more resolute!
help me!
he accepted their conditions confidently.
confidently he accepted their conditions.
in an anxious mood he helped the abbot.
he helped the abbot in an anxious mood.
without a doubt he helped the abbot.
he helped the abbot without a doubt.
confidently but not without some anxiety he helped the abbot.
he helped the abbot confidently but not without some anxiety.
he is crazy isn't he?
he isn't crazy is he?
he helps doesn't he?
he can help can he?
* he ought to help isn't he?
* he hasn't abandoned her mustn't he?

2. VP rules

he helps.
he helps out.
it is raining.
she abandons him.
she asks him out.
* she asks out him.
she gives him a message.
she gives him back the message.
she agrees with him.
she gives it back to him.
she answers to him for her actions.
he ended up at the abbey.
she puts it beside him.
it cost much anxiety.
he acquits himself well.
she anticipates that he will help.
it turns out that she knows him.
he promised her that he would help.
that he apologized amuses her.
it amuses her that he apologized.
it dawned on him that he ought to help.
she gets through to him that he should help.
she arranged for him to help.
she arranged with them for him to help.
he petitioned them that they let him appeal.
she figured out who helped.
she asks whether he helped.
they advised him what he should accept.
they asked him if he had accepted.
i would appreciate it if you could help me.

she figured out what to give him.
she reflected on whether to help.
he appears an able host.
he ended up an abbot.
she appears busy.
she ended up crazy.
it is beginning to rain.
it carried on raining.
he could do with being more confident.
she was banking on being able to help.
she might get around to helping him.
he will get caught.
he appears to her to be crazy.
she promised him to help.
it strikes me as conceivable that he would help.
she put him down as crazy.
she knows him to be crazy.
she appeals to him to help her.
it hurts her to abandon him.
to abandon him hurts her.
she was banking on him helping.
she makes lee help her.
she sees him fall over.
she gets him looked at.
she ought to help.
she may have been helping.
she is not the host.
she is in the mood.
she is not in the mood.
she is crazy.
she is not crazy.
there is an abbot in the abbey.
it is the abbot who dictates messages.
it is to the abbot that he gives the message.

3. AP rules

she is busy.
it is full up.
she is anxious about everything.
it is situated by the abbey.
he is accountable to us for his actions.
she is aware that he might not apologize.
that he is able to help is convenient.
it is convenient that he is able to help.
she was eager that he apologize.
it is necessary that he apologize.
he is eager for us to apologize.
he is ready to abandon her.
he is certain to help.
he is busy helping her.
he is confident of being resolute.
it is not clear who she abandoned.
it is not clear whether she abandoned him.
whether she abandoned him is not clear to me.
who she abandoned is not clear to me.
she was so busy.

she was much too busy.
he is more anxious.
they are so many.
they are too many.

4. NP rules

kim abandoned lee.
he abandoned her.
who abandoned her.
all is crazy.
half is crazy.
half are crazy.
both are crazy.
some are crazy.
many are crazy.
three are crazy.
none are crazy.
each is crazy.
neither is crazy.
one is crazy.
this mood is crazy.
the mood is crazy.
a mood is crazy.
the abbot 's mood is crazy.
kim 's discovery is gratifying.
kim 's is crazy.
some abbots are crazy.
all abbots are crazy.
three abbots are crazy.
the third abbot is crazy.
any three abbots are crazy.
nearly all abbots are crazy.
both abbots are crazy.
anxiety is permissible.
many abbots are crazy.
too many abbots are anxious.
the three abbots helped us.
too much anxiety might amaze him.
both of the abbots are crazy.
all of the abbots are crazy.
half of the abbots are crazy.
some of the abbots are crazy.
none of the abbots are crazy.
many of the abbots are crazy.
few of the abbots are crazy.
several of the abbots are crazy.
much of your anxiety is unnecessary.
three of the abbots are crazy.
most of the abbots are crazy.
more of the abbots are crazy.
half the abbots are crazy.
all the abbots are crazy.
both the abbots are crazy.
he is double my age.
he is twice my age.
each of the abbots is crazy.

one of the abbots is crazy.
either of the abbots is crazy.
neither of the abbots is crazy.
the back of the abbey is ablaze.
the message that he apologized didn't account for his mood.
he acknowledges the provision that he apologize.
they give him the message to abandon the abbey.
the busy abbot doesn't abandon his abbey.
* the busy helping abbot doesn't abandon the abbey.
the busy confident resolute abbot abdicates.
* the abbot busy is crazy.
the abbot busy helping lee is crazy.
the abbot eager that you help is crazy.
the abbot inside the abbey is crazy.
the abbot with crazy moods is here.
an acquaintance of lee 's is here.
the abbot to apologize to is absent.
the abbot agreeing with lee is crazy.
the abbot who abandoned lee is crazy.
the abbot who lee abandoned is crazy.
the abbot lee abandoned is crazy.
whichever he abandoned is crazy.
whichever abbot he abandoned is crazy.
whichever abbot appears scared is crazy.
sandy jones is here.
ms jones is crazy.
ms sandy jones is crazy.
kim jones esq is crazy.
mr kim jones esq is crazy.
the resolute would help.
the very resolute would help.
the too resolute would help.
the most resolute would help.

5. PP rules

he put it there.
he helped them down at the abbey.
he helped them because of their age.
this mood of lee 's is not very characteristic.
he couldn't help hearing that admission of the abbot 's.
because he was scared lee didn't arrange to help.
lee wasn't able to help although he had promised that he would.
since i am not appreciated i don't choose to help.

6. Rules involving WH and SLASH

6.1 VP rules

who was he abandoned by?
by whom was he abandoned?
who did she abandon her doubts about?
who did she ask out?
which message did she give to him?
who does she answer to for her actions?
where did she put her abacus?
what did it cost?

how well did his message come across?
what did she anticipate that he would do?
who did she anticipate would help?
who does it turn out that she knows?
who did she promise them that she would help?
who did it dawn on him that he should help?
who does it matter to her that she should help?
by whom was he asked what they should do?
how busy did she look?
so many doubts she is beginning to have!
who did he get abandoned by?
who did she arrange with him to help?
who did she promise to help?
who does he strike as crazy?
how crazy does she make him out?
what does it hurt her to do?
who could she hear apologizing?
who did she get them accepted by?
which abbey is he in?
in which abbey is he?
what is he likely to do?
how crazy is he?
what was there a message about?
who was it that he helped?
who was it who he helped?
how easily can she do it?
in which abbey did he see her?
how easily was he being abandoned?

6.2 AP rules

how confident is he?
how easily confident he is!
who is he eager to help?
who is he resolutely eager to help?
to whom is he accountable?
which abbot was he aware might not help?
who was it convenient that he could help?
to whom was it clear that he wouldn't apologize?
who is it crazy for him to help?
who is he certain to help?
who is he busy helping?

6.3 NP rules

whose abacus is this?
which of the abbots is here?
which abbot did you see?
the abbot my doubts about whom they are aware of is here.
which abbot do i have my doubts about?
which abbot do i agree with lee 's doubts about?
which abbot do I have some doubts about?
my doubts all of which are crazy don't help me.
who did he apologize for having his doubts about?
how many abbots can the abbey afford?
his messages many of which were not clear amazed me.
* who did they hear his admission that he abandoned?
who did he have a desire to help?

7. Comparatives

kim is busier than lee.
kim is more confident than lee.
kim is as busy as lee.
kim is busier than lee is.
kim is more confident than lee is.
kim is as busy as lee is.
kim is more eager than lee is anxious.
kim is as eager as lee is anxious.
kim has more anxieties than lee.
kim has as many anxieties as lee.
kim has more anxieties than lee has.
kim has as many anxieties as lee has.
kim has more anxieties than lee has doubts.
kim has as many anxieties as lee has doubts.
kim helps lee more than sandy helps.
more than sandy kim helps lee.
kim helps lee more than sandy.

8. Coordination

8.1 NP

kim and lee are here.
kim and lee and sandy are here.
kim lee and sandy are here.
both kim and sandy are here.
either kim or lee is here.
the abbot or his host has apologized.
the abbot or his host have apologized.
neither the abbots nor their hosts are here.
neither the abbot nor his host are here.
he helps not the abbot but his host.
the younger abbots and abbeys are inessential.
his crazy doubt or anxiety is not helping him.
his crazy doubts or anxieties are not helping him.
his desire and anxiety to help are convenient.
his desire or anxiety to help is convenient.

8.2 PP

he agrees with the abbot and with the host.
he agrees with the abbot or with the host.
he agrees both with the abbot and with the host.
he agrees neither with the abbot nor with the host.
he agrees with the abbot but not with the host.
* he agrees with the abbot but not on the host.
he puts it either in or outside the abbey.
he puts them both in and outside the abbey.
he puts them outside but not in the abbey.

8.3 AP

she appears both resolute and confident.
he appears neither resolute nor confident.
she appears both eager to help us and anxious to start.

he is too crazy and fearful.
she is so resolute and confident.
she is adamant and confident that she can help.
it is clear and certain that she can help.
that she can help is clear and certain.

8.4 VP

lee may help us or he may not help us.
lee will either help or not help.
lee will neither help nor allow kim to help.
that he won't help both amazes and amuses sandy.
kim will neither help nor abandon lee.
that he is crazy both amazes and amuses lee.
it amazes or amuses lee that he is crazy.

8.5 S

lee won't help but he will allow kim to help.
lee won't help and kim won't apologize.
either kim helps or lee helps.
lee wants to help and he will.
lee hasn't helped but he wants to.

8.6 WH and SLASH Coordinations

who will help and who won't help?
the abbot lee knows but kim doesn't know.
which abbot does lee know but kim doesn't know?
what does lee do confidently and kim do fearfully?
which abbot and which abbey did you see?
what do you have a desire to do but anxieties about?
how anxious and how scared can he be?
what is he scared to do but resolute that he will do?
in which abbey and with which abbot did he see them?
who will he do it with or without?
what does lee see and hear?
what will lee agree to but not do?

8.7 Coordination of Predicative Categories

he is not the abbot but his host.
he is the host but not the abbot.
he is in the abbey but not outside it.
he is not crazy but anxious.
he is crazy but not anxious.
he is crazy and anxious.
he is crazy or anxious.
he is crazy and in the abbey.
he is in the abbey but not crazy.