Technical Report

UCAM-CL-TR-662
ISSN 1476-2986

Number 662

UNIVERSITY OF
CAMBRIDGE

Computer Laboratory

An introduction to tag sequence grammars and the RASP system parser

Ted Briscoe

March 2006

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

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

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

http://www.cl.cam.ac.uk/TechReports/

ISSN 1476-2986

An introduction to tag sequence grammars and the RASP system parser

Ted Briscoe
Computer Laboratory
University of Cambridge
ejb@cl.cam.ac.uk

Abstract

This report describes the tag sequence grammars released as part of the Robust Accurate Statistical Parsing (RASP) system. It is intended to help users of RASP understand the linguistic and engineering rationale behind the grammars and prepare them to customize the system for their application. It also contains a fairly exhaustive list of references to extant work utilizing the RASP parser.

1 Introduction

The first tag sequence grammar (TSG) was written in 1991 in an afternoon as part of an experiment in grammar induction (Briscoe & Waegner, 1992). Since then, it has undergone many extensions and modifications. The basic idea behind the TSGs was that it is hard to develop a wide-coverage and robust parsing system which relies on rich lexical valency or subcategorization information, because it is hard to acquire such wide-coverage lexicons. Accordingly in 1995 we used a TSG to parse text in order to try to automatically acquire such lexical entries for the Alvey Natural Language Toolkit full grammar (Briscoe and Carroll, 1997). In the following decade, especially since the first public release of the robust accurate statistical parsing (RASP system in 2002 (Briscoe & Carroll, 2002)[1], the use of TSG parsed text has broadened to include experiments on topic classification (Bennett & Carbonell, 2005; Wang, 2005), sentiment classification (Khan, 2006), question answering (Leidner *et al*, 2003; Watson *et al*, 2003), text anonymization (Medlock, 2006), summarization (de Souza, 2005), information extraction (Callmeier *et al*, 2004; Grover *et al*, 2005; Weeds *et al*, 2004a; Vlachos *et al*, 2006), anaphora resolution (Preiss, 2003; Preiss & Briscoe, 2003; Preiss *et al*, 2004), word sense disambiguation (Clark & Weir, 2002; McCarthy *et al*, 2004; Preiss, 2004; Pustejovsky *et al*, 2004), cue phrase detection (Abdalla, 2005), identifying rhetorical relations (Sporleder and Lascarides, 2005), computing entailments (Andreevskaia *et al*, 2005), resolving metonymy (Nissim & Markert, 2003), extracting robust miminal recursion semantics (Copestake, 2003), detecting ellipsis

[1] This paper, describing the RASP system and announcing its public release, has been cited 74 times (Google Scholar 18/10/05).

(Nielsen, 2004), detecting telic roles (Yamada & Baldwin, 2004), detecting verb alternations (Tsang & Stevenson, 2004), detecting the countability of English nouns (Baldwin & Bond, 2003), detecting compositionality in phrasal verbs (Bannard, 2005; McCarthy *et al*, 2003), crosslingual PP attachment (Agirre *et al*, 2004), Japanese-English machine translation (Tanaka & Baldwin, 2003), measuring lexical and phrasal distributional similarity (Curran, 2003; Weeds *et al*, 2004b, 2005), measuring syntactic development in child language (Sagae *et al*, 2005; Buttery & Korhonen, 2005), as well as further work on valency acquisition (Baldwin, 2005; Korhonen, 2002, Korhonen *et al*, 2003; Yallop *et al*, 2005).

In this report, I describe and motivate the analyses adopted in these grammars and provide details of the various output representations available. The aim is to provide enough information that this report can act as a self-contained introduction to the current grammar for users of the RASP system (Briscoe and Carroll, 2002)[2]. For more detailed information, it provides references for further reading. The Grammar Development Environment (GDE) provides facilities for viewing all the grammar constructs as well as interactively parsing examples to view the derivations produced (see Carroll *et al* (1991); $RASP/manuals/gde). I also discuss performance and customization issues for new applications and datasets.

2 The Grammar Formalism

The grammar formalism is a subset of that used in the Alvey Natural Language Toolkit (ANLT) formalism (Briscoe *et al*, 1987) and TSGs have been developed in the ANLT Grammar Development Environment (Carroll *et al*, 1991).

The ANLT formalism defines a metagrammar, based on the notation of Generalized Phrase Structure Grammar (GPSG, Gazdar *et al*, 1985) which compiles into an object grammar which can be thought of as a syntactic variant of a Definite Clause Grammar, that is, a phrase structure grammar in which categories consist of sets of attributes with values which are unified by Prolog-style term unification. The object grammar is compiled from the metagrammar by 'expanding out' PS rules with additional features (see Carroll *et al*, 1991 for details). In the TSGs, I make use of feature declarations, feature defaults and propagation rules, category delcarations and phrase structure rules (i.e. no use of GPSG Metarules or ID/LP rules). Furthermore, all features take a finite-set of atomic, ground values, so the feature system is an abbreviatory device over the class of context-free (CF) grammars (though we do not compile out to CFGs). Below, I introduce and motivate the formalism informally – see Briscoe *et al* (1987) or Carroll *et al* (1991) for more precise and formal definitions.

2.1 Feature Declarations

Feature declarations take the following form:

```
FEATURE N{+, -}
```

[2] www.informatics.susx.ac.uk/research/nlp/rasp/

```
FEATURE PFORM{FOR, OF, WITH, WH, THAN, AS, PREP, PREPS}
FEATURE AUX{TO, BE, HAVE, DO, MODAL, CAT, -}
```

They define the set of allowable attributes together with their legal values. There are 41 such declarations in the current grammar and the attribute with the most values is VSUBCAT with 31. The attribute names (and values) are for the most part based closely on those used in GPSG and also the ANLT full grammar (Grover *et al*, 1993) so these works can be consulted for more detailed descriptions of the use of and motivation for these features.

2.2 Category Declarations

Category declarations take the following form:

```
CATEGORY CV : [N -, V +] => {PLU, BAR, VFORM, AUX, FIN, CONJ, QUOTE}.
CATEGORY CDET : [MINOR DET] => {PLU, POSS, WH}.
CATEGORY CVw2 : [~INV, N -, V +, BAR 0] => {VSUBCAT, PRT}.
```

They define the allowable attributes of a category or the legal extension of a set of attribute-value pairs. The antecedent can be a category pattern which specifies absence of an attribute (such as ĨNV above) or a disjunction of values of an attribute. Category declaration antecedent pattterns are matched with underspecified categories in PS rules during construction of the object grammar. Together with feature declarations, category declarations constitute the typing system which defines the possible forms of grammatical categories in the grammar.

2.3 Phrase Structure Rules

Phrase structure (PS) rules consist of a rule name, rule and optional semantic specification:

```
PSRULE T/txt-sc1 : Tp --> V2[FIN +, CONJ @c, IMP @i] (Ep).
PSRULE S/pp-np_s : V2[WH @w, INV @i] --> P2[WH -,
   PSUBCAT NP] ( +pco ) H2[FIN +, WH @w, INV @i]
   : 3 : (ncmod _ 3 1).
```

The first rule above, called 'T/txt-sc1', is one of two start rules in the grammar whose mother is the root category aliased 'Tp', and defined simply as (TOP +). It states that a matrix clause can consist of a finite sentence, optionally followed by end-of-sentence punctuation (Ep, e.g. *?*). The CONJ and IMP attributes appear with variable values to ensure that imperatives and sentences which begin with a conjunct can occur as matrix clauses. In effect, they prevent the application of default feature value declarations in this context (see below).

The second rule, 'S/pp-np_s', states that a matrix clause can consist of a preposed prepositional phrase (PP) with noun phrase (NP) complement, optionally followed by a comma, and then by a finite clause (the head daughter in this rule). The values of the (non-head) features WH and INV are bound between mother and head daughter in the rule as their propagation is not covered by more general feature propagation rules (see below). The category of the head daughter is abbreviated H2 (which expands to H(ead) +, BAR 2) to ensure that the HFC_V propagation rule applies. The attribute H and its values are a metagrammatical convenience and are removed from the object grammar after the metagrammar compilation process (Carroll *et al*, 1991). This rule also contains a semantic specification which states that the third daughter (3) is the semantic value of the rule and that the rule licenses inference of a grammatical relation of non-clausal modification (ncmod) between the head daughter and the first daughter (see § 7 for more details). Semantic specifications as well as PS rule patterns use the convention that the mother category is 0 and the daughter categories can be referenced numerically from left to right.

There are 678 PS rules in the current grammar of which 157 are marked as rare or peripheral as opposed to core rules of English grammar.

2.4 PS Rule Names

PS rule names in TSGs are intended to be mnemonic to aid grammar debugging. The general convention is that they consist of an upper case mother category followed by a slash and a sequence of lower case daughter categories delimited by underscores. I mostly use more traditional labels for categories in rule names and X-bar ones as abbreviations in the grammar proper. The daughter categories can also be given mnemonic 'featural' information separated by hyphens. So in 'S/pp-np_s' above, the rule has an S mother and PP and S daughters, while the PP is restricted to have a NP complement. The rule names are not intended to be an output representation, although the system can output trees with nodes decorated with rule names for debugging. Many changes to the grammar will result in rule name additions or modifications, therefore applications based on this output representation will tend to be unstable.

A final feature of rule names is that some end in '-r' to designate their peripheral or rare status. These rules are included to increase coverage of genre specific or otherwise marked constructions. The '-r' designation can be exploited when creating the probabilistic parser from the grammar and training data.

2.5 Feature Propagation Rules

Feature propagation rules propagate designated attribute values between mother and daughter categories in PS rules.

```
PROPRULE AGR1 : V2 --> [N +, V -], [H +, BAR 1]. PLU(1) = PLU(2).
PROPRULE WH1 : N1 --> [H +], U. WH(0) = WH(1).
```

The first rule above states that in any PS rule with V2 (S) mother, with a nominal first daughter and with a V1 (VP) head daughter, the attribute PLU must agree in value. The effect of this rule is to bind variable values on PLU between the two daughters or to propagate a specific value if one is specified in the PS rule. The second rule propagates the value of the attribute WH between head daughters and N1 mothers in all N1 rules with a head daughter. PS rule patterns in propagation rules are matched with PS rules during object grammar compilation.

2.6 Feature Default Rules

Default rules assign default values to attributes in PS rules also in terms of a pattern which matches a class of such rules.

```
DEFRULE CONJ1 : [] --> [N, V, ~CONJ], U. CONJ(1) = -.
DEFRULE AUX : V1 --> [H0], U. AUX(1) = -.
```

For instance the first rule above states that any daughter specified for the attributes N and V but unspecified for CONJ in any rule should be given a default value of '–'. It is to block application of this default rule that PS rule 'T/txt-sc1' above specifies CONJ with a variable value. The second default rule states that in any VP (V1) rule with a head daughter, that daughter is by default not an auxiliary verb. Default rules are applied after feature propagation rules during object grammar compilation – see Carroll *et al* (1991) for further details.

2.7 Features Sets and Aliases

Attributes can be organized into sets for the purposes of feature propagation in PS rules, and attribute-value pairs can be abbreviated by defining aliases to create a mnemonic CF 'backbone' in PS rules. For instance, the set declaration below, allows a succinct statement of verbal head feature propagation underneath,

```
SET VHEAD = {N, V, PLU, VFORM, AUX, FIN}
PROPRULE HFC_V : [V +, N -] --> [H +], U. F(0) = F(1), F in VHEAD.
```

enforcing binding of variables or identity of values on this set of attributes between mother and head daughter in all PS rules with S or VP mothers. The alias declaration below defines V2 (S) which can then be used as a mnemonic abbeviation in the rule underneath.

```
ALIAS V2 = [V +, N -, BAR 2, TXTCAT UNIT, TXT CL].
PSRULE S/np_vp : V2[WH @w, INV -] --> N2[WH @w] H1[FIN +].
```

Finally, TSGs contain WORD declarations which function as the lexicon by mapping preterminal to terminal categories of the grammar. In TSGs these preterminals are the PoS and punctuation tagset.

3 The PoS and Punctuation Tagset

TSGs exploit the comparative success of statistical part-of-speech (PoS) tagging (Church, 1988). The tagset utilized by the TSGs is based on the CLAWS tagset, as used in the Susanne Corpus (Sampson, 1995). The full definition of the tagset is available in these references, and Jurafsky and Martin (2000:Appendix C), the UCREL website at the University of Lancaster, and the RASP distribution files ($RASP/manuals/).

The full CLAWS tagset contains over 170 PoS and punctuation tags. However, the current grammar only utilizes 150 of these, of which around 50 are associated with an identical or subsuming lexical category in the current grammar. Since the development of the Penn Treebank (Marcus *et al*, 1993), it has been standard to use a tagset of around 50 labels following Church (1988), who argued that this tagset size was optimal with respect to local contextual resolution of ambiguity. In fact, there is no conclusive evidence that labels from the Penn Treebank tagset can be assigned any more accurately than those from CLAWS-style tagsets, and the CLAWS tagset contains useful additional information which can be exploited by a TSG[3] The tagger distributed with RASP can either be used in forced choice mode to select for each word the tag with highest posterior probability in the HMM or to return all tags with a posterior probabilility.

The tagger system contains a lexicon which has been developed to integrate with TSGs. The original training material for the tagger (LOB, SEC, Susanne) contained inconsistencies which typically resulted in additional but infrequent associations of tags with word-forms and omission of some additional tags which represent semi-productive lexical alternations. For instance, many closed class items such as *more* or *most* were assigned a wide variety of closed and sometimes open class tags (often as a consequence of confusion or error on the part of the original annotator or variation of practice across corpora). A systematic effort has been made to keep tag ambiguity in general, and for closed class items in particular, to a minimum, and to relocate some of the (real) ambiguity in the TSGs. Often low or even mid frequency open-class word forms occur in corpora only in their most frequent realization (e.g. *zip* as a noun, but not verb) or morphologically-related words are not present with a predictable tag (e.g. *blurred* as verb implies *blur* can be a verb). The majority of these omissions have been semi-automatically corrected. The tagger lexicon currently contains just over 50,000 word forms. The tagger utilizes a probabilistic unknown word model which has generally proved more accurate than attempting to include rarer words in the lexicon itself.

The word declarations in the grammar associate tags with lexical categories. Most closed-class words (except prepositions) have a single usually fully-specified such category:

```
WORD AT : DT[PLU @x, POSS -, WH -].    ; the, no
WORD DB : NO[NTYPE PART, PLU -, POSS -, WH -, CONJ -]. ; all, half
```

However, some may have several and may involve underspecification:

[3] Elworthy (1993, 1994) demonstrates essentially identical tagging accuracies using CLAWS tags with a 1st-order HMM tagger, and also argues that tagset size reduction can lead to worse performance since it reduces the informativeness of the surrounding context.

```
WORD DD : DT[PLU @x, WH -, POSS -],
   N2[QUOTE -, SCOLON -, NTYPE PRO, PLU @x, POSS -, WH -, CONJ -].
   ; any,enough,some,lot,rest
WORD CSA : PO[PSUBCAT @y, PFORM AS, CONJ -]. ; as
```

In the first case, as PLU is a binary-valued attribute, effectively 4 lexical categories are assigned to DD. CSA is unique to *as* but underspecifies 12 lexical categories, as PSUBCAT can take that number of values. The most ambiguous tags with respect to the current grammar are those for main verbs as they are all underspecified for VSUBCAT, which can take one of 31 values specifying allowable complementation via GPSG-style indexing with 218 PS rules expanding VP (V1) in the current grammar.

4 The Text Grammar

In addition to a grammar of the syntactic relations which hold inside a sentence, a practical system able to parse free text requires a text grammar which defines relations which hold between elements of a text sentence, defined as whatever occurs between an initial capitalized word and a full stop (i.e. the text unit returned by the RASP tokenizer for tagging and parsing). For instance, in *Kim fell – Sandy had pushed him – but luckily he was unhurt*, there are two sentences with syntactic sentences: *Kim fell but luckily was unhurt* and *Sandy had pushed him.* These two sentences stand in a discourse relation of elaboration which is signalled to some extent by the use of dashes.

The current grammar's treatment of punctuation and text grammar draws heavily on Nunberg's (1990) approach, with some modifications to ensure computational tractability and compatibility with the ANLT formalism. There are 53 PS rules with names beginning T, Taph (text adjunct phrase), Tacl (text adjunct clause), etc., all occurring in the first set of PS rules in the grammar file, which define how text units combine mediated by punctuation. For instance, the first PS rule below states that a text sentence can consist of a sentence followed by a non-clausal text adjunct with sentence-final punctuation.

```
PSRULE T/s_leta-cl : V2[FIN +, +ta] --> H2[-ta] Ta[+cl, +end]
  : 1 : (ta colon 1 2).
PSRULE Taph/dash- : Ta[+da, +end, TXT PH] --> +pda Tph[-da, -do, -sc]
```

While the second rule states that a sentence-final text adjunct can consist of a dash followed by a text phrase (in turn defined as any phrasal projection of the main grammar not itself containing dashes, semicolons or full stops). These two rules are used to parse a text sentence like *Kim fell – Sandy had pushed him.*

To a large extent the text grammar and main grammar are modular. However, there are some places where commas are introduced directly by PS rules of the main grammar in a construction-specific manner (see for example § 5.8 below). The text grammar is motivated and an earlier version described in considerably more detail in Briscoe (1994), while Briscoe and Carroll (1995) demonstrate that use of the text grammar with the main grammar reduces average ambiguity and extends coverage of the overall system. The treatment of quotation is dealt with in section 5.1.

5 The Main Grammar

The main grammar adopts the X-bar framework of Jackendoff as reinterpreted in GPSG. Phrases and clauses are treated as projections of (lexical) head words and a two-level (or bar) scheme is assumed so that the maximal projection of a lexical head has two intervening constituent types (X0, X1, X2). The four major lexical categories (N, V, A, P) all conform to this scheme (e.g. N0, N1, N2) and V1 (VP) is analyzed as the head of V2 (S). Other minor categories are mostly treated as specifiers or left-attachments to X2. Complements are right sisters of X0, and modifiers are right or left recursive sisters of X1. A more thorough introduction to the X-bar schema used in the TSGs can be found in Grover *et al* (1993). Note that the distinction between major and minor categories is distinct from that between open and closed class lexical items, as prepositions are a closed class but major lexical category.

The majority of PS rules in the current grammar respect this X-bar schema, using headedness to define properties of the mother via feature propagation and distinctions between level and directionality of attachment to distinguish arguments from adjuncts and specifiers, as originally proposed by Jackendoff. However, there are marked rules in the current grammar that are unheaded or otherwise override the schema to deal with peripheral or marked constructions. For instance, the following PS rule allows a degree premodifier of scalar adjectives, such as *very* or *so* to modify a progressive verb and form an AP, as in *a very damaging allegation.*

```
PSRULE AP/dg_ing-r : A2[MOD -] --> A0[ATYPE DG] V0[VFORM ING, FIN -]
   : 2 : (ncmod _ 2 1).
```

Many of these rules exist to compensate for omissions in the tagger lexicon and are therefore marked with '-r' so that an alternative derivation treating *damaging* as an adjective will be preferred where possible. However, in other cases such as coordination (see § 5.8) the inherent inadequacies of the X-bar framework when combined with strict unification of categories are circumvented by using unheaded rules.

5.1 Quotation

Quotation is treated somewhat differently from other punctuation because it is inherently not a (text) sentence-level phenomenon. The tagger normalizes different forms of quote mark to a single tag. The parser preprocessor removes unbalanced quote marks usually where the other mark crosses a sentence boundary as in *"They're here! Let's surprise them," Kim said.*. This is straightforward if there is a single quote mark in each sentence, but more difficult if there are three (or more). Therefore, intelligent preprocessing of the text at the document or paragraph level (before sentence boundary detection and tokenization) may be worthwhile if extra-sentential quotation information is to be retained and processed as accurately as possible.

Balanced quotes are handled by 21 PS rules at the end of the text grammar section of the grammar file. Some of these rules simply allow (major) categories and their projections to appear with quote marks around them and also allow nesting of quotation, as in: *"Kim*

'Slasher' Smith was arrested.". Conventions of quote mark interchanging are tokenized out by the tagger and not exploited by the parser, nor is the difference between begin/end quote marks for the same reason. Most quoted constituents optionally allow commas or end of sentence punctuation to occur inside the quote marks to handle quote transposition, as in: *Is he a "reasonable man?"*. Several of the rules combine quotation specific constructions with a requirement for balanced quote marks, as in: *"The situation," Kim opined, "is difficult."*. However in many cases these rules have non-quoted variants in the main grammar to handle cases where the quotation is extra-sentential and the marks will have been removed.

5.2 Clauses

There are 89 PS rules expanding V2 (S) in various ways and named 'S/...', covering (non-)finite clauses, small clauses, relative clauses, pre- and post-posing and modifying constructions, subject-auxiliary inversion, vocatives, tag questions, clefts, imperatives, locative inversion, sentential subjects, direct and indirect quotation, and so forth. Many of these rules are preceded in the grammar file by short examples illustrating their intended application.

Preposing and postposing PS rules, 'S/xp_s' or 'S/s_xp' are separated by type of PP etc to control ambiguity and some are marked '-r' to avoid spurious ambiguity where there is an alternative more likely derivation which cannot be distinguished (using only information in tag sequences). Postposing rules require an obligatory comma, 'S/s_pco_s'. The third group of rules handles vocatives, reflexives, tag questions, and similar phenomena, once again often requiring delimiting commas in order to avoid overlap with other derivations.

The fourth set of rules handle NP+VP combinations and agreement in finite and small clauses followed by some rules which handle clauses with different types of clausal or VP subjects, locative inversion and cleft constructions that cannot be handled by other more general rules. There is one rule of subject-auxiliary inversion 'S/sai' for main verbs, which treats the auxiliary as first daughter and (non-finite) clause as third daughter (with optional negative operator between) and special case rules for *be* and *ought*.

There is one rule which introduces sentential complements with complementizers, one for imperatives, and seven rules which deal with unbounded dependency questions (i.e. non-subject wh-questions). The final set of rules deal with clausal coordination (see § 5.8 below).

5.3 VP

Verb and adjacent particle combinations are constructed with a 'morphological' rule which combines a V0 and particle to form a V0 with PRT + (to avoid recursion). Otherwise, verbal complement frames are defined by 218 'V1/v...' rules which partition NP complements by noun type (NTYPE) defined in terms of CLAWS tagset nominal subtypes (see tags NX(X) in the WORD declaration of the grammar file). Breaking down the rules this way allows greater precision in the application of some rules, especially in contexts like ditransitive 'double NP' constructions where it is often difficult to distinguish NP boundaries (e.g. *(elect) Tony Blair the prime minister* vs. *(give) him it*). These rules also treat heavy-shifted variants, particle movement, and diathesis alternations as separate complementation patterns.

The second group of rules, 'V1/vp_xp' or 'V1/xp_vp' handles pre- and post-modification of V1 and the third set coordination of VP.

The next set of VP rules deals with auxiliaries. There are 35 rules, V1/be...' dealing with copular and auxiliary *be* along with 'VP-internal' modification of predicative NP objects of *be*. There are 11 rules, 'V1/do..' which handle auxiliary and main verb *do*, 22 for *have*, 'V1/have...', 2 for infinitive *to*, and 2 for modals. There are also a number of special rules for non-constituent coordination with *be* (e.g. *is a conservative and proud of it*) and '-r' rules allowing gaps and ellipsis with each type of auxiliary (e.g. *what do you do e?*). The latter are needed as auxiliaries are effectively subcategorized in the grammar, unlike main verbs.

5.4 Noun Phrases

The first set of rules deals with numbers and some tokens containing numbers, such as *$10M*, which receive a variety of PoS tags which are associated with the feature NTYPE NUM in the grammar. Thse rules allow complex numbers to be built out of tokenized components (e.g. *10 . 4*) (though the RASP tokenizer doesn't currently segment these), ordinals, ranges, dates, times, and so forth. Some variants of such phrases involve combination with a NTYPE TEMP token as head (e.g. *4th July, 1989*) or are coerced to this type if recognizable as a date, and so forth. Numbers can also be coerced to pronouns (e.g. *the twenty who came*).

The next set of rules license NP (N2) phrases without specifiers (determiners, etc) for a subset of NTYPEs, such as plural common nouns, names, etc, and NPs with initial determiners, partitives, and so forth (e.g. *all/half (of) the group*). Some of the latter rules are '-r' because many partitives have very flexible syntactic realization and thus a number of PoS tags, so these rules should only yield highly-ranked derivations if the tagger is used in forced choice mode. There follow three rules handling possessives including one '-r' rule which allows ellipsis (e.g. *Kim went to the butcher's*).

Pronouns have separate tags,PPX(X), and are NTYPE PRO/REFL. The next set of 15 rules deal with post- or pre-modification of pronouns (e.g. *(the) someone in the barn*) – these are mostly '-r' to force a preference for higher attachment of the postmodifiers where possible (e.g. *I met someone in the barn*). There follow 10 rules of more general NP postmodification, many of these are also quite marked as most postmodification and/or complementation is specified at the N1 level.

To cut down on ambiguity the rules dealing with nominal phrase combinations are broken down by NTYPE, for example licensing combinations of names and NPs (e.g. *girls' band Banamarama*), comma-less appositives *the singer James Browne* etc. There are 18 such often very specific rules and this is an area where further additions may be required to deal with specific sublanguages.

The next group of rules are unheaded as they mostly deal with morphological conversion (e.g. *the pick up*) or (elliptical) cases with adjectival heads (e.g. *the poor in the ghetto*). However, some rules capture additional marked specifiers or premodifiers (e.g. *about 10 pounds*).

Nominal phrases (N1) constructed from single nouns are subdivided by NTYPE to control ambiguity. (Notably pronouns have no N1 phrasal level and therefore are not accessible to the unmarked general rules of complementation and pre- or post-modification.) The second set of rules deals with AP premodification, possessives without specifiers, and unheaded cases that also need a N1 level analysis (e.g. *the tedious pick up*), these are all '-r' to avoid competition with the NP level analogues and to disprefer the additional misanalyses that can result from these (more general) rules.

There are 6 rules which deal with N0+N1 combinations which can't be subsumed by the NP-level analogue rules discussed above. These are again subdivided by NTYPE to avoid spurious ambiguities as much as possible. There follow 22 rules which deal with N0+N0 compounding, in the broadest sense, allowing many combinations of different NTYPES but excluding many others. Once again these rules may need supplementing for new sublanguages.

The next set of 6 rules deal with noun complementation or alternatively postmodification where a left-recursive N1 level rule would overgenerate. These rules handle *of* PP complements and sentential complements on various NTYPEs. The next 20 rules handle postmodification or occasional complementation of with verb phrases, and relative clauses. Finally, there are 13 rules to handle nominal coordination at N2, N1 and N0 level.

5.5 PP

There are 31 PS rules dealing with prepositional complementation for various types of preposition. Prepositions are distinguished in so far as the CLAWS tagset allows by the value of PFORM. For example, *with* is IW and (PFORM with), *as* is CSA and (PFORM as) etc. *than* and *of* are also (ADJ +) to prevent their attachment as adverbial modifiers. The bulk of prepositions though (128 in the current tagger lexicon), are tagged as II in the CLAWS scheme and are treated as (PFORM prep) in the grammar, so many PS rules underspecify PFORM value. The 53 subordinating conjunctions tagged CS and the 17 temporal conjunctions tagged ICS, such as *because*, *although*, *before* etc, are treated as prepositions with PFORM value 'preps'.

Prepositional complements are distinguished by the attribute PSUBCAT which takes 12 values. However, many P1 rules are further broken down to distinguish, for example, NTYPEs within (PSUBCAT NP) complements as reflected in the mnemonic rule names, e.g. 'P1/p_np_pl', P1/p_np-org' etc. These provide increased precision in parsing, as not all NP NTYPEs can occur as prepositional complements and they also allow induction of prepositional complementation from more reliable rules (e.g. 'P1/p_np-pro' *with him/her/it*), and higher precision marked rules, such as 'P1/p_np-poss-ellip-r' (*at the butcher's*). Some PSUBCAT values, such as 'sing' are often licensed as much by the verb as the preposition (*wonder/*wander about Kim leaving*). In a deeper grammar, these would be treated as part of the verbal complementation. TSGs factor the information between VSUBCAT and PSUBCAT in order to reduce overall ambiguity and to allow flexible combination of verbal and prepositional frames. However, the observed interactions of VSUBCAT and PSUBCAT, PFORM and specific verbs and prepositions can and should be exploited in parse selection and when inducing valency lexicons.

There are separate rules for *as* and *than* due to the large range of elliptical complementation in comparative and equative constructions and some marked rules for moved 'particles', often tagged as prepositions, which are also used in preposition stranding constructions. These treat such prepositions / particles as intransitive – therefore recovery of any complementation can only be done heuristically in the grammatical relations (GR) output or postprocessing phase (see section 7).

PP (P2) rules allow for premodification (*partly because ...*, *slightly after noon*) and MOD prevents these being treated as verbal arguments. Finally there are 7 rules of prepositional coordination.

5.6 AP

The AP rules generalize over adverbs and adjectives except where ADV +/- is specified in rules. The first 14 A1 rules handle premodification including some rare (idiomatic) cases (*very slow(ly)*, *trouble free*), premodified marked unheaded APs (*completely destroyed (city)*), and numerical (comparative) premodifiers (*one-third more difficult*).

There are 15 rules of adjectival complementation mostly dealing with PPs distinguished by PFORM including wh-PPs, but also clausal and predicative complements. These are followed by 2 rules for PP modifiers which are (MOD +) to prevent overlap with argument PPs. There are 7 rules for specific constructions involving adverbial premodification of adverbs (*completely accurately*), postmodification by A0, etc. Finally there are 16 rules handling A coordination including some for unlike categories in predicative complementation.

5.7 Modification

The attribute MOD is set to + when e.g. a N1 is postmodified and is used to constrain order of attachment of pre- and post-modifiers and thus reduce spurious ambiguity. It is also useful in some contexts (e.g. *is this afternoon not available*) to block attachment of 'long' postmodified NPs as interleaved adverbials, and so forth. The same feature is used in PP, AP and VP phrases in a similar manner.

5.8 Coordination

All the major categories have distinct coordination rules which follow an underlying binary- and right-branching (X → X X) schema where the first conjunct is (CONJ −), the second (CONJ +) and the mother unspecified for this attribute. Elsewhere in the grammar major categories default to (CONJ −). 'XP/cj-beg_xp' rules introduce coordinate markers at the beginning of coordinate constructions (*(both (... and ...))*). 'XP/cj-end' introduces conjunctions as sisters of (CONJ −) conjuncts (*... (and (...))*). There are 63 such rules in the current grammar as the rules must be specified for each major category separately and because additional rules relaxing conjunct agreement are specified in a category-specific way.

The main schema rules, ‘XP/xp_xp-coord’ are not headed and feature propagation is explicitly specified on a rule-by-rule basis to alleviate some of the problems of unification-based approaches to this construction. In particular, for many categories more relaxed ‘-r’ versions of the rules are given that do not force the same degree of ‘agreement’ between conjuncts (e.g. *he and I (were)*, *have you an alibi or did you murder Smith?*, etc).

5.9 Unbounded Dependencies

All wh-NPs involving subject dependencies in relative clauses, wh-questions, and so forth are treated as normal NPs. However, preposed non-subject wh-NPs, wh-PPs and some (topicalized) normal NPs are handled by 5 PS rules which combine a preposed wh-constituent with a clause (which may have undergone subject-auxiliary inversion). The location of the gap in the clause is not represented as the grammar has no access to (verbal) valency. However, the GR output does predict the role of the wh-constituent heuristically and by underspecification – e.g. there is usually only one verb but we cannot know whether it has one or more NP objects

The auxiliaries are treated somewhat differently as they are effectively subcategorized in the tagset, so the grammar contains explicit gap rules ‘V1/be_gap-r’, etc to allow for missing preposed arguments. However, there is still no analogue to the GPSG / HPSG gap features, which represent the dependency between preposed filler and gap.

6 Development Corpus

The development corpus in $RASP/prob/corpus/ indicates the current coverage of the grammar and contains tagged files that can be parsed using the GDE fparse utility (*.tag) or input to the RASP system (*.txt). The grs/ directory contains the correct GR output for the correct derivations (*.trees1). Studying the files of derivations and GR output is one good way to understand how the grammar works and what it is intended to produce.

7 Grammatical Relations

The current grammar outputs grammatical relations (GRs) based on a rule-to-rule encoding of GRs with PS rules. Historically, this developed from a procedural mapping from TSG derivations to a GR scheme close to that recommended for parser evaluation by EAGLES. However, this has been modified to make it more appropriate to a shallow semantic representation (Briscoe *et al*, 2002) which might form the input to further semantic processing such as robust minimal recursion semantics construction or to ‘event’ extraction. The scheme is similar to the F-structure of LFG, but is used by RASP purely as an output representation with no effect on the underlying space of possible derivations.

The main changes over the EAGLES-derived scheme (which is described on the web at www.informatics.susx.ac.uk/research/nlp/carroll/grdescription/index.html) are that most GRs are now factored into simple binary lexical relations, and coordination is not distributed across conjuncts but rather the coordinator is treated as (semantic) head – this avoids problems of interaction with scope in examples like: *Every man smiled or laughed* i.e. (conj or smiled) (conj or laughed) (ncsubj or man _) – which does not entail that every man smiled and laughed.

The new scheme also adds several relations, notably text adjunct (ta) which encodes information conveyed by punctuation. The aim is that the GR scheme is a factored representation of a directed graph which is 'almost' acyclic. Most cycles are between adjacent nodes and involve disjoint relations in which head and dependent lexical items are reversed. For example in *the hidden chest* we output a non-clausal modifier (ncmod) relation between head (*chest*) and dependent (*hidden*) and a non-clausal surface subject / underlying object relation between head (*hidden*) and dependent (*chest*) i.e. (ncsubj hidden chest obj). However, we also create cycles to capture unbounded dependencies between nominal heads and (relative) clause postmodifiers, for example in *ideas (that) linguists (want to) promote*, there is an underspecified obj (dobj/obj2/iobj) relation (obj promote ideas) as well as (cmod (that) ideas promote). However, given that this is an unbounded dependency it can involve non-adjacent nodes: (cmod (that) ideas want), (xcomp want promote), (obj promote ideas).

All GRs are of the following form:

(GR-type optional-subtype head dependent optional-initial-GR)

In practice, only non-clausal subjects take the optional initial GR field and only some modifier and complement relations take the optional subtype field. We also use a single feature (passive head) which is required to facilitate recovery of the initial-GR field for non-clausal subjects. There are 17 fully-specified GR types, however, these are organized into an inheritance hierarchy shown in Figure 1 which facilitates underspecification of relations e.g. with some unbounded dependency constructions (see section 5.9) and also cross-parser evaluation. In addition to the main GR types, there are subtypes of some GRs which indicate the presence of some grammatical markers such as infinitive *to*, complementizer *that* and so forth. The intention is to output a representation which conveys as much as possible about the likely underlying predicate-argument structure of the input sentence. To this end, certain aspects of the output are default, especially default subtype/initial-GR values (_) and relations in unbounded dependency constructions (arg_mod, obj), and others are frequency-based heuristics that may need overriding when more specific lexical information is available, as in *Kim promised Lee to go* (ncsubj go Lee _) → (ncsubj go Kim _) – see section 7.2 below.

Further information can be obtained by inspecting the grammar file and comments associated with rules. In the GDE, the GR output can be viewed from the parser command line using 'v(iew) sem(antics)' as the mechanism used to construct GRs is a simple modification of that used to produce the compositional semantics of the ANLT full grammar

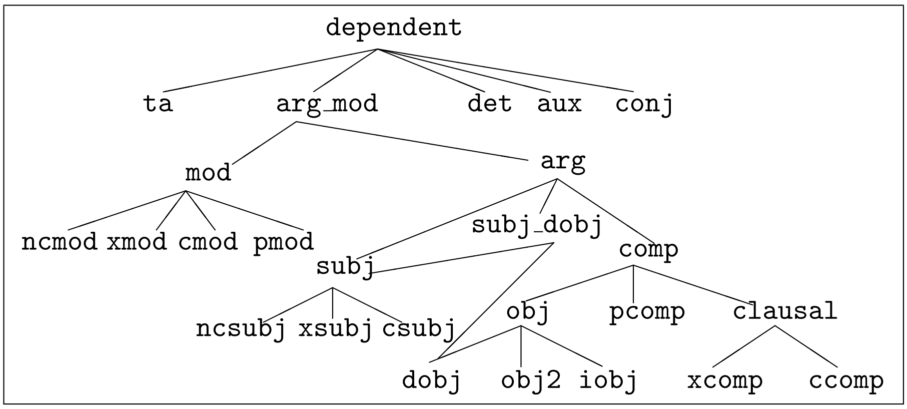

Figure 1: The GR hierarchy

(Grover *et al*, 1993). The following contains a brief description of each GR type, its subtypes and intended use.

conj encodes relations between a coordinator and the head of a conjunct. There will be as many such binary relations as there are conjuncts of a specific coordinator. It has no additional fields.
Kim likes oranges, apples, and satsumas or clementines

```
(ncsubj likes Kim _) (dobj likes and)
(conj and oranges) (conj and apples) (conj and or)
(conj or satsumas) (conj or clementines)
```

aux encodes relations between main verbs as (semantic) head and auxiliary dependents. It has no additional fields. There will be as many such binary relations as there are auxiliaries. If a copular or main verb form of an auxiliary is present then it is the head of any such aux relation. The head of aux can be ellip(tical) as in *Kim will.*
Kim has been sleeping

```
(ncsubj sleeping Kim _) (aux sleeping has) (aux sleeping been)
```

det encodes a binary relation between articles, quantifiers, partitives and other single word forms which can begin NPs and the head of the NP. It has no additional fields.
Some men came

```
(det men Some) (ncsubj came men _)
```

ncmod encodes binary relations between non-clausal modifiers and their heads. There are subtypes: default (_), part(itive), prt(particle), poss(essive), num(ber), ta(text adjunct), and ij(interjection). The default case covers most pre-/post-modification.
the old man in the barn slept

```
(ncmod _ man old) (ncmod _ man in) (dobj in barn)
```

Numbers are identified as special types of modifier where possible. Possessives are treated as relations between head and dependent nouns:
the butcher's shop

```
(ncmod poss shop butcher)
```

where the head can be ellip(tical). Partitive predeterminers are (ncmod part men all) *all the men*). Verbal particles are (ncmod prt look up) (*look up the word*).

xmod encodes binary unsaturated predicative relations between modifiers (VPs, APs) and heads. There are subtypes default (_) and 'to', the latter is used when the modifier is an infinitive VP (though the current grammar doesn't always recover it):
who to talk to

```
(xmod to who talk) (iobj talk to) (dobj to who)
```

cmod encodes binary saturated relations between clausal (S) modifiers and heads. There are subtypes default (_) and complementizer 'that': *although he came, Kim left*

```
(cmod _ left although) (ccomp although came)
```

pmod encodes binary relations between PP modifiers with PP complements and heads:
he went, off into the darkness

```
(pmod went off) (pcomp off into) (dobj into darkness)
```

ncsubj encodes binary relations between non-clausal subjects (NPs, PPs) and their verbal heads. There are no subtypes but four initial GR values: default/subj (_), underlying object (obj), raising subject (rais) and inverted (inv) which is used for locative (PP, AdvP) inversion and quote inversion (*said Kim*):
the upset man

```
(ncsubj upset man obj) (passive upset)
```

All passives get the passive feature but the overwriting of the default ncsubj subtype is often left to the GR inference stage (see section 7.2 below. (This is because of limitations of the rule-to-rule translation mechanism.) This relation is also used for understood subjects of unsaturated predicative complements and some modifiers – these assignments are heuristic and based on the assumption that most transitive/intransitive verbs are object/subject equi, as in *Kim wants to go* (ncsubj go Kim _). The raising subtype is thus never inserted by the grammar.

xsubj encodes binary relations between unsaturated predicative subjects (VP, AP) and verbal heads. This relation only has non-default subtype value inverted (inv) used for extraposition examples like *to go appears difficult*:
leaving matters

```
(xsubj matters leaving _)
```

csubj is a binary relation between saturated clausal subjects (S/V2) and verbal heads. The subtype slot is filled by the complementizer if the clause is finite and left empty for non-finite 'small clauses' like *her coming matters*:
that he came matters

```
(csubj matters came that) (ncsubj came he)
```

dobj is a binary relation between verbal or prepositional head and the head of the NP to its immediate right. Thus, it doesn't distinguish between themes and goals with dative alternation verbs:
She gave it to Kim

```
(dobj gave it) (ncsubj gave She _) (iobj gave to) (dobj to Kim)
```

obj2 is a binary relation between verbal heads and the head of the second NP in a double object construction:
She gave Kim toys

```
(obj2 gave toys) (dobj gave Kim) (ncsubj gave She _)
```

The NP immediately to the right of a verb in a passive construction is sometimes recognized correctly as obj2 (e.g. when in a reduced relative) but not when in a main clause. However, the presence of (passive given) in the GR set for *Kim was given it* can be used to trigger an overwrite rule – see section 7.2 below.

iobj is a binary relation between a head and the preposition of a PP argument when the PP complement is a NP:
Kim flew to Paris from Geneva

```
(ncsubj flew Kim _) (iobj flew to) (iobj flew from)
(dobj to Paris) (dobj from Geneva)
```

pcomp is a binary relation between a head and the preposition of a PP argument when the PP complement is itself a PP:
Kim climbed through into the attic

```
(ncsubj climbed Kim _) (pcomp climbed through)
(pcomp through into) (dobj into attic)
```

xcomp is a binary relation between a head and an unsaturated VP complement. It has subtypes: default (_) and 'to', the latter indicating an infinitival complement. However, the current grammar doesn't always insert 'to' when the complement is infinitival:
Kim thought of leaving

```
(ncsubj thought Kim _) (xcomp _ thought of) (xcomp _ of leaving)
```

ccomp is a binary relation between a head and the head of a saturated clausal complement, either finite, subjunctive, headed by a wh-element or a non-finite 'small clause'. It has subtypes: default (_) and 'that'. The head of the dependent clause is usually the verb but can be the subject of the 'small clause':
Kim asked about him playing rugby

```
(ncsubj asked Kim _) (ccomp _ asked about) (ccomp _ about him)
(ncsubj playing him _) (dobj playing rugby)
```

ta is a binary relation between a head and the head of a text adjunct delimited by some punctuation (see section 4). It has subtypes: 'quote', 'brack'(et), 'dash' 'colon', 'comma', 'bal'(anced), 'end', 'echo', 'tag' (questions), 'refl'(exive), 'voc'(ative). The first five subtypes are self-explanatory. Balanced text adjuncts have matching delimiting punctuation (e.g. commas or dashes at both boundaries). End text adjuncts usually occur sentence-finally and thus the matching punctuation mark at the right boundary is promoted to a full stop. The remaining subtypes attempt to infer some of the semantic/discourse import of a comma from information in the PoS tagset concerning nominal lexical types. They are sometimes wrong in the current grammar, so could more conservatively all be mapped to 'comma':
He made the discovery: Kim was the abbot; Lee was the host.

```
(ncsubj made He _)
(dobj made discovery)
(ta colon discovery was)
(ncsubj was Lee _)
(xcomp _ was host)
(det host the)
(ncsubj was Kim _)
(xcomp _ was abbot)
(det abbot the)
(det discovery the)
```

Underspecification is used by the current parser, especially in the analysis of unbounded dependencies, so that **obj**, **arg** and **arg_mod** may be output and also may be further specified as described in section 7.2. **mod**, **subj**, **comp** and **subj_or_dobj** are provided primarily for cross parser evaluation purposes.

7.1 Mapping from the EAGLES Evaluation Scheme GRs

The first release of RASP supported GR output in a format very similar to that proposed by the EAGLES parser evaluation working group (see www.informatics.susx.ac.uk/research/nlp/carroll/grdescription/index.html). The following rules describe the mapping from the old scheme to the new scheme. The mapping will not be perfect as there are more subtypes on some GRs in the new scheme (e.g. part, num, ta on ncmod), new GRs (e.g. ta), and more information (especially control and unbounded relations) extracted.

Lower case variables, x, y, ... range over word| lemma-affix-tag tokens which represent the heads/dependents in GRs and/or over subtype values of specific GRs. Where the values of these variables need restricting, I use the convention that x=(_A) matches any word/lemma with PoS tag prefix A, x=(_*A*) any word/lemma with PoS infix A, while x=(_) indicates the literal (subtype default) value 'underscore'. I also use x=(A|B) as a disjunction operator over tag and token components.

1. $\forall$ x=(_|poss|prt),y,z (ncmod x y z) $\Rightarrow$ (ncmod x y z)
2. $\forall$ x=(_I),y,z=(_N) (ncmod x y z) $\Rightarrow$ (ncmod _ y x) $\wedge$ (dobj x z)
3. $\forall$ x=(_I),y,z=(_I) (ncmod x y z) $\Rightarrow$ (pmod _ y x) $\wedge$ (pcomp x z)
4. $\forall$ x=(_|to),y,z (xmod x y z) $\Rightarrow$ (xmod x y z)
5. $\forall$ x=(_|that),y,z=(_V) (cmod x y z) $\Rightarrow$ (cmod x y z)
6. $\forall$ x=(_I|C),y,z (cmod x y z) $\Rightarrow$ (cmod _ y x) $\wedge$ (ccomp x z)
7. $\forall$ x=(_*Q*),y,z (cmod x y z) $\Rightarrow$ (cmod _ y x) $\wedge$ (arg x z)
8. $\forall$ x,y,z (ncsubj x y z) $\Rightarrow$ (ncsubj x y z)
9. $\forall$ x,y (ncsubj x y obj) $\Rightarrow$ (passive x)
10. $\forall$ x,y,z (xsubj x y z) $\Rightarrow$ (xsubj x y z)
11. $\forall$ x,y,z (csubj x y z) $\Rightarrow$ (csubj x y z)
12. $\forall$ x,y,z (dobj x y z) $\Rightarrow$ (dobj x y)
13. $\forall$ x,y (obj2 x y) $\Rightarrow$ (obj2 x y)
14. $\forall$ x,y,z(_N) (iobj x y z) $\Rightarrow$ (iobj y x) $\wedge$ (dobj x z)
15. $\forall$ x,y,z(_I) (iobj x y z) $\Rightarrow$ (pcomp y x) $\wedge$ (pcomp x z)
16. $\forall$ x=(_|to),y,z (xcomp x y z) $\Rightarrow$ (xcomp x y z)
17. $\forall$ x=(_|that),y,z=(_V) (ccomp x y z) $\Rightarrow$ (ccomp x y z)
18. $\forall$ x=(_I|C),y,z (ccomp x y z) $\Rightarrow$ (ccomp _ y x) $\wedge$ (ccomp _ x z)
19. $\forall$ x,y,z (arg_mod x y z) $\Rightarrow$ (ncmod _ y x) $\wedge$ (dobj x z)
20. $\forall$ x,y,z,... (conj x y z ...) $\Rightarrow$ (conj x y) $\wedge$ (conj x z) $\wedge$ (conj x ...) ...
21. $\forall$ x,y (detmod _ x y) $\Rightarrow$ (det x y)
22. $\forall$ x,y (aux _ x y) $\Rightarrow$ (aux x y)

7.2 GR Inference

The new GR extraction process relies on purely local information available within a local instantiated tree/PS rule. In some cases, it is possible to D(elete), C(orrect) or A(dd) to the resulting GRs using rules which look at subsets of the GR output including L(emmas)(see section 7.1 for other notational conventions). The following is not an exhaustive list of such rules. In particular, there are many more rules similar to those beginning with 10) below which add or further specify missing GRs in unbounded dependency constructions. Such rules require a valency dictionary and could be used to probabilistically extend GR sets if the valency dictionary included conditional probabilities of the form $P(valency \mid verb)$.

1. **Passive1:** ∀ x,y (passive x) ∧ (ncsubj x y _) ⇒ C:(ncsubj x y obj)

2. **Passive2:** ∀ x,y (ncsubj x y obj) ∧ ¬ (passive x) ⇒ A:(passive x)

3. **Passive3:** ∀ x,y,z ((ncsubj x y obj) ∨ (passive x)) ∧ (dobj x z) ⇒ C:(obj2 x z)

4. **Not-passive1:** ∀ x,y,z (aux x have) ∧ ((ncsubj x y obj) ∨ (passive x)) ⇒ C:(ncsubj x y _) ∧ D:(passive x)

5. **Not-passive2:** ∀ x,y,z (aux x have) ∧ ((ncsubj x y obj) ∨ (passive x)) ∧ (obj2 x z) ⇒ C:(dobj x z) ∧ C:(ncsubj x y _) ∧ D:(passive x)

6. **Predicative:** ∀ x=(_VB),y,z (xcomp _ x y) ∧ (ncsubj x z _) ⇒ C:(ncsubj y z _)

7. **SRaising:** ∀ x=(L_V),y,z,w,w′ (dobj x y) ∧ (ncsubj x z _) ∧ ¬ (passive x) ∧ ¬ (xcomp w x w′) ⇒ C:(xcomp _ x y) ∧ C:(ncsubj y z _), L=become,seem,...

8. **ORaising:** ∀ x=(L_V),y,z,w (dobj x y) ∧ (xcomp w x z) ⇒ D:(dobj x y) ∧ A:(ncsubj z y _), L=believe,suppose,...

9. **SControl:** ∀ x=(L_V),y,z,w,w′ ¬ (dobj x y) ∧ (xcomp w x z) ∧ (ncsubj x w′ _) ⇒ A:(ncsubj z w′ _), L=try,want,...

10. **OControl:** ∀ x=(L_V),y,z,w (dobj x y) ∧ (xcomp w x z) ⇒ A:(ncsubj z y _), L=want,expect,...

11. **Promise:** ∀ x=(L_V),y,z,w,w′ (dobj x y) ∧ (xcomp w x z) ∧ (ncsubj x w′ _) ⇒ A:(ncsubj z w′ _), L=promise,...

12. **Scomp2Rel1** ∀ x,y=(_N),z=(L_V),w, w′ (ccomp x y z) ∧ (ncsubj z w _) ∧ ¬ (dobj z w′) ⇒ C:(cmod x y z) ∧ A:(dobj z y), L=seek,... (oblig. transitive)

13. **Scomp2Rel2** ∀ x,y=(_N),z,w=(L_V),w′, w″ (ccomp x y z) ∧ (xcomp w′ z w) ∧ ¬ (dobj w w″) ⇒ C:(cmod x y z) ∧ A:(dobj w y), L=seek,... (oblig. transitive)

14. **Rel1-UB** ∀ x,y=(_N),z=(L_V),w, w′ (cmod x y z) ∧ (ncsubj z w _) ∧ ¬ (dobj z w′) ⇒ A:(dobj z y), L=seek,... (oblig. transitive)

15. **Rel2-UB** $\forall$ x,y=(_N),z,w=(L_V),w′, w″ (cmod x y z) $\wedge$ (xcomp w′ z w) $\wedge$ $\neg$ (dobj w w″) $\Rightarrow$ A:(dobj w y), L=seek,. . . (oblig. transitive)
16. **Rel-Inf-UB** $\forall$ x,y=(_N),z=(L_V) (xmod x y z) $\wedge$ (obj z y) $\Rightarrow$ C:(dobj z y), L=seek,. . . (oblig. transitive)

8 Other Output Representations

In this section, I give an example of RASP's various outputs – further examples can be found at www.informatics.susx.ac.uk/research/nlp/rasp/offline-demo.html For the input sentence, *I can can a can* the system produces GRs:

```
(|ncsubj| |can_VV0| I_PPIS1 _)
(|aux| |can_VV0| |can_VM|)
(|dobj| |can_VV0| |can_NN1|)
(|det| |can_NN1| |a_AT1|)
```

In order to distinguish multiple occurrences of the same form in a sentence and so word order can be reconstructed unambiguously from GR output, by default, the word sequence in each sentence is numbered: 1-n in the output:

```
(|ncsubj| |can:3_VV0| I:1_PPIS1 _)
(|aux| |can:3_VV0| |can:2_VM|)
(|dobj| |can:3_VV0| |can:5_NN1|)
(|det| |can:5_NN1| |a:4_AT1|)
```

In addition to GR output descibed in section 7 above, there are other output formats many of which can be selected from the command line using the RASP shell script ($RASP/scripts/rasp.sh).

Weighted GRs are GRs with weights computed from the probabilities of the set of derivations in the top n derivations which yield this GR. This output mode allows the user to define the trade-off between precision and recall by setting a weight threshold for further processing. However, the set of GRs output is no longer consistent so further processing is required to select a consistent and complete set of GRs for the input (see Carroll & Briscoe, 2002). The definition of consistency and completeness will be very similar to that for LFG F-structures but is complicated by the fact that a few words like complementizer *that* and infinitive marker *to* appear as GR subtype values as opposed to as nodes in the graph. A further caveat is that when no complete parses can be found for an input, weighted GRs are computed from a single sequence of subanalyses (and thus all appear with weight 1). For instance, the weighted GRs for the example sentence are:

```
0.231392       (|ncsubj| |can_VV0| I_ZZ1 _)
 1.0            (|dobj| |can_VV0| |can_NN1|)
 1.0            (|aux| |can_VV0| |can_VM|)
 0.768608       (|ncsubj| |can_VV0| I_PPIS1 _)
 1.0            (|det| |can_NN1| |a_AT1|)
```

If multiple tags for *can* were not filtered out, then further weighted GRs would be present.

Grammar Rules as labels of the nodes in a derivation can be output primarily for debugging purposes. Rule names mostly follow the convention 'M/d1-f1-f2-fN_d2_dN' where M is the mother category, d1...dN are the daughter categories and f1...fN are salient features of daughters (see the TSG file comments for more details on naming conventions). It is a bad idea to base further processing on these names as they invariably change when the grammar is modified. The labelled bracketing with grammar rule names as nodes for the example sentence is:

```
(|T/txt-sc1/-+|
 (|S/np_vp| I_PPIS1
  (|V1/modal_bse/--| |can_VM|
   (|V1/v_np| |can_VV0| (|NP/det_n1| |a_AT1| (|N1/n| |can_NN1|)))))
 (|End-punct3/-| ._.))
```

PTB-style (Penn Treebank) labels on the nodes of derivations can be selected if such similarity is useful. However, the tree topology will be different (and more informative) in RASP as it uses the X-bar scheme (see section 5).

Shallow Phrasal labels on the nodes of derivations emulate the Susanne (Sampson, 1995) treebank scheme though again not the tree topology.

Alias labels on the nodes of derivations use the alias declarations from the grammar file located in $RASP/prob/data/tsgX. For instance,

```
ALIAS V1 = [V +, N -, BAR 1].
```

so nodes labelled V1 will be subsumed by this aliased feature set.

Other representations of node labels in derivations are available and can be selected from within Lisp using the functions linked to the 'view' command in the GDE, including the full featural representation of each node. However, in general we recommend GRs, and it is possible to specify many tasks that are often taken to require trees in terms of GRs (e.g. for anaphora resolution see Preiss & Briscoe, 2003).

9 Performance and Accuracy

Training on about 4K sentences paired with unlabelled brackets from the Susanne treebank, and testing on 560 sentences from the Penn Treebank reannotated with GRs (part of the Parc DepBank700 reannotated from section 23 of the Wall Street Journal in the Penn Treebank) the parser achieves a microaveraged F1-score of 74.9%. RASP took 153secs. to load and parse these sentences on a 1.8GHz Opteron CPU. See $RASP/prob/greval for more details of the evaluation scheme, data, etc.

Using RASP in tag thresholding mode yields little improvement in accuracy on this test set but may be worthwhile for datasets with a large number of unknown words (for the PoS tagger) at some cost to speed. Using the subcategorization option is likely to yield some small improvements in accuracy at little cost to performance, so is recommended. Using the inside-outside algorithm variant of weighted GR extraction will increase accuracy at little cost to efficiency and is, therefore, recommended (see parser options in the RASP shell scripts).

Improving parse selection accuracy by lexicalizing the current model, building a model over GRs, and/or using discriminative techniques over the parse forest is a current area of research (Sloane, 2005; Watson *et al*, 2005).

10 Customization

10.1 Tokenization / Sentence Boundary Detection

Sentences which are not already split and marked using the hat symbol will be split by the RASP sentence boundary detector (see $RASP/sentence) and then passed to the tokenizer (see $RASP/tokenise). Idiosyncracies of input can lead to failure at this point with dramatic effects on parser performance. Unusual sequences of punctuation marks, such as '(A):-' or '`-----`', and archaic abbreviations, such as *'twas* or *'tis* will not be treated correctly by the current modules.

Therefore it is worth looking carefully at the output of this stage and either preprocesssing with global substitutions in a batch editor to make such sequences compatible with the current modules (and tagger lexicon) or modifying the modules directly (e.g. `-----` → `--`; or 'twas → it was).

10.2 GR Output

Modifying the syntactic part of the grammar rules is not possible as the public release does not yet allow you to retrain the probabilistic parser. However, altering the GR output from (sets of) rules is possible and may be desirable if, for example, it is preferable to have a less informative set of GRs which are guaranteed to yield directed acyclic graphs. The description of the semantic formalism used to generate GRs is given in Carroll *et al* (1991) and Grover *et al* (1993) in more detail. However, to find cycles in the GRs it is only necessary to find PS rules which specify more than one GR output with identical daughter variables (usually reversed as head/dependent), for example:

```
PSRULE N1/n_ppart : N1[POSS -, MOD -] --> H0[NTYPE NORM] V1[VFORM
   PPART] : 1 : (xmod _ 1 2) : (ncsubj 2 1 obj) : (passive 2).
PSRULE NP/np-whpro_inf : N2[WH +, POSS -, MOD +] --> H2[WH +, NTYPE PRO]
   V1[VFORM INF, FIN -] : 1 : (xmod to 1 2) : (obj 2 1).
```

Removing the second of these GRs from the output specification is usually most appropriate and will remove cycles, at some cost to informativeness.

10.3 Grammar Tuning

Though it is not possible to add rules to the grammar or retrain the grammar, it is possible to remove rules by customizing the RASP shell script with a list of unwanted rule names (see $RASP/scripts/README for details). This can be useful because the overall accuracy of the parser may be improved by deletion of rules that will rarely apply correctly within a particular text type or genre, but which may nevertheless occur incorrectly in significant numbers of highly ranked parses. Removing such rules will not cause the parser to fail outright, as the fragmentary parse mechanism will be invoked in cases where the only possible parse required the deleted rule.

Many of the -r rules in the grammar are there to cover quite specific constructions which occur rarely in many types of text. For example, text which does not contain a high proportion of questions or elliptical constructions with auxiliaries (as in fictional dialogue) may be parsed more accurately without the four rules with names ending in '_gap-r'. By examining incorrect but highly ranked parses on specific data, it should be possible to identify candidate rules for removal.

10.4 Further Processing

Many people are tempted to use trees labelled with grammar rule names as the output they will work with (as it appears informative and is the default tree output representation). This output is very useful for debugging and understanding the grammar. However, it changes almost every time the grammar is modified. Therefore, it is far better to work with a relatively stable output scheme. The GRs are the most stable output available as a command line option in the distribution. Derivations containing category aliases, or Penn Treebank style output are more stable than those containing rule names but may still change if modifications to the grammar alter tree topology. These can be selected as a parameter to the RASP shell script.

11 References

Abdalla, R.M. (2005) *Syntactic and lexical variants of cue phrases,* M.Phil Dissertation, University of Cambridge Computer Laboratory.

Abney, S. (1996) 'Part-of-speech tagging and partial parsing' in Church, K. *et al* (eds.), *Corpus-based Methods in Language and Speech,* Kluwer, Dordrecht.

Eneko Agirre, Aitziber Atutxa, Koldo Gojenola, Kepa Sarasola (2004) 'Exploring portability of syntactic information from English to Basque', *Proceedings of the LREC'04,* Lisbon.

Alina Andreevskaia, Zhuoyan Li and Sabine Bergler (2005) 'Can Shallow Predicate Argument Structures Determine Entailment?', *Proceedings of the Pascal Challenge Workshop on Recognizing Entailments,* Southampton.

Baldwin, T. (2005) ‘Bootstrapping Deep Lexical Resources: Resources for Courses’, *Proceedings of the ACL SIGLEX Workshop on Deep Lexical Acquisition,* Ann Arbor, pp. 67–76.

Baldwin, T. and Bond, F. (2003) ‘Learning the Countability of English Nouns from Corpus Data’, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics,* Sapporo, Japan, pp. 463–470.

Bannard, C. (2005) ‘Learning about the Meaning of Verb Particle Constructions from Corpora’, *Journal of Computer Speech and Language, vol.19.4,*

Bennett, P. and Carbonell, J. (2005) ‘Detecting action items in e-mail’, *Proceedings of the 28th Int. ACM SIGIR R&D in Information Retrieval,* Salvador, Brazil.

Briscoe, E.J. (1994) *Parsing (with) Punctuation, etc.,* Xerox European Research Laboratory, Grenoble, MLTT-TR-002 // www.cl.cam.ac.uk/ ejb/punct-pos-parsing.ps.gz.

Briscoe, E.J. and J. Carroll (1995) ‘Developing and Evaluating a Probabilistic LR Parser of Part-of-Speech and Punctuation Labels’, *Proceedings of the ACL/SIGPARSE 4th Int. Workshop on Parsing Technologies (IWPT95),* Prague / Karlovy Vary, Czech Republic, pp. 48–58.

Briscoe, E.J. and J. Carroll (1997) ‘Automatic extraction of subcategorisation from corpora’, *Proceedings of the 5th ACL Conf. on Applied Nat. Lg. Proc.,* Morgan Kaufmann, pp. 356–363.

Briscoe, E.J. and J. Carroll (2002) ‘Robust accurate statistical annotation of general text’, *Proceedings of the 3rd Int. Conf. on Language Resources and Evaluation (LREC’02),* Las Palmas, Gran Canaria, pp. 1499–1504.

Briscoe, E., J. Carroll, J. Graham and A. Copestake (2002) ‘Relational evaluation schemes’, *Proceedings of the LREC’02 Workshop, Beyond PARSEVAL: Towards Improved Evaluation Measures for Parsing Systems,* Gran Canaria.

Briscoe, E.J., C. Grover, B.K. Boguraev and J. Carroll (1987) ‘A formalism and environment for the development of a large grammar of English’, *Proceedings of the 10th Int. Joint Conf. on Artificial Intelligence,* Milan, Italy, pp. 703–708.

Briscoe, E.J. and N. Waegner (1992) ‘Robust stochastic parsing using the inside-outside algorithm’, *Proceedings of the AAAI Workshop on Probabilistically-Based NLP Techniques ,* San Jose, Ca, pp. 39–53.

Buttery, P. and Korhonen A. (2005) ‘Large-scale analysis of verb subcategorization differences between child directed speech and adult speech’, *Proceedings of the Verb Workshop 2005, Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes,* Saarland University.

Callmeier, U., A Eisele, U Schafer, and M Siegel (2004) ‘The DeepThought Core Architecture Framework’, *Proceedings of the LREC’04,* Lisbon.

Carroll, J. and E.J. Briscoe (2002) ‘High precision extraction of grammatical relations’, *Proceedings of the 19th Int. Conf. on Computational Linguistics (COLING’02),* Taipei, Taiwan, pp. 134–140.

Carroll, J., E. Briscoe and C. Grover (1991) *A development environment for large natural language grammars,* Cambridge University Computer Laboratory, TR-233
ftp.cl.cam.ac.uk/nltools/reports/gde.ps.

Church, K. (1988) 'A stochastic parts program and noun phrase parser for unrestricted text', *Proceedings of the 2nd Conference on Applied Natural Language Processing,* Austin, Texas, pp. 136–143.

Clark, S. and Weir, D. (2002) 'Class-Based Probability Estimation using a Semantic Hierarchy', *Computational Linguistics, vol.28.2,* 187–206.

Copestake, A. (2003) *Report on the Design of RMRS,* Deep Thought EU project Deliverable D1.1a
www.project-deepthought.net/.

Curran, J. (2003) *From distributional to semantic similarity,* Phd Dissertation, Dept. of Informatics, University of Edinburgh.

De Souza, S. (2005) *Simple discourse topic summarisation,* M.Phil Dissertation, University of Cambridge Computer Laboratory.

Elworthy, D. (1993) *Part-of-speech tagging and phrasal tagging,* Acquilex-II Working Paper 10, Cambridge University Computer Laboratory ($RASP/tag/documentation).

Elworthy, D. (1994) 'Does Baum-Welch re-estimation help taggers?', *Proceedings of the 4th ACL Conference on Applied NLP,* Stuttgart, Germany, pp. 53–58.

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

Grover, C., E.J. Briscoe and J. Carroll (1993) *The Alvey Natural Language Tools Grammar (4th Release),* Cambridge University Computer Laboratory, TR-284
ftp.cl.cam.ac.uk/nltools/reports/grammar.ps.

Grover, C., Lascarides, A. and Lapata, M. (2005) 'A comparison of parsing technologies for the biomedical domain', *Natural Language Engineering, vol.11.1,* 27–65.

Jurafsky, D. and Martin, J. (2000) *Speech and Language Processing,* Prentice-Hall.

Kaplan, R., S. Riezler, T. H. King, J. T. Maxwell III, A. Vasserman and R. Crouch (2004) 'Speed and accuracy in shallow and deep stochastic parsing', *Proceedings of the Human Language Technology Conference and the 4th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04),* Boston, MA.

Khan, S. (2006) 'Negation in sentiment analysis: employing contextual valence shifters to enhance classification accuracy', *Proceedings of the ACL-Coling'06,* submitted.

Korhonen, A. (2002) *Subcategorization Acquisition,* Techical Report UCAM-CL-TR-530, Computer Laboratory, University of Cambridge.

Korhonen, Anna, Yuval Krymolowski and Zvika Marx (2003) ‘Clustering Polysemic Subcategorization Frame Distributions Semantically’, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics,* Sapporo, Japan, pp. 64–71.

Leidner, Jochen L., Johan Bos, Tiphaine Dalmas, James R. Curran, Stephen Clark, Colin J. Bannard, Bonnie Webber, Mark Steedman (2003) ‘QED: The Edinburgh TREC-2003 Question Answering System’, *Proceedings of the TREC’03 QA Track,* NIST.

McCarthy, D., and J Carroll (2003) ‘Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences’, *Computational Linguistics, vol.29.4,* 639–654.

McCarthy, D., B Keller, and J Carroll (2003) ‘Detecting a continuum of compositionality in phrasal verbs’, *Proceedings of the ACL-2003 Workshop on Multiword Expressions,* Sapporo, Japan.

McCarthy, D., R. Koeling, J. Weeds and J. Carroll (2004) ‘Finding predominant word senses in untagged text’, *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, pp. 280–287.

Medlock, B. (2006) *Anonymisation of informal text,* LREC’06, in press.

Nielsen, L. (2004) ‘Verb Phrase Ellipsis detection using Automatically Parsed Text’, *Proceedings of the COLING’04,* Geneva.

Nissim, M., and Markert, K. (2003) ‘Syntactic Features and Word Similarity for Supervised Metonymy Resolution’, *Proceedings of the 41st Ann. Mtg of Assoc. for Comp. Linguistics,* Sapporo, Japan.

Nunberg, G. (1990) *The linguistics of punctuation,* CSLI Lecture Notes 18, Stanford, CA.

Preiss, J. (2003) ‘Using Grammatical Relations to Compare Parsers’, *Proceedings of the Eur. Assoc. Comp. Linguistic,* Budapest, pp. 291–298.

Preiss, J. (2004) ‘Probabilistic Word Sense Disambiguation’, *Journal of Computer Speech and Language, vol.18.3,* 319–337.

Preiss, J. and E.J. Briscoe (2003) ‘Intermediate parsing for anaphora resolution? Implementing the Lappin and Leass non-coreference filters’, *Proceedings of the Proc. of the ACL03 Workshop on Anaphora Resolution,* Morgan Kaufmann, pp. 1–6.

Pustejovsky, James, Patrick Hanks, and Anna Rumshisky (2004) ‘Automated Induction of Sense in Context’, *Proceedings of the Coling’04,* Geneva.

Sagae,K., A Lavie, B MacWhinney (2005) ‘Automatic Measurement of Syntactic Development in Child Language’, *Proceedings of the 43rd Annual Meeting of the ACL,* Ann Arbor, pp. 197–204.

Sampson, G. (1995) *English for the Computer,* Oxford University Press, Oxford, UK.

Sporleder, C. and Lascarides, A. (2005) ‘Exploiting Linguistic Cues to Classify Rhetorical Relations’, *Proceedings of the Recent Advances in Natural Language Processing (RANLP-05),* Borovets, Bulgaria.

Tsang V. and S. Stevenson (2004) 'Using selectional profile distance to detect verb alternations', *Proceedings of the HLT/NAACL 2004 Workshop on Computational Lexical Semantics,* Boston.

Vlachos, A., Gasperin, C., Lewin, I., Briscoe, E. J. (2006) 'Bootstrapping the recognition and anaphoric linking of named entities in Drosophila articles', *Proceedings of the Pacific Symposium in Biocomputing,* Hawaii.

Wang, D. (2005) *Classifying text using linguistic features,* M.Phil Dissertation, University of Cambridge Computer Laboratory.

Watson, R., Preiss, J. and Briscoe, E.J. (2003) 'The Contribution of Domain-independent Robust Pronominal Anaphora Resolution to Open-Domain Question-Answering', *Proceedings of the Int. Symposium on Reference Resolution and its Applicati on to Question-Answering and Summarisation,* Venice.

Watson, R., J. Carroll and E. Briscoe (2005) 'Efficient extraction of grammatical relations', *Proceedings of the 9th Int. Workshop on Parsing Technologies (IWPT'05),* Vancouver, Ca..

Weeds, J., B Keller, D Weir, I Wakeman, J Rimmer (2004a) 'Natural language expression of user policies in pervasive computing environments', *Proceedings of the OntoLex 2004 (LREC Workshop on Ontologies and Lexicons),* Lisbon.

Weeds, J., D Weir, and D McCarthy (2004b) 'Characterising Measures of Lexical Distributional Similarity', *Proceedings of the Coling'04,* Geneva.

Weeds J., D Weir, and B Keller (2005) 'The Distributional Similarity of Sub-Parses', *Proceedings of the ACL SIGLEX Workshop on Deep Lexical Acquisition,* Ann Arbor, pp. 7–13.

Yallop, J., Korhonen, A. and Briscoe, E.J. (2005) 'Automatic acquisition of adjectival subcategorisation from corpora', *Proceedings of the 43rd Assoc. for Comp. Ling.,* Morgan Kaufmann, pp. 614–621.

Yamada, I. and Baldwin, T. (2004) 'Automatic Discovery of Telic and Agentive Roles from Corpus Data', *Proceedings of the 18th Pacific-Asia Conf. on NLP,* Tokyo, Japan, pp. 115–126.