# Engineering a Wide-Coverage Lexicalized Grammar

J. Carroll, N. Nicolov, O. Shaumyan, M. Smets & D. Weir
School of Cognitive and Computing Sciences
University of Sussex
Brighton, BN1 6RG
UK

## Abstract

*We discuss a number of practical issues that have arisen in the development of a wide-coverage lexicalized grammar for English. In particular, we consider the way in which the design of the grammar and of its encoding was influenced by issues relating to the size of the grammar.*

## 1. Introduction

Hand-crafting a wide-coverage grammar is a difficult task, requiring consideration of a seemingly endless number of constructions in an attempt to produce a treatment that is as uniform and comprehensive as possible. In this paper we discuss a number of practical issues that have arisen in the development of a wide-coverage lexicalized grammar for English: the LEXSYS grammar. In particular, we consider the way in which the design of the grammar and of its encoding—from the viewpoint both of the grammar writer and of the parsing mechanism—was influenced by issues relating to the size of the grammar.

One criterion that is often used as a judge of grammar quality is the extent to which 'linguistic generalizations' have been captured. Generally speaking, concern over this issue leads to a preference for smaller rather than larger grammars. A second reason for preferring smaller grammar sizes is on the basis of parsing efficiency, since the running time of parsing algorithms generally depends on the size of the grammar.

However, a rather different criterion determining grammar quality has to do with the analyses that the grammar assigns to sentences: in particular, the extent to which they provide a good basis for further, perhaps deeper processing. It is not necessarily the case that this criterion is compatible with the desire to minimize grammar size.

In developing the LEXSYS grammar we have explored the consequences of giving the grammar writer the freedom to write a grammar that maximizes analysis quality without any regard for grammar size. In the next three sections we present detailed statistics for the current LEXSYS grammar that give an indication of what the grammar contains, its current size, and why it has grown to this size.

In order to ease the process of engineering such a large grammar, we have made use of the lexical knowledge representation language DATR (Evans & Gazdar, 1996) to compactly encode the elementary trees (Evans *et al.*, 1995; Smets & Evans, 1998). In Section 5 we present some figures that show how the size of the encoding of the grammar has increased during the grammar development process as the number and complexity of elementary trees has grown.

We have addressed problems that result from trying to parse with such a large grammar by using a technique proposed by (Evans & Weir, 1997) and (Evans & Weir, 1998) in which all the trees that each word can anchor are compactly represented using a collection of finite state automata.

In Section 6 we give some data that shows the extent to which this technique is successful in compacting the grammar.

## 2. Coverage of the LEXSYS Grammar

The LEXSYS grammar has roughly the same coverage as the Alvey NL Tools grammar (Grover *et al.*, 1993), and adopts the same set of subcategorization frames as in the Alvey lexicon. There are at present 143 families in the grammar. Each family contains the base tree of the family, and definitions of lexical rules which derive trees from the base tree. There are currently 88 lexical rules. Possible rule combinations are determined automatically (see (Smets & Evans, 1998)).

There are 7 **noun** and **pronoun** families. The noun families include trees for bare nouns, for small clauses headed by a noun, for noun-noun modifiers and for coordination. Coordination can be at the N, $\bar{\text{N}}$ or NP levels. There are 19 **adjective** families, distinguished according to the position of the adjective and its subcategorization frames. Trees derived by lexical rules include small clauses headed by an adjective, comparative constructions, trees with unbounded dependencies for adjectives which subcategorize for a complement (***wh***-questions, relative clauses, topicalization), a tree for *tough*-movement, and trees for coordination.

**Numerals** also anchor adjective trees. Rules derive from the base tree uses of numerals as pronouns and nouns, and coordination of cardinal numbers (for example, *hundred and ten*). However, the grammar does not as yet have a complete account of complex numerals. For ordinals, there are rules to derive fractions with complement, fractions without complement, and the use of ordinals as degree specifiers.

**Adverbs** are distinguished according to whether they are complements or modifiers. Modifier trees differ according to the modified category and the relative position of the adverb and its argument. Rules derive coordinated structures headed by adverbs, and also adverb distribution. Long distance dependencies possibly involving adverbs (for example, *How did he behave*) are handled in the PP modifier family.[1]

The grammar contains an account of constituent and sentential negation (but in the latter disregarding scope issues arising when an adverb comes in between the auxiliary and the negation).

**Specifier** families include families for determiners, quantifying pre-determiners and genitive determiners. There is also a family for adjective and adverb specifiers.

**Prepositions** followed by an NP are divided into two families: a family for case-marking prepositions and a family for predicative prepositions. These two types of prepositions differ in their semantic content, and syntactically also: case-marking prepositions do not head PP-modifiers. The case-marking preposition family includes trees for long-distance dependencies with preposition stranding (*wh*-questions, relative clause, *tough*-constructions) and trees for coordination. The family of predicative prepositions inherits these trees, and also contains trees for adjunct preposition phrases and long-distance dependencies involving adjunct PPs. There are also families for prepositions introducing Ss, VPs, PPs and AP. There are two families for **complementizers** (introducing an S or a VP).

The 94 **verb** families constitute the bulk of the grammar. Verb families include trees[2] for gerunds (nominal and verbal), long-distance dependencies (topicalization, relative clause and *wh*-questions), VP complements, VP complements for *tough*-constructions, small clauses (headed by a present participle or a passive verb), *for-to* clauses, extraposition, imperative, passive with or without *by*, inversion (for auxiliaries and modals), VP-ellipsis (after auxiliaries and modals), dative alternation, movement of particles, and coordination (at V, VP and S).

Finally, we have recently extended the grammar to include semantic features capturing predicate

[1] It would be redundant also to have such a rule in the adverb family.

[2] Of course, these constructions are not relevant for every single family.

argument structures. We have not implemented quantification yet. The grammar adopts a semantic representation inspired by the Minimal Recursion Semantics (MRS) framework (Copestake *et al.*, unpublished). MRS representations are flat lists of elementary predications, with relations between predications being expressed through coindexation.

## 3. Localization of Syntactic Dependencies

The LEXSYS grammar has been designed to localize syntactic dependencies, not only unbounded dependencies between filler and gap, but agreement relations, case and mode of the clause, etc. (Carroll *et al.*, 1999). One immediate advantage is that there is no need for feature percolation during parsing: all syntactic features are grounded during anchoring. There are, however, a few cases where all syntactic features cannot be localized in the same tree. This happens when the values of syntactic features are determined by more than one constituent.

This is the case, for example, in raising constructions: the subject raising verb agrees with its syntactic subject but the complement of the raising verb (adjective or verb) determines the category of the subject. In such cases, feature percolation is needed, unless one define trees for all the possible feature combinations. This is what we have done in the grammar, and 9 more trees are needed to that effect.

In *there*-constructions, the NP following the verb (*be*) determines the agreement of the verb. This does not represent a problem if the dependency is local. However, if a subject raising verb comes in between *there* and the rest of the sentence, agreement cannot be determined locally anymore. We need one more tree to cover both possible instantiations of agreement features.

Finally, PP phrases can involve a *wh*-NP or a *rel*-NP, thus must be specified as such. Because the head of PPs does not set that feature, feature percolation would be needed between the NP and the root of the PP. In the grammar, we define three PP trees, one for each possible instantiation of that feature. Thus, two more trees are needed than if we had feature percolation.

In all the above cases, the specification of all possible feature combinations does not involve the creation of many more trees. However, from a linguistic point of view, we do miss generalizations.

With coordination, however, the problem is not the loss of linguistic generalizations, but the substantial increase in the number of trees. Indeed, coordination[3] trees are anchored by the head of one of the coordinated constituents. The advantage of this is that constraints on the coordination phrase are defined at anchoring. But the disadvantage is that this doubles the number of trees in the grammar: every structure can occur in coordination.

## 4. Anchored Trees

The previous two sections discussed the coverage of the grammar, and how some decisions have increased the number of *unanchored* trees. Another important property of the grammar is the number of trees that result from *anchoring* with lexical items.

We find that some verbs induce a very large number of anchored trees: for example, *get* results in 2847 trees, *put* 3465, *come* 2656, and *turn* 1425. To illustrate why, consider *get*. First, *get* has 17 different subcategorization frames (it can be transitive, ditransitive, it can have a prepositional complement, be followed by one or more particles, etc.). It therefore belongs to 17 different families, and each family contains a number of trees (for example, the V_PP family, selected by *get*, has 33 trees, and the V_NP_PPto family contains 146 trees).

Moreover, when a *lexical item* anchors a tree, features get grounded, and different feature instantiations characterize different trees. For example, *get* can be followed by one of 12 different prepositions which means that there are at least 12 × 33 trees for the single subcategorization

[3]Only same constituent coordination has been implemented so far.

| # trees in set | # sets | # merged states (mean) | # minimized states (mean) | ratio merged / minimized |
|---|---|---|---|---|
| 1–10 | 112 | 17.9 | 6.9 | 2.6 |
| 11–20 | 83 | 53.9 | 13.1 | 4.1 |
| 21–50 | 69 | 133 | 18.1 | 7.4 |
| 51–100 | 47 | 364 | 28.1 | 13.0 |
| 101–200 | 68 | 687 | 33.0 | 20.8 |
| 201–500 | 56 | 1815 | 42.8 | 42.4 |
| 501–1000 | 23 | 3654 | 48.9 | 74.7 |
| 1001–5000 | 16 | 10912 | 60.1 | 181.5 |
| Totals | 474 | 927.7 | 23.5 | 39.4 |

Table 1: Grammar compaction statistics

frame V_PP. Similarly, there are 16 different particles which can follow *get*, and this also multiplies the number of trees.

Finally, there are other features that get instantiated and are responsible for the creation of new trees, such as agreement features of the anchor, verb form feature of the anchor and of its verbal complement. Thus the different instantiations of features together with the various subcategorization frames that a word selects explain the very high number of trees anchored by some individual words.

## 5. Encoding for Grammar Development

Following (Evans *et al.*, 1995) and (Smets & Evans, 1998) the LEXSYS grammar is encoded using DATR, a non-monotonic knowledge representation language.

In 1998, the grammar contained 620 trees organized into 44 tree families and produced using 35 rules. This grammar was encoded in 2200 DATR statements, giving an average of 3.55 DATR statements per tree. The grammar currently contains around 4000 trees in 143 families produced with 88 rules. This grammar is encoded with around 5300[4] DATR statements, giving an average of 1.325 statements per tree. Thus, as the grammar has grown the number of DATR statements needed to encode it has grown, but not as rapidly.

## 6. Encoding for Parsing

Following (Evans & Weir, 1997) and (Evans & Weir, 1998) each elementary tree is encodec as a finite state automaton that specifies an accepting traversal of the tree from anchor to root For each input word, the set of all the alternative trees that can anchor an input word can b captured in just one such automaton, which can be minimized in the standard way, and the used for parsing.

In order to assess the extent to which this technique alleviates the problem of grammar size we produced automata for the words appearing in the 1426 sentences (mean length 5.70 words forming the Alvey NL Tools grammar development test suite. Each sentence was processe by a morphological analyser, and the result was then used in conjunction with the lexicon t determine for each word in the sentence the complete set of anchored trees, feature valu being determined by the morphological analyser or lexicon as appropriate. 474 distinct sets anchored trees ('tree sets') were produced in this way, ranging in size from 1 to 3465 tree The total number of anchored trees was 24198, with a mean of 175.5 trees in each tree s

[4]We have excluded from this figure around 700 DATR statements that specify the semantics associated w elementary trees.

| # occurrences | # sets | # trees in sets (mean) | # minimized states in sets (mean) |
|---|---|---|---|
| 1 | 98 | 256 | 25.8 |
| 2–5 | 178 | 205 | 26.6 |
| 6–10 | 68 | 182 | 23.0 |
| 11–20 | 56 | 83 | 19.6 |
| 21–50 | 48 | 64 | 16.7 |
| 51–100 | 12 | 84 | 21.2 |
| 101–200 | 9 | 54 | 11.2 |
| 201–500 | 3 | 21 | 13.0 |
| 501–1000 | 2 | 5 | 6.0 |

Table 2: Occurrences of tree sets in test sentences

Before parsing, the trees in each tree set are stripped of their anchor, merged into a single automaton and minimized; at parse time the relevant automaton is retrieved and the appropriate anchoring lexical item inserted. Table 1 shows what happens when the tree sets are converted into automata and minimized, giving figures for the distribution of tree sets, mean numbers of merged and minimized states in each tree set, and ratios of numbers of merged and minimized states.

What is not clear from Table 1 is how often each of the 474 distinct tree sets occurred in the test sentences. This is shown in Table 2 which gives the numbers and mean sizes of tree sets (number of trees and minimized states) relative to the number of times they occurred in the test suite sentences. This shows that the larger tree sets tend to occur less often than small ones, and that very few of those tree sets containing more than 100 trees anchored more than 10 of the more than 8100 word tokens in the test sentences.

The results we have presented in this section appear to show that by encoding the anchoring possibilities for words with minimized automata we are able to alleviate the grammar size problem to a considerable extent.

## References

CARROLL J., NICOLOV N., SHAUMYAN O., SMETS M. & WEIR D. (1999). Parsing with an extended domain of locality. In *Proceedings of the Eighth Conference of the European Chapter of the Association for Computational Linguistics*, p. 217–224.

COPESTAKE A., FLICKINGER D., SAG I. & POLLARD C. (unpublished). Minimal recursion semantics: An introduction.

EVANS R. & GAZDAR G. (1996). DATR: A language for lexical knowledge representation. *Computational Linguistics*, **22** (2 p.), 167–216.

EVANS R., GAZDAR G. & WEIR D. (1995). Encoding lexicalized Tree Adjoining Grammars with a nonmonotonic inheritance hierarchy. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*, p. 77–84.

EVANS R. & WEIR D. (1997). Automaton-based parsing for lexicalized grammars. In *Proceedings of the Fifth International Workshop on Parsing Technologies*, p. 66–76.

EVANS R. & WEIR D. (1998). A structure-sharing parser for lexicalized grammars. In *Proceedings of the 36th Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, p. 372–378.

GROVER C., CARROLL J. & BRISCOE E. (1993). *The Alvey Natural Language Tools grammar (4th*

*release)*. Technical Report 284, Cambridge University, Computer Laboratory.

SMETS M. & EVANS R. (1998). A compact encoding of a DTG grammar. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks*.