Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars

Ted Briscoe*
University of Cambridge

John Carroll*
University of Cambridge

We describe work toward the construction of a very wide-coverage probabilistic parsing system for natural language (NL), based on LR parsing techniques. The system is intended to rank the large number of syntactic analyses produced by NL grammars according to the frequency of occurrence of the individual rules deployed in each analysis. We discuss a fully automatic procedure for constructing an LR parse table from a unification-based grammar formalism, and consider the suitability of alternative LALR(1) parse table construction methods for large grammars. The parse table is used as the basis for two parsers; a user-driven interactive system that provides a computationally tractable and labor-efficient method of supervised training of the statistical information required to drive the probabilistic parser. The latter is constructed by associating probabilities with the LR parse table directly. This technique is superior to parsers based on probabilistic lexical tagging or probabilistic context-free grammar because it allows for a more context-dependent probabilistic language model, as well as use of a more linguistically adequate grammar formalism. We compare the performance of an optimized variant of Tomita's (1987) generalized LR parsing algorithm to an (efficiently indexed and optimized) chart parser. We report promising results of a pilot study training on 150 noun definitions from the Longman Dictionary of Contemporary English *(LDOCE) and retesting on these plus a further 55 definitions. Finally, we discuss limitations of the current system and possible extensions to deal with lexical (syntactic and semantic) frequency of occurrence.*

1. Wide-Coverage Parsing of Natural Language

The task of syntactically analyzing substantial corpora of naturally occurring text and transcribed speech has become a focus of recent work. Analyzed corpora would be of great benefit in the gathering of statistical data regarding language use, for example to train speech recognition devices, in more general linguistic research, and as a first step toward robust wide-coverage semantic interpretation. The Alvey Natural Language Tools (ANLT) system is a wide-coverage lexical, morphological, and syntactic analysis system for English (Briscoe et al. 1987). Previous work has demonstrated that the ANLT system is, in principle, able to assign the correct parse to a high proportion of English noun phrases drawn from a variety of corpora. The goal of the work reported here is to develop a practical parser capable of returning probabilistically highly ranked analyses (from the usually large number of syntactically legitimate possibilities) for material drawn from a specific corpus on the basis of minimal (supervised) training and manual modification.

* University of Cambridge, Computer Laboratory, Pembroke Street, Cambridge, CB2 3QG, UK. (Tel. +44-223-334600), (ejb / jac @cl.cam.ac.uk).

The first issue to consider is what the analysis will be used for and what constraints this places on its form. The corpus analysis literature contains a variety of proposals, ranging from part-of-speech tagging to assignment of a unique, sophisticated syntactic analysis. Our eventual goal is to recover a semantically and pragmatically appropriate syntactic analysis capable of supporting semantic interpretation. Two stringent requirements follow immediately: firstly, the analyses assigned must determinately represent the syntactic relations that hold between all constituents in the input; secondly, they must be drawn from an *a priori* defined, well-formed set of possible syntactic analyses (such as the set defined by a generative grammar). Otherwise, semantic interpretation of the resultant analyses cannot be guaranteed to be (structurally) unambiguous, and the semantic operations defined (over syntactic configurations) cannot be guaranteed to match and yield an interpretation. These requirements immediately suggest that approaches that recover only lexical tags (e.g. de Rose 1988) or a syntactic analysis that is the 'closest fit' to some previously defined set of possible analyses (e.g. Sampson, Haigh, and Atwell 1989), are inadequate (taken alone).

Pioneering approaches to corpus analysis proceeded on the assumption that computationally tractable generative grammars of sufficiently general coverage could not be developed (see, for example, papers in Garside, Leech, and Sampson 1987). However, the development of wide-coverage declarative and computationally tractable grammars makes this assumption questionable. For example, the ANLT word and sentence grammar (Grover et al. 1989; Carroll and Grover 1989) consists of an English lexicon of approximately 40,000 lexemes and a 'compiled' fixed-arity term unification grammar containing around 700 phrase structure rules. Taylor, Grover, and Briscoe (1989) demonstrate that an earlier version of this grammar was capable of assigning the correct analysis to 96.8% of a corpus of 10,000 noun phrases extracted (without regard for their internal form) from a variety of corpora. However, although Taylor, Grover, and Briscoe show that the ANLT grammar has very wide coverage, they abstract away from issues of lexical idiosyncrasy by formimg equivalence classes of noun phrases and parsing a single token of each class, and they do not address the issues of 1) tuning a grammar to a particular corpus or sublanguage 2) selecting the correct analysis from the set licensed by the grammar and 3) providing reliable analyses of input outside the coverage of the grammar. Firstly, it is clear that vocabulary, idiom, and conventionalized constructions used in, say, legal language and dictionary definitions, will differ both in terms of the range and frequency of words and constructions deployed. Secondly, Church and Patil (1982) demonstrate that for a realistic grammar parsing realistic input, the set of possible analyses licensed by the grammar can be in the thousands. Finally, it is extremely unlikely that any generative grammar will ever be capable of correctly analyzing all naturally occurring input, even when tuned for a particular corpus or sublanguage (if only because of the synchronic idealization implicit in the assumption that the set of grammatical sentences of a language is well formed.)

In this paper, we describe our approach to the first and second problems and make some preliminary remarks concerning the third (far harder) problem. Our approach to grammar tuning is based on a semi-automatic parsing phase during which additions to the grammar are made manually and statistical information concerning the frequency of use of grammar rules is acquired. Using this statistical information and modified grammar, a breadth-first probabilistic parser is constructed. The latter is capable of ranking the possible parses identified by the grammar in a useful (and efficient) manner. However, (unseen) sentences whose correct analysis is outside the coverage of the grammar remain a problem. The feasibility and usefulness of our approach has been investigated in a preliminary way by analyzing a small corpus of noun definitions drawn from the *Longman Dictionary of Contemporary English* (LDOCE) (Procter 1978). This corpus was chosen because the vocabulary employed is restricted (to approximately 2,000 morphemes), average definition length is about 10 words (with a maximum of around 30), and each definition is independent, allowing us to ignore phenomena such as ellipsis. In addition, the language of definitions represents a recognizable sublanguage, allowing us to explore the task of tuning a general purpose grammar. The results reported below suggest that probabilistic information concerning the frequency of occurrence of syntactic rules correlates in a useful (though not absolute) way with the semantically and pragmatically most plausible analysis.

In Section 2, we briefly review extant work on probabilistic approaches to corpus analysis and parsing and argue the need for a more refined probabilistic model to distinguish distinct derivations. Section 3 discusses work on LR parsing of natural language and presents our technique for automatic construction of LR parsers for unification-based grammars. Section 4 presents the method and results for constructing a LALR(1) parse table for the ANLT grammar and discusses these in the light of both computational complexity and other empirical results concerning parse table size and construction time. Section 5 motivates our interactive and incremental approach to semi-automatic production of a disambiguated training corpus and describes the variant of the LR parser used for this task. Section 6 describes our implementation of a breadth-first LR parser and compares its performance empirically to a highly optimized chart parser for the same grammar, suggesting that (optimized) LR parsing is more efficient in practice for the ANLT grammar despite exponential worst case complexity results. Section 7 explains the technique we employ for deriving a probabilistic version of the LR parse table from the training corpus, and demonstrates that this leads to a more refined and parse-context–dependent probabilistic model capable of distinguishing derivations that in a probabilistic context-free model would be equally probable. Section 8 describes and presents the results of our first experiment parsing LDOCE noun definitions, and Section 9 draws some preliminary conclusions and outlines ways in which the work described should be modified and extended.

2. Probabilistic Approaches to Parsing

In the field of speech recognition, statistical techniques based on hidden Markov modeling are well established (see e.g. Holmes 1988:129f for an introduction). The two main algorithms utilized are the Viterbi (1967) algorithm and the Baum-Welch algorithm (Baum 1972). These algorithms provide polynomial solutions to the tasks of finding the most probable derivation for a given input and a stochastic regular grammar, and of performing iterative re-estimation of the parameters of a (hidden) stochastic regular grammar by considering all possible derivations over a corpus of inputs, respectively. Baker (1982) demonstrates that Baum-Welch re-estimation can be extended to context-free grammars (CFGs) in Chomsky Normal Form (CNF). Fujisaki et al. (1989) demonstrate that the Viterbi algorithm can be used in conjunction with the CYK parsing algorithm and a CFG in CNF to efficiently select the most probable derivation of a given input. Kupiec (1991) extends Baum-Welch re-estimation to arbitrary (non-CNF) CFGs. Baum-Welch re-estimation can be used with restricted or unrestricted grammars/models in the sense that some of the parameters corresponding to possible productions over a given (non-)terminal category set/set of states can be given an initial probability of zero. Unrestricted grammars/models quickly become impractical because the number of parameters requiring estimation becomes large and these algorithms are polynomial in the length of the input and number of free parameters.

Typically, in applications of Markov modeling in speech recognition, the derivation used to analyze a given input is not of interest; rather what is sought is the best (most likely) model of the input. In any application of these or similar techniques to parsing, though, the derivation selected is of prime interest. Baum (1972) proves that Baum-Welch re-estimation will converge to a local optimum in the sense that the initial probabilities will be modified to increase the likelihood of the corpus given the grammar and 'stabilize' within some threshold after a number of iterations over the training corpus. However, there is no guarantee that the global optimum will be found, and the *a priori* initial probabilities chosen are critical for convergence on useful probabilities (e.g. Lari and Young 1990). The main application of these techniques to written input has been in the robust, lexical tagging of corpora with part-of-speech labels (e.g. Garside, Leech, and Sampson 1987; de Rose 1988; Meteer, Schwartz, and Weischedel 1991; Cutting et al. 1992).

Fujisaki et al. (1989) describe a corpus analysis experiment using a probabilistic CNF CFG containing 7550 rules on a corpus of 4206 sentences (with an average sentence length of approximately 11 words). The unsupervised training process involved automatically assigning probabilities to each CF rule on the basis of their frequency of occurrence in all possible analyses of each sentence of the corpus. These probabilities were iteratively re-estimated using a variant of the Baum-Welch algorithm, and the Viterbi algorithm was used in conjunction with the CYK parsing algorithm to efficiently select the most probable analysis after training. Thus the model was restricted in that many of the possible parameters (rules) defined over the (non-)terminal category set were initially set to zero and training was used only to estimate new probabilities for a set of predefined rules. Fujisaki et al. suggest that the stable probabilities will model semantic and pragmatic constraints in the corpus, but this will only be so if these correlate with the frequency of rules in correct analyses, and also if the 'noise' in the training data created by the incorrect parses is effectively factored out. Whether this is so will depend on the number of 'false positive' examples with only incorrect analyses, the degree of heterogeneity in the training corpus, and so forth. Fujisaki et al. report some results based on testing the parser on the corpus used for training. In 72 out of 84 sentences examined, the most probable analysis was also the correct analysis. Of the remainder, 6 were false positives and did not receive a correct parse, while the other 6 did but it was not the most probable. A success rate (per sentence) of 85% is apparently impressive, but it is difficult to evaluate properly in the absence of full details concerning the nature of the corpus. For example, if the corpus contains many simple and similar constructions, unsupervised training is more likely to converge quickly on a useful set of probabilities.

Sharman, Jelinek, and Mercer (1990) conducted a similar experiment with a grammar in ID/LP format (Gazdar et al. 1985; Sharman 1989). ID/LP grammars separate the two types of information encoded in CF rules—immediate dominance and immediate precedence—into two rule types that together define a CFG. This allows probabilities concerning dominance, associated with ID rules, to be factored out from those concerning precedence, associated with LP rules. In this experiment, a supervised training regime was employed. A grammar containing 100 terminals and 16 nonterminals and initial probabilities based on the frequency of ID and LP relations was extracted from a manually parsed corpus of about one million words of text. The resulting probabilistic ID/LP grammar was used to parse 42 sentences of 30 words or less drawn from the same corpus. In addition, lexical syntactic probabilities were integrated with the probability of the ID/LP relations to rank parses. Eighteen of the parses were identical to the original manual analyses, while a further 19 were 'similar,' yielding a success rate of 88%. What is noticeable about this experiment is that the results are no better than Fujisaki et al.'s unsupervised training experiment discussed above, despite the use of supervised training and a more sophisticated grammatical model. It is likely that these differences derive from the corpus material used for training and testing, and that the results reported by Fujisaki et al. will not be achieved with all corpora.

Pereira and Schabes (1992) report an experiment using Baum-Welch re-estimation to infer a grammar and associated rule probabilities from a category set containing 15 nonterminals and 48 terminals, corresponding to the Penn Treebank lexical tagset (Santorini 1990). The training data was 770 sentences, represented as tag sequences, drawn from the treebank. They trained the system in an unsupervised mode and also in a 'semi-supervised' mode, in which the manually parsed version of the corpus was used to constrain the set of analyses used during re-estimation. In supervised training analyses were accepted if they produced bracketings consistent but not necessarily identical with those assigned manually. They demonstrate that in supervised mode, training not only converges faster but also results in a grammar in which the most probable analysis is compatible with the manually assigned analysis of further test sentences drawn from the tree bank in a much greater percentage of cases—78% as opposed to 35%. This result indicates very clearly the importance of supervised training, particularly in a context where the grammar itself is being inferred in addition to the probability of individual rules.

In our work, we are concerned to utilize the existing wide-coverage ANLT grammar; therefore, we have concentrated initially on exploring how an adequate probabilistic model can be derived for a unification-based grammar and trained in a supervised mode to effectively select useful analyses from the large space of syntactically legitimate possibilities. There are several inherent problems with probabilistic CFG (including ID/LP)-based systems. Firstly, although CFG is an adequate model of the majority of constructions occurring in natural language (Gazdar and Mellish 1989), it is clear that wide-coverage CFGs will need to be very large indeed, and this will lead to difficulties of (manual) development of consistent grammars and, possibly, to computational intractability at parse time (particularly during the already computationally expensive training phase). Secondly, associating probabilities with CF rules means that information about the probability of a rule applying at a particular point in a parse derivation is lost. This leads to complications distinguishing the probability of different derivations when the same rule can be applied several times in more than one way. Grammar 1 below is an example of a probabilistic CFG, in which each production is associated with a probability and the probabilities of all rules expanding a given nonterminal category sum to one.

Grammar 1

```
 1)   S' → S        (1.0)
 2)   S → NP VP     (1.0)
 3)   VP → Vt NP    (.4)
 4)   VP → Vi       (.6)
 5)   NP → ProNP    (.4)
 6)   NP → Det N    (.3)
 7)   NP → NP PP    (.3)
 8)   N → N N       (.3)
 9)   PP → P NP     (1.0)
10)   N → N@        (.7)
```

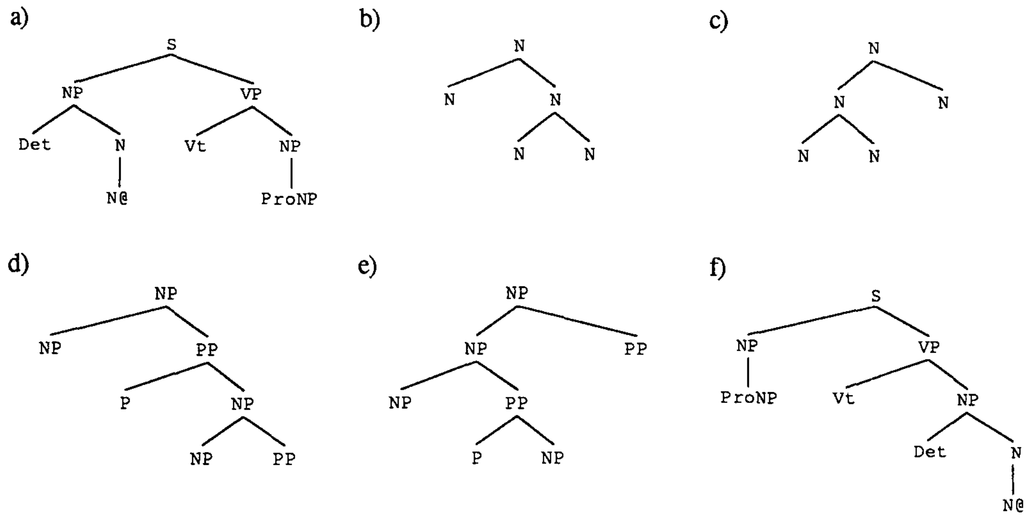

Figure 1
Probabilistic context-free derivations.

The probability of a particular parse is the product of the probabilities of each rule used in the derivation. Thus the probability of parse a) in Figure 1 is 0.0336. The probability of parse b) or c) must be identical though (0.09), because the same rule is applied twice in each case. Similarly, the probability of d) and e) is also identical (0.09) for essentially the same reason. However, these rules are natural treatments of noun compounding and prepositional phrase (PP) attachment in English, and the different derivations correlate with different interpretations. For example, b) would be an appropriate analysis for *toy coffee grinder*, while c) would be appropriate for *cat food tin*, and each of d) and e) yields one of the two possible interpretations of *the man in the park with the telescope*. We want to keep these structural configurations probabilistically distinct in case there are structurally conditioned differences in their frequency of occurrence; as would be predicted, for example, by the theory of parsing strategies (e.g. Frazier 1988). Fujisaki et al. (1989) propose a rather inelegant solution for the noun compound case, which involves creating 5582 instances of 4 morphosyntactically identical rules for classes of word forms with distinct bracketing behavior in noun–noun compounds. However, we would like to avoid enlarging the grammar and eventually to integrate probabilistic lexical information with probabilistic structural information in a more modular fashion.

Probabilistic CFGs also will not model the context dependence of rule use; for example, an NP is more likely to be expanded as a pronoun in subject position than elsewhere (e.g. Magerman and Marcus 1991), but only one global probability can be associated with the relevant CF production. Thus the probabilistic CFG model predicts (incorrectly) that a) and f) will have the same probability of occurrence. These considerations suggest that we need a technique that allows use of a more adequate grammatical formalism than CFG and a more context-dependent probabilistic model. Our approach is to use the LR parsing technique as a natural way to obtain a finite-state representation of a non-finite–state grammar incorporating information about parse context. In the following sections, we introduce the LR parser and in Section 8 we demonstrate that LR parse tables do provide an appropriate amount of contextual information to solve the problems described above.

3. LR Parsing in a Unification-Based Grammar Framework

The heart of the LR parsing technique is the parse table construction algorithm, which is the most complex and computationally expensive aspect of LR parsing. Much of the attraction of the technique stems from the fact that the real work takes place in a precompilation phase and the run time behavior of the resulting parser is relatively simple and directed. An LR parser finds the 'rightmost derivation in reverse,' for a given string and CF grammar. The precompilation process results in a parser control mechanism that enables the parser to identify the 'handle,' or appropriate substring in the input to reduce, and the appropriate rule of the grammar with which to perform the reduction. The control information is standardly encoded as a parse table with rows representing parse states, and columns terminal and nonterminal symbols of the grammar. This representation defines a finite-state automaton. Figure 2 gives the LALR(1) parse table for Grammar 1. (LALR(1) is the most commonly used variant of LR since it usually provides the best trade-off between directed rule invocation and parse table size.) If the grammar is in the appropriate LR class (a stronger restriction than being an unambiguous CFG), the automaton will be deterministic; however, some algorithms for parse table construction are also able to build nondeterministic automata containing action conflicts for ambiguous CFGs. Parse table construction is discussed further in Section 4.

Actions

State	$	Det	N@	P	ProNP	Vi	Vt
0		s3			s2		
1	r1						
2	r5			r5		r5	r5
3			s4				
4	r10		r10	r10		r10	r10
5	r6		s4	r6		r6	r6
6	r8		r8/s4	r8		r8	r8
7				s8		s13	s11
8		s3			s2		
9	r9			r9/s8		r9	r9
10	r7			r7		r7	r7
11		s3			s2		
12	r3			s8			
13	r4						
14	r2						
15	acc						

Gotos

State	N	NP	PP	S	S'	VP
0		7		1	15	
1						
2						
3	5					
4						
5	6					
6	6					
7			10			14
8		9				
9			10			
10						
11		12				
12			10			
13						
14						
15						

Figure 2
LALR(1) parse table for Grammar 1.

3.1 Creating LR Parse Tables from Unification Grammars

Tomita (1987) describes a system for nondeterministic LR parsing of context-free grammars consisting of atomic categories, in which each CF production may be augmented with a set of tests (which perform similar types of operations to those available in a unification grammar). At parse time, whenever a sequence of constituents is about to be reduced into a higher-level constituent using a production, the augmentation associated with the production is invoked to check syntactic or semantic constraints such as agreement, pass attribute values between constituents, and construct a representation of the higher-level constituent. (This is the standard approach to parsing with attribute grammars). The parser is driven by an LR parse table; however, the table is constructed solely from the CF portion of the grammar, and so none of the extra information embodied in the augmentations is taken into account during its construction. Thus the predictive power of the parser to select the appropriate rule given a specific parse history is limited to the CF portion of the grammar, which must be defined manually by the grammar writer. This requirement places a greater load on the grammar writer and is inconsistent with most recent unification-based grammar formalisms, which represent grammatical categories entirely as feature bundles (e.g. Gazdar et al. 1985; Pollard and Sag 1987; Zeevat, Calder, and Klein 1987). In addition, it violates the principle that grammatical formalisms should be declarative and defined independently of parsing procedure, since different definitions of the CF portion of the grammar will, at least, effect the efficiency of the resulting parser and might, in principle, lead to nontermination on certain inputs in a manner similar to that described by Shieber (1985).

In what follows, we will assume that the unification-based grammars we are considering are represented in the ANLT object grammar formalism (Briscoe et al. 1987). This formalism is a notational variant of Definite Clause Grammar (e.g. Pereira and Warren 1980), in which rules consist of a mother category and one or more daughter categories, defining possible phrase structure configurations. Categories consist of sets of feature name-value pairs, with the possibility of variable values, which may be bound within a rule, and of category-valued features. Categories are combined using fixed-arity term unification (Prolog-style). The results and techniques we report below should generalize to many other unification-based formalisms. An example of a possible ANLT object grammar rule is:

```
[N -, V +, BAR 2, PER x, PLU y, VFORM z] →
    [N +, V -, BAR 2, PER x, PLU y, CASE Nom]
    [N -, V +, BAR 1, PER x, PLU y, VFORM z]
```

This rule provides a (simple) analysis of the structure of English clauses, corresponding to S → NP VP, using a feature system based loosely on that of GPSG (Gazdar et al. 1985). In Tomita's LR parsing framework, each such rule must be manually converted into a rule of the following form in which some subpart of each category has been replaced by an atomic symbol.

```
Vb[BAR 2, PER x, PLU y, VFORM z] →
    Nn[BAR 2, PER x, PLU y, CASE Nom]
    Vb[BAR 1, PER x, PLU y, VFORM z]
```

However, it is not obvious which features should be so replaced—why not include BAR and CASE? It will be difficult for the grammar writer to make such substitutions in a consistent way, and still more difficult to make them in an optimal way for the purposes of LR parsing, since both steps involve consideration and comparison of all the categories mentioned in each rule of the grammar.

Constructing the LR parse table directly and automatically from a unification grammar would avoid these drawbacks. In this case, the LR parse table would be based on complex categories, with unification of complex categories taking the place of equality of atomic ones in the standard LR parse table construction algorithm (Osborne 1990; Nakazawa 1991). However, this approach is computationally prohibitively expensive: Osborne (1990:26) reports that his implementation (in HP Common Lisp on a Hewlett Packard 9000/350) takes almost 24 hours to construct the LR(0) states for a unification grammar of just 75 productions.

3.2 Constructing a CF Backbone from a Unification Grammar

Our approach, described below, not only extracts unification information from complex categories, but is computationally tractable for realistic sized grammars and also safe from inconsistency. We start with a unification grammar and automatically construct a CF 'backbone' of rules containing categories with atomic names and an associated 'residue' of feature name-value pairs. Each backbone grammar rule is generally in direct one-to-one correspondence with a single unification grammar rule. The LR parse table is then constructed from the CF backbone grammar. The parser is driven by this table, but in addition when reducing a sequence of constituents the parser performs the unifications specified in the relevant unification grammar rule to form the category representing the higher-level constituent, and the derivation fails if one of the unifications fails. Our parser is thus similar to Tomita's (1987), except that it performs unifications rather than invoking CF rule augmentations; however, the main difference between our approach and Tomita's is the way in which the CF grammar that drives the parser comes into being.

Even though a unification grammar will be, at best, equivalent to a very large (and at worst, if features are employed in recursive or cyclic ways, possibly infinite) set of atomic-category CF productions, in practice we have obtained LR parsers that perform well from backbone grammars containing only about 30% more productions than the original unification grammar. The construction method ensures that for any given grammar the CF backbone captures at least as much information as the optimal CFG that contains the same number of rules as the unification grammar. Thus the construction method guarantees that the resulting LR parser will terminate and will be as predictive as the source grammar in principle allows.

Building the backbone grammar is a two-stage process:

1. Compute the largest maximally specific set (in terms of subsumption) of disjoint categories covering the whole grammar and assign to each category a distinct atomic category name. That is:

```
initialize disjoint-set to be empty;
for each category C in grammar
  let disjoint-merge be the categories in disjoint-set
    which unify with C;
  if disjoint-merge is empty
    then add C to disjoint-set;
    else replace all elements of disjoint-merge in disjoint-set
      with the single most specific category which subsumes C
      and all categories in disjoint-merge;
assign a distinct name to each category in disjoint-set.
```

```
[N -, V +, BAR 2, PER x, PLU y, VFORM z] -->
   [N +, V -, BAR 2, PER x, PLU y, CASE Nom]
   [N -, V +, BAR 1, PER x, PLU y, VFORM z]

[N +, V -, BAR 2, PER x, PLU y, CASE c] -->
   [SUBCAT DET]
   [N +, V -, BAR 1, PER x, PLU y, CASE c]

[N -, V +, BAR 1, PER x, PLU y, VFORM z] -->
   [N -, V +, BAR 0, PER x, PLU y, VFORM z, SUBCAT INTRANS]

[N -, V +, BAR 1, PER x, PLU y, VFORM z] -->
   [N -, V +, BAR 0, PER x, PLU y, VFORM z, SUBCAT TRANS]
   [N +, V -, BAR 2, PER x, PLU y, CASE Acc]
```

Figure 3
ANLT object grammar rules.

```
{S-1:    [N -, V +, BAR 2, PER x, PLU y, VFORM z]
 NP-2:   [N +, V -, BAR 2, PER x, PLU y, CASE c]
 VP-3:   [N -, V +, BAR 1, PER x, PLU y, VFORM z]
 DET-4:  [SUBCAT DET]
 N1-5:   [N +, V -, BAR 1, PER x, PLU y, CASE c]
 V-6:    [N -, V +, BAR 0, PER x, PLU y, VFORM z, SUBCAT INTRANS]
 V-7:    [N -, V +, BAR 0, PER x, PLU y, VFORM z, SUBCAT TRANS]}
```

Figure 4
Categories in disjoint-set.

```
S-1  --> NP-2 VP-3
NP-2 --> DET-4 N1-5
VP-3 --> V-6
VP-3 --> V-7 NP-2
```

Figure 5
Backbone grammar corresponding to object grammar.

2. For each unification grammar rule, create a backbone grammar rule containing atomic categories, each atomic category being the name assigned to the category in the disjoint category set that unifies with the corresponding category in the unification grammar rule:

```
for each rule R of form C1 → C2 ... Cn in unification grammar
  add a rule B of form B1 → B2 ... Bn to backbone grammar
    where Bi is the name assigned to the (single) category in
      disjoint-set which unifies with Ci, for i=1, n.
```

For example, for the rules in Figure 3 (corresponding loosely to `S` → `NP VP`, `NP` → `Vi` and `VP` → `Vt NP`), step 1 would create the disjoint-set shown in Figure 4. (Note that the value for `CASE` on the `NP` categories in the grammar has 'collapsed' down to a

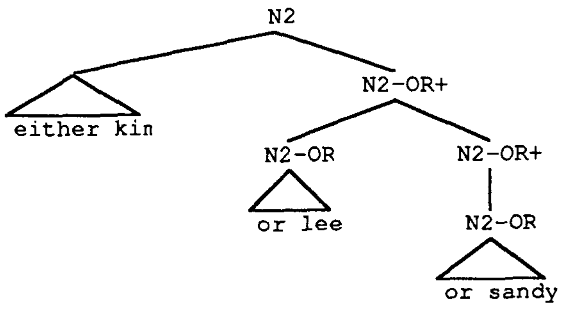

Figure 6
Backbone parse tree for *either kim or lee or sandy* using rule `N2 --> N2[CONJ EITHER], N2[CONJ OR]+`.

variable, but that the two `V` categories remain distinct). Figure 5 shows the backbone rules that would be built in step 2.

Algorithms for creating LR parse tables assume that the terminal vocabulary of the grammar is distinct from the nonterminal one, so the procedure described above will not deal properly with a unification grammar rule whose mother category is assumed elsewhere in the grammar to be a lexical category. The modification we make is to automatically associate two different atomic categories, one terminal and one nonterminal, with such categories, and to augment the backbone grammar with a unary rule expanding the nonterminal category to the terminal.

Two other aspects of the ANLT grammar formalism require further minor elaborations to the basic algorithm: firstly, a rule may introduce a gap by including the feature specification `[NULL +]` on the gapped daughter—for each such daughter an extra rule is added to the backbone grammar expanding the gap category to the null string; secondly, the formalism allows Kleene star and plus operators (Gazdar et al. 1985)—in the ANLT grammar these operators are utilized in rules for coordination. A rule containing Kleene star daughters is treated as two rules: one omitting the daughters concerned and one with the daughters being Kleene plus. A new nonterminal category is created for each distinct Kleene plus category, and two extra rules are added to the backbone grammar to form a right-branching binary tree structure for it; a parser can easily be modified to flatten this out during processing into the intended flat sequence of categories. Figure 6 gives an example of what such a backbone tree looks like. Grammars written in other, more low-level unification grammar formalisms, such as PATR-II (Shieber 1984), commonly employ treatments of the type just described to deal with phenomena such as gapping, coordination, and compounding. However, this method both allows the grammar writer to continue to use the full facilities of the ANLT formalism and allows the algorithmic derivation of an appropriate backbone grammar to support LR parsing.

The major task of the backbone grammar is to encode sufficient information (in the atomic categoried CF rules) from the unification grammar to constrain the application of the latter's rules at parse time. The nearly one-to-one mapping of unification grammar rules to backbone grammar rules described above works quite well for the ANLT grammar, with only a couple of exceptions that create spurious shift-reduce conflicts during parsing, resulting in an unacceptable degradation in performance. The phenomena concerned are coordination and unbounded dependency constructions.

In the ANLT grammar three very general rules are used to form nominal, adjectival, and prepositional phrases following a conjunction; the categories in these rules lead to otherwise disjoint categories for conjuncts being merged, giving rise to a set of overly general backbone grammar rules. For example, the rule in the ANLT grammar for forming a noun phrase conjunct introduced by a conjunction is

```
N2[CONJ @con] --> [SUBCAT @con, CONJN +], H2.
```

The variable value for the `CONJ` feature in the mother means that all `N2` categories specified for this feature (e.g. `N2[CONJ EITHER]`, `N2[CONJ NULL]`) are generalized to the same category. This results in the backbone rules, when parsing *either kim or lee helps*, being unable, after forming a `N2[CONJ EITHER]` for *either kim*, to discriminate between the alternatives of preparing to iterate this constituent (as in the phrase *kim, lee, or sandy helps* where *kim* would be `N2[CONJ NULL]`), or shifting the next word *or* to start a new constituent. We solve this problem by declaring `CONJ` to be a feature that may not have a variable value in an element of the disjoint category set. This directs the system to expand out each unification grammar rule that has a category containing this feature with a variable value into a number of rules fully specified for the feature, and to create backbone rules for each of these. There are eight possible values for `CONJ` in the grammar, so the general rule for forming a nominal conjunct given above, for example, ends up being represented by a set of eight specialized backbone grammar rules.

In the grammar, unbounded dependency constructions (UBCs) are analyzed by propagating the preposed constituent through the parse tree as the value of the `SLASH` feature, to link it with the 'gap' that appears in the constituent's normal position. All nonlexical major categories contain the feature, rules in the grammar propagating it between mother and a single daughter; other daughters are marked `[SLASH [NOSLASH +]]` indicating that the daughter is not 'gapped.' Backbone grammar construction would normally lose the information in the unification grammar about where gaps are allowed to occur, significantly degrading the performance of a parser. To carry the information over into the backbone we declare that wherever `SLASH` occurs with a variable value, the value should be expanded out into two values: `[NOSLASH +]`, and a notional value unifying with anything except `[NOSLASH +]`. We have also experimented with a smaller grammar employing 'gap threading' (e.g. Pereira and Shieber 1987), an alternative treatment of UBCs. We were able to use the same techniques for expanding out and inference on the values of the (in this case atomic) features used for threading the gaps to produce a backbone grammar (and parse table) that had the same constraining power with respect to gaps as the original grammar.

To date, we have not attempted to compute CF backbones for grammars written in formalisms with minimal phrase structure components and (almost) completely general categories, such as HPSG (Pollard and Sag 1987) and UCG (Zeevat, Calder, and Klein 1987); more extensive inference on patterns of possible unification within nested categories and appropriate expanding-out of the categories concerned would be necessary for an LR parser to work effectively. This and other areas of complexity in unification-based formalisms need further investigation before we can claim to have developed a system capable of producing a useful LR parse table for any unification-based grammar. In particular, declaring certain category-valued features so that they cannot take variable values may lead to nontermination in the backbone construction for some grammars. However, it should be possible to restrict the set of features that are considered in category-valued features in an analogous way to Shieber's (1985) restrictors for Earley's (1970) algorithm, so that a parse table can still be constructed.

4. Building LR Parse Tables for Large NL Grammars

The backbone grammar generated from the ANLT grammar is large: it contains almost 500 distinct categories and more than 1600 productions. When we construct the LALR(1) parse table, we therefore require an algorithm with practical time and space requirements. In the LR parsing literature there are essentially two approaches to constructing LALR(1) parse tables. One approach is graph-based (DeRemer and Pennello 1982), transforming the parse table construction problem to a set of well-known directed graph problems, which in turn are solvable by efficient algorithms. Unfortunately this approach does not work for grammars that are not LR(*k*) for any *k* (DeRemer and Pennello 1982:633), for example, ambiguous grammars. We therefore broadly follow the alternative approach of Aho, Sethi, and Ullman (1986), but with a number of optimizations:

1. Constructing the LR(0) sets of items: we compute LR(0) states containing only kernel items (the item `[S' --> . S]`, where `S'` is the start symbol, and all items that have a symbol to the left of the dot), since nonkernel items can be cached in a table and retrieved only if needed. Being able to partition the items in this way is especially useful with the ANLT grammar, since the mean number of kernel items in each LR(0) set is about 9, whereas the mean number of nonkernel items per state is more than 400.

2. Computing the LALR(1) lookaheads for each item: the conventional approach is to compute the LR(1) closure of each kernel item in order to determine the lookaheads that are generated *spontaneously* and those that *propagate* from other items. However, in an initial implementation we found that the LR(1) closure operation as described by Aho et al. was too expensive to be practicable for the number and size of LR(0) states we deal with, even with schemes for caching the closures of nonkernel items once they had been computed. Instead, we have moved to an algorithm devised by Kristensen and Madsen (1981), which avoids performing the LR(1) closure operation. The crucial advantage of this algorithm is the ability, at any stage in the computation, to tell whether the calculation of the lookahead set for a particular item has been completed, is underway, or has not yet started. This means that even partially computed lookahead sets can be cached (with the computation yet to be done explicitly marked), and that items whose lookahead sets are found to subsume those of others are able to just copy the results from the subsumed sets.

3. Constructing the parse table: the LALR(1) parse table is derived straightforwardly from the lookahead sets, although to keep the size of the parse table within reasonable bounds we chose appropriate data structures to represent the goto entries and shift and reduce actions. For the ANLT backbone grammar there are approximately 150,000 goto entries (nonterminal—state pairs), 440,000 shift actions (terminal—state pairs), and 670,000 reduce actions (terminal—rule-number pairs); however, of the goto entries only 2,600 are distinct and of the shift actions only 1,100 are distinct; most states contain just reduce or just shift actions, and in any one state very few different rules are involved

in reduce actions.[1] The majority of states contain just reduce or just shift actions, and in any one state very few different rules are involved in reduce actions. Taking advantage of the characteristics of this distribution, in each state we represent (in Common Lisp)

(a) a set of goto entries as a list of (nonterminal—state) conses sorted into a canonical order, list elements and tails of lists shared where possible between states,
(b) a set of shift actions as a list containing a single (large) integer (the list shared when possible between states), where if the state shifts to state *s* on lookahead *t*, the element indexed by *t* in an auxiliary array will contain *s* together with a number *n*, and bit *n* in the binary representation of the integer will be 1,
(c) a set of reduce actions as, for each rule involved, a cons whose second element is the rule number and whose first is a bit-vector (shared when possible between states) whose *n*th bit is 1 if the reduce should occur with the *n*th terminal as lookahead,
(d) an accept action as a cons with the first element being the lookahead symbol.

For the grammars we have investigated, this representation achieves a similar order of space saving to the *comb vector* representation suggested by Aho, Sethi, and Ullman (1986:244ff) for unambiguous grammars (see Klein and Martin [1989] for a survey of representation techniques). The parse table for the ANLT grammar occupies approximately 360 Kbytes of memory, and so represents each action (shift, reduce, or goto) in an average of less than 2.3 bits. In contrast to conventional techniques, though, we maintain a faithful representation of the parse table, not replacing error entries with more convenient nonerror ones in order to save extra space. Our parsers are thus able to detect failures as soon as theoretically possible, an important efficiency feature when parsing nondeterministically with ambiguous grammars, and a time-saving feature when parsing interactively with them (see next section).

Table 1 compares the size of the LALR(1) parse table for the ANLT grammar with others reported in the literature. From these figures, the ANLT grammar is more than twice the size of Tomita's (combined morphological and syntactic) grammar for Japanese (Tomita 1987:45). The grammar itself is about one order of magnitude bigger than that of a typical programming language, but the LALR(1) parse table, in terms of number of actions, is two orders of magnitude bigger. Although Tomita (1984:357) anticipates LR parsing techniques being applied to large NL grammars written in formalisms such as GPSG, the sizes of parse tables for such grammars grow more rapidly than he predicts. However, for large real-world NL grammars such as the ANLT, the table size is still quite manageable despite Johnson's (1989) worst-case complexity result of the number of LR(0) states being exponential on grammar size (leading to a parser with exponentially bad time performance). We have, therefore, not found it necessary to use Schabes' (1991a) LR-like tables (with number of states guaranteed to be polynomial even in the worst case).

1 Of the 3,710 states, 2,200 contain at least 1 action conflict, with a median of 34 conflicts per state. There are a total of 230,000 shift-reduce conflicts and 220,000 reduce-reduce conflicts, fairly uniformly distributed across the terminal lookahead symbols. In half of the latter conflicts, the rules involved have an identical number of daughters. One implication of this finding is that an approach to conflict resolution such as that of Shieber (1983) where reduce-reduce conflicts are resolved in favor of the longer reduction may not suffice to select a unique analysis for realistic NL grammars.

Table 1
Sizes of grammar and LALR(1) parse tables.

Grammar	Number of CFG rules/categories	Number of LR(0) states	Number of kernel items	Total number of actions
Pascal[2]	158 / 124	275	?	2883
Modula-2[3]	227 / 194	373	420	3238
Tomita, Japanese	800 / ?	?	?	?
ANLT (689 PS rules)	1641 / 496	3710	34836	1258451

Table 2
Timings for LALR(1) parse table construction (in seconds of CPU time on a Sparc-Server 390 running Sun Common Lisp).

Grammar	Backbone computation	LR(0) state construction	lookahead computation	parse table construction
ANLT	150	710	4200	780

As might be expected, and Table 2 illustrates, parse table construction for large grammars is CPU-intensive. As a rough guide, Grosch (1990) quotes LALR(1) table construction for a grammar for Modula-2 taking from about 5 to 50 seconds, so scaling up two orders of magnitude, our timings for the ANLT grammar fall in the expected region.

5. Interactive Incremental Deterministic Parsing

5.1 Constructing a Disambiguated Training Corpus

The major problem with attempting to employ a disambiguated training corpus is to find a way of constructing this corpus in an error-free and resource-efficient fashion. Even manual assignment of lexical categories is slow, labor-intensive, and error-prone. The greater complexity of constructing a complete parse makes the totally manual approach very unattractive, if not impractical. Sampson (1987:83) reports that it took 2 person-years to produce the 'LOB tree bank' of 50,000 words. Furthermore, in that project, no attempt was made to ensure that the analyses were well formed with respect to a generative grammar. Attempting to manually construct analyses consistent with a grammar of any size and sophistication would place an enormous additional load on the analyst. Leech and Garside (1991) discuss the problems that arise in manual parsing of corpora concerning accuracy and consistency of analyses across time and analyst, the labor-intensive nature of producing detailed analyses, and so forth. They advocate an approach in which simple 'skeleton' parses are produced by hand from previously tagged material, with checking for consistency between analysts. These skeleton analyses can then be augmented automatically with further information implicit in the lexical tags. While this approach may well be the best that can be achieved

2 Figures given by Klein and Martin (1989).
3 Grammar from Spector (1983) with optionality expanded out; statistics taken from a parse table constructed by the second author.

with fully manual techniques, it is still unsatisfactory in several respects. Firstly, the analyses are crude, while we would like to automatically parse with a grammar capable of assigning sophisticated semantically interpretable ones; but it is not clear how to train an existing grammar with such unrelated analyses. Secondly, the quality of any grammar obtained automatically from the parsed corpus is likely to be poor because of the lack of any rigorous checks on the form of the skeleton parses. Such a grammar might, in principle, be trained from the parsed corpus, but there are still likely to be small mismatches between the actual analysis assigned manually and any assigned automatically. For these reasons, we decided to attempt to produce a training corpus using the grammar that we wished ultimately to train. As long as the method employed ensured that any analysis assigned was a member of the set defined by the grammar, these problems during training should not arise.

Following our experience of constructing a substantial lexicon for the ANLT grammar from unreliable and indeterminate data (Carroll and Grover 1989), we decided to construct the disambiguated training corpus semi-automatically, restricting manual interaction to selection between alternatives defined by the ANLT grammar. One obvious technique would be to generate all possible parses with a conventional parser and to have the analyst select the correct parse from the set returned (or reject them all). However, this approach places a great load on the analyst, who will routinely need to examine large numbers of parses for given sentences. In addition, computation of all possible analyses is likely to be expensive and, in the limit, intractable.

Briscoe (1987) demonstrates that the structure of the search space in parse derivations makes a left-to-right, incremental mode of parse selection most efficient. For example, in noun compounds analyzed using a recursive binary-branching rule (N $\rightarrow$ N N) the number of analyses correlates with the Catalan series (Church and Patil, 1982),[4] so a 3-word compound has 2 analyses, 4 has 5, 5 has 14, 9 has 1430, and so forth. However, Briscoe (1987:154f) shows that with a simple bounded context parser (with one word lookahead) set up to request help whenever a parse indeterminacy arises, it is possible to select any of the 14 analyses of a 5-word compound with a maximum of 5 interactions and any of the 1430 analyses of a 9-word compound with around 13 interactions. In general, resolution of the first indeterminacy in the input will rule out approximately half the potential analyses, resolution of the next, half of the remaining ones, and so on. For 'worst case' CF ambiguities (with $O(n^3)$ complexity) this approach to parse selection appears empirically to involve numbers of interactions that increase at little more than linear rate with respect to the length of the input. It is possible to exploit this insight in two ways. One method would be to compute all possible analyses represented as a (packed) parse forest and ask the user to select between competing subanalyses that have been incorporated into a successful analysis of the input. In this way, only genuine global syntactic ambiguities would need to be considered by the user. However, the disadvantage of this approach is that it relies on a prior (and perhaps CPU-intensive) on-line computation of the full set of analyses. The second method involves incremental interaction with the parser during the parse to guide it through the search space of possibilities. This has the advantage of being guaranteed to be computationally tractable but the potential disadvantage of requiring the user to resolve many local syntactic ambiguities that will not be

4 The nth Catalan number is given by

$$C_n = \binom{2n}{n} \frac{1}{n+1}.$$

incorporated into a successful analysis. Nevertheless, using LR techniques this problem can be minimized and, because we do not wish to develop a system that must be able to compute all possible analyses (at some stage) in order to return the most plausible one, we have chosen the latter incremental method.

5.2 The Interactive LR Parsing System

The interactive incremental parsing system that we implemented asks the user for a decision at each choice point during the parse. However, to be usable in practice, such a system must avoid, as far as possible, presenting the user with spurious choices that could be ruled out either by using more of the left context or by looking at words yet to be parsed. Our approach goes some way to addressing these points, since the parser is as predictive as the backbone grammar and LR technique allow, and the LALR(1) parse table allows one word lookahead to resolve some ambiguities (although, of course, the resolution of a local ambiguity may potentially involve an unlimited amount of lookahead; e.g. Briscoe 1987:125ff). In fact, LR parsing is the most effectively predictive parsing technique for which an automatic compilation procedure is known, but this is somewhat undermined by our use of features, which will block some derivations so that the valid prefix property will no longer hold (e.g. Schabes 1991b). Extensions to the LR technique, for example those using LR-regular grammars (Culic and Cohen 1973; Bermudez 1991), might be used to further cut down on interactions; however, computation of the parse tables to drive such extended LR parsers may prove intractable for large NL grammars (Hektoen 1991).

An LR parser faces an indeterminacy when it enters a state in which there is more than one possible action, given the current lookahead. In a particular state there cannot be more than one shift or accept action, but there can be several reduce actions, each specifying a reduction with a different rule. When parsing, each shift or reduce choice must lead to a different final structure, and so the indeterminacy represents a point of syntactic ambiguity (although it may not correspond to a genuinely global syntactic ambiguity in the input, on account of the limited amount of lookahead).

In the ANLT grammar and lexicon, lexical ambiguity is at least as pervasive as structural ambiguity. A naive implementation of an interactive LR parser would ask the user the correct category for each ambiguous word as it was shifted; many open-class words are assigned upwards of twenty lexical categories by the ANLT lexicon with comparatively fine distinctions between them, so this strategy would be completely impracticable. To avoid asking the user about lexical ambiguity, we use the technique of *preterminal delaying* (Shieber 1983), in which the assignment of an atomic preterminal category to a lexical item is not made until the choice is forced by the use of a particular production in a later reduce action. After shifting an ambiguous lexical item, the parser enters a state corresponding to the union of states that would be entered on shifting the individual lexical categories. (Each union of states will in practice be small, since it being otherwise would imply that the current context was completely failing to constrain the following input). Since, in general, several unification grammar categories for a single word may be subsumed by a single atomic preterminal category, we extend Shieber's technique so that it deals with a grammar containing complex categories by associating a set of alternative analyses with each state (not just one), and letting the choice between them be forced by later reduce actions, just as with atomic preterminal categories.

In order not to overload the user with spurious choices concerning local ambiguities, the parser does not request help immediately after it reaches a parse action conflict. Instead the parser pursues each option in a limited breadth-first fashion and only requests help with analysis paths that remain active. In our current system this

Table 3
Amount of user interaction parsing *the act of putting an end to something* with the ANLT grammar using different amounts of action conflict lookahead.

action conflict lookahead	number of choices	mean number of options in each choice
none	6	2.3
1 choice	5	2.2
2 choices	3	2.0
3 choices	2	2.0
4 choices	1	2.0

type of lookahead is limited to up to four indeterminacies ahead. Such checking is cheap in terms of machine resources and very effective in cutting down both the number of choice points the user is forced to consider and also the average number of options in each one. Table 3 shows the reduction in user interaction achieved by increasing the amount of lookahead in our system. Computation of the backbone grammar generates extra rules (as previously described to deal with lexical categories used as rule mothers and daughters specified to be repeatable an indefinite number of times) that do not correspond directly to single unification grammar rules. At choice points, reductions involving these rules are not presented to the user; instead the system applies the reductions automatically, proceeding until the next shift action or choice point is reached, including these options in those presented to the user.

The final set of measures taken to reduce the amount of interaction required with the user is to ask if the phrase being parsed contains one or more gaps or instances of coordination before presenting choices involving either of these phenomena, blocking consideration of rules on the basis of the presence of particular feature-value pairs. Figure 7 shows the system parsing a phrase with a four-choice lookahead. The resulting parse tree is displayed with category aliases substituted for the actual complex categories.

The requests for manual selection of the analysis path are displayed to the analyst in as terse a manner as possible, and require knowledge of the ANLT grammar and lexicon to be resolved effectively. Figure 8 summarizes the amount of interaction required in the experiment reported below for parsing a set of 150 LDOCE noun definitions with the ANLT grammar. To date, the largest number of interactions we have observed for a single phrase is 55 for the (30-word) LDOCE definition for *youth hostel*:

> *a hostel for usu young people walking around country areas on holiday for which they pay small amounts of money to the youth hostels association or the international yha.*

Achieving the correct analysis interactively took the first author about 40 minutes (including the addition of two lexical entries). Definitions of this length will often have many hundreds or even thousands of parses; computing just the parse forest for this definition takes of the order of two hours of CPU time (on a DEC 3100 Unix workstation). Since in a more general corpus of written material the average sentence length is likely to be 30–40 words, this example illustrates clearly the problems with any approach based on *post hoc* on-line selection of the correct parse. However, using

```
Parse>>  the act of putting an end to something

Are there any gaps in this phrase?  n

Ambiguity in state 27/193 with (of putting an end to
something $) remaining in buffer. Analysis so far is the, act.
1: Shift word 'of' onto stack.
2: Reduce end 1 analyses with rule N1/N (giving category N1-9).

Which choice (1 - 2 / abort / finish)?  2

2384 msec CPU
2700 unifications, 2025 failures, 1 parse

(N2 the
    (N2 (N1 act
            (P2 (P1 of
                    (VP (V putting) (N2 an (N2 (N1 end)))
                        (P2 (P1 to (N2 something))))))))
```

Figure 7
An interactive parse.

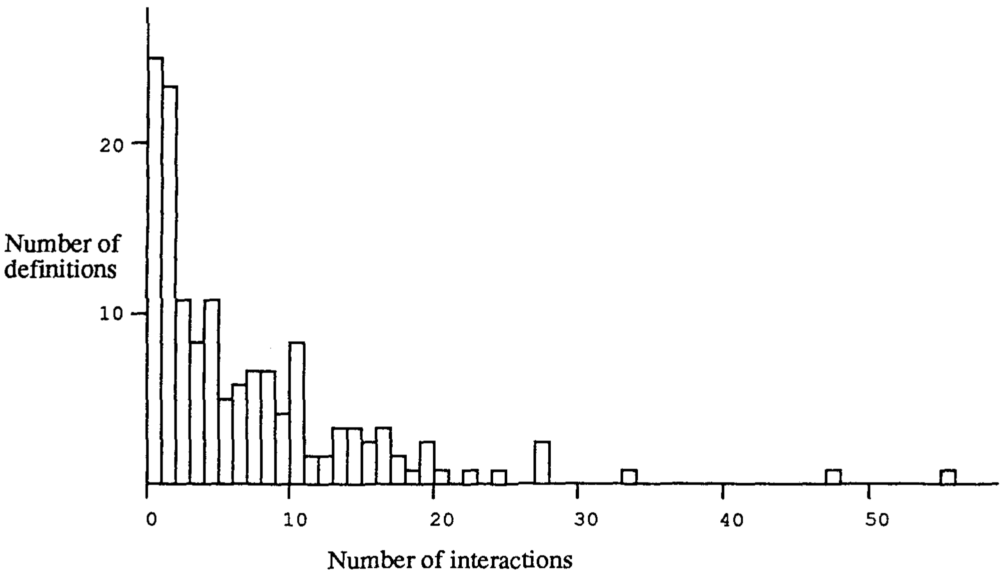

Figure 8
Numbers of definitions requiring particular amounts of interaction.

the incremental approach to semi-automatic parsing we have been able to demonstrate that the correct analysis is among this set. Furthermore, a probabilistic parser such as the one described later may well be able to compute this analysis in a tractable fashion by extracting it from the parse forest. (To date, the largest example for which we have been able to compute all analyses had approximately 2500).

The parse histories resulting from semi-automatic parsing are automatically stored and can be used to derive the probabilistic information that will guide the parser after training. We return to a discussion of the manner in which this information is utilized in Section 7.

6. Non-Deterministic LR Parsing with Unification Grammars

As well as building an interactive parsing system incorporating the ANLT grammar (described above), we have implemented a breadth-first, nondeterministic LR parser for unification grammars. This parser is integrated with the Grammar Development Environment (GDE; Carroll et al. 1988) in the ANLT system, and provided as an alternative parser for use with stable grammars for batch parsing of large bodies of text. The existing chart parser, although slower, has been retained since it is more suited to grammar development, because of the speed with which modifications to the grammar can be compiled and its better debugging facilities (Boguraev et al. 1988).

Our nondeterministic LR parser is based on Kipps' (1989) reformulation of Tomita's (1987) parsing algorithm and uses a graph-structured stack in the same way. Our parser is driven by the LALR(1) state table computed from the backbone grammar, but in addition on each reduction the parser performs the unifications appropriate to the unification grammar version of the backbone rule involved. The analysis being pursued fails if one of the unifications fails. The parser performs sub-analysis sharing (where if two or more trees have a common sub-analysis, that sub-analysis is represented only once), and local ambiguity packing (in which sub-analyses that have the same top node and cover the same input have their top nodes merged, being treated by higher level structures as a single sub-analysis). However, we generalize the technique of atomic category packing described by Tomita, driven by atomic category names, to complex feature-based categories following Alshawi (1992): the packing of sub-analyses is driven by the subsumption relationship between the feature values in their top nodes. An analysis is only packed into one that has already been found if its top node is subsumed by, or is equal to that of the one already found. An analysis, once packed, will thus never need to be unpacked during parsing (as in Tomita's system) since the value of each feature will always be uniquely determined.

Our use of local ambiguity packing does not in practice seem to result in exponentially bad performance with respect to sentence length (cf. Johnson 1989) since we have been able to generate packed parse forests for sentences of over 30 words having many thousands of parses. We have implemented a unification version of Schabes' (1991a) chart-based LR-like parser (which is polynomial in sentence length for CF grammars), but experiments with the ANLT grammar suggest that it offers no practical advantages over our Tomita-style parser, and Schabes' table construction algorithm yields less fine-grained and, therefore, less predictive parse tables. Nevertheless, searching the parse forest exhaustively to recover each distinct analysis proved computationally intractable for sentences over about 22 words in length. Wright, Wrigley, and Sharman (1991) describe a Viterbi-like algorithm for unpacking parse forests containing probabilities of (sub-)analyses to find the n-best analyses, but this approach does not generalize (except in a heuristic way) to our approach in which unification failure on the different extensions of packed nodes (resulting from differing super- or sub-

Table 4
Chart and LR parse times for the LDOCE definition *the state of being away or of not being present* with the ANLT grammar (in CPU seconds on a DEC 3100).

Parser	Parse time
GDE (bottom-up) chart parser	7.9
LR semi-automatic (with 4-choice lookahead)	6.0
LR nondeterministic	5.8

analyses) cannot be computed 'locally.' In subsequent work (Carroll and Briscoe 1992) we have developed such a heuristic technique for best-first search of the parse forest which, in practice, makes the recovery of the most probable analyses much more efficient (allowing analysis of sentences containing over 30 words).

We noticed during preliminary experiments with our unification LR parser that it was often the case that the same unifications were being performed repeatedly, even during the course of a single reduce action. The duplication was happening in cases where two or more pairs of states in the graph-structured stack had identical complex categories between them (for example due to backbone grammar ambiguity). During a reduction with a given rule, the categories between each pair of states in a backwards traversal of the stack are collected and unified with the appropriate daughters of the rule. Identical categories appearing here between traversed pairs of states leads to duplication of unifications. By caching unification results we eliminated this wasted effort and improved the initially poor performance of the parser by a factor of about three.

As for actual parse times, Table 4 compares those for the GDE chart parser, the semi-automatic, user-directed LR parser, and the nondeterministic LR parser. Our general experience is that although the nondeterministic LR parser is only around 30–50% faster than the chart parser, it often generates as little as a third the amount of garbage. (The relatively modest speed advantage compared with the substantial space saving appears to be due to the larger overheads involved in LR parsing). Efficient use of space is obviously an important factor for practical parsing of long and ambiguous texts.

7. LR Parsing with Probabilistic Disambiguation

Several researchers (Wright and Wrigley 1989; Wright 1990; Ng and Tomita 1991; Wright, Wrigley, and Sharman 1991) have proposed using LR parsers as a practical method of parsing with a probabilistic context-free grammar. This approach assumes that probabilities are already associated with a CFG and describes techniques for distributing those probabilities around the LR parse table in such a way that a probabilistic ranking of alternative analyses can be computed quickly at parse time, and probabilities assigned to analyses will be identical to those defined by the original probabilistic CFG. However, our method of constructing the training corpus allows us to associate probabilities with an LR parse table directly, rather than simply with rules of the grammar. An LR parse state encodes information about the left and right context of the current parse. Deriving probabilities relative to the parse context will allow the probabilistic parser to distinguish situations in which identical rules reapply in different ways across different derivations or apply with differing probabilities in different contexts.

Semi-automatic parsing of the training corpus yields a set of LR parse histories that are used to construct the probabilistic version of the LALR(1) parse table. The parse table is a nondeterministic finite-state automaton so it is possible to apply Markov modeling techniques to the parse table (in a way analogous to their application to lexical tagging or CFGs). Each row of the parse table corresponds to the possible transitions out of the state represented by that row, and each transition is associated with a particular lookahead item and a parse action. Nondeterminism arises when more than one action, and hence transition, is possible given a particular lookahead item. The most straightforward technique for associating probabilities with the parse table is to assign a probability to each action in the action part of the table (e.g. Wright 1990).[5] If probabilities are associated directly with the parse table rather than derived from a probabilistic CFG or equivalent global pairing of probabilities to rules, then the resulting probabilistic model will be more sensitive to parse context. For example, in a derivation for the sentence *he loves her* using Grammar 1, the distinction between reducing the first pronoun and second pronoun to `NP`—using rule 5 (`NP --> ProNP`)—can be maintained in terms of the different lookahead items paired with the reduce actions relating to this rule (in state 5 of the parse table in Figure 2); in the first case, the lookahead item will be `Vi`, and in the second `$` (the end of sentence marker). However, this approach does not make maximal use of the context encoded into a transition in the parse table, and it is possible to devise situations in which the reduction of a pronoun in subject position and elsewhere would be indistinguishable in terms of lookahead alone; for example, if we added appropriate rules for adverbs to Grammar 1, then this reduction would be possible with lookahead `Adv` in sentences such as *he passionately loves her* and *he loves her passionately*.

A slightly less obvious approach is to further subdivide reduce actions according to the state reached after the reduce action has applied. This state is used together with the resultant nonterminal to define the state transition in the goto part of the parse table. Thus, this move corresponds to associating probabilities with transitions in the automaton rather than with actions in the action part of the table. For example, a reduction of pronoun to `NP` in subject position in the parse table for Grammar 1 in Figure 2 always results in the parser returning to state 0 (from which the goto table deterministically prescribes a transition to state 7 with nonterminal `NP`). Reduction to `NP` of a pronoun in object position always results in the parser returning to state 11. Thus training on a corpus with more subject than nonsubject pronominal `NP`s will now result in a probabilistic preference for reductions that return to 'pre-subject' states with 'post-subject' lookaheads. Of course, this does not mean that it will be impossible to devise grammars in which reductions cannot be kept distinct that might, in principle, have different frequencies of occurrence. However, this approach appears to be the natural stochastic, probabilistic model that emerges when using a LALR(1) table. Any further sensitivity to context would require sensitivity to patterns in larger sections of a parse derivation than can be defined in terms of such a table.

The probabilities required to create the probabilistic version of the parse table can be derived from the set of parse histories resulting from the training phase described in Section 5, by computing the frequency with which each transition from a particular state has been taken and converting these to probabilities such that the probabilities

5 In our implementation, the probabilities are actually stored separately from the parse table to ensure that otherwise-sharable transitions in the table can still be represented compactly even if their probabilities differ.

State	$	Det	N@	P	ProNP	Vi	Vt
0		s3 (.50)			s2 (.50)		
1	r1 (0 .83)						
2	r5			r5 (8 .33)		r5	r5 (0 .50)
3			s4 (1.00)				
4	r10 (3 .11 6 .11)		r10 (3 .17 5 .22)	r10 (3 .11)		r10 (3 .11 5 .11)	r10
5	r6 (8 .13 11 .13)		s4 (.33)	r6 (11 .13)		r6 (0 .20)	r6
6	r8 (3 .17 5 .17)		r8 (3 .25) s4 (.17)	r8		r8 (3 .17)	r8
7				s8		s13 (.43)	s11 (.43)
8		s3 (.50)			s2 (.50)		
9	r9 (12 .40)			r9 (12 .40) s8		r9	r9
10	r7 (11 .40)			r7 (11 .40)		r7	r7
11		s3 (.75)			s2		
12	r3 (7 .43)			s8 (.43)			
13	r4 (7 .75)						
14	r2 (0 .84)						
15	acc (1.00)						

Figure 9
A probabilistic version of the parse table for Grammar 1.

assigned to each transition from a given state sum to one. In Figure 9 we show a probabilistic LALR(1) parse table for Grammar 1 derived from a simple, partial (and artificial) training phase. In this version of the table a probability is associated with each shift action in the standard way, but separate probabilities are associated with reduce

```
                0) 0                     (Det)
                1) 0 Det 3               (N@1)   .50
                2) 0 Det 3 N@ 4          (N@2) 1.00
                3) 0 Det 3 N             (N@2)
                4) 0 Det 3 N 5           (N@2)   .17
                5) 0 Det 3 N 5 N@ 4      (N@3)   .33
                6) 0 Det 3 N 5 N         (N@3)
                7) 0 Det 3 N 5 N 6       (N@3)   .22

 8a) 0 Det 3 N 5 N 6 N@ 4  (Vi)   .17      8b) 0 Det 3 N            (N@3)
 9a) 0 Det 3 N 5 N 6 N     (Vi)            9b) 0 Det 3 N 5          (N@3)   .25
10a) 0 Det 3 N 5 N 6 N 6   (Vi)   .06     10b) 0 Det 3 N 5 N@ 4     (Vi)    .33
11a) 0 Det 3 N 5 N         (Vi)           11b) 0 Det 3 N 5 N        (Vi)
12a) 0 Det 3 N 5 N  6      (Vi)   .08     12b) 0 Det 3 N 5 N 6      (Vi)    .11
13a) 0 Det 3 N             (Vi)           13b) 0 Det 3 N            (Vi)

               14) 0 Det 3 N 5           (Vi)    .17
               15) 0 NP                  (Vi)
               16) 0 NP 7                (Vi)    .20
               17) 0 NP 7 Vi 13          ($)     .43
               18) 0 NP 7 VP             ($)
               19) 0 NP 7 VP 14          ($)     .75
               20) 0 S                   ($)
               21) 0 S 1                 ($)     .83
               22) 0 S'                  ($)
               23) 0 S' 15               ($)    1.00
```

Figure 10
Parse derivations for *the winter holiday camp closed.*

actions, depending on the state reached after the action; for example, in state 4 with lookahead N@ the probability of reducing with rule 10 is 0.17 if the state reached is 3 and 0.22 if the state reached is 5. The actions that have no associated probabilities are ones that have not been utilized during the training phase; each is assigned a smoothed probability that is the reciprocal of the result of adding one to the total number of observations of actions actually taken in that state. Differential probabilities are thus assigned to unseen events in a manner analogous to the Good-Turing technique. For this reason, the explicit probabilities for each row add up to less than one. The goto part of the table is not shown because it is always deterministic and, therefore, we do not associate probabilities with goto transitions.

The difference between our approach and one based on probabilistic CFG can be brought out by considering various probabilistic derivations using the probabilistic parse table for Grammar 1. Assuming that we are using probabilities simply to rank parses, we can compute the total probability of an analysis by multiplying together the probabilities of each transition we take during its derivation. In Figure 10, we give the two possible complete derivations for a sentence such as *the winter holiday camp closed* consisting of a determiner, three nouns, and an intransitive verb. The ambiguity concerns whether the noun compound is left- or right-branching, and, as we saw in Section 2, a probabilistic CFG cannot distinguish these two derivations. The probability of each step can be read off the action table and is shown after the lookahead item in the figure.

In step 8 a shift-reduce conflict occurs so the stack 'splits' while the left- and right-branching analyses of the noun compound are constructed. The a) branch corresponds to the right-branching derivation and the product of the probabilities is 4.6×10^{-8}, while the product for the left-branching b) derivation is 5.1×10^{-7}. Since the table was constructed from parse histories with a preponderance of left-branching structures this is the desired result. In practice, this technique is able to distinguish and train accurately on 3 of the 5 possible structures for a 4-word noun–noun compound; but it inaccurately prefers a completely left-branching analysis over structures of the form $((n\ n)(n\ n))$ and $((n\ (nn))\ n)$. Once we move to 5-word noun–noun compounds, performance degrades further. However, this level of performance on such structural configurations is likely to be adequate, because correct resolution of most ambiguity in such constructions is likely to be dominated by the actual lexical items that occur in individual texts. Nevertheless, if there are systematic structural tendencies evident in corpora (for example, Frazier's [1988] parsing strategies predict a preference for left-branching analyses of such compounds), then the probabilistic model is sensitive enough to discriminate them.[6]

In practice, we take the geometric mean of the probabilities rather than their product to rank parse derivations. Otherwise, it would be difficult to prevent the system from always developing a bias in favor of analyses involving fewer rules or equivalently 'smaller' trees, almost regardless of the training material. Of course, the need for this step reflects the fact that, although the model is more context-dependent than probabilistic CFG, it is by no means a perfect probabilistic model of NL.[7] For example, the stochastic nature of the model and the fact that the entire left context of a parse derivation is not encoded in LR state information means that the probabilistic model cannot take account of, say, the pattern of resolution of earlier conflicts in the current derivation. Another respect in which the model is approximate is that we are associating probabilities with the context-free backbone of the unification grammar. Successful unification of features at parse time does not affect the probability of a (partial) analysis, while unification failure, in effect, sets the probability of any such analysis to zero. As long as we only use the probabilistic model to rank successful analyses, this is not particularly problematic. However, parser control regimes that attempt some form of best-first search using probabilistic information associated with transitions might not yield the desired result given this property. For example, it is not possible to use Viterbi-style optimization of search for the maximally probable parse because this derivation may contain a sub-analysis that will be pruned locally before a subsequent unification failure renders the current most probable analysis impossible.

In general, the current breadth-first probabilistic parser is more efficient than its nonprobabilistic counterpart described in the previous section. In contrast to the parser described by Ng and Tomita (1991), our probabilistic parser is able to merge (state and stack) configurations and in all cases still maintain a full record of all the probabilities computed up to that point, since it associates probabilities with partial analyses of the input so far rather than with nodes in the graph-structured stack. We are currently

6 Although we define our probabilistic model relative to the LR parsing technique, it is likely that there is an equivalent encoding in purely grammatical terms. In general our approach corresponds to making the probability of rule application conditional on other rules having applied during the parse derivation (e.g. Magerman and Marcus 1991) and the lexical category of the next word; for example, it would be possible to create a grammatical representation of the probabilistic model that emerges from a LR(0) table by assigning a set of probabilities associated with rule numbers to each right-hand side category in each rule of a CFG that would encode the probability of a rule being used to expand that category in that context.

7 Magerman and Marcus (1991) argue that it is reasonable to use the geometric mean when computing the probability of two or more sub-analyses because the independence assumptions that motivate using products do not hold for such an approximate model. In Carroll and Briscoe (1992) we present a more motivated technique for normalizing the probability of competing sub-analyses in the parse forest.

experimenting with techniques for probabilistically unpacking the packed parse forest to recover the first few most probable derivations without the need for exhaustive search or full expansion.

8. Parsing LDOCE Noun Definitions

In order to test the techniques and ideas described in previous sections, we undertook a preliminary experiment using a subset of LDOCE noun definitions as our test corpus. (The reasons for choosing this corpus are discussed in the introduction.) A corpus of approximately 32,000 noun definitions was created from LDOCE by extracting the definition fields and normalizing the definitions to remove punctuation, font control information, and so forth.[8] A lexicon was created for this corpus by extracting the appropriate lemmas and matching these against entries in the ANLT lexicon. The 10,600 resultant entries were loaded into the ANLT morphological system (Ritchie et al. 1987) and this sublexicon and the full ANLT grammar formed the starting point for the training process.

A total of 246 definitions, selected without regard for their syntactic form, were parsed semi-automatically using the parser described in Section 5. During this process, further rules and lexical entries were created for some definitions that failed to parse. Of the total number, 150 were successfully parsed and 63 lexical entries and 14 rules were added. Some of the rules required reflected general inadequacies in the ANLT grammar; for example, we added rules to deal with new partitives and prepositional phrase and verb complementation. However, 7 of these rules cover relatively idiosyncratic properties of the definition sublanguage; for example, the postmodification of pronouns by relative clause and prepositional phrase in definitions beginning *something that...*, *that of...*, parenthetical phrases headed by adverbs, such as *the period... esp the period*, and coordinations without explicit conjunctions ending with *etc.*, and so forth. Further special rules will be required to deal with brackets in definitions to cover conventions such as *a man (monk) or woman (nun) who lives in a monastery*, which we ignored for this test. Nevertheless, the number of new rules required is not great and the need for most was identified very early in the training process. Lexical entries are more problematic, since there is little sign that the number of new entries required will tail off. However, many of the entries required reflect systematic inadequacies in the ANLT lexicon rather than idiosyncrasies of the corpus. It took approximately one person-month to produce this training corpus. As a rough guide, it takes an average of 15 seconds to resolve a single interaction with the parser. However, the time a parse takes can often be lengthened by incorrect choices (and the consequent need to back up manually) and by the process of adding lexical entries and occasional rules.

The resultant parse histories were used to construct the probabilistic parser (as described in the previous section). This parser was then used to reparse the training corpus, and the most highly ranked analyses were automatically compared with the original parse histories. We have been able to reparse in a breadth-first fashion all but 3 of the 150 definitions that were parsed manually.[9] (These three are each over

8 The corpus contains about 17,000 unique headwords and 13,500 distinct word forms in the definitions. Its perplexity (PP) measures based on bigram and trigram word models and an estimate of an infinite model were PP(2) = 104, PP(3) = 41, and PP(inf) = 8 (Sharman 1991).

9 The results we report here are from using the latest versions of the ANLT grammar and LR parsing system. Briscoe and Carroll (1991) report an earlier version of this experiment using different versions of the grammar and parser in which results differed in minor ways. Carroll and Briscoe (1992) report a third version of the experiment in which results were improved slightly through the use of a better normalization and parse forest unpacking technique.

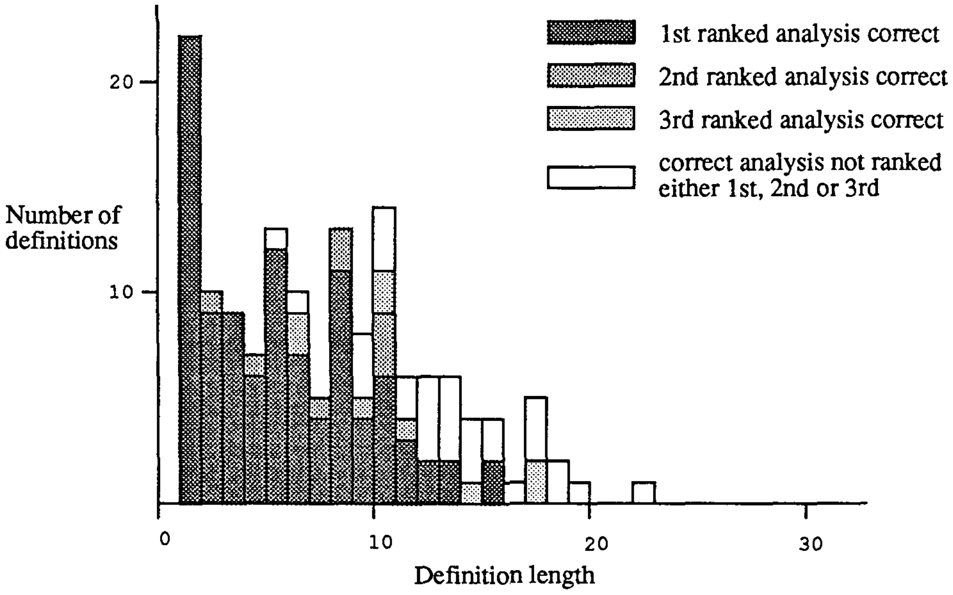

Figure 11
Correctness of results for reparsed definitions with respect to length.

25 words in length.) There are 22 definitions one word in length: all of these trivially receive correct analyses. There are 89 definitions between two and ten words in length inclusive (mean length 6.2). Of these, in 68 cases the correct analysis (as defined by the training corpus) is also the most highly ranked. In 13 of the 21 remaining cases the correct analysis is the second or third most highly ranked analysis. Looking at these 21 cases in more detail, in 8 there is an inappropriate structural preference for 'low' or 'local' attachment (see Kimball 1973), in 4, an inappropriate preference for compounds, and in 6 of the remaining 9 cases, the highest ranked result contains a misanalysis of a single constituent two or three words in length. If these results are interpreted in terms of a goodness of fit measure such as that of Sampson, Haigh, and Atwell (1989), the measure would be better than 96%. If we take correct parse/sentence as our measure then the result is 76%. For definitions longer than 10 words this latter figure tails off, mainly due to misapplication of such statistically induced, but nevertheless structural, attachment preferences. Figure 11 summarizes these results.

We also parsed a further 55 LDOCE noun definitions not drawn from the training corpus, each containing up to 10 words (mean length 5.7). Of these, in 41 cases the correct parse is the most highly ranked, in 6 cases it is the second or third most highly ranked, and in the remaining 8 cases it is not in the first three analyses. This yields a correct parse/sentence measure of 75%. Examination of the failures again reveals that a preference for local attachment of postmodifiers accounts for 5 cases, a preference for compounds for 1, and the misanalysis of a single constituent for 2. The others are mostly caused by the lack of lexical entries with appropriate SUBCAT features. In Figure 12 we show the analysis for the unseen definition of *affectation*, which has 20 parses of which the most highly ranked is correct.

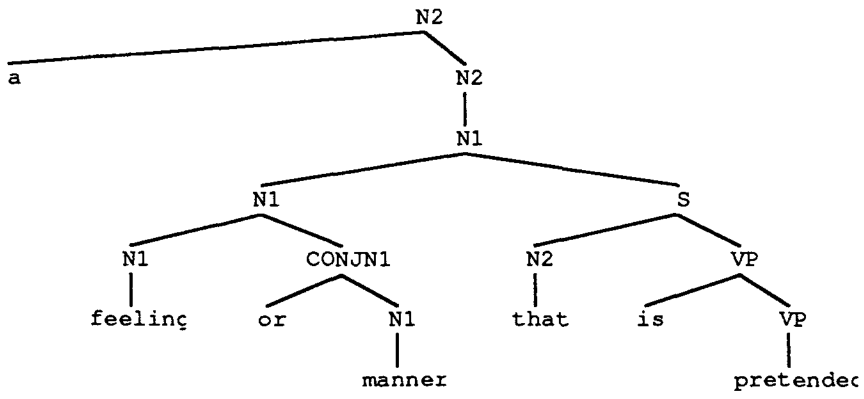

Figure 12
Parse tree for *a feeling or manner that is pretended.*

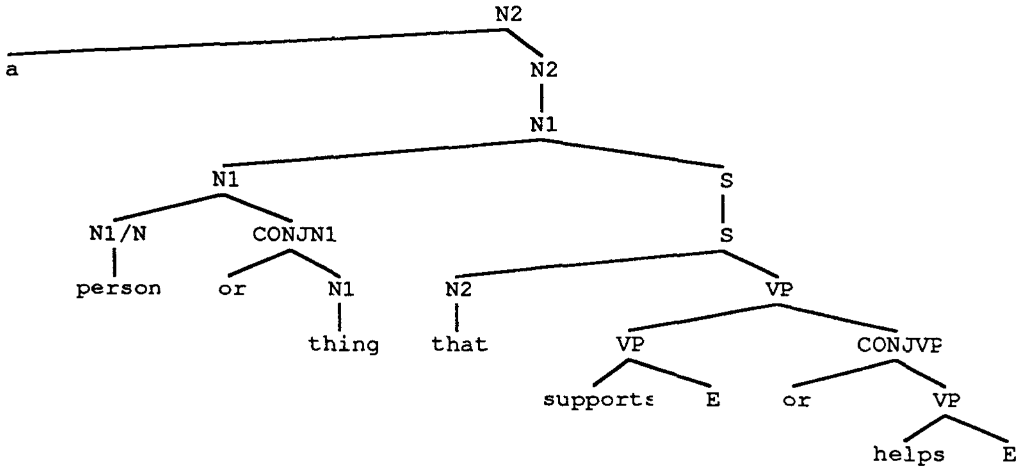

Figure 13
Parse tree for *a person or thing that supports or helps.*

Figure 13 shows the highest-ranked analysis assigned to one definition of *aid*. This is an example of a false positive which, in this case, is caused by the lack of a lexical entry for *support* as an intransitive verb. Consequently, the parser finds, and ranks highest, an analysis in which *supports* and *helps* are treated as transitive verbs forming verb phrases with object NP gaps, and *that supports or helps* as a zero relative clause with *that* analyzed as a prenominal subject—compare *a person or thing that* **that** *supports or helps*. It is difficult to fault this analysis and the same is true for the other false positives we have looked at. Such false positives present the biggest challenge to the type of system we are attempting to develop. One hopeful sign is that the analyses assigned such examples appear to have low probabilities relative to most probable correct analyses of other examples. However, considerably more data will be required before we can decide whether this trend is robust enough to provide the basis for automatic identification of false positives.

Using a manually disambiguated training corpus and manually tuned grammar appears feasible with the definitions sublanguage. Results comparable to those obtained by Fujisaki et al. (1989) and Sharman, Jelinek, and Mercer (1990) are possible on the basis of a quite modest amount of manual effort and a very much smaller training corpus, because the parse histories contain little 'noise' and usefully reflect the semantically and pragmatically appropriate analysis in the training corpus, and because the number of failures of coverage were reduced to some extent by adding the rules specifically motivated by the training corpus. Unlike Fujisaki et al. or Sharman, Jelinek, and Mercer, we did not integrate information about lexemes into the rule probabilities or make use of lexical syntactic probability. It seems likely that the structural preference for local attachment might be overruled in appropriate contexts if lexeme (or better, word sense) information were taken into account. The slightly worse results (relative to mean definition length) obtained for the unseen data appear to be caused more by the nonexistence of a correct analysis in a number of cases, rather than by a marked decline in the usefulness of the rule probabilities. This again highlights the need to deal effectively with examples outside the coverage of the grammar.

9. Conclusions and Further Work

The system that we have developed offers partial and practical solutions to two of the three problems of corpus analysis we identified in the introduction. The problem of tuning an existing grammar to a particular corpus or sublanguage is addressed partly by manual extensions to the grammar and lexicon during the semi-automatic training phase and partly by use of statistical information regarding frequency of rule use gathered during this phase. The results of the experiment reported in the last section suggest that syntactic peculiarities of a sublanguage or corpus surface quite rapidly, so that manual additions to the grammar during the training phase are practical. However, lexical idiosyncrasies are far less likely to be exhausted during the training phase, suggesting that it will be necessary to develop an automatic method of dealing with them. In addition, the current system does not take account of differing frequencies of occurrence of lexical entries; for example, in the LOB corpus the verb *believe* occurs with a finite sentential complement in 90% of citations, although it is grammatical with at least five further patterns of complementation. This type of lexical information, which will very likely vary between sublanguages, should be integrated into the probabilistic model. This will be straightforward in terms of the model, since it merely involves associating probabilities with each distinct lexical entry for a lexeme and carrying these forward in the computation of the likelihood of each parse. However, the acquisition of the statistical information from which these probabilities can be derived is more problematic. Existing lexical taggers are unable to assign tags that reliably encode subcategorization information. It seems likely that automatic acquisition of such information must await successful techniques for robust parsing of, at least, phrases in corpora (though Brent [1991] claims to be able to recognize some subcategorization patterns using large quantities of untagged text).

The task of selecting the correct analysis from the set licensed by the grammar is also partially solved by the system. It is clear from the results of the preliminary experiment reported in the previous section that it is possible to make the semantically and pragmatically correct analysis highly ranked, and even most highly ranked in many cases, just by exploiting the frequency of occurrence of the syntactic rules in the training data. However, it is also clear that this approach will not succeed in all cases; for example, in the experiment the system appears to have developed a preference for local attachment of prepositional phrases (PPs), which is inappropriate in a significant number of cases. It is not surprising that probabilities based solely on the frequency of syntactic rules are not capable of resolving this type of ambiguity; in an example such as *John saw the man on Monday again* it is the temporal interpretation of *Monday* that favors the adverbial interpretation (and thus nonlocal attachment). Such examples are syntactically identical to ones such as *John saw the man on the bus again,* in which the possibility of a locative interpretation creates a mild preference for the adjectival reading and local attachment. To select the correct analysis in such cases it will be necessary to integrate information concerning word sense collocations into the probabilistic analysis. In this case, we are interested in collocations between the head of a PP complement, a preposition and the head of the phrase being postmodified. In general, these words will not be adjacent in the text, so it will not be possible to use existing approaches unmodified (e.g. Church and Hanks 1989), because these apply to adjacent words in unanalyzed text. Hindle and Rooth (1991) report good results using a mutual information measure of collocation applied within such a structurally defined context, and their approach should carry over to our framework straightforwardly.

One way of integrating 'structural' collocational information into the system presented above would be to make use of the semantic component of the (ANLT) grammar. This component pairs logical forms with each distinct syntactic analysis that represent, among other things, the predicate-argument structure of the input. In the resolution of PP attachment and similar ambiguities, it is 'collocation' at this level of representation that appears to be most relevant. Integrating a probabilistic ranking of the resultant logical forms with the probabilistic ranking of the distinct syntactic analyses presents no problems, in principle. However, once again, the acquisition of the relevant statistical information will be difficult, because it will require considerable quantities of analyzed text as training material. One way to ameliorate the problem might be to reduce the size of the 'vocabulary' for which statistics need to be gathered by replacing lexical items with their superordinate terms (or a disjunction of such terms in the case of ambiguity). Copestake (1990, 1992) describes a program capable of extracting the genus term of a definition from an LDOCE definition, resolving the sense of such terms, and constructing hierarchical taxonomies of the resulting word senses. Taxonomies of this form might be used to replace PP complement heads and postmodified heads in corpus data with a smaller number of superordinate concepts. This would make the statistical data concerning trigrams of head–preposition–head less sparse (cf. Gale and Church 1990) and easier to gather from a corpus. Nevertheless, it will only be possible to gather such data from determinately syntactically analyzed material.

The third problem of dealing usefully with examples outside the coverage of the grammar even after training is not addressed by the system we have developed. Nevertheless, the results of the preliminary experiment for unseen examples indicate that it is a significant problem, at least with respect to lexical entries. A large part of the problem with such examples is identifying them automatically. Some such examples will not receive any parse and will, therefore, be easy to spot. Many, though, will receive incorrect parses (one of which will be automatically ranked as the most probable) and can, therefore, only be identified manually (or perhaps on the basis of relative improbability). Jensen et al. (1983) describe an approach to parsing such examples based on parse 'fitting' or rule 'relaxation' to deal with 'ill-formed' input. An approach of this type might work with input that receives no parse, but cannot help with the identification of those that only receive an incorrect one. In addition, it involves annotating each grammar rule about what should be relaxed and requires that semantic interpretation can be extended to 'fitted' or partial parses (e.g. Pollack and Pereira 1988).

Sampson, Haigh, and Atwell (1989) propose a more thorough-going probabilistic approach in which the parser uses a statistically defined measure of 'closest fit' to the set of analyses contained in a 'tree bank' of training data to assign an analysis. This approach attempts to ensure that analyses of new data will conform as closely as possible to existing ones, but does not require that analyses assigned are well formed with respect to any given generative grammar implicit in the tree bank analyses. Sampson, Haigh, and Atwell report some preliminary results for a parser of this type that uses the technique of simulated annealing to assign the closest fitting analysis on the basis of initial training on the LOB treebank and automatic updating of its statistical data on the basis of further parsed examples. Sampson, Haigh, and Atwell give their results in terms of a similarity measure with respect to correct analyses assigned by hand. For a 13-sentence sample the mean similarity measure was 80%, and only one example received a fully correct analysis. These results suggest that the technique is not reliable enough for practical corpus analysis, to date. In addition, the analyses assigned, on the basis of the LOB treebank scheme, are not syntactically determinate (for example, syntactic relations in unbounded dependency constructions are not represented).

A more promising approach with similar potential robustness would be to infer a probabilistic grammar using Baum-Welch re-estimation from a given training corpus and predefined category set, following Lari and Young (1990) and Pereira and Schabes (1992). This approach has the advantage that the resulting grammar defines a well-defined set of analyses for which rules of compositional interpretation might be developed. However, the technique is limited in several ways; firstly, such grammars are restricted to small (maximum about 15 nonterminal) CNF CFGs because of the computational cost of iterative re-estimation with an algorithm polynomial in sentence length and nonterminal category size; and secondly, because some form of supervised training will be essential if the analyses assigned by the grammar are to be linguistically motivated. Immediate prospects for applying such techniques to realistic NL grammars do not seem promising—the ANLT backbone grammar discussed in Section 4 contains almost 500 categories. However, Briscoe and Waegner (1992) describe an experiment in which, firstly, Baum-Welch re-estimation was used in conjunction with other more linguistically motivated constraints on the class of grammars that could be inferred, such as 'headedness'; and secondly, initial probabilities were heavily biased in favor of manually coded, linguistically highly plausible rules. This approach resulted in a simple tag sequence grammar often able to assign coherent and semantically/pragmatically plausible analyses to tag sequences drawn from the Spoken English Corpus. By combining such techniques and relaxing the CNF constraint, for example, by adopting the trellis algorithm version of Baum-Welch re-estimation (Kupiec 1991), it might be possible to create a computationally tractable system operating with a realistic NL grammar that would only infer a new rule from a finite space of linguistically motivated possibilities in the face of parse failure or improbability. In the shorter term, such techniques combined with simple tag sequence grammars might yield robust phrase-level 'skeleton' parsers that could be used as corpus analysis tools.

The utility of the system reported here would be considerably improved by a more tractable approach to probabilistically unpacking the packed parse forest than exhaustive search. Finding the n-best analyses would allow us to recover analyses for longer sentences where a parse forest is constructed and would make the approach generally more efficient. Carroll and Briscoe (1992) present a heuristic algorithm for parse forest unpacking that interleaves normalization of competing sub-analyses with best-first extraction of the n most probable analyses. Normalization of competing sub-analyses with respect to the longest derivation both allows us to prune the search probabilistically and to treat the probability of analyses as the product of the probability of their sub-analyses, without biasing the system in favor of shorter derivations. This modified version of the system presented here is able to return analyses for sentences over 31 words in length, yields slightly better results on a replication of the experiment reported in Section 8, and the resultant parser is approximately three times faster at returning the three highest-ranked parsers than that presented here.

In conclusion, the main positive points of the paper are that 1) LR parse tables can be used to define a more context-dependent and adequate probabilistic model of NL, 2) predictive LR parse tables can be constructed automatically from unification-based grammars in standard notation, 3) effective parse table construction and representation techniques can be defined for realistically sized ambiguous NL grammars, 4) semi-automatic LR based parse techniques can be used to efficiently construct training corpora, and 5) the LR parser and ANLT grammar jointly define a useful probabilistic model into which probabilities concerning lexical subcategorization and structurally defined word sense collocations could be integrated.

Acknowledgments

This research is supported by SERC/DTI-IED project 4/1/1261 'Extensions to the Alvey Natural Language Tools' and by ESPRIT BRA 3030 'Acquisition of Lexical Information from Machine-Readable Dictionaries.' We would like to thank Longman Group Ltd. for allowing us access to the LDOCE MRD and Ann Copestake and Antonio Sanfilippo for considerable help in the analysis of the LDOCE noun definition corpus. Richard Sharman kindly calculated the perplexity measures for this corpus. In addition, Hiyan Alshawi, David Weir, and Steve Young have helped clarify our thinking and made several suggestions that have influenced the way this research has developed. Alex Lascarides and four anonymous reviewers' comments on earlier drafts were very helpful to us in preparing the final version. All errors and mistakes remain our responsibility.

References

Aho, A.; Sethi, R.; and Ullman, J. (1986). *Compilers: Principles, Techniques and Tools.* Addison-Wesley.

Alshawi, H., ed. (1992). *The Core Language Engine.* MIT Press.

Baker, J. (1982). "Trainable grammars for speech recognition." Speech Communication Papers for the 97th Meeting of the Acoustical Society of America, 547–550.

Baum, L. (1972). "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes." *Inequalities*, **III**, 1–8.

Bermudez, M. (1991). "A unifying model for lookahead LR parsing." *Computer Languages*, **16**(2), 167–178.

Boguraev, B.; Carroll, J.; Briscoe, E.; and Grover, C. (1988). "Software support for practical grammar development." In *Proceedings, 12th International Conference on Computational Linguistics*, Budapest, Hungary, 54–58.

Brent, M. (1991). "Automatic acquisition of subcategorization frames from untagged text." In *Proceedings, 29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, CA, 209–214.

Briscoe, E. (1987). *Modelling Human Speech Comprehension: A Computational Approach.* Ellis Horwood and Wiley.

Briscoe, E.; Grover, C.; Boguraev, B.; and Carroll, J. (1987). "A formalism and environment for the development of a large grammar of English. In *Proceedings, 10th International Joint Conference on Artificial Intelligence*, Milan, Italy, 703–708.

Briscoe, E., and Carroll, J. (1991). "Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars." Technical Report 224, Computer Laboratory, Cambridge University.

Briscoe, E., and Waegner, N. (1992). "Robust stochastic parsing using the inside-outside algorithm." In *Proceedings, AAAI Workshop on Statistically-based NLP Techniques*, San Jose, CA.

Carroll, J., and Grover, C. (1989). "The derivation of a large computational lexicon for English." In *Computational Lexicography for Natural Language*

Processing, edited by B. Boguraev and E. Briscoe, 117–134. Longman.

Carroll, J.; Boguraev, B.; Grover, C.; and Briscoe, E. (1988). "The grammar development environment: a user guide." Technical Report 127, Computer Laboratory, Cambridge University.

Carroll, J., and Briscoe, E. (1992). "Probabilistic normalisation and unpacking of packed parse forests for unification-based grammars." In *Proceedings, AAAI Fall Symposium on Probabilistic Approaches to Natural Language,* Cambridge, MA, 33–38.

Chapman, N. (1987). *LR Parsing: Theory and Practice.* Cambridge University Press.

Church, K., and Hanks, P. (1989). "Word association norms, mutual information, and lexicography." In *Proceedings, 27th Annual Meeting of the Association for Computational Linguistics,* Vancouver, British Columbia, 76–83.

Church, K., and Patil, R. (1982). "Coping with syntactic ambiguity or how to put the block in the box on the table." *Computational Linguistics,* **8**, 139–149.

Copestake, A. (1990). "An approach to building the hierarchical element of a lexical knowledge base from a machine-readable dictionary." In *Proceedings, Workshop on Inheritance in Natural Language Processing,* Tilburg, 19–29.

Copestake, A. (1992). "Defaults in lexical representation." In *Default Inheritance in Unification-based Approaches to the Lexicon,* edited by E. Briscoe, A. Copestake, and V. de Paiva. Cambridge University Press.

Culic, K. II, and Cohen, R. (1973). "LR-regular grammars—an extension of LR(k) grammars." *Journal of Computer and System Sciences,* **7**, 66–96.

Cutting, D.; Kupiec, J.; Pedersen, J.; and Sibun, P. (1992). "A practical part-of-speech tagger." *3rd Applied ACL,* Trento, Italy, 133–140.

DeRemer, F., and Pennello, T. (1982). "Efficient computation of LALR(1) look-ahead sets." *ACM Transactions on Programming Languages and Systems,* **4**(4), 615–649.

de Rose, S. (1988). "Grammatical category disambiguation by statistical optimization." *Computational Linguistics,* **14**(1), 31–39.

Earley, J. (1970). "An efficient context-free parsing algorithm." *Communications of the ACM,* **13**(2), 94–102.

Frazier, L. (1988). "Grammar and language processing." In *Linguistics: The Cambridge Survey,* Volume 2, edited by F. Newmeyer, 14–45. Cambridge University Press.

Fujisaki, T.; Jelinek, F.; Cocke, J.; Black, E.; and Nishino, T. (1989). "A probabilistic method for sentence disambiguation." In *Proceedings, 1st International Workshop on Parsing Technologies,* Carnegie-Mellon University, Pittsburgh, PA, 105–114.

Gale, W., and Church, K. (1990). "Poor estimates of context are worse than none." *DARPA Speech and Natural Language Workshop,* Hidden Valley, PA, 283–287.

Garside, R.; Leech, G.; and Sampson, G. (1987). *The Computational Analysis of English: A Corpus-Based Approach.* Longman.

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

Gazdar, G., and Mellish, C. (1989). *Natural Language Processing in Lisp.* Addison-Wesley.

Grosch, J. (1990). "Lalr—A generator for efficient parsers." *Software—Practice and Experience,* **20**(11), 1115–1135.

Grover, C.; Briscoe, E.; Carroll, J.; and Boguraev, B. (1989). "The Alvey natural language tools grammar (second release)." Technical Report 162, Computer Laboratory, Cambridge University.

Hektoen, E. (1991). "LAR processing for natural language grammars." Masters dissertation, Computer Laboratory, Cambridge University, Cambridge, U.K.

Hindle, D., and Rooth, M. (1991). "Structural ambiguity and lexical relations." In *Proceedings, 29th Annual Meeting of the Association for Computational Linguistics,* Berkeley, CA, 229–236.

Holmes, J. (1988). *Speech Synthesis and Recognition.* Van Nostrand Reinhold. Workingham, UK.

Jensen, K.; Heidorn, G.; Miller, L.; and Ravin, Y. (1983). "Parse fitting and prose fixing: getting a hold on ill-formedness." *Computational Linguistics,* **9**, 147–153.

Johnson, M. (1989). "The computational complexity of Tomita's algorithm." In *Proceedings, 1st International Workshop on Parsing Technologies,* Carnegie-Mellon University, Pittsburgh, PA, 203–208.

Kimball, J. (1973). "Seven principles of surface structure parsing in natural language." *Cognition,* **2**, 15–47.

Kipps, J. (1989). "Analysis of Tomita's algorithm for general context-free parsing." In *Proceedings, 1st International Workshop on Parsing Technologies,* Carnegie-Mellon University, Pittsburgh, PA, 193–202.

Klein, E., and Martin, M. (1989). "The parser generating system PGSU." *Software—Practice and Experience,* **19**(11), 1015–1028.

Kristensen, B., and Madsen, O. (1981). "Methods for computing LALR(k) lookahead." *ACM Transactions on Programming Languages and Systems*, **3**(1), 60–82.

Kupiec, J. (1991). "A trellis-based algorithm for estimating the parameters of a hidden stochastic context-free grammar." *DARPA Speech and Natural Language Workshop*, Asilomar, CA.

Lari, K., and Young, S. (1990). "The estimation of stochastic context-free grammars using the Inside-Outside algorithm." *Computer Speech and Language Processing*, **4**, 35–56.

Leech, G., and Garside, R. (1991). "Running a grammar factory: the production of syntactically analysed corpora or 'treebanks'." In *English Computer Corpora: Selected Papers and Bibliography*, edited by S. Johansson and A. Strenstrom. Mouton de Gruyter.

Magerman, D., and Marcus, M. (1991). "Pearl: A probabilistic chart parser." In *Proceedings, 2nd International Workshop on Parsing Technologies*, Cancun, Mexico, 193–199.

Meteer, M.; Schwartz, R.; and Weischedel, R. (1991). "POST: Using probabilities in language processing." In *Proceedings, 12th International Joint Conference on Artificial Intelligence*, Sydney, Australia, 960–965.

Nakazawa, T. (1991). "An extended LR parsing algorithm for grammars using feature-based syntactic categories." In *Proceedings, 5th European Conference of the Association for Computational Linguistics*, Berlin, Germany, 69–74.

Ng, S.-K., and Tomita, M. (1991). "Probabilistic parsing for general context-free grammars." In *Proceedings, 2nd International Workshop on Parsing Technologies*, Cancun, Mexico, 154–163.

Osborne, M. (1990). "Corpus parsing." Masters dissertation, University of Cambridge, Cambridge, U.K.

Pereira, F., and Shieber, S. (1987). *Prolog and Natural Language Analysis*. University of Chicago Press.

Pereira, F., and Warren, D. (1980). "Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks." *Artificial Intelligence*, **13**(3), 231–278.

Pereira, F., and Schabes, Y. (1992). "Inside-outside re-estimation for partially bracketed corpora." In *Proceedings, 30th Annual Meeting of the Association for Computational Linguistics*, Newark, DE, 128–135.

Pollack, M., and Pereira, F. (1988). "An integrated framework for semantic and pragmatic interpretation." In *Proceedings, 26th Annual Conference of the Association for Computational Linguistics*, Buffalo, NY, 75–86.

Pollard, C., and Sag, I. (1987). *Information-Based Syntax and Semantics: Volume 1—Fundamentals*. Chicago University Press.

Procter, P., ed. (1978). *The Longman Dictionary of Contemporary English*. Longman.

Ritchie, G.; Pulman, S.; Russell, G.; and Black, A. (1987). "A computational framework for lexical description." *Computational Linguistics*, **13**, 290–307.

Sampson, G. (1987). "The grammatical database and parsing scheme." In *The Computational Analysis of English: A Corpus-Based Approach*, edited by R. Garside; G. Leech; and G. Sampson, 82–96. Longman.

Sampson, G.; Haigh, R.; and Atwell, E. (1989). "Natural language analysis by stochastic optimization: a progress report on Project APRIL." *Journal of Experimental and Theoretical Artificial Intelligence*, **1**, 271–287.

Santorini, B. (1990). "Penn treebank tagging and parsing manual." University of Pennsylvania, CIS Dept. Unpublished manuscript.

Schabes, Y. (1991a). "Polynomial time and space shift-reduce parsing of arbitrary context-free grammars." In *Proceedings, 29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, CA, 106–113.

Schabes, Y. (1991b). "The valid prefix property and left to right parsing of tree-adjoining grammar." In *Proceedings, 2nd International Workshop on Parsing Technologies*, Cancun, Mexico, 21–30.

Sharman, R. (1989). "Probabilistic ID/LP grammars for English." Report 217, IBM UK Scientific Centre, Winchester, England.

Sharman, R. (1991). "An introduction to language modelling." Unpublished manuscript, IBM UK Scientific Centre, Winchester, England.

Sharman, R.; Jelinek, F.; and Mercer, R. (1990). "Generating a grammar for statistical training." *DARPA Speech and Natural Language Workshop*, Hidden Valley, PA, 267–274.

Shieber, S. (1983). "Disambiguation by a shift-reduce parsing technique." In *Proceedings, 21st Annual Meeting of the Association for Computational Linguistics*, MIT, Cambridge, MA, 113–118.

Shieber, S. (1984). "The design of a

computer language for linguistic information." In *Proceedings, 10th International Conference on Computational Linguistics*, Stanford, CA, 362–366.

Shieber, S. (1985). "Using restriction to extend parsing algorithms for complex feature-based formalisms." In *Proceedings, 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, IL, 145–152.

Spector, D. (1983). "Lexing and parsing modula-2." *SIGPLAN Notices*, **18**(10), 25–32.

Taylor, L.; Grover, C.; and Briscoe, E. (1989). "The syntactic regularity of English noun phrases." In *Proceedings, 4th European Meeting of the Association for Computational Linguistics*, UMIST, Manchester, U.K., 256–263.

Tomita, M. (1984). "Parsers for natural languages." In *Proceedings, 10th International Conference on Computational Linguistics*, Stanford, CA, 354–357.

Tomita, M. (1987). "An efficient augmented context-free parsing algorithm." *Computational Linguistics*, **13**(1), 31–46.

Viterbi, A. (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE Trans. Information Theory*, IT-13, 260–269.

Wright, J. (1990). "LR parsing of probabilistic grammars with input uncertainty for speech recognition." *Computer Speech and Language*, **4**, 297–323.

Wright, J., and Wrigley, E. (1989). "Probabilistic LR parsing for speech recognition." In *Proceedings, 1st International Workshop on Parsing Technologies*, Carnegie-Mellon University, Pittsburgh, PA, 193–202.

Wright, J.; Wrigley, E.; and Sharman, R. (1991). "Adaptive probabilistic generalized LR parsing." In *Proceedings, 2nd International Workshop on Parsing Technologies*, Cancun, Mexico, 154–163.

Zeevat, H.; Calder, J.; and Klein, E. (1987). Unification categorial grammar. In *Edinburgh Working Papers in Cognitive Science, 1: Categorial Grammar, Unification Grammar, and Parsing*, edited by N. Haddock, E. Klein, and G. Morrill, Centre for Cognitive Science, Edinburgh University, U.K., 195–222.