The Acquisition and Use of Context-Dependent Grammars for English

Robert F. Simmons*
University of Texas

Yeong-Ho Yu†
University of Texas

This paper introduces a paradigm of context-dependent grammar (CDG) and an acquisition system that, through interactive teaching sessions, accumulates the CDG rules. The resulting context-sensitive rules are used by a stack-based, shift/reduce parser to compute unambiguous syntactic structures of sentences. The acquisition system and parser have been applied to the phrase structure and case analyses of 345 sentences, mainly from newswire stories, with 99% accuracy. Extrapolation from our current grammar predicts that about 25 thousand CDG rule examples will be sufficient to train the system in phrase structure analysis of most news stories. Overall, this research concludes that CDG is a computationally and conceptually tractable approach for the construction of sentence grammar for large subsets of natural language text.

1. Introduction

An enduring goal for natural language processing (NLP) researchers has been to construct computer programs that can read narrative, descriptive texts such as newspaper stories and translate them into knowledge structures that can answer questions, classify the content, and provide summaries or other useful abstractions of the text. An essential aspect of any such NLP system is parsing—to translate the indefinitely long, recursively embedded strings of words into definite ordered structures of constituent elements. Despite decades of research, parsing remains a difficult computation that often results in incomplete, ambiguous structures; and computational grammars for natural languages remain notably incomplete. In this paper we suggest that a solution to these problems may be found in the use of context-sensitive rules applied by a deterministic shift/reduce parser.

A system is described for rapid acquisition of a context-sensitive grammar based on ordinary news text. The resulting grammar is accessed by deterministic, bottom-up parsers to compute phrase structure or case analyses of texts that the grammars cover. The acquisition system allows a linguist to teach a CDG grammar by showing examples of parsing successive constituents of sentences. At this writing, 16,275 example constituents have been shown to the system and used to parse 345 sentences ranging from 10 to 60 words in length achieving 99% accuracy. These examples compress to a grammar of 3,843 rules that are equally effective in parsing. Extrapolation from our data suggests that acquiring an almost complete phrase structure grammar for AP Wire text will require about 25,000 example rules. The procedure is further demonstrated to apply directly to computing superficial case analyses from English sentences.

* Department of Computer Sciences, AI Lab, University of Texas, Austin TX 78712. E-mail @cs.texas.edu
† Boeing Helicopter Computer Svces, Philadelphia, PA

One of the first lessons in natural or formal language analysis is the Chomsky (1957) hierarchy of formal grammars, which classifies grammar forms from unrestricted rewrite rules, through context-sensitive, context-free, and the most restricted, regular grammars. It is usually conceded that pure, context-free grammars are not powerful enough to account for the syntactic analysis of natural languages (NL) such as English, Japanese, or Dutch, and most NL research in computational linguistics has used either augmented context-free or ad hoc grammars. The conventional wisdom is that context-sensitive grammars probably would be too large and conceptually and computationally untractable. There is also an unspoken supposition that the use of a context-sensitive grammar implies using the kind of complex parser required for parsing a fully context-sensitive language.

However, NL research based on simulated neural networks took a context-based approach. One of the first hints came from the striking finding from Sejnowski and Rosenberg's NETtalk (1988), that seven-character contexts were largely sufficient to map each character of a printed word into its corresponding phoneme—where each character actually maps in various contexts into several different phonemes. For accomplishing linguistic case analyses McClelland and Kawamoto (1986) and Miikkulainen and Dyer (1989) used the entire context of phrases and sentences to map string contexts into case structures. Robert Allen (1987) mapped nine-word sentences of English into Spanish translations, and Yu and Simmons (1990) accomplished comparable context-sensitive translations between English and German simple sentences. It was apparent that the contexts in which a word occurred provided information to a neural network that was sufficient to select correct word sense and syntactic structure for otherwise ambiguous usages of language.

In order to solve a problem of accepting indefinitely long, complex sentences in a fixed-size neural network, Simmons and Yu (1990) showed a method for training a network to act as a context-sensitive grammar. A sequential program accessed that grammar with a deterministic, single-path parser and accurately parsed descriptive texts. Continuing that research, 2,000 rules were accumulated and a network was trained using a back-propagation method. The training of this network required ten days of continuous computation on a Symbolics Lisp Machine. We observed that the training cost increased by more than the square of the number of training examples and calculated that 10,000–20,000 rules might well tax a supercomputer. So we decided that storing the grammar in a hash table would form a far less expensive option, provided we could define a selection algorithm comparable to that provided by the trained neural network.

In this paper we describe such a selection formula to select rules for context-sensitive parsing, a system for acquiring context-sensitive rules, and experiments in analysis and application of the grammar to ordinary newspaper text. We show that the application of context-sensitive rules by a deterministic shift/reduce parser is a conceptually and computationally tractable approach to NLP that may allow us to accumulate practical grammars for large subsets of English texts.

2. Context-Dependent Parsing

In NL research most interest has centered on context-free grammars (CFG), augmented with feature tests and transformations, used to describe the phrase structure of sentences. There is a broad literature on Generalized Phrase Structure Grammar (Gazdar et al. 1985), Unification Grammars of various types (Shieber 1986), and Augmented

Transition Networks (J. Allen 1987). Gazdar (1988) calls attention to a subcategory of context-sensitive grammars called indexed languages and illustrates some applicability to natural languages, and Joshi illustrates an application of "mild context-sensitivity" (Joshi 1987), but in general, NL computation with context-sensitive grammars is a largely unexplored area.

While a few advanced NLP laboratories have developed grammars and parsing capabilities for significantly large subsets of natural language,[1] it cannot be denied that massive effort was required and that the results are plagued by ambiguous interpretations. These grammars are typically a context-free form, augmented by complex feature tests, transformations, and occasionally, arbitrary programs. The combination of even an efficient parser with such intricate grammars may greatly increase computational complexity of the parsing system (Tomita 1985). It is extremely difficult to write and maintain such grammars, and they must frequently be revised and retested to ensure internal consistency as new rules are added. We argue here that an acquisition system for accumulating context-sensitive rules and their application by a deterministic shift/reduce parser will greatly simplify the process of constructing and maintaining natural language parsing systems.

Although we use context-sensitive rules of the form

$$uXv \rightarrow uYv$$

they are interpreted by a shift/reduce parser with the result that they can be applied successfully to the LR(k) subset of context-free languages. Unless the parser is augmented to include shifts in both directions, the system cannot parse context-sensitive languages. It is an open question as to whether English is or is not context-sensitive, but it definitely includes discontinuous constituents that may be separated by indefinitely many symbols. For this reason, future developments of the system may require operations beyond shift and reduce in the parser. To avoid the easy misinterpretation that our present system applies to context-sensitive languages, we call it Context-Dependent Grammar (CDG).

We begin with the simple notion of a shift/reduce parser. Given a stack and an input string of symbols, the shift/reduce parser may only shift a symbol to the stack (Figure 1a) or reduce n symbols on the stack by rewriting them as a single symbol (Figure 1b). We further constrain the parser to reduce no more than two symbols on the stack to a single symbol. The parsing terminates when the stack contains only a single root element and the input string is empty. Usually this class of parser applies a CFG to a sentence, but it is equally applicable to CDG.

2.1 CDG Rule Forms

The theoretical viewpoint is that the parse of a sentence is a sequence of states, each composed of a condition of the stack and the input string. The sequence ends successfully when the stack contains only the root element (e.g. SNT), and the input string is

1 Notable examples include the large augmented CFGs at IBM Yorktown Hts, the Univ. of Pennsylvania, and the Linguistic Research Ctr. at the Univ. of Texas.

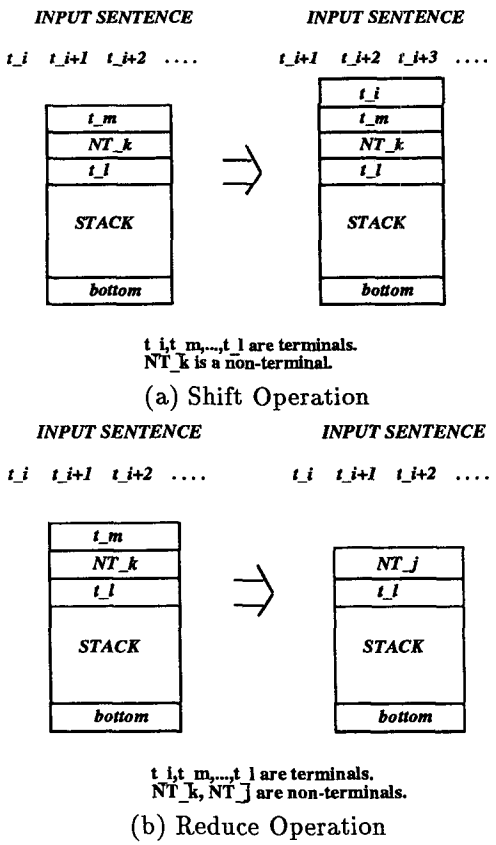


Figure 1
Shift/reduce parser.

empty. Each state can be seen as the left half of a context-sensitive rule whose right half is the succeeding state.

$$stack_s input_s \Rightarrow stack_{s+1} input_{s+1}$$

However, sentences may be of any length and are often more than forty words, so the resulting strings and stacks would form very cumbersome rules of variable lengths. To avoid this difficulty, the stack and input parts of a rule are limited to five symbols each. In the following example the stack and input parts are separated by the symbol "*," as the idea is applied to the sentence "The old man from Spain ate fish." The symbol _ stands for blank, art for article, adj for adjective, p for preposition, n for noun, and v for verb. The syntactic classes are assigned by dictionary lookup in a context-sensitive dictionary.[2]

```
The old man from Spain ate fish
art adj n    p     n    v   n

_ _ _ _ _ * art adj n p n
_ _ _ _ art * adj n p n v
_ _ _ art adj * n p n v n
_ _ art adj n * p n v n _
_ _ _ art np * p n v n _
_ _ _ _ np * p n v n _
_ _ _ np p * n v n _ _
_ _ np p n * v n _ _ _
_ _ _ np pp * v n _ _ _
_ _ _ _ np * v n _ _ _
_ _ _ np v * n _ _ _ _
_ _ np v n * _ _ _ _ _
_ _ _ np vp * _ _ _ _ _
_ _ _ _ snt * _ _ _ _ _
```

The analysis terminates with an empty input string and the single symbol "snt" on the stack, successfully completing the parse. Note that the first four operations can be described as shifts followed by the two reductions, adj n → np, and art np → np. Subsequently the p and n were shifted onto the stack and then reduced to a pp; then the np and pp on the stack were reduced to an np, followed by the shifting of v and n, their reduction to vp, and a final reduction of np vp → snt. Illustrations similar to this are often used to introduce the concept of parsing in AI texts on natural language (e.g. J. Allen 1987).

We could perfectly well record the grammar in pairs of successive states as follows:

_ _ _ np p * n v n _ _ → _ _ np p n * v n _ _ _
_ _ np p n * v n _ _ _ → _ _ _ np pp * v n _ _ _

but some economy can be achieved by recording the operation and possible label as the right half of a rule. So for the example immediately above, we record:

_ _ _ np p * n v n _ _ → (S)
_ _ np p n * v n _ _ _ → (R pp)

where S shifts and (R pp) replaces the top two elements of the stack with pp to form the next state of the parse.

Thus a *windowed context* of ten symbols is created as the left half of a rule and an operation as the right half. Note that if the stack were limited to the top two elements, and the input to a single element, the rule system would reduce to a binary rule CFG.

The example in Figure 2 shows how a sentence "Treatment is a complete rest and a special diet" is parsed by a context sensitive shift/reduce parser. Terminal symbols are lowercase, while nonterminals are uppercase. The shaded areas represent the parts

2 Described in Section 7.3.

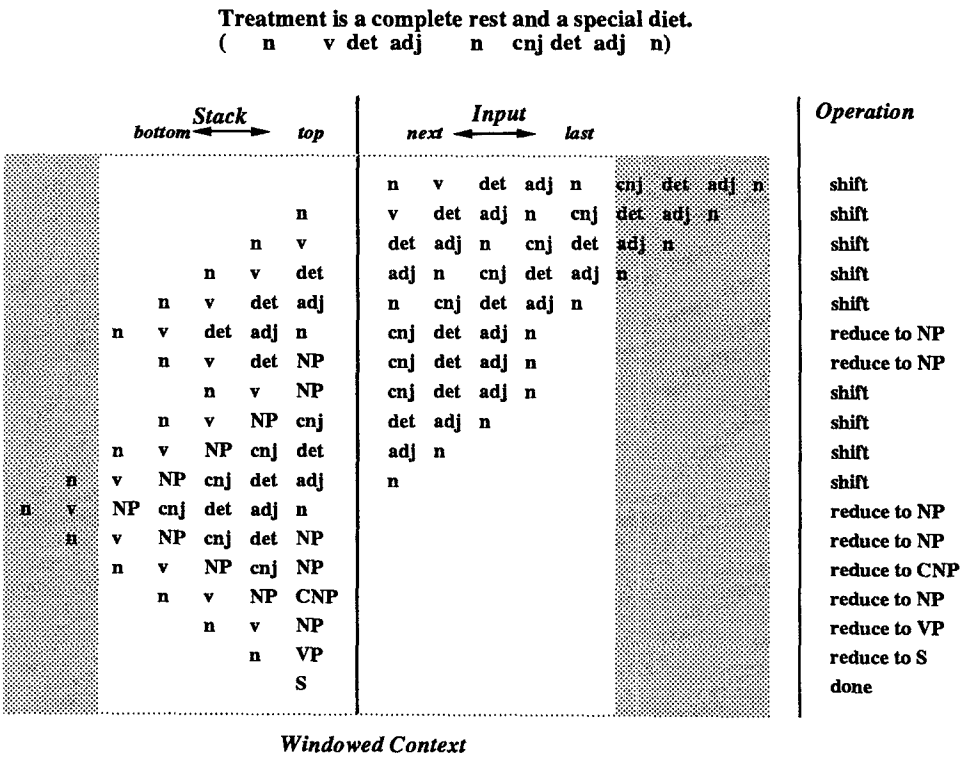


Figure 2
An example of windowed context.

of the context invisible to the system. The next operation is solely decided by the windowed context. It can be observed that the last state in the analysis is the single symbol SNT—the designated root symbol, on the stack along with an empty input string, successfully completing the parse.

And this is the CDG form of rule used in the phrase structure analysis.

2.2 Algorithm for the Shift/Reduce Parser

The parser accepts a string of syntactic word classes as its input and forms a ten-symbol vector, five symbols each from the stack and the input string. It looks up this vector as the left half of a production in the grammar and interprets the right half of the production as an instruction to modify the stack and input sequences to construct the next state of the parse. To accomplish these tasks, it maintains two stacks, one for the input string and one for the syntactic constituents. These stacks may be arbitrarily large.

An algorithm for the parser is described in Figure 3. The most important part of this algorithm is to find an applicable CDG rule from the grammar. Finding such a rule is based on the current windowed context. If there is a rule whose left side exactly matches the current windowed context, that rule will be applied. However, realistically, it is often the case that there is no exact match with any rule. Therefore, it is necessary to find a rule that *best* matches the current context.

CD-SR-Parser(Input,Cdg)

Input is a string of syntactic classes for the given sentence.
Cdg is the given CDG grammar rules.

```
Stack := empty
do until(Input = empty and Stack = (SNT))
        Windowed-context := Append(Top_five(stack),First_five(input))
        Operation := Consult_CDG(Window-context,Cdg)
        if First(Operation) = SHIFT
        then Stack := Push(First(Input),Stack)
             Input := Rest(Input)
        else Stack := Push(Second(Operation),Pop(Pop(Stack)))
end do
```

The functions, Top_five *and* First_five, *return the lists of top (or first) five elements of the Stack and the Input respectively. If there are not enough elements, these procedures pad with blanks. The function* Append *concatenates two lists into one.* Consult_CDG *consults the given CDG rules to find the next operation to take. The details of this function are the subject of the next section.* Push *and* Pop *add or delete one element to/from a stack while* First *and* Second *return the first or second elements of a list, respectively.* Rest *returns the given list minus the first element.*

Figure 3
Context-sensitive shift reduce parser.

2.3 Consulting the CDG Rules

There are two related issues in consulting the CDG rules. One is the computational representation of CDG rules, and the other is the method for selecting an applicable rule.

In the traditional CFG paradigms, a CFG rule is applicable if the left-hand side of the rule exactly matches the top elements of the stack. However, in our CDG paradigm, a perfect match between the left side of a CDG rule and the current state cannot be assured, and in most cases, a partial match must suffice for the rule to be applied. Since many rules may partially match the current context, the *best* matching rule should be selected.

One way to do this is to use a neural network. Through the back-propagation algorithm (Rumelhart, Hinton, and Williams 1986), a feed-forward network can be trained to memorize the CDG rules. After successful training, the network can be used to retrieve the best matching rule. However, this approach based on a neural network usually takes considerable training time. For instance, in our previous experiment (Simmons and Yu 1990), training a network for about 2,000 CDG rules took several days of computation. Therefore, this approach has an intrinsic problem for scaling up, at least on the present generation of neural net simulation software.

Another method is based on a hash table in which every CDG rule is stored according to its top two elements of the stack—the fourth and fifth elements of the left half of the rule. Given the current windowed context, the top two elements of the stack are used to retrieve all the relevant rules from the hash table.

We use no more than 64 word and phrase class symbols, so there can be no more than 4,096 possible pairs. The effect is to divide the large number of rules into no more than 4,096 subgroups, each of which will have a manageable subset. In fact, with 16,275 rules we discovered that we have only 823 pairs and the average number of rules per subgroup is 19.8; however, for frequently occurring pairs the number of rules in the subgroups can be much larger. The problem is to determine what scoring formula should be used to find the rule that best matches a parsing context.

Sejnowski and Rosenberg (1988) analyzed the weight matrix that resulted from training NETtalk and discovered a triangular function with the apex centered at the character in the window and the weights falling off in proportion to distance from that character. We decided that the best matching rule in our system would follow a similar pattern with maximum weights for the top two elements on the stack with weights decreasing in both directions with distance from those positions. The scoring function we use is developed as follows:

Let $\mathcal{R}$ be the set of vectors $\{R_1, R_2, \ldots, R_n\}$
where R_i is the vector $[r_1, r_2, \ldots, r_{10}]$
Let C be the vector $[c_1, c_2, \ldots, c_{10}]$
Let $\mu(c_i, r_i)$ be a matching function whose value is 1 if $c_i = r_i$, and 0 otherwise.

$\mathcal{R}$ is the entire set of rules, R_i is (the left half of) a particular rule, and C is the parse context.

Then $\mathcal{R}'$ is the subset of $\mathcal{R}$ where if $R_i \in \mathcal{R}'$ then $\mu(r_{i4}, c_4) \cdot \mu(r_{i5}, c_5) = 1$.

Access of the hash table with the top two elements of the stack, c_4, c_5 produces the set $\mathcal{R}'$.

We can now define the scoring function for each $R_i \in \mathcal{R}'$.

$$Score = \sum_{i=1}^{3} \mu(c_i, r_i) \cdot i + \sum_{i=6}^{10} \mu(c_i, r_i)(11 - i)$$

The first summation scores the matches between the stack elements of the rule and the current context, and the second summation scores the matches between the elements in the input string. If two items of the rule and context match, the total score is increased by the weight assigned to that position. The maximum score for a perfect match is 21 according to the above formula.

From several experiments, varying the length of vector and the weights, particularly those assigned to blanks, it has been determined that this formula gave the best performance among those tested. More importantly, it has worked well in the current phrase structure and case analysis experiments.

It was an unexpected surprise to us[3] that using context-sensitive productions, an elementary, deterministic, parsing algorithm proved adequate to provide 99% correct, unambiguous analyses for the entire text studied.

3. Grammar Acquisition for CDG

Constructing an augmented phrase structure grammar of whatever type—unification, GPSG, or ATN—is a painful process usually involving a well-trained linguistic team of several people. These types of grammar require that a CFG recognition rule such

3 But perhaps not to Marcus (1980) and Berwick (1985), who promote the study of deterministic parsing.

as np vp → snt be supported by such additional information as the fact that the np and vp agree in number, that the np is characterized by particular features such as *count, animate,* etc., and that the vp can or cannot accept certain types of complements. The additional features make the rules exceedingly complex and difficult to prepare and debug. College students can be taught easily to make a phrase structure tree to represent a sentence, but it requires considerable linguistic training to deal successfully with a feature grammar.

We have seen in the preceding section that a CFG is derived from recording the successive states of the parses of sentences. Thus it was natural for us to develop an interactive acquisition system that would assist a linguist (or a student) in constructing such parses to produce easily large sets of example CFG rules.[4] The system continued to evolve as a consequence of our use until we had included capabilities to:

- read in text and data files
- compile dictionary and grammar tables from completed text files
- select a sentence to continue processing or revise
- look up words in a dictionary to suggest the syntactic class for the word in context when assigning syntactic classes to the words in a sentence
- compare each state of the parse with rules in the current grammar to predict the shift/reduce operation. A carriage return signals that the user accepts the prompt, or the typing in of the desired operation overrides it.
- compute and display the parse tree from the local grammar after completion of each sentence, or from the global total grammar at any time
- provide backing up and editing capability to correct errors
- print help messages and guide the user
- compile dictionary and grammar entries at the completion of each sentence, insuring no duplicate entries
- save completed or partially completed grammar files.

The resulting tool, GRAMAQ, enables a linguist to construct a context-sensitive grammar for a text corpus at the rate of several sentences per hour. Thousands of rules are accumulated with only weeks of effort in contrast to the years required for a comparable system of augmented CFG rules. About ten weeks of effort were required to produce the 16,275 rules on which this study is based. Since GRAMAQ's prompts become more accurate as the dictionary and grammar grow in size, there is a positive acceleration in the speed of grammar accumulation and the linguist's task gradually converges to one of alert supervision of the system's prompts.

A slightly different version of GRAMAQ is Caseaq, which uses operations that create case constituents to accumulate a context-sensitive grammar that transforms

4 Starting with an Emacs editor, it was fairly easy to read in a file of sentences and to assign each word its syntactic class according to its context. Then the asterisk was inserted at the beginning of the syntactic string, the string was copied to the next line, the asterisk moved if a shift operation was indicated, or the top two symbols on the stack were rewritten if a reduce was required—just as we constructed the example in the preceding section. Naturally enough, we soon made Emacs macros to help us, and then escalated to a Lisp program that would print the *stack-*-string* and interpret our shift/reduce commands to produce a new state of the parse.

Text	*States*	*Sentences*	*Wds/Snt*	*Mn-Wds/Snt*
Hepatitis	236	12	4-19	10.3
Measles	316	10	4-25	16.3
News Story	470	10	9-51	23.5
APWire-Robots	1005	21	11-53	26.0
APWire-Rocket	1437	25	8-47	29.2
APWire-Shuttle	598	14	12-32	21.9
Total	4062	92	4-53	22.8

Table 1
Characteristics of a sample of the text corpus.

sentences directly to case structures with no intermediate stage of phrase structure trees. It has the same functionality as GRAMAQ but allows the linguist user to specify a case argument and value as the transformation of syntactic elements on the stack, and to rename the head of such a constituent by a syntactic label. Figure 9 in Section 7.3 illustrates the acquisition of case grammar.

4. Experiments with CDG

There are a number of critical questions that need be answered if the claim that CDG grammars are useful is to be supported.

- Can they be used to obtain accurate parses for real texts?
- Do they reduce ambiguity in the parsing process?
- How well do the rules generalize to new texts?
- How large must a CFG be to encompass the syntactic structures for most newspaper text?

4.1 Parsing and Ambiguity with CDG

Over the course of this study we accumulated 345 sentences mainly from newswire texts. The first two articles were brief disease descriptions from a youth encyclopedia; the remaining fifteen were newspaper articles from February 1989 using the terms "star wars," "SDI," or "Strategic Defense Initiative." Table 1 characterizes typical articles by the number of CDG rules or states, number of sentences, the range of sentence lengths, and the average number of words per sentence.

We developed our approach to acquiring and parsing context-sensitive grammars on the first two simple texts, and then used GRAMAQ to redo those texts and to construct productions for the news stories. The total text numbered 345 sentences, which accumulated 16,275 context-sensitive rules—an average of 47 per sentence.

The parser embodying the algorithm illustrated earlier in Figure 1 was augmented to compare the constituents it constructed with those prescribed during grammar acquisition by the linguist. In parsing the 345 sentences, 335 parses exactly matched the linguist's original judgement. In nine cases in which differences occurred, the parses were judged correct, but slightly different sequences of parse states occurred. The tenth case clearly made an attachment error—of an introductory adverbial phrase in the sentence "Hours later, Baghdad announced. . . ." This was mistakenly attached to "Baghdad." This evaluation shows that the grammar was in precise agreement with

Another mission soon scheduled that also would have priority over the shuttle is the first firing of a trident two intercontinental range missile from a submerged submarine.

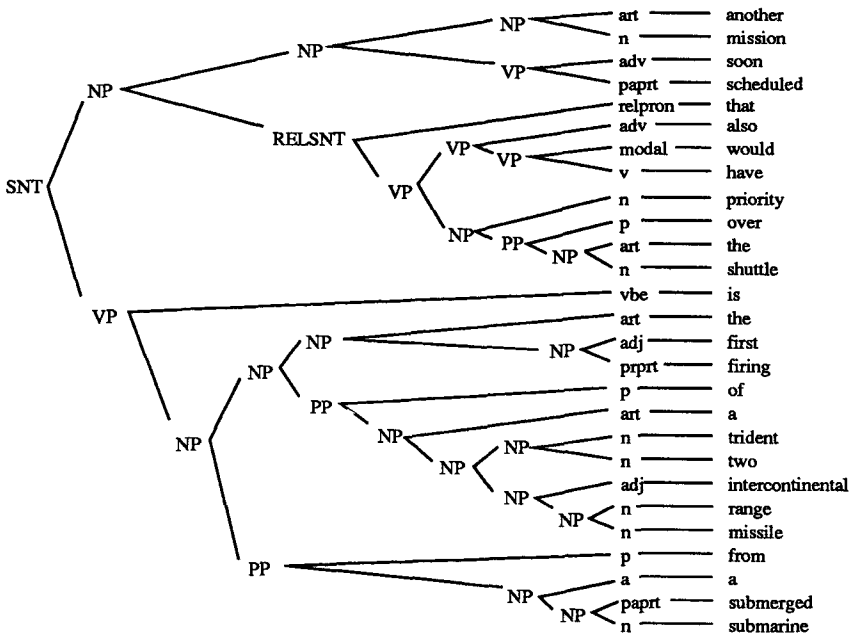


Figure 4
Sentence parse.

the linguist 97% of the time and completed correct parses in 99.7% of the 345 sentences from which it was derived. Since our primary interest was in evaluating the effectiveness of the CDG, all these evaluations were based on using correct syntactic classes for the words in the sentences. The context-sensitive dictionary lookup procedure described in Section 7.3 is 99.5% accurate, but it assigns 40 word classes incorrectly. As a consequence, using this procedure would result in a reduction of about 10% accuracy in parsing.

An output of a sentence from the parser is displayed as a tree in Figure 4. Since the whole mechanism is coded in Lisp, the actual output of the system is a nested list that is then printed as a tree.

Notice in this figure that the PP at the bottom modifies the NP composed of "the first firing of a trident two intercontinental range missile" not just the word "firing." Since the parsing is bottom-up, left-to-right, the constituents are formed in the natural order of words encountered in the sentence and the terminals of the tree can be read top-to-bottom to give their ordering in the sentence.

Although 345 sentences totaling 8594 words is a small selection from the infinite set of possible English sentences, it is large enough to assure us that the CDG is a reasonable form of grammar. Since the deterministic parsing algorithm selects a single interpretation, which we have seen almost perfectly agrees with the linguist's parsings, it is apparent that, at least for this size text sample, there is little difficulty with ambiguous interpretations.

5. Generalization of CDG

The purpose of accumulating sample rules from texts is to achieve a grammar general enough to analyze new texts it has never seen. To be useful, the grammar must generalize. There are at least three aspects of generalization to be considered.

- How well does the grammar generalize at the sentence level? That is, how well does the grammar parse new sentences that it has not previously experienced?
- How well does the grammar generalize at the operation level? That is, how well does the grammar predict the correct Shift/Reduce operation during acquisition of new sentences?
- How much does the rule retention strategy affect generalization? For instance, when the grammar predicts the same output as a new rule does, and the new rule is not saved, how well does the resulting grammar parse?

5.1 Generalization at the Sentence Level

The complete parse of a sentence is a sequence of states recognized by the grammar (whether it be CDG or any other). If all the constituents of the new sentence can be recognized, the new sentence can be parsed correctly. It will be seen in a later paragraph that with 16,275 rules, the grammar predicts the output of new rules correctly about 85% of the time. For the average sentence with 47 states, only 85% or about 40 states can be expected to be predicted correctly; consequently the deterministic parse will frequently fail. In fact, 5 of 14 new sentences parsed correctly in a brief experiment that used a grammar based on 320 sentences to attempt to parse the new, 20-sentence text. Considering that only a single path was followed by the deterministic parser, we predicted that a multiple-path parser would perform somewhat better for this aspect of generalization. In fact, our initial experiments with a beam search parser resulted in successful parses of 15 of the 20 new sentences using the same grammar based on the 320 sentences.

5.2 Generalization at the Operation Level

This level of generalization is of central significance to the grammar acquisition system. When GRAMAQ looks up a state in the grammar it finds the best matching state with the same top two elements on the stack, and offers the right half of this rule as its suggestion to the linguist. How often is this prediction correct?

To answer this question we compiled the grammar of 16,275 rules in cumulative increments of 1,017 rules using a procedure, *union-grammar*, that would only add a rule to the grammar if the grammar did not already predict its operation. We call the result a "minimal-grammar," and it contains 3,843 rules. The black line of Figure 5 shows that with the first 1,000 rules 40% were new; with an accumulation of 5,000, 18% were new rules. By the time 16,000 rules have been accumulated, the curve has flattened to an average of 16% new rules added. This means that the acquisition system will make correct prompts about 84% of the time and the linguist will only need to correct the system's suggestions about 3 or 4 times in 20 context presentations.

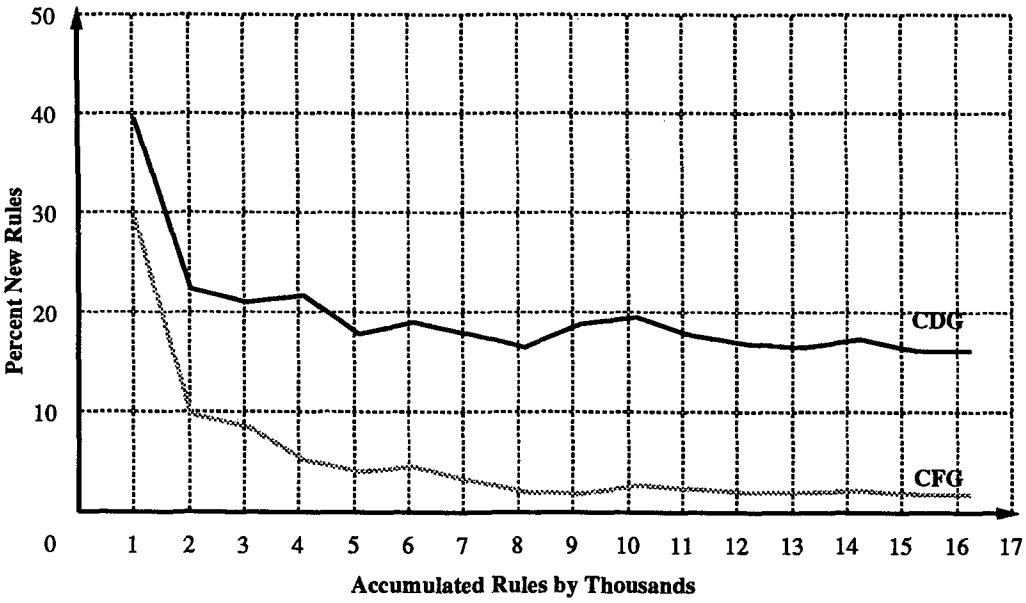


Figure 5
Generalization of CDG rules.

5.3 Rule Retention and Generalization

If two parsing grammars account equally well for the same sentences, the one with fewer rules is less redundant, more abstract, and the one to be preferred. We used the *union-grammar* procedure to produce and study the minimal grammar for the 16,275 rules (rule-examples) derived from the sample text. Union-grammar records a new rule for a rule-example:[5]

1. if best matching rule has an operation that doesn't match
2. if best matching rule ties with another rule whose operation does not match
3. if 2 is true, and score = 21 we have a full contradiction and list the rule as an error.

Six contradictions occurred in the grammar; five were inconsistent treatments of "SNT" followed by one or more punctuation marks, while the sixth offered both a shift and a "pp" for a preposition-noun followed by a preposition. The latter case is an attachment ambiguity not resolvable by syntax.

In the first pass as shown in Table 2, the text resulted in 3,194 rules compared with 16,275 possible rules. That is, 13,081 possible CDG rules were not retained because already existing rules would match and predict the operation. However, using those rules to parse the same text gave very poor results: zero correct parses at the sentence level. Therefore, the process of compiling a minimal grammar was repeated starting with those 3,194 rules. This time only 619 new rules were added. The purpose of this

5 These definite conditions are due to an analysis by Mark Ring.

Pass	*Unretained*	*Retained*	*Total Rules*
1	13081	3194	16275
2	15656	619	16275
3	16245	18	16275
4	16275	0	16275

Table 2
Four passes with minimal grammar.

repetition is to get rid of the effect that the rules added later change the predictions made earlier. Finally, in a fourth repetition of the process no rules were new.

The resulting grammar of 3,843 rules succeeds in parsing the text with only occasional minor errors in attaching constituents. It is to be emphasized that the unretained rules are *similar* but not identical to those in the minimal grammar.

We can observe that this technique of minimal retention by "unioning" new rules to the grammar results in a compression of the order 16,275/3,843 or 4.2 to 1, without increase in error. If this ratio holds for larger grammars, then if the linguist accumulates 40,000 training-example rules to account for the syntax of a given subset of language, that grammar can be compressed automatically to about 10,000 rules that will accomplish the same task.

6. Predicting the Size of CDGs

When any kind of acquisition system is used to accumulate knowledge, one very interesting question is, when will the knowledge be complete enough for the intended application? In our case, how many CDG rules will be sufficient to cover almost all newswire stories? To answer this question, an extrapolation can be used to find a point when the solid line of Figure 5 intersects with the y-axis. However, the CDG curve is descending too slowly to make a reliable extrapolation.

Therefore, another question was investigated instead: when will the CDG rules include a complete set of CFG rules? Note that a CDG rule is equivalent to a CFG rule if the context is limited to the top two elements of the stack. What the other elements in the context accomplish is to make one rule preferable to another that has the same top two elements of the stack, but a different context.

We allow 64 symbols in our phrase structure analysis. That means, there are 64^2 possible combinations for the top two elements of the stack. For each combination, there are 65 possible operations:[6] a shift or a reduction to another symbol. Among 16,275 CDG rules, we studied how many different CFG rules can be derived by eliminating the context. We found 844 different CFG rules that used 600 different left-side pairs of symbols. This shows that a given context free pair of symbols averages 1.4 different operations.[7]

Then, as we did with CDG rules, we measured how many new CFG rules were added in an accumulative fashion. The shaded line of Figure 5 shows the result.

6 Actually, there are fewer than 65 possible operations since the stack elements can be reduced only to *nonterminal* symbols.

7 We actually use only 48 different symbols, so only 48^2 or 2,304 combinations could have occurred. The fraction 600/2,304 yields .26, the proportion of the combinatoric space that is actually used, so far.

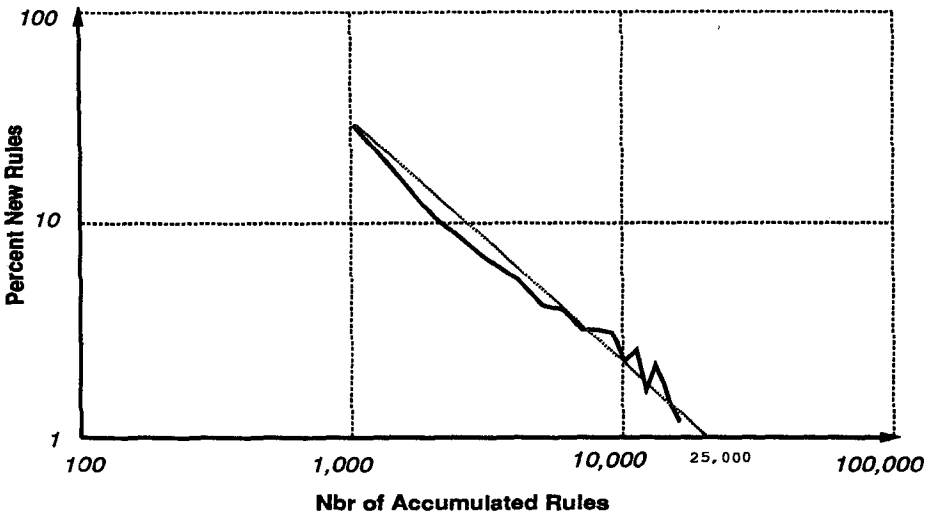


Figure 6
Log-log plot of new CFG rules.

Notice that the line has descended to about 1.5% errors at 16,000 rules. To make an extrapolation easier, a log-log graph shows the same data in Figure 6. From this graph, it can be predicted that, after about 25,000 CDG rules are accumulated, the grammar will encompass a CFG component that is 99% complete. Beyond this point, additional CDG rules will add almost no new CFG rules, but only fine-tune the grammar so that it can resolve ambiguities more effectively.

Also, it is our belief that, after the CDG reaches that point, a multi-path, beam-search parser will be able to parse most newswire stories very reliably. This belief is based on our initial experiment that used a beam search parser to test generalization of the grammar to find parses for fifteen out of twenty new sentences.

7. Acquiring Case Grammar

Explicating the phrase structure constituents of sentences is an essential aspect in computer recognition of meaning. Case analysis organizes the constituents into a hierarchical structure of labeled propositions. The propositions can be used directly to answer questions and are the basis of schemas, scripts, and frames that are used to add meaning to otherwise inexplicit texts. As a result of the experiments with acquiring CDG and exploring its properties for parsing phrase structures, we became fairly confident that we could generalize the system to acquisition and parsing based on a grammar that would compute syntactic case structures *directly* from syntactic strings. Direct translation from string to structure is supported by neural network experiments such as those by McClelland and Kawamoto (1986), Miikkulainen and Dyer (1989), Yu and Simmons (1990), and Leow and Simmons (1990). We reasoned that if we could acquire case grammar with something approaching the simplicity of acquiring phrase structure rules, the result could be of great value for NL applications.

7.1 Case Structure

Cook (1989) reviewed twenty years of linguistic research on case analysis of natural language sentences. He synthesized the various theories into a system that depends on the subclassification of verbs into twelve categories, and it is apparent from his review that with a fine subcategorization of verbs and nominals, case analysis can be accomplished as a purely syntactic operation—subject to the limitations of attachment ambiguities that are not resolvable by syntax. This conclusion is somewhat at variance with those AI approaches that require a syntactic analysis to be followed by a semantic operation that filters and transforms syntactic constituents to compute case-labeled propositions (e.g. Rim 1990), but it is consistent with the neural network experience of directly mapping from sentence to case structure, and with the AI research that seeks to integrate syntactic and semantic processing while translating sentences to propositional structures.

Linguistic theories of case structure have been concerned only with single propositions headed by verb predications; they have been largely silent with regard to the structure of noun phrases and the relations among embedded and sequential propositions. Additional conventions for managing these complications have been developed in Simmons (1984) and Alterman (1985) and are used here.

The central notion of a case analysis is to translate sentence strings into a nested structure of case relations (or predicates) where each relation has a head term and an indefinite number of labeled arguments. An argument may itself be a case relation. Thus a sentence, as in the examples below, forms a tree of case relations.

The old man from Spain ate fish.

```
(eat Agt (man Mod old From spain) Obj fish)
```

Another mission scheduled soon is the first firing of a trident missile from a submerged submarine.

```
(is Obj1 (mission Mod another Obj* (scheduled Vmod soon))
    Obj2 (firing Mod first Det the Of (missile Nmod trident Det a)
                 From (submarine Mod submerged Det a)))
```

Note that *mission* is in *Obj** relation to *scheduled*. This means the object of *scheduled* is *mission*, and the expression can be read as "another mission *such that* mission is scheduled soon." An asterisk as a suffix to a label always signals the reverse direction for the label.

There is a small set of case relations for verb arguments, such as *verbmodifier, agent, object, beneficiary, experiencer, location, state, time, direction,* etc. For nouns there are *determiner, modifier, quantifier, amount, nounmodifier, preposition,* and reverse verb relations, *agt*, obj*, ben**, etc. Prepositions and conjunctions are usually used directly as argument labels while sentence conjunctions such as *because, while, before, after,* etc. are represented as heads of propositions that relate two other propositions with the labels *preceding, post, antecedent,* and *consequent*. For example, "Because she ate fish and chips earlier, Mary was not hungry."

```
(because Ante (ate Agt she Obj (fish And chips) Vmod earlier)
         Conse (was Vmod not Obj1 mary State hungry))
```

Verbs are subcategorized as *vao, vabo, vo, va, vhav, vbe* where a is agent, o is object, b is beneficiary and vhav is a form of *have* and vbe a form of *be*. So far, only the

subcategory of *time* has been necessary in subcategorizing nouns to accomplish this form of case analysis, but in general, a lexical semantics is required to resolve syntactic attachment ambiguities. The complete set of case relations is presumed to be small, but no one has yet claimed a complete enumeration of them.

Other case systems such as those taught by Schank (1980) and Jackendoff (1983) classify predicate names into such primitives as *Do, Event, Thing, Mtrans, Ptrans, Go, Action,* etc., to approximate some form of "language of thought" but the present approach is less ambitious, proposing merely to represent in a fairly formal fashion the organization of the words in a sentence. Subsequent operations on this admittedly superficial class of case structures, when augmented with a system of shallow lexical semantics, have been shown to accomplish question answering, focus tracking of topics throughout a text, automatic outlining, and summarization of texts (Seo 1990; Rim 1990). One strong constraint on this type of analysis is that the resulting case structure must maintain *all* information present in the text so that the text may be exactly reconstituted from the analysis.

7.2 Syntactic Analysis of Case Structure

We've seen earlier that a shift/reduce-rename operation is sufficient to parse most sentences into phrase structures. Case structure, however, requires transformations in addition to these operations. To form a case structure it is frequently necessary to change the order of constituents and to insert case labels. Following Jackendoff's principle of *grammatical constraint,* which argues essentially that semantic interpretation is frequently reflected in the syntactic form, case transformations are accomplished as each syntactic constituent is discovered. Thus when a verb, say *throw* and an NP, say *coconuts* are on top of the stack, one must not only create a VP, but also decide the case, *Obj,* and form the constituent, (*throw Obj coconuts*). This can be accomplished in customary approaches to parsing by using augmented context free recognition rules of the form:

$$\text{VP} \rightarrow \text{VP NP / 1 obj 2}$$

where the numbers following the slash refer to the text dominated by the syntactic class in the referenced position, (ordered left-to-right) in the right half of the rule. The resulting constituents can be accumulated to form the case analysis of a sentence (Simmons 1984).

We develop augmented context-sensitive rules following the same principle. Let us look again at the example "The old man from Spain ate fish," this time to develop case relations.

```
* art adj n from n vao n ; shift
art * adj n from n vao n ; shift
art adj * n from n vao n ; shift
art adj n * from n vao n ; 1 mod 2  (man Mod old)
art n * from n vao n     ; 1 det 2  (man Mod old Det the)
n * from n vao n         ; shift
n from * n vao n         ; shift
n from n * vao n         ; 3 2 1    (man Mod old Det the From spain)
n * vao n                ; shift
n vao * n                ; 2 agt 1  (ate Agt (man Mod old ... )
vao * n                  ; shift
vao n *                  ; 1 obj 2  (ate Agt (man ...) Obj fish)
```

Stack			Case-Transform		
	adj	n	n	mod	adj
	n1	n2	n2	nmod	n1
	n	vao	1	agt	2
	n	vo	1	obj	2
	vbe	v	1	vbe	2 vpasv
	vabo	n	2	ben	1 vao
	n	vpasv	1	obj	2
v..	prep	n	3	2	1
n	prep	n	3	2	1
vpasv	by	n	1	prep	2
	snt	because	1	conse	2
	because	snt	2	ante	1
n	and	n	1	2	3
	snt	after	1	pre	2
	after	snt	2	post	1

Table 3
Some typical case transformations for syntactic constituents

In this example the case transformation immediately follows the semicolon, and the result of the transformation is shown in parentheses further to the right. The result in the final constituent is:

(ate Agt (man Mod old Det the From spain) Obj fish).

Note that we did not rename the syntactic constituents as NP or VP in this example, because we were not interested in showing the phrase structure tree. Renaming in case analysis need only be done when it is necessary to pass on information accumulated from an earlier constituent.

For example, in "fish were eaten by birds," the CS parse is as follows:

```
* n vbe ppart by n ; shift
n * vbe ppart by n ; shift
n vbe * ppart by n ; shift
n vbe ppart * by n ; 1 vbe 2, vpasv  (eaten Vbe were)
n vpasv * by n     ; 1 obj 2  (eaten Vbe were Obj fish)
vpasv * by n       ; shift
vpasv by * n       ; shift
vpasv by n *  ; 1 prep 2 (birds Prep by)
vpasv n *     ; 2 agt 1 (eaten Vbe were Obj fish Agt (birds Prep by))
```

Here, it was necessary to rename the combination of a past participle and its auxiliary as a passive verb, *vpasv*, so that the syntactic subject and object could be recognized as *Obj* and *Agent*, respectively. We also chose to use the argument name *Prep* to form (*birds Prep by*) so that we could then call that constituent *Agent*.

We can see that the *reduce* operation has become a *reduce-transform-rename* operation where numbers refer to elements of the stack, the second term provides a case argument label, the ordering provides a transformation, and an optional fourth element may rename the constituent. A sample of typical case transformations is shown associated with the top elements of the stack in Table 3. In this table, the first element of the stack is in the third position in the left side of the table, and the number 1 refers to that position, 2 to the second, and 3 to the first. As an aid to the reader the first two

```
CS-CASE-Parser(input,cdg)

Input is a string of syntactic classes for the given sentence.
Cdg is the given CDG grammar rules.

stack := empty
outputstack := empty
do until(input = empty and 2nd(stack) = blank)
        window-context := append(top_five(stack),first_five(input))
        operation := consult_CDG(window-context,cdg)
        if first(operation) = SHIFT
        then stack := push(first(input),stack)
             input := rest(input)
        else stack := push(select(operation),pop(pop(stack)))
             outputstack := make_constituent(operation,outputstack)
end do
```

Figure 7
Algorithm for case parse.

entries in the table refer literally by symbol rather than by reference to the stack. The symbols *vao* and *vabo* are subclasses of verbs that take, respectively, agent and object; and agent, beneficiary, and object. The symbol *v..* refers to any verb. Forms of the verb *be* are referred to as *vbe*, and passivization is marked by relabeling a verb by adding the suffix *-pasv*.

Parsing case structures

From the discussion above we may observe that the flow of control in accomplishing a case parse is identical to that of a phrase structure parse. The difference lies in the fact that when a constituent is recognized (see Figure 7):

- in phrase structure, a new name is substituted for its stack elements, and a constituent is formed by listing the name and its elements
- in case analysis, a case transformation is applied to designated elements on the stack to construct a constituent, and the head (i.e. the first element of the transformation) is substituted for its elements—unless a new name is provided for that substitution.

Consequently the algorithm used in phrase structure analysis is easily adapted to case analysis. The difference lies in interpreting and applying the operation to make a new constituent and a new stack.

In the algorithm shown above, we revise the stack by attaching either the head of the new constituent, or its new name, to the stack resulting from the removal of all elements in the new constituent. The function *select* chooses either a new name if present, or the first element, the head of the operation. *Makeconstituent* applies the transformation rule to form a new constituent from the output stack and pushes the constituent onto the output stack, which is first reduced by removing the elements used in the constituent. Again, the algorithm is a deterministic, first (best) path parser

with behavior essentially the same as the phrase structure parser. But this version accomplishes transformations to construct a case structure analysis.

7.3 Acquisition System for Case Grammar

The acquisition system, like the parser, required only minor revisions to accept case grammar. It must apply a shift or any transformation to construct the new stack-string for the linguist user, and it must record the shift or transformation as the right half of a context-sensitive rule—still composed of a ten-symbol left half and an operation as the right half. Consequently, the system will be illustrated in Figure 9 rather than described in detail.

Earlier we mentioned the context-sensitive dictionary. This is compiled by associating with each word the linguist's in-context assignments of each syntactic word class in which it is experienced. When the dictionary is built, the occurrence frequencies of each word class are accumulated for each word. A primitive grammar of four-tuples terminating with each word class is also formed and hashed in a table of syntactic paths. The procedure to determine a word class in context,

- first obtains the candidates from the dictionary.
- For each candidate wc, it forms a four-tuple, vec, by adding it to the cdr of each immediately preceding vec, stored in IPC.
- Each such vec is tested against the table of syntactic paths;
  - — if it has been seen previously, it is added to the list of IPCs,
  - — otherwise it is eliminated.
- If the union of first elements of the IPC list is a single word class, that is the choice. If not, the word's most frequent word class among the union of surviving classes for the word is chosen.

The effect of this procedure is to examine a context of plus and minus three words to determine the word class in question. Although a larger context based on five-tuple paths is slightly more effective, there is a tradeoff between accuracy and storage requirements.

The word class selection procedure was tested on the 8,310 words of the 345-sentence sample of text. A score of 99.52% correct was achieved, with 8,270 words correctly assigned. As a comparison, the most frequent category for a word resulted in 8,137 correct assignments for a score of 97.52%. Although there are only 3,298 word types with an average of 3.7 tokens per type, the occurrence of single word class usages for words in this sample is very high, thus accounting for the effectiveness of the simpler heuristic of assignment of the most frequent category. However, since the effect of misassignment of word class can often ruin the parse, the use of the more complex procedure is amply justified. Analysis of the 40 errors in word class assignment showed 7 confusions of nouns and verbs that will certainly cause errors in parsing; other confusions of adjective/noun, and adverb/preposition are less devastating, but still serious enough to require further improvements in the procedure.

The word class selection procedure is adequate to form the prompts in the lexical acquisition phase, but the statistics on parsing effectiveness given earlier depend on perfect word class assignments.

Shown in Figure 8 is the system's presentation of a sentence and its requests for each word's syntactic class. The protocol in Figure 9 shows the acquisition of shift

Lexical Acquisition: The system prompts for syntactic classes are in capitals. The user accepts the system's prompt with a carriage return, cr or types in a syntactic class in lower case. We show user's responses in bold-face, using **cr** for carriage return. Other abbreviations are wc for word class, y or n for yes or no, and b for backup.

(THE LAUNCH OF DISCOVERY AND ITS FIVE ASTRONAUTS HAS BEEN DELAYED AT-LEAST TWO DAYS UNTIL MARCH ELEVENTH BECAUSE-OF A CRUSHED ELECTRICAL PART ON A MAIN ENGINE COMMA OFFICIALS SAID)
process this one? - y or n **y**
THE cr for default else wc or b default is: ART **cr**
LAUNCH cr for default else wc or b **cr** ;user made an error since there was no default
LAUNCH cr for default else wc or b **n** ;system repeated the question
OF cr for default else wc or b default is: OF **cr**
DISCOVERY cr for default else wc or b **n**
AND cr for default else wc or b default is: CONJ **cr**
ITS cr for default else wc or b **b** ;user decided to redo "and"
AND cr for default else wc or b default is: CONJ **and**
ITS cr for default else wc or b **ppron**
⋮ **skipping most of the sentence...**
A cr for default else wc or b default is: ART **cr**
MAIN cr for default else wc or b **n**
ENGINE cr for default else wc or b **n**
COMMA cr for default else wc or b default is: COMMA **cr**
OFFICIALS cr for default else wc or b **n**
SAID cr for default else wc or b **vao**

Figure 8
Illustration of dictionary acquisition.

and transformation rules for the sentence. What we notice in this second protocol is that the stack shows syntactic labels but the input string presented to the linguist is in English. As the system constructs a CS rule, however, the vector containing five elements of stack and five of input string is composed entirely of syntactic classes. The English input string better enables the linguist to maintain the meaningful context he or she uses to analyze the sentence. About five to ten minutes were required to make the judgments for this sentence. Appendix A shows the rules acquired in the session.

When rules for the sentence were completed, the system added the new syntactic classes and rules to the grammar, then offered to parse the sentence. The resulting parse is shown in Figure 10.

The case acquisition system was used on the texts described earlier in Table 1 to accumulate 3,700 example CDG case rules. Because the case transformations refer to three stack elements and the number of case labels is large, we expected and found that a much larger sample of text would be required to obtain the levels of generalization seen in the phrase structure experiments.

Accumulated in increments of 400 rules, the case curve flattens at about 2,400 rules with an average of 33% error in prediction compared to the 20% found in analysis of the same number of phrase structure rules. The compressed or minimal grammar for this set of case rules reduces the 3,700 rules to 1,633, a compression ratio in this case of 2.3 examples accounted for by each rule. The resulting compressed grammar parses the texts with 99% accuracy. These statistics are from our initial study of a case grammar, and they should be taken only as preliminary estimates of what a more thorough study may show.

Case-Grammar Acquisition: The options are h for a help message, b for backup one state, s for shift, case-trans for a case transformation, and cr for carriage return to accept a system prompt. System prompts are capitalized in parentheses, user responses are in lower case. Where no apparent response is shown, the user did a carriage return to accept the prompt. The first line shows the syntactic classes for the words in the sentence.

(ART N OF N AND PPRON ADJ N VHAV VBE VAO AT-LEAST ADJ N UNTIL N N BECAUSE-OF ART PPART ADJ N ON ART N N COMMA N VAO)
(* THE LAUNCH OF DISCOVERY AND ITS FIVE ASTRONAUTS HAS BEEN DELAYED AT-LEAST TWO DAYS UNTIL MARCH ELEVENTH BECAUSE-OF A CRUSHED ELECTRICAL PART ON A MAIN ENGINE COMMA OFFICIALS SAID)
options-are h b s case-trans or cr for default: (S)
(ART * LAUNCH OF DISCOVERY AND ITS ... SAID)
options-are h b s case-trans or cr for default: (S)
(ART N * OF DISCOVERY AND ITS FIVE ... SAID)
options-are h b s case-trans or cr for default: (S) 1 det 2
(N * OF DISCOVERY AND ITS FIVE ... SAID)
options-are h b s case-trans or cr for default: (S)

: **skipping several shifts**
(N OF N AND PPRON ADJ N * HAS BEEN DELAYED AT-LEAST ... SAID)
options-are h b s case-trans or cr for default: (S) 1 mod 2
(N OF N AND PPRON N * HAS BEEN DELAYED AT-LEAST ... SAID)
options-are h b s case-trans or cr for default: NIL 1 possby 2
(N OF N AND N * HAS BEEN DELAYED AT-LEAST ... SAID)
options-are h b s case-trans or cr for default: NIL 3 2 1
(N OF N * HAS BEEN DELAYED AT-LEAST TWO ... SAID)
options-are h b s case-trans or cr for default: (3 2 1)
(N * HAS BEEN DELAYED AT-LEAST TWO ... SAID)
options-are h b s case-trans or cr for default: (S)
(N VHAV * BEEN DELAYED AT-LEAST TWO DAYS ... SAID)
options-are h b s case-trans or cr for default: (1 OBJ 2) s
(N VHAV VBE * DELAYED AT-LEAST TWO DAYS ... SAID)
options-are h b s case-trans or cr for default: (S) 1 aux 2
(N VBE * DELAYED AT-LEAST TWO DAYS UNTIL ... SAID)
options-are h b s case-trans or cr for default: (S)
(N VBE VAO * AT-LEAST TWO DAYS UNTIL MARCH ... SAID)
options-are h b s case-trans or cr for default: (1 VBE 2 VAOPASV)
(N VAOPASV * AT-LEAST TWO DAYS UNTIL MARCH ... SAID)
options-are h b s case-trans or cr for default: (1 OBJ 2)

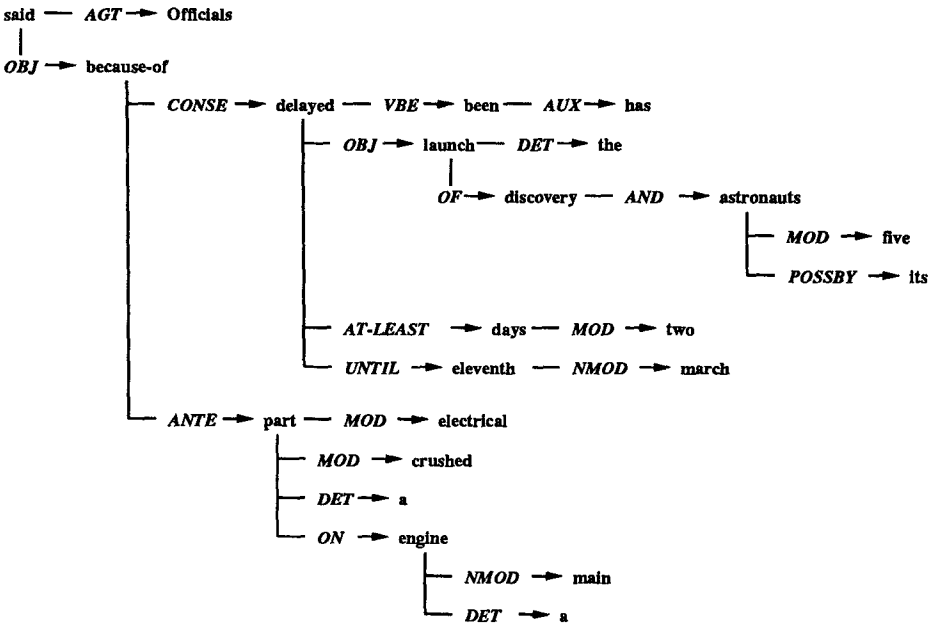
: **skipping now to BECAUSE**
(VAOPASV UNTIL N * BECAUSE-OF A CRUSHED ELECTRICAL ... SAID)
options-are h b s case-trans or cr for default: (S) 3 2 1
(VAOPASV * BECAUSE-OF A CRUSHED ELECTRICAL PART ... SAID)
options-are h b s case-trans or cr for default: (S)
(VAOPASV BECAUSE-OF * A CRUSHED ELECTRICAL PART ON .. SAID)
options-are h b s case-trans or cr for default: NIL 1 conse 2
(BECAUSE-OF * A CRUSHED ELECTRICAL PART ON ... SAID)
options-are h b s case-trans or cr for default: NIL s

: **skipping now to the end**
(BECAUSE-OF COMMA N VAO *)
options-are h b s case-trans or cr for default: (1 OBJ 2) 1 agt 2
(BECAUSE-OF COMMA VAO *)
options-are h b s case-trans or cr for default: NIL 1 obj 3

Figure 9
Illustration of case grammar acquisition.

8. Discussion and Conclusions

It seems remarkable that although the theory of context-sensitive grammars appeared in Chomsky (1957), formal context-sensitive rules seem not to have been used pre-



The launch of discovery and its five astronauts has been delayed at-least two days until march eleventh because-of a crushed electrical part on a main engine comma officials said.

Figure 10
Case analysis of a sentence.

viously in computational parsing. As researchers we seem simply to have assumed, without experimentation, that context-sensitive grammars would be too large and cumbersome to be a practical approach to automatic parsing. In fact, context-sensitive, binary phrase structure rules with a context composed of the preceding three stack symbols and the next five input symbols,

$$stack_{1-3} \text{ binary-rule } input_{1-5} \rightarrow \text{operation}$$

provide several encouraging properties.

- The linguist uses the full context of the sentence to make a simple decision: either shift a new element onto the stack or combine the top two elements into a phrase category.
- The system compiles a CS rule composed of ten symbols, the top five elements of the stack and the next five elements of the input string. The context of the embedded binary rule specializes that rule for use in similar environments, thus providing selection criteria to the parser for the choice of *shift* or *reduce*, and for assigning the phrase name that has most frequently been used in similar environments. The context provides a simple but powerful approach to *preference parsing*.

- As a result, a deterministic bottom-up parser is notably successful in finding precisely the parse tree that the linguist who constructed the analysis of a sentence had in mind—and this is true whether the grammar is stored as a trained neural network or in the form of hash-table entries.
- Despite the large combinatoric space for selecting 1 of 64 symbols in each of 10 slots in the rules—64^{10} possible patterns—experiments in accumulating phrase structure grammar suggest that a fairly complete grammar will require only about 25,000 CS rules.
- It is also the case that when redundant rules are removed the CS grammar is reduced by a factor of four and still maintains its accuracy in parsing.
- Because of the simplicity and regular form of the rule structure, it has proved possible to construct an acquisition system that greatly facilitates the accumulation of grammar. The acquisition system presents contexts and suggests operations that have previously been used with similar contexts; thus it helps the linguist to maintain consistency of judgments.
- Parsing with context-sensitive rules generalizes from phrase structure rewriting rules to the transformational rules required by case analysis. Since the case analysis rules retain a regular, simple form, the acquisition system also generalizes to case grammar.

Despite such advantageous properties, a few cautions should be noted. First, the deterministic parsing algorithm is sufficient to apply the CDG to the sentences from which the grammar was derived, but to accomplish effective generalization to new sentences, a bandwidth parsing algorithm that follows multiple parsing paths is superior. Second, the 99% accuracy of the parsing will deteriorate markedly if the dictionary lookup makes errors in word assignment. Thirdly, the shift/reduce parsing is unable to give correct analyses for such embedded discontinuous constituents as "I saw the man yesterday who ...." Finally, the actual parsing structures that we have presented here are skeletal. We did not mark mood, aspect or tense of verbs, number for nouns, or deal with long distance dependencies. We do not resolve pronoun references; and we do not complete ellipses in conjunctive and other constructions.

Each of these shortcomings is the subject of continuing research. For the present, the output of the case parser provides the nested, labeled, propositional structures which, supported by a semantic knowledge base, we have customarily used to accomplish focus-tracking of topics through a continuous text to compute labeled outlines and other forms of discourse structure (Seo 1990; Rim 1990; Alterman 1985). During this process of discourse analysis, some degapping, completion of ellipsis, and pronoun resolution is accomplished.

8.1 Conclusions

From the studies presented in this paper we conclude:

1. Context-Dependent Grammars (CDGs) are computationally and conceptually tractable formalisms that can be composed easily by a linguist and effectively used by a deterministic parser to compute phrase structures and case analyses for subsets of newspaper English.

2. The contextual portions of the CDG rules and the scoring formula that selects the rule that best matches the parsing context allow a deterministic parser to provide *preferred* parses, reflecting the linguist's meaning-based judgments.
3. The CDG acquisition system described earlier simplifies linguistic judgments and greatly improves a linguist's ability to construct relatively large grammars rapidly.
4. Although a deterministic, bottom-up parser has been sufficient to provide highly accurate parses for the 345-sentence sample of news text studied here, we believe that a multi-path parser proves superior in its ability to analyze sentences beyond the sample on which the grammar was developed.
5. With 3,843 compressed CDG rules, the acquisition system is about 85% accurate in suggesting the correct parsing for constituents from texts it has not experienced.
6. For phrase structure analysis, the context-free core of the CS rules will be 99% complete when we have accumulated about 25,000 CS rules. At that point it should be possible for a multi-path parser to find a satisfactory analysis for almost all news story sentences.

We have shown that the acquisition and parsing techniques apply also to CDG grammars for computing structures of case propositions to represent sentences. In this application, however, much more research is needed to better define linguistic systems for case analysis, and for their application to higher levels of natural language understanding.

Acknowledgments

This work was partially supported by the Army Research Office under contract DAAG29-84-K-0060.

References

Alterman, Richard (1985). "A dictionary based on concept coherence," *Artificial Intelligence*, 25, 153–186.

Allen, James (1987). *Natural Language Understanding*. Benjamin Cummings.

Allen, Robert (1987). "Several studies on natural language and back propagation." In *Proceedings, International Conference on Neural Networks*. San Diego.

Berwick, Robert C. (1985). *The Acquisition of Syntactic Knowledge*, Vol. 2, 335–341. MIT Press.

Chomsky, Noam (1957). *Syntactic Structures*. Mouton.

Cook, Walter (1989). *Case Grammar Theory*. Georgetown University Press.

Gazdar, Gerald (1988). "Applicability of indexed grammars to natural languages." In *Linguistic Theory and Computer Applications*, edited by P. Whitelock, et al., Academic Press, 37–67.

Gazdar, Gerald, and Mellish, Chris (1989). *Natural Language Processing in LISP*. Addison-Wesley.

Gazdar, Gerald; Klein, E.; and Pullum, G.; and Sag, I. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press.

Jackendoff, Ray (1983). *Semantics and Cognition*, MIT Press, Cambridge, Mass., 1983.

Joshi, Aravind (1987). "An introduction to tree-adjoining grammars" In *Mathematics of Language*, edited by A. Manaster-Ramer, 87–114. John Benjamins.

Leow, Wee-Keng, and Simmons, R. F. (1990). "A constraint satisfaction network for case analysis," AI Technical Report AI90-129, Department of Computer Science, University of Texas, Austin.

Marcus, M. P. (1980). *A Theory of Syntactic Recognition for Natural Language*. MIT Press.

McClelland, J. L., and Kawamoto, A. H. (1986). "Mechanisms of sentence processing: Assigning roles to constituents." In *Parallel Distributed*

Processing, Vol. 2, edited by J. L. McClelland and D. E. Rumelhart, MIT Press, 272–326.

Miikkulainen, Risto, and Dyer, M. (1989). "A modular neural network architecture for sequential paraphrasing of script-based stories," Artificial Intelligence Lab., Department of Computer Science, UCLA.

Rim, Hae-Chang (1990). *Computing outlines from descriptive texts*. Doctoral dissertation, University of Texas, Austin.

Rumelhart, David E.; Hinton, G. E.; and Williams, R. J. (1986). "Learning internal representations by error propagation." In *Parallel Distributed Processing*, edited by D. E. Rumelhart and J. L. McClelland, MIT Press, 318–362.

Schank, Roger C. (1980). "Language and memory," *Cognitive Science*, 4(3).

Seo, Jungyun (1990). *Text driven construction of discourse structures for understanding descriptive texts*. Doctoral dissertation, University of Texas, Austin.

Sejnowski, Terrence J., and Rosenberg, C. (1988). "NETtalk: A parallel network that learns to read aloud." In *Neurocomputing*, edited by Anderson and Rosenfeld, MIT Press.

Shieber, Stuart M. (1986). *An Introduction to Unification Based Approaches to Grammar*. University of Chicago Press.

Simmons, Robert F. (1984). *Computations from the English*. Prentice Hall.

Simmons, Robert F., and Yu, Yeong-Ho (1990). "Training a neural network to be a context sensitive grammar." In *Proceedings, 5th Rocky Mountain AI Conference*. Las Cruces, NM.

Tomita, M. (1985). *Efficient Parsing for Natural Language*. Kluwer Academic Publishers.

Yu, Yeong-Ho, and Simmons, R. F. (1990). "Descending epsilon in back-propagation: A technique for better generalization," in press, *Proc. Int. Jt. Conf. Neural Networks*, San Diego.

Appendix A. Rules from the Case Acquisition Session

Blanks in the 10-symbol vectors are signified by the letter B.

((THE LAUNCH OF DISCOVERY AND ITS FIVE ASTRONAUTS HAS BEEN DELAYED AT-LEAST TWO DAYS UNTIL MARCH ELEVENTH BECAUSE-OF A CRUSHED ELECTRICAL PART ON A MAIN ENGINE COMMA OFFICIALS SAID)
(ART N OF N AND PPRON ADJ N VHAV VBE VAO AT-LEAST ADJ N UNTIL N N BECAUSE-OF ART PPART ADJ N ON ART N N COMMA N VAO)

```
(((B B B B B ART N OF N AND) (S))
 ((B B B B ART N OF N AND PPRON) (S))
 ((B B B ART N OF N AND PPRON ADJ) (1 DET 2))
 ((B B B B N OF N AND PPRON ADJ) (S))
 ((B B B N OF N AND PPRON ADJ N) (S))
 ((B B N OF N AND PPRON ADJ N VHAV) (S))
 ((B N OF N AND PPRON ADJ N VHAV VBE) (S))
 ((N OF N AND PPRON ADJ N VHAV VBE VAO) (S))
 ((OF N AND PPRON ADJ N VHAV VBE VAO AT-LEAST) (S))
 ((N AND PPRON ADJ N VHAV VBE VAO AT-LEAST ADJ) (1 MOD 2))
 ((OF N AND PPRON N VHAV VBE VAO AT-LEAST ADJ) (1 POSSBY 2))
 ((N OF N AND N VHAV VBE VAO AT-LEAST ADJ) (3 2 1))
 ((B B N OF N VHAV VBE VAO AT-LEAST ADJ) (3 2 1))
 ((B B B B N VHAV VBE VAO AT-LEAST ADJ) (S))
 ((B B B N VHAV VBE VAO AT-LEAST ADJ N) (S))
 ((B B N VHAV VBE VAO AT-LEAST ADJ N UNTIL) (1 AUX 2))
 ((B B B N VBE VAO AT-LEAST ADJ N UNTIL) (S))
 ((B B N VBE VAO AT-LEAST ADJ N UNTIL N) (1 VBE 2 VAOPASV))
 ((B B B N VAOPASV AT-LEAST ADJ N UNTIL N) (1 OBJ 2))
 ((B B B B VAOPASV AT-LEAST ADJ N UNTIL N) (S))
 ((B B B VAOPASV AT-LEAST ADJ N UNTIL N N) (S))
 ((B B VAOPASV AT-LEAST ADJ N UNTIL N N BECAUSE-OF) (S))
 ((B VAOPASV AT-LEAST ADJ N UNTIL N N BECAUSE-OF ART) (1 MOD 2))
 ((B B VAOPASV AT-LEAST N UNTIL N N BECAUSE-OF ART) (3 2 1))
 ((B B B B VAOPASV UNTIL N N BECAUSE-OF ART) (S))
 ((B B B VAOPASV UNTIL N N BECAUSE-OF ART PPART) (S))
 ((B B VAOPASV UNTIL N N BECAUSE-OF ART PPART ADJ) (S))
 ((B VAOPASV UNTIL N N BECAUSE-OF ART PPART ADJ N) (1 NMOD 2))
 ((B B VAOPASV UNTIL N BECAUSE-OF ART PPART ADJ N) (3 2 1))
 ((B B B B VAOPASV BECAUSE-OF ART PPART ADJ N) (S))
 ((B B B VAOPASV BECAUSE-OF ART PPART ADJ N ON)
  (1 CONSE 2))
 ((B B B B BECAUSE-OF ART PPART ADJ N ON) (S))
 ((B B B BECAUSE-OF ART PPART ADJ N ON ART) (S))
 ((B B BECAUSE-OF ART PPART ADJ N ON ART N) (S))
 ((B BECAUSE-OF ART PPART ADJ N ON ART N N) (S))
 ((BECAUSE-OF ART PPART ADJ N ON ART N N COMMA) (1 MOD 2))
 ((B BECAUSE-OF ART PPART N ON ART N N COMMA) (1 MOD 2))
```

```
((B B BECAUSE-OF ART N ON ART N N COMMA) (1 DET 2))
((B B B BECAUSE-OF N ON ART N N COMMA) (S))
((B B BECAUSE-OF N ON ART N N COMMA N) (S))
((B BECAUSE-OF N ON ART N N COMMA N VAO) (S))
((BECAUSE-OF N ON ART N N COMMA N VAO B) (S))
((N ON ART N N COMMA N VAO B B) (1 NMOD 2))
((BECAUSE-OF N ON ART N COMMA N VAO B B) (1 DET 2))
((B BECAUSE-OF N ON N COMMA N VAO B B) (3 2 1))
((B B B BECAUSE-OF N COMMA N VAO B B) (2 ANTE 1))
((B B B B BECAUSE-OF COMMA N VAO B B) (S))
((B B B BECAUSE-OF COMMA N VAO B B B) (S))
((B B BECAUSE-OF COMMA N VAO B B B B) (S))
((B BECAUSE-OF COMMA N VAO B B B B B) (1 AGT 2))
((B B BECAUSE-OF COMMA VAO B B B B B) (1 OBJ 3))))
```