The Acquisition and Application of Context Sensitive Grammar for English

Robert F. Simmons and Yeong-Ho Yu @cs.texas.edu

Department of Computer Sciences, AI Lab
University of Texas, Austin Tx 78712

Abstract

A system is described for acquiring a context-sensitive, phrase structure grammar which is applied by a best-path, bottom-up, deterministic parser. The grammar was based on English news stories and a high degree of success in parsing is reported. Overall, this research concludes that CSG is a computationally and conceptually tractable approach to the construction of phrase structure grammar for news story text.[1]

1 Introduction

Although many papers report natural language processing systems based in part on syntactic analysis, their authors typically do not emphasize the complexity of the parsing and grammar acquisition processes that were involved. The casual reader might suppose that parsing is a well understood, minor aspect in such research. In fact, parsers for natural language are generally very complicated programs with complexity at best of $O(n^3)$ where n is the number of words in a sentence. The grammars they usually use are technically, "augmented context free" where the simplicity of the context-free form is augmented by feature tests, transformations, and occasionally arbitrary programs. The combination of even an efficient parser with such intricate grammars may greatly increase the computational complexity of the system [Tomita 1985]. It is extremely difficult to write such grammars and they must frequently be revised to maintain internal consistency when applied to new texts. In this paper we present an alternative approach using context-sensitive grammar to enable preference parsing and rapid acquisition of CSG from example parsings of newspaper stories.

Chomsky[1957] defined a hierarchy of grammars including context-free and context-sensitive ones. For natural language a grammar distinguishes terminal, single element constituents such as parts of speech from non-terminals which are phrase-names such as NP, VP, ADVPH, or SNT[2] signifying multiple constituents.

A context-free grammar production is characterized as a rewrite rule where a non-terminal element as a left-side is rewritten as multiple symbols on the right.

Snt → NP + VP

Such rules may be augmented by constraints to limit their application to relevant contexts.

Snt → NP + VP / anim(np), agree(nbr(np),nbr(vp))

To the right of the slash mark, the constraints are applied by an interpretive program and even arbitrary code may be included; in this case the interpreter would recognize that the NP must be animate and there must be agreement in number between the NP and the VP. Since this is such a flexible and expressive approach, its many variations have found much use in application to natural language applications and there is a broad literature on Augmented Phrase Structure Grammar [Gazdar et. al. 1985], Unification Grammars of various types [Shieber 1986], and Augmented Transition Networks [Allen, J. 1987, Simmons 1984].

In context-sensitive grammars, the productions are restricted to rewrite rules of the form,

$$uXv \rightarrow uYv$$

where u and v are context strings of terminals or nonterminals, and X is a non-terminal and Y is a non-empty string . That is, the symbol X may be rewritten as as the string Y in the context $u \cdots v$. More generally, the right-hand side of a context-sensitive rule must contain at least as many symbols as the left-hand side.

Excepting Joshi's Tree Adjoining Grammars which are shown to be "mildly context-sensitive," [Joshi 1987] context-sensitive grammars found little or no use among natural language processing (NLP) researchers until the reoccurrance of interest in Neural Network computation. One of the first suggestions of their potential utility came from Sejnowski and Rosenberg's NETtalk [1988], where 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 Miikulainen and

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

[2] NounPhrase, VerbPhrase, AdverbialPhrase, Sentence

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 context sensitive translations between English and German. 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.

An explicit use of context-sensitive grammar was developed by Simmons and Yu [1990] to solve the problem of accepting indefinitely long, recursively embedded strings of language for training a neural network. However although the resulting neural network was trained as a satisfactory grammar, there was a problem of scale-up. Training the network for even 2000 rules took several days, and it was foreseen that the cost of training for 10-20 thousand rules would be prohibitive. This led us to investigate the hypothesis that storing a context-sensitive grammar in a hash-table and accessing it using a scoring function to select the rule that best matched a sentence context would be a superior approach.

In this paper we describe a series of experiments in acquiring context-sensitive grammars (CSG) from newspaper stories, and a deterministic parsing system that uses a scoring function to select the best matching context sensitive rules from a hash-table. We have accumulated 4000 rules from 92 sentences and found the resulting CSG to be remarkably accurate in computing exactly the parse structures that were preferred by the linguist who based the grammar on his understanding of the text. We show that the resulting grammar generalizes well to new text and compresses to a fraction of the example training rules.

2 Context-Sensitive Parsing

The simplest form of parser applies two operations *shift* or *reduce* to an input string and a stack. A sequence of elements on the stack may be reduced — rewritten as a single symbol, or a new element may be shifted from the input to the stack. Whenever a reduce occurs, a subtree of the parse is constructed, dominated by the new symbol and placed on the stack. The input and the stack may both be arbitrarily long, but the parser need only consult the top elements of the stack and of the input. The parse is complete when the input string is empty and the stack contains only the root symbol of the parse tree. Such a simple approach to parsing has been used frequently to introduce methods of CFG parsing in texts on computer analysis of natural language [J. Allen 1987], but it works equally well with CSG. In our application to phrase structure analysis, we further constrain the reduce operation to refer to only the top two elements of the stack

2.1 Phrase Structure Analysis with CFG

For shift/reduce parsing, a phrase structure analysis takes the form of a sequence of states, each comprising a condition of the stack and the input string. The final state in the parse is an empty input string and a stack containing only the root symbol, SNT. In an unambiguous analysis, each state is followed by exactly one other; thus each state can be viewed as the left-half of a CSG production whose right-half is the succeeding state.

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

News story sentences, however, may be very long, sometimes exceeding fifty words and the resulting parse states would make cumbersome rules of varying lengths. To obtain manageable rules we limit the stack and input parts of the state to five symbols each, forming a ten symbol pattern for each state of the parse. In the example of Figure 1 we separate the stack and input parts with the symbol "*", as we illustrate the basic idea on the sentence "The late launch from Alaska delayed interception." The symbol `b` 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.

The analysis terminates successfully with an empty input string and the single symbol "snt" on the stack. 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.

The grammar could now be recorded as pairs of successive states as below:

b b b np p * n v n b b → b b np p n * v n b b b
b b np p n * v n b b b → b b b np pp * v n b b b

but some economy can be achieved by summarizing the right-half of a rule as the operations, shift or reduce, that produce it from the left-half. So for the example immediately above, we record:

b b b np p * n v n b b → (S)

b b np p n * v n b b b → (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 we create a *windowed context* of 10 symbols 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 CFG; thus this CSG embeds a CFG.

```
The late launch from Alaska
art  adj    n    p     n

delayed interception.
 v       n
b b b b b * art adj n p n
b b b b art * adj n p n v
b b b art adj * n p n v n
b b art adj n * p n v n b
b b b art np * p n v n b
b b b b np * p n v n b
b b b np p * n v n b b
b b np p n * v n b b b
b b b np pp * v n b b b
b b b b np * v n b b b
b b b np v * n b b b b
b b np v n * b b b b b
b b b np vp * b b b b b
b b b b snt * b b b b b
```

Figure 1: Successive Stack/Input States in a Parse

2.2 Algorithm for Shift/Reduce Parser

The algorithm used by the Shift/Reduce parser is described in Figure 2. Essentially, the algorithm shifts elements from the input onto the stack under the control of the CSG productions. It can be observed that unlike most grammars which include only rules for reductions, this one has rules for recognizing shifts as well. The reductions always apply to the top two elements of the stack and it is often the case that in one context a pair of stack elements lead to a shift, but in another context the same pair can be reduced.

An essential aspect of this algorithm is to consult the CFG to find the left-half of a rule that matches the sentence context. The most important part of the rule is the top two stack elements, but for any such pair there may be multiple contexts leading to shifts or various reductions, so it is the other eight context elements that decide which rule is most applicable to the current state of the parse. Since many thousands of contexts can exist, an exact match cannot always be expected and therefore a scoring function must be used to discover the best matching rule.

```
CS-SR-Parser(Input,Csg)

Input is a string of syntactic classes
Csg is the given CSG production rules.


Stack := empty
do until(Input = empty and Stack = (SNT))
        Windowed-context := Append(Top_five(stack),First_five(input))
        Operation := Consult_CSG(Window-context,Csg)
        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_CSG consults the given CSG 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 2: Context Sensitive Shift Reduce Parser

One of the exciting aspects of neural network research is the ability of a trained NN system to discover closest matches from a set of patterns to a given one. We studied Sejnowski and Rosenberg's [1988] analyses of the weight matrices resulting from training NETtalk. They reported that the weight matrix had maximum weights relating the character in the central window to the output phoneme, with weights for the surrounding context characters falling off with distance from the central window. We designed a similar function with maximum weights being assigned to the top two stack elements and weights decreasing in both directions with distance from those positions. The scoring function 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 $\mu(r_{i4}, c_4) \cdot \mu(r_{i5}, c_5) = 1$.
The statement above is achieved by accessing the hash table with the top two elements of the stack, c_4, c_5 to produce 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 while 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.

3 Experiments with CSG

To support the claim that CSG systems are an improvement over Augmented CFG, a number of questions need be answered.

- Can they be acquired easily?
- Do they reduce ambiguity in phrase structure analysis?
- How well do CSG rules generalize to new texts?
- How large is the CSG that encompasses most of the syntactic structures in news stories?

3.1 Acquisition of CSG

It has been shown that our CSG productions are essentially a recording of the states from parsing sentences. Thus it was easy to construct a grammar acquisition system to present the successive states of a sentence to a linguist user, accepting and recording the linguist's judgements of shift or reduce. This system has evolved to a sophisticated grammar acquisition/editing program that prompts the user on the basis of the rules best fitting the current sentence context. It's lexicon also suggests the choice of syntactic class for words in context. Generally it reduces the linguistic task of constructing a grammar to the much simpler task of deciding for a given context whether to shift input or to rewrite the top elements of the stack as a new constituent. It reduces a vastly complex task of grammar writing to relatively simple, concrete judgements that can be made easily and reliably.

Using the acquisition system, it has been possible for linguist users to provide example parses at the rate of two or three sentences per hour. The system collects the resulting states in the form of CSG productions, allows the user to edit them, and to use them for examining the resulting phrase structure tree for a sentence. To obtain the 4000+ rules examined below required only about four man-weeks of effort (much of which was initial training time.)

3.2 Reduced Ambiguity in Parsing

Over the course of this study six texts were accumulated. The first two were brief disease descriptions from a youth encyclopedia; the remaining four were newspaper texts. Figure 1 characterizes each article by the number of CSG rules or states, number of sentences, the range of sentence lengths, and the average number of words per sentence.

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 the Text Corpus

It can be seen that the news stories were fairly complex texts with average sentence lengths ranging from 22 to 29 words per sentence. A total of 92 sentences in over 2000 words of text resulted in 4062 CSG productions.

It was noted earlier that in each CFG production there is an embedded context-free rule and that the primary function of the other eight symbols for parsing is to select the rule that best applies to the current sentence state. When the linguist makes the judgement of shift or reduce, he or she is considering the entire meaning of the sentence to do so, and is therefore specifying a semantically *preferred* parse. The parsing system has access only to limited syntactic information, five syntactic symbols on the stack, and five input word classes and the parsing algorithm follows only a single path. How well does it work?

The CSG was used to parse the entire 92 sentences with the algorithm described in Figure 2 augmented with instrumentation to compare the constituents the parser found with those the linguist prescribed. 88 of the 92 sentences exactly matched the linguist's parse. The other four cases resulted in perfectly reasonable complete parse trees that differed in minor ways from the linguist's pre-

scription. As to whether any of the 92 parses are truly "correct", that is a question that linguists could only decide after considerable study and discussion. Our claim is only that the grammars we write provide our own preferred interpretations — useful and meaningful segmentation of sentences into trees of syntactic constituents.

Figure 3 displays the tree of a sentence as analyzed by the parser using CSG. It is a very pleasant surprise to discover that using context sensitive productions, an elementary, deterministic, parsing algorithm is adequate to provide (almost) perfectly correct, unambiguous analyses for the entire text studied.

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.

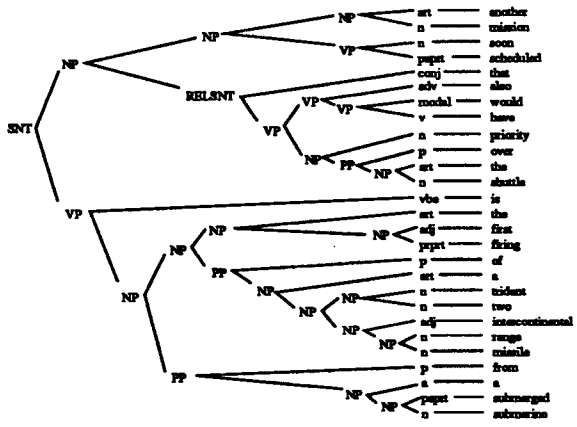

Figure 3: Sentence Parse

3.3 Generalization of CSG

One of the first questions considered was what percent of new constituents would be recognized by various accumulations of CSG. We used a system called *union-grammar* that would only add a rule to the grammar if the grammar did not already predict its operation. The black line of Figure 4 shows successive accumulations of 400-rule segments of the grammar after randomizing the ordering of the rules. Of the first 400 CS rules 50% were new; and for an accumulation of 800, only 35% were new. When 2000 rules had been experienced the curve is flattening to an average of 20% new rules. This curve tells us that if the acquisition system uses the current grammar to suggest operations to the linguist, it will be correct about 4 out of 5 times and so reduce the linguist's efforts accordingly. The curve also suggests that our collection of rule examples has about 80% redundancy in that earlier rules can predict newcomers at that level of accuracy. On the down-side, though, it shows that only 80% of the constituents of a new sentence will be recognized, and thus the probability of a correct parse for a sentence never seen before is very small. We experimented with a grammar of 3000 rules to attempt to parse the new shuttle text, but found that only 2 of 14 new sentences were parsed correctly.

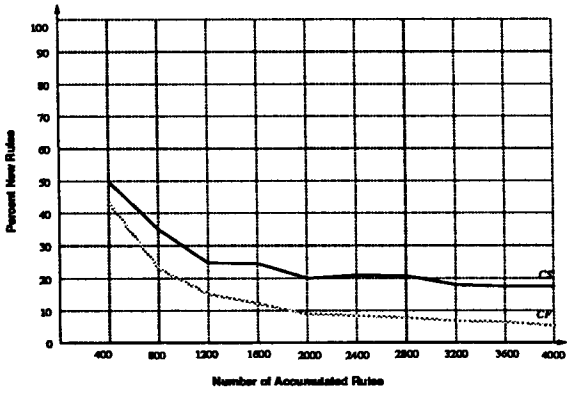

Figure 4: Generalization of CSG Rules

If two parsing grammars equally well account for the same sentences, the one with fewer rules is less redundant, more general, and the one to be preferred. We used *union-grammar* to construct the "minimal grammar" with successive passes through 3430 rules, as shown in Figure2. The first pass found 856 rules would account for the rest. A second pass of the 3430 rules against the 856 extracted by the first pass resulted in the addition of 26 more rules, adding rules that although recognized by earlier rules found interference as a result of later ones. The remaining 8 rules discovered in the next pass are apparently identical patterns resulting in differing operations — contradictories that need to be studied and resolved. The resulting minimal grammar totaling 895 rules succeeds in parsing the texts with only occasional minor differences from the linguist's prescriptions. We must emphasize that the unretained rules are not identical but only *similar* to those in the minimal grammar.

Pass	*Unretained*	*Retained*	*Total Rules*
1	2574	856	3430
2	3404	26	3430
3	3422	8	3430
4	3425	5	3430

Table 2: Four Passes with Minimal Grammar

3.4 Estimated Size of Completed CSG

A question, central to the whole argument for the utility of CSG, is how many rules will be required to account for the range of structures found in news story text? Refer again to Figure 4 to try to estimate when the black line, CS, will intersect the abscissa. It is apparent that more data is needed to make a reliable prediction.

Let us consider the gray line, labeled CF that shows how many new context-free rules are accumulated for 400 CSG rule increments. This line rapidly decreases to about 5% new CFG rules at the accumulation of 4000 CSG productions. We must recall that it is the embedded context-free binary rule that is carrying the most weight in determining a constituent, so let us notice some of the CFG properties.

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[3]: a shift or a reduction to another symbol. Among 4000 CSG rules, we studied how many different CFG rules can be derived by eliminating the context. We found 551 different CFG rules that used 421 different left-side pairs of symbols. This shows that a given context free pair of symbols averages 1.3 different operations.

Then, as we did with CSG rules, we measured how many new CFG rules were added in an accumulative fashion. The shaded line of Figure 4 shows the result. Notice that the line has descended to about 5% errors at 4000 rules. To make an extrapolation easier, a log-log graph shows the same data in Figure 5. From this graph, it can be predicted that, after about 25000 CSG rules are accumulated, the grammar will encompass an almost complete CFG component. Beyond this point, additional CSG rules will add 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 CSG reaches that point, a multi-path, beam-search parser would be

[3] Actually, there are many fewer than 65 possible operations since the stack elements can be reduced only to *non-terminal* symbols.

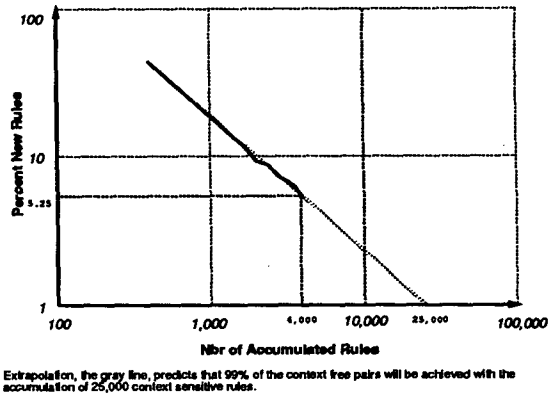

Figure 5: Log-Log Plot of New CFG Rules

able to parse most newswire stories very reliably. This belief is based on our observation that most failures in parsing new sentences with a single-path parser result from a *dead-end* sequence; i.e., by making a wrong choice in the middle, the parsing ends up with a state where no rule is applicable. The beam-search parser should be able to recover from this failure and produce a reasonable parse.

4 Discussion and Conclusions

Neural network research showed us the power of contextual elements for selecting preferred word-sense and parse-structure in context. But since NN training is still a laborious, computation-intensive process that does not scale well into tens of thousands of patterns, we chose to study context-sensitive grammar in the ordinary context of sequential parsing with a hash-table representation of the grammar, and a scoring function to select the rule most applicable to a current sentence context. We find that context-sensitive, binary phrase structure rules with a context comprising the three preceding stack symbols and the oncoming five input symbols,

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

provide unexpected advantages for acquisition, the computation of preferred parsings, and generalization.

A linguist constructs a CSG with the acquisition system by demonstrating successive states in parsing sentences. The acquisition system presents the state resulting from each shift/reduce operation that the linguist prescribes, and it uses the total grammar so far accumulated to find the best matching rule and so prompt the linguist for the next decision. As a result CSG acquisition is a rapid process that requires only that a linguist decide for a given state to reduce the top two elements of the stack, or to shift a new input element onto the stack. Since the current grammar is about 80% accurate in its predictions, the linguist's task is reduced by the prompts to an alert observation and occasional correction of the acquisition system's choices.

The parser is a bottom-up, deterministic, shift/reduce program that finds a best sequence of parse states for a sentence according to the CSG. When we instrument the parser to compare the constituents it finds with those originally prescribed by a linguist, we discover almost perfect correspondence. We observe that the linguist used judgements based on understanding the meaning of the sentence and that the parser using the contextual elements of the state and matching rules can successfully reconstruct the linguist's parse, thus providing a purely syntactic approach to preference parsing.

The generalization capabilities of the CSG are strong. With the accumulation 2-3 thousand example rules, the system is able to predict correctly 80% of subsequent parse states. When the grammar is compressed by storing only rules that the accumulation does not already correctly predict, we observe a compression from 3430 to 895 rules, a ratio of 3.8 to 1. We extrapolate from the accumulation of our present 4000 rules to predict that about 25 thousand rule examples should approach completion of the CF grammar for the syntactic structures usually found in news stories. For additional fine tuning of the context selection we might suppose we create a total of 40 thousand example rules. Then if the 3.8/1 compression ratio holds for this large a set of rules, we could expect our final grammar to be reduced from 40 to about 10 thousand context sensitive rules.

In view of the large combinatoric space provided by the ten symbol parse states — it could be as large as 64^{10} — our prediction of 25-40 thousand examples as mainly sufficient for news stories seems contra-intuitive. But our present grammar seems to have accumulated 95% of the binary context free rules — 551 of about 4096 possible binaries or 13% of the possibility space. If 551 is in fact 95% then the total number of binary rules is about 580 or only 14% of the combinatoric space for binary rules. In the compressed grammar, there are only 421 different left-side patterns for the 551 rules, and we can notice that each context-free pair of symbols averages only 1.3 different operations. We interpret this to mean that we need only enough context patterns to distinguish the different operations associated with binary combinations of the top two stack elements; since there are fewer than an average of two, it appears reasonable to expect that the context-sensitive portion of the grammar will not be excessively large.

We conclude,

- Context sensitive grammar is a conceptually and computationally tractable approach to unambiguous parsing of news stories.
- The context of the CSG rules in conjunction with a scoring formula that selects the rule best matching the current sentence context allow a deterministic parser to provide *preferred* parses reflecting a linguist's meaning-based judgements.
- The CSG acquisition system simplifies a linguist's judgements and allows rapid accumulation of large grammars.
- CSG grammar generalizes in a satisfactory fashion and our studies predict that a nearly-complete accounting for syntactic phrase structures of news stories can be accomplished with about 25 thousand example rules.

REFERENCES

Allen, Robert, "Several Studies on Natural Language and Back Propagation", Proc. Int. Conf. on Neural Networks, San Diego, Calif., 1987.

Allen, James, *Natural Language Understanding,* Benjamin Cummings, Menlo Park, Calif., 1987. .

Chomsky, Noam, *Syntactic Structures,* Mouton, The Hague, 1957.

Gazdar, Gerald, Klein E., Pullum G., and Sag I., *Generalized Phrase Structure Grammar*, Harvard Univ. Press, Boston, 1985.

Joshi, Aravind K., "An Introduction to Tree Adjoining Grammars." In Manaster-Ramer (Ed.),*Mathematics of Language*, John Benjamins, Amsterdam, Netherlands, 1985.

McClelland, J.L., and Kawamoto, A.H., "Mechanisms of Sentence Processing: Assigning Roles to Constituents," In McClelland J. L. and Rumelhart, D. E., *Parallel Distributed Processing,* Vol. 2. 1986.

Miikkulainen, Risto, and Dyer, M., "A Modular Neural Network Architecture for Sequential Paraphrasing of Script-Based Stories", Artif. Intell. Lab., Dept. Comp. Sci., UCLA, 1989.

Shieber, Stuart M., *An Introduction to Unification Based Approaches to Grammar*, Chicago Univ. Press, Chicago, 1986.

Sejnowski, Terrence J., and Rosenberg, C., "NETtalk: A Parallel Network that Learns to Read Aloud", in Anderson and Rosenfeld (Eds.) *Neurocomputing*, MIT Press., Cambridge Mass., 1988.

Simmons, Robert F. *Computations from the English*, Prentice-Hall, Engelwood Cliffs, New Jersey, 1984.

Simmons, Robert F. and Yu, Yeong-Ho, "Training a Neural Network to be a Context Sensitive Grammar," Proc. 5th Rocky Mountain AI Conf. Las Cruces, N.M., 1990.

Tomita, M. *Efficient Parsing for Natural Language*, Kluwer Academic Publishers, Boston, Ma., 1985.

Yu, Yeong-Ho, and Simmons, R.F. "Descending Epsilon in Back-Propagation: A Technique for Better Generalization," In Press, *Proc. Int. Jt. Conf. Neural Networks*, San Diego, Calif., 1990.