16

1965 International Conference on Computational Linguistics

# SETS OF GRAMMARS BETWEEN CONTEXT-FREE AND CONTEXT-SENSITIVE

Peter Kugel

Technical Operations Research

South Avenue
Burlington, Mass.
U. S. A.

## A B S T R A C T

We discuss some sets of grammars whose generative power lies between that of the set of context-free grammars and that of the set of context-sensitive grammars. These sets are developed by subjecting generators of context-sensitive grammars to abstract versions of a "hardware" restriction to which the users of natural languages, unlike the describers of natural languages, might be subject.

The notion of a formal grammar was first introduced to provide formal models of techniques used by the describers of natural languages (linguists) (1). Later, formal grammars have been used as models of the capabilities of users of natural languages (See (2) for a review). Language users differ from language describers in being subject to restrictions on the amount of "hardware" that they have available to them and the amount of time that they have to perform their operations. Where the linguist has available (at least theoretically) an unlimited amount of material with which (pencil) and on which (paper) to store his intermediate results, it is probable that the internal organization of the natural language user may not permit him the use of such unlimited resources.

Therefore, when one uses a formal grammar as a model of the language user, one may consider the effects of subjecting such grammars to abstract versions of certain types of hardware limitations. One model in this vein is t hat of Yngve (3) which considers the natural language user to be like a device capable of dealing with context-free languages and then subjects it to further limitations. However, there are reasons for thinking that natural language users may have available to them powers beyond those of the context-free grammars. According to current views, these additional powers are those that are required to construct transformational grammars. Among these one might include the ability to permute the order of elements in a string and the ability to erase elements (4).

The ability to effect the permutation of elements is a property of context-sensitive grammars. However, context-sensitive grammars have additional drawbacks as models for the capabilities of the users of natural languages (1). Permitting erasure as an element the generation of a phrase marker has the difficulty that it is not always clear whether the resulting rewriting systems generate only recursive sets of strings.* These considerations suggest that one

*Thus, any semi-Thue system (For a definition see (5), p. 84) can be looked at as a context-free grammar which permits the shortening of strings (erasure). But semi-Thue systems are capable of generating non-recursive sets of strings ((5), Theorem 2.6, p. 93).

might want some context-sensitivity and some erasure but not enough to produce the undesirable features of context-sensitive grammar or of semi-Thue systems.

One way of getting at such grammars might be to consider a device for generating context-sensitive languages and subjecting it to abstract versions of t he types of hardware limitations to which the users of natural language users might be sjubect.

Assume that users of natural languages are information processing systems organized in the manner of the present-day digital computer. They have a storage unit (memory),a processing unit, and some input/output equipment. One way of suggesting the roles of these parts is to say that they correspond roughly to those parts of the handling of a natural language that are described by the semantic, syntactic and phonetic components of a language description respectively. Since our concern in this paper is largely with the syntactic component, we will consider limitations on the effects of limitations on the processing unit.

Suppose that the processing unit has the machinery for applying the rewriting rules of context-sensitive grammar, but that this application has to be done by changing the state of something like a register in the arithmetic unit of a present-day computer. Such a register can be looked at as a sequence of pigeon holes into which symbols can be placed. A rule then is applied to change the contents of the pigeonholes and the results are returned to the memory or output. To say that the registers have a given size is to say that there is only a fixed number of such pigeon-holes*. Such an assumption finds a formal analogue in the notion of a formal grammar as a restriction on the length of the strings that can appear on either side of the arrow in a rewriting rule. To say that a register has only n pigeon-holes is to say that the strings on either side of the arrow can contain at most n symbols.** However, such a restriction does not accomplish much that

*We are also assuming that there is no way of doing anything like multiple precision arithmetic.

**Or, equivalently, that the string on the right hand side of the arrow can contain at most n symbols.

is of interest, for it is easy to prove that:

Theorem 1: The set of all grammars that contain strings of no more than two symbols on either side of their rewriting rules has the generative power of the set of context-sensitive grammars. The set of all grammars that contain no more than three symbols in any rewriting rule has the generative power of the set of context-free grammars.*

It is clear from an examination of the proof of the first part of this theorem that the restriction on the length of the strings used in stating rules of the grammar is overcome by introducing new letters. Such an introduction of additional letters is common in proofs of theorems about formal grammars and it is reasonable so long as one is considering these grammars as models of the procedures used by language describers who have available to them a medium (pencil marks on paper) which is unlimited not only in amount, but which permits an unlimited variety of symbols within a given space (at least in theory).

The fact that language users might have to represent their grammatical catagories in a discrete rather than continuous medium suggests that one might limit the number of available (distinct) symbols that can appear in a rule of grammar. However, this restriction also is of no great interest since we can prove the following:

Theorem 2: There is a sense** in which the generative power of grammars whose rules can be expressed using only two distinct symbols in its vocabulary is equivalent to the set of all context-sensitive grammars.

Suppose, therefore, that one attempts to limit both of these simultaneously. Thus, let us define a "grammar of size (m, p)" as a grammar whose rules are constructed of strings (on either side of the rewriting rule's arrows) such that no string contains more than m occurrences of letters and such that the non-

*Definitions and proofs of theorems can be found in the appendix.

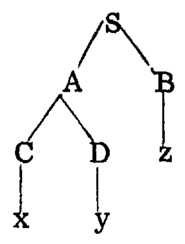
**Explicated in the appendix.

terminal vocabulary of the grammars contains no more than p distinct letters. Let us first consider such grammars as augmented simply by dictionaries. These grammars turn out to be curious hybrids. For one thing, given a size, there is only a finite number of grammars of that size (if one equates straightforward reletterings of the same grammars). Furthermore:

Theorem 3: The set of grammars of size (m, p) with dictionaries, for sufficiently large m and p, cannot generate all context-free languages and can generate some languages which are not context free.

Nevertheless, it is obvious that the union of the grammars of size (m, p) for all values of m and p has the generative power of the set of all context-sensitive grammars (since any context-sensitive grammar has some finite size).

These grammars are not particularly interesting because we have put limits on the amount of recursion that can appear in them. This can be overcome by permitting some recursion either in a pre- or post-processor, limiting recursion to context-free rules only. Thus, we are led to consider systems consisting of three parts in tandem. The first part is a context-free grammar, the second part is a grammar of size (m, p), and the third is a dictionary. Although such systems appear to be rather *ad hoc*, one can give some arguments for considering them. The arguments for the two grammars in tandem are roughly those for a context-free grammar followed by a transformational component. If we allow erasure in the final processing we can permit our intermediate string generated by the context-free grammar to be the phrase marker in something approximating Polish notation. Thus, the phrase marker:



could be represented by the string SACxDyBz. The context-sensitive grammar of restricted size could operate on these markers in the manner of a transformational component. The dictionary would contain rules of the form $X\rightarrow$, $A\rightarrow$, etc., to erase the non-terminal symbols. This argument suggests that if one wants such a system as a model for a natural language user one might consider different primitive operations in the part of the system that was to represent the transformational component. Thus, using the suggestion of (4) one might permit not only what we have been calling grammar rules but also rules which permute the order of strings directly such as rules of the form: $XYZ\rightarrow ZYX$. By making these primitive one makes them cost less of the "size" of the underlying grammar. The argument for allowing something like a dictionary is that something of this sort appears to be required for the phonetic component of a language description anyway.

Let us call such systems "grammar systems of size (m, p)." Those systems which have primitive permutation rules we might call "permutation systems." We can prove:

<u>Theorem 4:</u> Grammar systems define infinite hierarchies of languages $L_0 \ldots L_i \ldots$ such that (a) $L_0$ is the set of context-free languages; (2) $L_i \subsetneq L_j$ for j sufficiently greater than i and (3) the sum of the $L_i$ for all i is the set of context-sensitive languages.

We have suggested that if a natural language user is organized like a present-day digital computer he might find that the size of the registers in what corresponds to his "processing unit" might have an effect on the kinds of languages with which he could deal. We have given a rather preliminary sketch of how this might occur. Such effects appear however, to be critically dependent on the "machine code" of such a system, and in view of the current lack of knowledge as to what this code might be, it is not clear whether the kinds of notions that we have discussed have any applications in computational linguistics, even if the underlying notion of some sort of a "register" limitation applies to the competence of natural language users.

## APPENDIX

This appendix contains definitions of some of the terms used in the body of the paper and proofs of the theorems. We begin by defining some basic notions. A rewriting rule is a rule of the form $PhQ \rightarrow PHQ$ where P, Q, h, and H are (possibly empty*) strings. If h is a single letter and H is a non-empty string of letters a rewriting rule is called a grammar rule. A grammar (or a context-sensitive grammar) G is a single letter S together with a finite set of grammar rules. The alphabet of G is the set of all letters in rewriting rules of G. The non-terminal vocabulary of G is the set of all letters appearing on the left hand side of some grammar rule in G. The terminal vocabulary of G is the alphabet of G minus the non-terminal vocabulary of G. We will assume that S is always in the non-terminal vocabulary of G.

A set of rewriting rules which contains no non-terminal letters on the right-hand side of a rewriting rule is called a dictionary. If P and Q in all the grammar rules of G are empty, then G is a context-free grammar. A derivation of a string $S_n$ in a grammar G is a sequence of strings $S_1, \ldots, S_n$ such that $S_1$ is S and such that $S_{i+1}$ is the result of replacing some sequence of letters L in $S_i$ by a sequence of letters L' such that $L \rightarrow L'$ is one of the grammar rules of G. The language generated by a grammar G is the set of all strings M such that there exists a derivation of M in G, and such that M consists of only letters in the terminal vocabulary of G. Two sets of grammars that generate the same sets of languages are said to have the same generative power.

Theorem 1: (a) The set of grammars , none of whose rules contain more than four letters, has the same generative power as the set of context-sensitive grammars. (b) The set of grammars none of whose rules contain more than three letters has the generative power of the set of all context free grammars.

*PhQ, however, is not empty (i.e., not all of P, h, and Q can be empty).

Proof: (a) In order to prove this part of the theorem we need only present an effective procedure for replacing each of the rules of an arbitrary context-sensitive grammar G with a set of rules containing no more than two letters on either side of the arrow, and such that the generative power of the resulting grammar G' remains the same. Consider an arbitrary rule of the form $L \rightarrow R$, where $L = a_1 \ldots a_i \ldots a_j$ and $R = a_1 \ldots a_{i-1} b_1 \ldots b_k a_{i+1} \ldots a_j$. Replace this by the following new rules in which the $c_m$ and $d_m$ are new letters not in the alphabet of G:

<u>Rules</u> <u>Effect of Added Rules</u>[*]

$$\left.\begin{array}{l} a_1 a_2 \rightarrow a_1 c_2 \\ c_2 a_3 \rightarrow c_2 c_3 \\ \vdots \\ c_n a_{n+1} \rightarrow c_n c_{n+1} \\ \vdots \\ c_{i-1} a_i \rightarrow c_{i-1} c_i \end{array}\right\} \quad a_1 \ldots a_i (a_{i+1} \ldots a_j) \rightarrow a_1 c_2 \ldots c_i (a_{i+1} \ldots a_j)$$

$$\left.\begin{array}{l} a_{j-1} a_j \rightarrow d_{j-1} a_j \\ a_{j-2} d_{j-1} \rightarrow d_{j-2} d_{j-1} \\ \vdots \\ a_n d_{n+1} \rightarrow d_n d_{n+1} \\ \vdots \\ c_i d_{i-1} \rightarrow d_i d_{i-1} \end{array}\right\} \quad (a_1 c_2 \ldots) c_i a_{i+1} \ldots a_j \rightarrow (c_1 \ldots) d_i \ldots d_{j-1} a_j$$

[*] In schematizing the effects of a sequence of rules we have assumed an order in their application. However, where the order of application is arbitrary parts of the strings might be different if the order of application were different. These parts are indicated by surrounding them with parentheses.

| Rules | Effect of Added Rules |
|---|---|
| $\begin{array}{l} d_i \to b_1 d'_{i+1} \\ \vdots \\ d'_{i+n} \to b_{n-1} d'_{i+n+1} \\ \vdots \\ d'_{i+k-1} \to b_k \end{array}$ | $(\ldots)d_i(\ldots) \mapsto (\ldots)b_1 \ldots b_c(\ldots)$ |
| $\begin{array}{l} c_2 \to a_2 \\ \vdots \\ c_{i-1} \to a_{i-1} \end{array}$ | $a_1 c_2 \ldots c_{i-1}(\ldots) \mapsto a_1 \ldots a_{i-1}(\ldots)$ |
| $\begin{array}{l} d_{i+1} \to a_{i+1} \\ \vdots \\ d_j \to a_j \end{array}$ | $(\ldots)d_{i+1} \ldots d_j \to (\ldots)a_{i+1} \ldots a_j$ |

The equivalence in the other direction (i.e., the fact that all four letter grammars are at most context sensitive) is obvious.

(b) Because of the definition of a "grammar rule" rules containing three letters can only be of the form $a \to bc$ (and not $ab \to c$) so clearly all three letter rules are context-free. To produce a three letter equivalent of a longer context-free rule, say $a \to a_1 \ldots a_n$ one replaces it by the rules $a \to a_1 a'_2, \ldots, a'_i \to a_i a'_{i+1}, \ldots a'_n \to a_n$ where the $a'_j$ are new letters.

Theorem 2: The set of grammars containing only two letter together with a dictionary has the generative power of the set of all context sensitive grammars.

Proof: Let the two letters be 0 and 1. Again, it is only necessary to provide an effective procedure for replacing any rule in a given context-sensitive grammar with a new set of rules containing only two letter, plus some dictionary rules.

Suppose that G contains m rules and that the alphabet of G contains n letters. Let each rule be of the form $L^i \rightarrow R^i$ (for the i-th rule). We construct G' as follows:

To replace each rule $L^i \rightarrow R^i$ we add new rules as follows:

Rewrite each letter $a_j$ in $L^i$ by the string:

$$\underbrace{011\ldots11}_{m \text{ times}}00\underbrace{011\ldots\overset{\text{jth position}}{0}\ldots110}_{n \text{ times}} \qquad (= a'_j)$$

The first replacing rule takes the revised $L^i$ into $0111..\overset{\text{ith position}}{0}\ldots11000\ldots$ . The effect of this is to tag the string as being subjected to rule number i. The second replacing rule takes the 0 in the n-tuple of ones of the letter being replaced, and it turns it into a 1. If the only effect of the rule is to simply replace this letter by another letter, the rest of the new rules place the 0 in the n-tuple appropriately and then erase the tag in the left-most m—tuple to signal the end of the application of the rule. If the replaced letter is expanded then the replacements are added one letter at a time and the process is finished off by "untagging" the left-most m-tuple in the replacement for $L^i$.

The dictionary has the job of translating back into the vocabulary of G. It lacks any procedures for dealing with letters whose m-tuple is not all one's so that no intermediate product of a rule can be terminal. The dictionary is simply the set of rules $a'_j \rightarrow a_j$ for each $a_j$ in the terminal vocabulary of G. It is clear that G' generates exactly the same language as G.

This proof suggests a problem that might be of some interest. In devising a procedure for reducing the number of letters in what is, in some sense, a program, one is required to add new rules. These rules introduce intermediate products (strings), and the basic problem in the proof was that of devising a way in which these intermediate products can be prevented from being caught up by rules other than those that are intended to apply to them. We have used an extremely straightforward technique for doing this but this technique is costly in the size of the required strings.

One might ask what more efficient general procedures there are for such reduction. A reason for asking this question (other than a theoretic interest) is that the world as seen by a biological organism can be looked at as consisting of an arbitrary alphabet, the units (or letters) of which are the basic percepts of that organism. However, the organism's brain might have a fixed alphabet into which the processing of this (probably larger) alphabet has to be encoded. Such encoding would probably have to be done by an algorithm that avoided this crossing of intermediate products.

We define <u>a grammar of size (m, p)</u> as a set of grammar rules which has a non-terminal vocabulary of no more than m letters and such that no rule contains a string of more than p occurrences of letters on the right hand side of the arrow.

Theorem 3: The set of grammars of size (m, p) plus an arbitrary number of dictionary rules for sufficiently large m and p, cannot generate all context-free languages and can generate some languages that are not context-free.

Proof: Consider the language that consists of the strings

$$a_i \overbrace{b_i \ldots b_i}^{b_i \text{ repeated an arbitrary number of times}} a_i$$

for some range r of i, $(1 \leq i \leq r)$. If $r > \sum_{i=1}^{i=p} im$ then this language cannot be generated by a grammar of size (m, p) since all the recursion must be in the context-sensitive part. But there are only $\sum_{i=1}^{i=p} im$ distinct left hand sides of such rules so that the grammar must generate some string of the form $a_i b_j \ldots b_j a_i$ for $i \neq j$. Since any context sensitive grammar is a grammar of size (m, p) and since Chomsky has proved that not all context-sensitive languages are context free (6), it is obvious that there are languages generated by grammars of size (m, p) for sufficiently large (m, p) that are not context-free. We define a <u>grammar system of size (m, p)</u> as three rewriting systems, the first of which is a context-free grammar, the second of which is a grammar of size (m, p) and the third of which is a dictionary. The language generated by such a system is defined in the obvious way.

Theorem 4: The sets of languages generated by grammar systems of size (mxp),where mxp = y, define a hierarchy of languages $L_y$ such that (a) $L_0$ is the set of context-free languages, (b) $L_i \subsetneq L_j$ for j sufficiently greater than i, and (c) such that the sum of the $L_y$ is the set of context-sensitive languages.

Proof: The set of languages whose strings are of the form PhP, where h is a fixed string and P are arbitrary strings on a given alphabet A, are context-sensitive and not context-free (6). Therefore, in a grammar system which generates such a language,the part that generates such strings must be in the context-sensitive part. Although the dictionary can introduce arbitrary new letters it cannot insure that if the substitution for some given letter $a_i$ is to be $a_j$ at one time and $a_k$ at another, that the substitutions in a given string will be uniform (i.e., always $a_j$ and never $a_k$) for the entire length of an arbitrarily long string. Therefore, the rules of the context-sensitive part of the grammar system generating PhP must have different letters (or distinct strings representing different letters) in the left-hand side of its rules. But in the grammar generating the copy of a given string P there must be at least one rule to produce the effect of copying each letter of A. If we let the alphabet of A be larger than mxp, then this cannot occur in a grammar of size (m, p).

Therefore, for every grammar of size (m, p) there is a context-sensitive language that cannot be generated by a grammar system limited to a grammar of that size. But clearly this language can be generated by a system having a grammar of some finite size. This proves part (b). Part (a) of the theorem is proved by observing that the set of context free languages are generated by a grammar system with a grammar of size (0, 0). This is so because the content-sensitive part is empty and the amount of erasure that can be produced by any dictionary is always finite and therefore its effect can be incorporated into a context-free grammar. Part (c) of the theorem is obvious.

## R E F E R E N C E S

1. Chomsky, N., Syntactic Structures, Mouton, 1957.
2. Miller, G. A. and Chomsky, N., "Finitary Models of Language Users" in Handbook of Mathematical Psychology, (Luce, Bush and Galanter eds.) Volume 2, John Wiley, 1963.
3. Yngve, V. H., "A Model and an Hypothesis for Language Structure", Proceedings of the American Philosophical Society, Volume 104, pp. 444-466, 1960.
4. Fraser, J. B., "Some Remarks on Elementary Transformations", in Quarterly Progress Report of the Research Laboratory of Electronics (M. I. T.), No. 71.
5. Davis, M. , Computability and Unsolvability, McGraw- Hill, 1958.
6. Chomsky, N., "On Certain Formal Properties of Grammars", Information and Control, Volume 2, pp. 137-167, 1959.