[From: *Information Processing 1965*: Proceedings of the IFIP Congress 65 (Washington, D.C.: Spartan Books, 1965)]

AUTOMATIC PROCESSING OF NATURAL AND FORMAL LANGUAGES

A. G. OETTINGER
Computation Laboratory, Harvard University

Since IFIP CONGRESS-62 the pace of research and development efforts in natural and formal language processing has noticeably speeded up.

THEORY

In the most outstanding theoretical advance, theories of language and of automata have been joined through one series of theorems establishing within each realm hierarchies of increasing power and by another series of theorems correlating machines with the languages they may either generate or recognize.[1,2,3]

Machines are ordered in terms of computing power by placing one above the other if everything that can be computed by the lower machine can also be computed by the upper machine but there exists a computation which the upper can do that the lower cannot. The two best known types of machines—finite state automata and Turing machines—respectively mark the lower and upper boundary of the range of computing power. A finite state machine, as its name implies, is one whose configuration or state at any one time is selected from among a finite number of possible states. In the absence of other restrictions, any machine with an infinite number of states or, alternatively, a storage capacity which, though finite at any time, can keep growing beyond any bound as required in the course of operation essentially has the computing power of a Turing machine.

Although real computing machines are obviously always finite, describing them in the formalism of the theory of finite state machines does not provide as clear an insight into their properties as one might wish. This is one reason for the growth, within the theory of automata, of descriptions of machines in which a basically finite state control element is supplemented by an actually or potentially infinite storage device, usually an abstraction of the notion of n tapes capable of growing beyond any bound and subject to a variety of restrictions on reading or writing ability and on direction of motion.

Most interesting among these in the present context is the so-called pushdown store automaton which, typically, is a finite state device augmented by a tape on which both reading and writing is permitted, motion is allowed in both directions but scanning is limited to only the one symbol immediately on top of the tape, visualized as extending downward potentially without limit.[3] A symbol further down on the tape may be reached but only at the price of erasing all symbols above it. This access discipline is often described as "last in first out."

Machines are further distinguished as deterministic or nondeterministic. Roughly speaking, a machine is deterministic if the function of current state and current input which determines the next state and the next reading or writing action is single-valued and nondeterministic if this function is multivalued.

In addition to or instead of restrictions on the amount of storage or number of states allowable, machines may also be restricted as to the time (usually measured as number of steps) allowable in a computation. One category of *time-restricted machine* of current interest is the *real-time machine* which, given input at a constant rate, must produce a result within a fixed delay after entry of the last input item. In the last three years research on both hardware-restricted and time-restricted machines has produced a somewhat bewildering partial ordering of machine power and, since time-restricted machines are only beginning to be explored, further developments in this area may be expected.

During the same period the description of grammars and of abstract machines by what logicians call *productions,* linguists *rewrite rules,* and computer people *Backus normal form* has become increasingly widespread. A production is an expression of the form

$$\alpha_1\omega\alpha_2 \rightarrow \alpha_1\chi\alpha_2 \qquad (1)$$

in which α_1, α_2, ω, and χ, represent arbitrary and, except for ω, possible null strings of symbols.

A grammar G = (I,T,S,P) has disjoint sets of intermediate symbols (I) and terminal symbols (T), a distinguished starting symbol ($S \in I$), and a set of productions (P).

It turns out that no generality is lost if, in (1), ω is restricted to a single symbol. A grammar which includes productions of this type is the most general possible type of grammar. The class of context-dependent phrase structure grammars is described by restricting χ to be a non-empty string. The familiar context free phrase structure grammars have productions with the additional restriction that α_1 and α_2 are null. A grammar with productions of the context free variety obeying the further restriction that χ contains at most one terminal or one nonterminal symbol is a finite state grammar.

The connection between automata theory and formal linguistics may now be made apparent. If we consider a typical production $X \rightarrow aY$ of a finite state language and interpret it as specifying that a machine in state X emits symbol a when going to state Y, a close relation between finite state automata and finite state languages becomes immediately plausible. In fact, the finite state languages are precisely the regular sets defined in Kleene's classical study of finite state automata.[4] In a similar vein, the languages described by grammars with unrestricted productions are precisely those that can be generated by unrestricted Turing machines.

The basic theoretical interest of such equivalence results lies in the consequent introduction of linguistic results into proofs in automata theory and vice versa. An interesting and also practical connection between the two fields was made independently by Chomsky[2] and Evey[3] who proved that the set of all languages that can either be accepted or generated by nondeterministic pushdown store automata is precisely the set of all context free phrase structure languages. Grammars of this type are especially important because:

- they underlie, although they do not by any means complete, the description of most of the current higher-level programming languages such as ALGOL;
- all presently operating systems of syntactic analysis for natural languages are based on context free phrase structure grammars;
- they are the only grammars whose theory is well understood and whose practice is well established;
- even transformational grammars are built on a context free phrase structure base because, in generation, the domain of a transformation is given by a phrase marker which specifies the phrase structure of sentences in the set to which the given transformation may be applied; in analysis or recognition likewise, the application of an inverse transformation requires prior assignment of one or more potential phrase structures (surface structures) to the sentence being analyzed.

The past few years have thus been characterized by an interesting three-way exchange among logicians, automata theorists and mathematical linguists. Proofs of unsolvability based on Post's proof of the unsolvability of the correspondence problem[5] have diffused into linguistics and into automata theory. Questions about the ambiguity problem for grammars or on the nature of the intersections on unions of classes of languages, have, when settled or shown to be unsolvable, yielded through reinterpretation equivalent results about automata and vice-versa.

The rapidly growing interest in the algebraic formulation of linguistic problems promises further interesting developments in this area. Curiously enough in the light of the fruitful interaction between mathematical linguistics and *hardware restricted* automata, there seems to be little promise of interaction between linguistics and time-restricted machines. For example, very simple real-time machines will accept languages that are context dependent, but not context free, while there seem to be context free languages that are arbitrarily high in certain real-time hierarchies.[6]

AUTOMATIC LANGUAGE TRANSLATION

Automatic language translation no longer holds the center of the stage. Progress continues to be made in the matter of machine aids to translation, but today no one seriously imagines that fully-automatic, high-quality machine translation is just around the corner.[7] The automatic parsing of English, Russian and other languages continues to be a subject of active study throughout the world.

Some serious scholars[8] have a continuing faith in the eventual feasibility of economical, fully automatic translation but they recognize that considerable basic knowledge necessary for this goal to be reached is still lacking and they properly refuse to guess how soon such knowledge might be acquired. In their dedication to a continuing search for basic knowledge, and to the application of this knowledge wherever possible and worth while, such scholars are indistinguishable from those who hold that, in general, human intervention will continue to be essential and that machine-aided translation or, more broadly, machine-aided language data processing is a more reasonable long-range goal.

Meanwhile, a dwindling lunatic fringe continue their periodic claims of perfect translation just around the corner, and many who have been unsuccessful in translating from Russian to English, English to Russian or in any other direction are now flocking to the study of translation from or to Chinese.[9,10] One system long and loudly advertised as an automatic translation system[9,10] is now in experimental operation for the United States Air Force but it acts—by necessity rather than by choice—as a machine aid to translation. Output of this system is placed in the hands of bilingual post-editors who prepare final translations with the original text, the machine output, and a variety of other aids at their command.

Meanwhile, projects more deliberately aimed at providing appropriate assistance to human translators have been undertaken by the German government and the European Coal and Steel Community. It is recognized that technical competence in the subject matter being translated is the most important quality of a good translator, yet that precise mastery of a broad range of specialized vocabulary is difficult even for one with a good basic command of the language. One goal is therefore to supply the translator with up-to-date and accurate descriptions of specialized technical words and phrases with which he is unlikely to be familiar. The operation of such a semi-automatic dictionary system also is likely to improve the accuracy and the uniformity of the translations produced by a group of translators by aiding in the sharing of solutions to terminological problems. Contemporary advances in time-sharing, on-line operations and display techniques open further avenues for the exploration of useful man-machine interaction for similar purposes.

SYNTAX

Syntax and the construction of parsing systems continues to be an active object of study. Since a recent survey[11] describes in some detail a wide variety of different attacks on this problem, no attempt will be made to repeat the details here.

The most valuable theoretical insight into these diverse systems gained in the past three years is the realization that their diversity is, in a sense, only superficial: with a very few exceptions all systems of syntactic recognition or analysis (as opposed to generation) developed to date are now known to have the power of context free phrase structure grammars. For instance, it is now known that the introduction of discontinuous constituents normally does not affect the abstract power of a grammar, in the sense that any language describable by a grammar that has such rules is also describable by a grammar lacking them. Similarly, dependency grammars and predictive grammars are essentially context free phrase structure grammars,[12] although the latter are prototypically interpreted as immediate constituent grammars.

This abstract kinship among grammars implies that any language describable by one is describable by the others and that, in this sense, none is to be preferred to any other. This does not mean that all context free phrase structure grammars have precisely the same properties. While there is no difference in abstract descriptive power, some differences do remain in terms of the relative perspicuity of the properties of a language, some of which may be evident in one grammar but obscure in another and vice versa. Differences in processing efficiency also remain.

Consider, for example, the simple phrase structure grammar of Fig. la which generates a rudimentary (Łukasiewicz) prefix notation, as for example the formula $\alpha = + y + yy = ((y + y) + y)$. This form of context free, phrase structure grammar has an obvious interpretation as an immediate constituent grammar: the structure of a as generated by the grammar of Fig. la is shown by the solid lines of Fig. 1c, where it is evident that *a* is an S, made up of a "+" linking two S's, each of which, in turn, has finer structure. A dependency tree describing the same formula is shown in Fig. 1d. The interpretation of this tree is that a has a main element " + " linking the "y" with a compound element whose own main term, in turn, is a "+". Figure 1d may be obtained from Fig. 1c by identifying each intermediate symbol with the terminal symbol immediately beneath it and erasing the terminal node, as shown by the dotted lines.

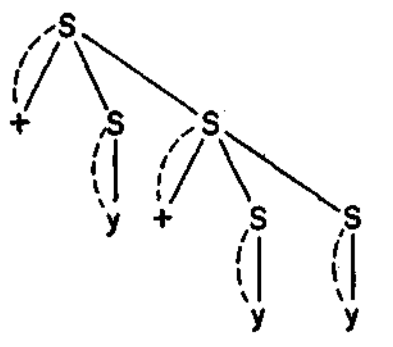
The ease of going from Fig. 1c to Fig. 1d and back does not generalize but the correspondence between the two does generalize, while the different flavor of the two representations also remains. Where immediate constituent analysis in general introduces a hierarchy of abstract constructions which have no overt representations in the terminal string itself, only terminal symbols appear in the dependency tree. Typically, an immediate constituent analysis will refer to such terms as *sentence, object, noun phrase, prepositional phrase* and so on, but a dependency tree will mention only *nouns, verbs, adjectives* and so on. Where the apex of an immediate constituent tree might be a symbol for *sentence,* the apex of a dependency tree will name the main verb

$$G = (I, T, S, P)$$
$$I = \{S\},\quad T = \{+, y\},\quad S = S$$
$$P = \left\{ \begin{array}{l} S \rightarrow y \\ S \rightarrow +\ SS \end{array} \right\}$$

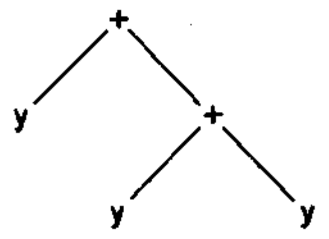
(a)

$$G' = (I, T, S, P')$$
$$I = \{S\},\quad T = \{+, y\},\quad S = S$$
$$P' = \left\{ \begin{array}{ll} S \rightarrow y & S \rightarrow Zy \\ Z \rightarrow +y & Z \rightarrow +Zy \\ Z \rightarrow Z+y & Z \rightarrow Z+Zy \end{array} \right\}$$

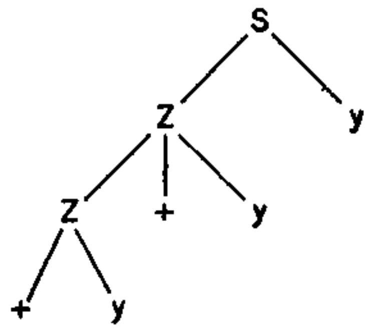
(b)



(c)



(d)



(e)

Figure 1 — Representations of α = + y + yy.

of the sentence. These different representations thus emphasize different aspects of sentence structure: the immediate constituent analysis focuses on a construction, the dependency analysis focuses on the head of that construction; for instance, the main verb of a sentence is treated as the head of that sentence and all other words or constructions depend on it.

In the example of Fig. 1, the head of a construction and the construction itself are so closely related that the mapping back and forth is trivial. In general practice it is not always obvious how to choose the head of a construction or what constructions to recognize. To build a dependency analysis or to deal with immediate constituents or to choose still another variant is a matter of taste and of objectives.

The language generated by the grammar of Fig. la may also be generated by that of Fig. 1b; the parsing of a according to the latter is shown in Fig. le. By comparing Fig. 1c with Fig. le, it may be observed that while the grammar "G" ascribes a left-to-right structure to a, the grammar "G"' imparts a right-to-left structure.

This observation, which generalizes fairly readily, is interesting on two accounts. First, it demonstrates that there is nothing absolutely left-to-right or right-to-left about a language which may have a grammar "going either way." Arguments over whether scanning and parsing should go from left to right, right to left, inside out, or upside down, are settled in one sense: any way is *possible.* It is also clear, however, that grammar "G"' is more complex than grammar "G" in that it has three times as many productions. From this point of view, one might therefore argue that the language to which α belongs is fundamentally a left-to-right language. One might further conclude that most natural languages have a mixture of intrinsically left-handed and intrinsically right-handed constructions, thus accounting for some of the controversy over preferred directions of scanning which has occurred in the recent past. Although some attempts have already been made to study this question more deeply[12,13,14] further exploration seems desirable.

Except for details of notation the grammar G is in a recently described standard form[12,13] into which any context free phrase structure grammar may be cast. The right-hand side of productions of standard form grammars is characterized by the absence of terminal symbols anywhere but in the first position, where a terminal symbol is mandatory. Every method of sentence structure determination can be regarded as the inverse of some generative grammar. There is a particularly easy solution of the inversion problem for generative grammars in standard form and the recognition algorithm for sentences generated by such a grammar uses a particularly simple form of pushdown store machine.

Since the usual grammar for prefix notation, (e.g., Fig. la) may be regarded as the very prototype of a standard form grammar, it is easy to see why pushdown stores or stacks have become so prevalent in the design of compilers for computer languages.[13,15-17] The theory of standard form grammars also accounts fully for the operation of predictive recognition algorithms.[13,18-20] In terms of Fig. la, the current prediction is identified with the left-hand symbol of a production, the currently scanned word form or word class with the one and only terminal symbol beginning the right-hand side of the production, and the new predictions added to the pushdown store or prediction pool with the intermediate symbols on the right-hand side of the production. As is obvious from Fig. la and 1c, when a particular context free phrase structure grammar is in standard form, the immediate constituent structure and the structure obtained through predictive analysis are indistinguishable.

In general, it has been shown not only that a standard form grammar exists for any arbitrary context free phrase structure grammar, but that there is a constructive algorithm for obtaining a standard form grammar from the given grammar; furthermore, ambiguity can be preserved in passing from arbitrary form to standard form, meaning that whatever the number of structures a given string may have according to the original grammar, it will have the same number of structures according to the corresponding standard form grammar. Of course, just as the three trees of Figs. 1c, 1d and le differ from one another, so, in general, would the tree produced by a standard form analyzer differ from all three even though it would, in this particular case, be practically indistinguishable from that of Fig. 1c and *a fortiriori* from that of Fig. 1d.

The raw output of a predictive analyzer consequently is a structural description of a type that emphasizes aspects of sentence structure differing from those stressed by other representations. Nevertheless, in practice, properties evident in one form of structural representation usually may be extracted from any of the other equivalent forms without excessive labor. For instance the raw output of the Kuno/Oettinger predictive analyzer is converted by an editing program not only from internal machine representation into printable symbols but also into a form approximating a conventional immediate constituent representation.

The Kuno/Oettinger multiple path predictive analyzer, first described publicly at IFIP CONGRESS-62,[18] has been constantly improved since. Grammar rules have been corrected, extended and refined. The dictionary has been enlarged. Most significant for an analyzer which is designed to supply all of the structures of an ambiguous sentence that are implicit in a given grammar, is the degree of control achieved over the combinatorial explosion inherent in the operation of what is essentially a nondeterministic pushdown store machine which explores all possible avenues of analysis (Table 1). Processing time has been decreased steadily and, recently, the operation of the analyzer became tape-limited on an IBM 7094 following the introduction of means for recognizing well-formed substrings the first time they occur and not thereafter.

While the details of the process are too complicated to recount here,[20] the basic idea is illustrated in Fig. 2. Consider a sentence analyzed as a main clause with a nested subordinate clause. If the main clause has m possible structures and the subordinate clause n, there will be m × n distinct analyses. It is obvious, however, that these mn analyses are completely described by giving the m analyses of the main clause and the n analyses of the subordinate clause. The labor of analysis should therefore be governed by m + n rather than by mn. The catch is that the existence of the subordinate clause is obvious only after the sentence has been analyzed; it is not clear *a priori* that the nested clause can be detected and analyzed independently with less labor than that required to analyze the whole sentence. In other words, an amount of labor governed by (m × n - (m + n)) might have to be expended to discover the existence of the decomposition of the sentence. Fortunately, however, Table 1 shows that this may not be the case.

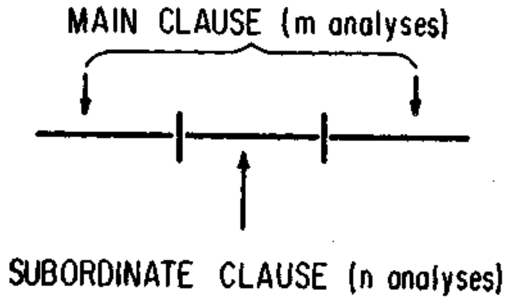


Figure 2—Complex sentence with mn analyses.

Neither the theory nor the practice of this new development are yet fully explored. Nevertheless, a capability for processing sentences of nontrivial length and complexity in a few seconds, coupled with recent developments in time-sharing, on-line operations and display techniques, opens up some interesting possibilities. First of all, processing time may be further reduced if a person makes a preliminary scan of the sentence to strike out irrelevant homographs. For example, while an automatic dictionary must indicate that "time," as in "time flies like an arrow," may serve as a noun, as an attribute or as an imperative verb, a person looking at the sentence would find it easy to strike out the last two possibilities prior to analysis. If this is done for a whole sentence, the number of possible search paths, hence the total processing time, are enormously reduced. Further exploration of on-line editing techniques for this purpose would therefore be of great interest.

Next, as well-formed substrings are identified in the course of analysis, they could be displayed to a person who also has the whole text before him. The

Table 1—Predictive Analyzer; Comparative Processing Times

No. of words in test sentence	No. of analyses	1963-FJCC (SHARE version)[19]	With well-formed substring identification[20]
38	94	42.4 *mins*	0.2 *mins*
32	18	9.8	0.1
35	12	9.0	0.1
30	118	7.6	0.3
25	136	7.2	0.3
20	71	2.7	0.2
27	1	.7	0.1
23	31	.5	0.1
29	2	.5	0.1
30	17	1.2	0.2
25	5	1.1	0.1
20	16	0.7	0.1
25	72	0.5	0.2
23	7	0.2	0.1
16	3	0.1	0.0
17	4	0.1	0.0
18	1	0.1	0.1
17	1	0.0	0.0
14	4	0.0	0.1
		87.5 *mins*	2.6 *mins*

elimination of syntactically acceptable but semantically absurd constructions could therefore be carried out on-line thereby not only speeding up the editing of the output, but also economizing analysis time by enabling the machine to reject any constructions in which an already rejected well-formed substring appears. With several persons using the same machine, the idea of holding up the analysis of a given sentence to let a person make a decision while letting others proceed seems reducible to economical practice. The machine might also be used to help the human editor keep track of previous decisions and use these to guide him when he comes to the same or a similar situation further on in a text. Unrestricted natural language input to information processors thus seems, through this type of man-machine interaction, to be closer to realizability.

While syntactic ambiguity is often ill-regarded, the predictive syntactic analyzer's ability to recognize ambiguous constructions has been turned to advantage in at least one experimental application of international interest.[21] When sentences of the Nuclear Test Ban Treaty were subjected to a version of this analyzer, several interesting ambiguities were detected and called to the attention of the authorities. The possibility of refining the technique and extending it to the analysis of other legal and commercial documents is an attractive speculation.

SEMANTICS; QUESTION-ANSWERING

The increasing practicality of context free phrase structure recognizers opens the door toward serious experimental evaluation of the promise of an additional transformational superstructure. At least two attempts in this direction are known to be underway.[22,23] It would be helpful if the theory of transformational grammars were to become bathed in illumination of the same intensity as that now shining on context free phrase structure grammars. However, for any transformational grammar that has the power of unrestricted Turing machines, most important questions are abstractly unsolvable. What position restricted transformational grammars might occupy in the hierarchy between context-free phrase structure grammars and unrestricted production grammars is as yet ill understood.

In any case, with syntactic recognizers promising to deliver syntactic structures at a high enough rate and with great enough economy to provide ample experimental material for further investigation, the intensive experimental exploration of semantic questions seems ripe for approach. The literature of semantic studies to date consists mainly of philosophical analysis and fragmentary linguistic speculation, on the one hand, and of reports on limited experiments with very sharply-restricted semantic universes on the other hand.

The latter type of study has been undertaken chiefly with the hope of developing devices which might answer questions posed in anything from a highly formatted language to as unrestricted a form of English as the very limited special purpose syntactic analyzers usually constructed in connection with such experiments will allow. More than fifteen studies along such lines have been reported in the last few years and cannot be described here in any detail. A recent survey[24] restricted to this matter will, however, guide any interested reader to the growing literature on the subject. While several of these experiments are extremely interesting and some seem potentially valuable within very limited practical and theoretical restrictions, it is, at present, very hard to see how any of them can be extended much beyond the boundaries presently set by the experimenters. Nevertheless, this is an area which deserves and undoubtedly will receive much further attention.

One question that question-answering systems raise but which hardly seems to have been studied is that of the stability of restricted subsets of natural languages. Cursory personal observation and some anecdotal evidence suggest that attempts to define a regular and easily manageable subset of a natural language fail through an instability which causes these subsets to drift either in the direction of a formal mathematical notation as unlike the vernacular as possible, or else in the opposite direction toward full use of the unrestricted vernacular. In the former case, machine manipulation is simplified, but the expected advantages of communication with a machine in a vernacular are obviously lost. In the latter case, one drifts back to all of the problems of dealing with unrestricted natural language.

In spite of this, it is by no means clear whether or not stable subsets of a vernacular could be defined and maintained, if not in complete stability at least in some controllable and followable drifting state of quasi-stability. Linguists to whom one broaches this question regard it as a version of the problem of language universals and throw up their hands if indeed they understand the question at all. Nevertheless, it seems reasonable to expect that an intelligent scientific and engineering approach to the question of guidelines for language synthesis or restriction could at least illuminate it further, if not solve it. Repeated none-to-successful attempts to fabricate such languages have already been sufficiently expensive in time, energy and money to merit more rational guidance. Even in the more regular domain of formal and programming languages, many unsolved practical and theoretical problems remain. For example, the matter of recovery from error in the course of compilation[17] remains in a quasi-mystical experimental state, although some early results, applicable only to the simplest of languages[15] suggest that further formal study of this problem could be worth while.

REFERENCES

1. P. C. Fischer, "On Computability by Certain Classes of Restricted Turing Machines," *Switching Circuit Theory and Logical Design,* Proceedings of the 4th Annual Symposium, IEEE, New York, 1963.
2. N. Chomsky, "Formal Properties of Grammars," *Handbook of Mathematical Psychology* (R. R. Bush, E. H. Galanter and R. D. Luce, eds.), v. 2, John Wiley and Sons, New York, 1963.
3. R. J. Evey, *The Theory and Applications of Pushdown Store Machines,* Doctoral thesis, Harvard University, 1963.
4. S. C. Kleene, "Representation of Events in Nerve Nets and Finite Automata," *Automata Studies* (C. E. Shannon and J. McCarthy, eds.), Princeton University Press, 1956.
5. E. L. Post, "A Variant of a Recursively Unsolvable Problem," *Bulletin of the American Mathematical Society,* v. 52, pp. 264-268, 1946.
6. P. C. Fischer, private communication.
7. A. G. Oettinger, "The State of the Art of Automatic Language Translation," *Sprachkunde und Informationsverarbeitung,* v. 2, November, 1963. pp.17-32.
8. V. H. Yngve, "Implications of Mechanical Translation Research," *Proc. American Philosophical Society,* v. 108, no. 4, August, 1964. pp.275-281.
9. G. W. King, and H. W. Chang, "Machine Translation of Chinese," *Scientific American,* June, 1963. pp. 124-135.
10. G. W. King, and A. G. Oettinger, letters to the Editor on "Machine Translation of Chinese," *Scientific American,* October, 1963. pp. 8-11.
11. D. G. Bobrow, "Syntactic Analysis of English by Computer—A Survey," *AFIPS Conference Proceedings,* v. 24, Spartan, Baltimore, 1963.
12. S. Greibach, *Inverses of Phrase Structure Generators,* Doctoral thesis, Harvard University, 1963.
13. S. Greibach, "Formal Parsing Systems," *Communications of the ACM,* v. 7, August, 1964. pp.499-504.
14. T, V. Griffiths and S. R. Petrick, "On the Relative Efficiencies of Context Free Grammar Recognizers," submitted for publication.
15. A. G. Oettinger, "Automatic Syntactic Analysis and the Pushdown Store," *Structure of Language and Its Mathematical Aspects* (R. Jakobson, ed.), Proc. Symposia in Applied Mathematics, v. 12, American Mathematical Society, Providence, Rhode Island, 1961.
16. K. Samelson and F. L. Bauer, "Sequential Formula Translation," *Communications of the ACM,* v. 3, no. 2, 1960. pp. 76-83.
17. R. W. Floyd, "The Syntax of Programming Languages—A Survey," *IEEE Transactions on Electronic Computers,* v. EC-13, August, 1964. pp.346-353.
18. S. Kuno and A. G. Oettinger, "Multiple-Path Syntactic Analyzer," *Information Processing-62,* North-Holland, Amsterdam, 1963.
19. S. Kuno and A. G. Oettinger, "Syntactic Structure and Ambiguity of English," *AFIPS Conference Proceedings,* v. 24, Spartan, Baltimore, 1963.
20. S. Kuno, "The Predictive Analyzer and a Path Elimination Technique," submitted for publication.
21. R. A. Langevin and M. F. Owens, "Computer Analysis of the Nuclear Test Ban Treaty," *Science,* v. 146, November 27, 1964. pp. 1186-1189.
22. S. R. Petrick, "Recognition Procedure for Transformational Grammars," to appear in *Information System Sciences: Proceedings of the Second Congress,* Spartan, Baltimore.
23. B. Hall, private communication.
24. R. F. Simmons, "Answering English Questions by Computer: A Survey," *Communications of ACM,* v. 8, January, 1965. pp. 53-70.