CONTEXT-FREENESS AND THE COMPUTER PROCESSING OF HUMAN LANGUAGES

Geoffrey K. Pullum

Cowell College
University of California, Santa Cruz
Santa Cruz, California 95064

ABSTRACT

Context-free grammars, far from having insufficient expressive power for the description of human languages, may be overly powerful, along three dimensions; (1) weak generative capacity: there exists an interesting proper subset of the CFL's, the profligate CFL's, within which no human language appears to fall; (2) strong generative capacity: human languages can be appropriately described in terms of a proper subset of the CF-PSG's, namely those with the ECPO property; (3) time complexity: the recent controversy about the importance of a low deterministic polynomial time bound on the recognition problem for human languages is misdirected, since an appropriately restrictive theory would guarantee even more, namely a linear bound.

0. INTRODUCTION

Many computationally inclined linguists appear to think that in order to achieve adequate grammars for human languages we need a bit more power than is offered by context-free phrase structure grammars (CF-PSG's), though not a whole lot more. In this paper, I am concerned with the defense of a more conservative view: that even CF-PSG's should be regarded as too powerful, in three computationally relevant respects: weak generative capacity, strong generative capacity, and time complexity of recognition. All three of these matters should be of concern to theoretical linguists; the study of what mathematically definable classes human languages fall into does not exhaust scientific linguistics, but it can hardly be claimed to be irrelevant to it. And it should be obvious that all three issues also have some payoff in terms of certain computationally interesting, if rather indirect, implications.

1. WEAK GENERATIVE CAPACITY

Weak generative capacity (WGC) results are held by some linguists (e.g. Chomsky (1981)) to be unimportant. Nonetheless, they cannot be ignored by linguists who are interested in setting their work in a context of (even potential) computational implementation (which, of course, some linguists are not). To paraphrase Montague, we might say that linguistically (as opposed to psycholinguistically) there is no important theoretical difference between natural languages and high-level programming languages. Mediating programs (e.g. a compiler or interpreter), of considerable complexity, will be needed for the interpretation of computer input in either Prolog or Japanese. In the latter case the level of complexity will be much higher, but the assumption is that we are talking quantitatively, not qualitatively. And if we are seriously interested in the computational properties of either kind of language, we will be interested in their language-theoretic properties, as well as properties of the grammars that define them and the parsers that accept them.

The most important language-theoretic class considered by designers of programming languages, compilers, etc. is the context-free languages (CFL's). Ginsburg (1980, 7) goes so far as to say on behalf of formal language theorists, "We live or die on the context-free languages.") The class of CFL's is very rich. Although there are simply definable languages well known to be non-CF, linguists often take CFL's to be non-CF in error. Several examples are cited in Pullum and Gazdar (1982). For another example, see Dowty, Wall and Peters (1980; p.81), where exercise 3 invites the reader to prove a certain artificial language non-CF. The exercise is impossible, for the language <u>is</u> a CFL, as noted by William H. Baxter (personal communication to Gerald Gazdar).

From this point on, it will be useful to be able to refer to certain types of formal language by names. I shall use the terms defined in (1) thru (3), among others.

(1) Triple Counting Languages:
languages that can be mapped by a homomorphism onto some language of the form

$\{a^n b^n c^n \mid n \geq 1\}$

(2) String Matching Languages:
languages that can be mapped by a homomorphism onto some language of the form

{xx|x is in some infinite language A}

(3) String Contrasting Languages:
languages that can be mapped by a homomorphism onto some language of the form

{xcy|x and y are in some infinite language A and $x \neq y$}

Programming languages are virtually always designed to be CF, except that there is a moot point concerning the implications of obligatory initial declaration of variables as in ALGOL or Pascal, since if variables (identifiers) can be alphanumeric strings of arbitrary length, a syntactic guarantee that each variable has been declared is tantamount to a syntax for a string matching language. The following view seems a sensible one

to take about such cases: languages like ALGOL or Pascal are CF, but not all ALGOL or Pascal programs compile or run. Programs using undeclared variables make no sense either to the compiler or to the CPU. But they are still programs, provided they conform in all other ways to the syntax of the language in question, just as a program which always goes into an infinite loop and thus never gives any output is a program. Aho and Ullmann (1977, 140) take such a view:

> the syntax of ALGOL...does not get down to the level of characters in a name. Instead, all names are represented by a token such as id, and it is left to the bookkeeping phase of the compiler to keep track of declarations and uses of particular names.

The bookkeeping has to be done, of course, even in the case of languages like LISP whose syntax does not demand a list of declarations at the start of each program.

Various efforts have been made in the linguistic literature to show that some human language has an infinite, appropriately extractable subset that is a triple counting language or a string matching language. (By appropriately extractable I mean isolable via either homomorphism or intersection with a regular set.) But all the published claims of this sort are fallacious (Pullum and Gazdar 1982). This lends plausibility to the hypothesis that human languages are all CF. Stronger claims than this (e.g. that human languages are regular, or finite cardinality) have seldom seriously defended. I now want to propose one, however.

I propose that human languages are never profligate CFL's in the sense given by the following definition.

(i) A CFL is profligate if all CF-PSG's generating it have nonterminal vocabularies strictly larger than their terminal vocabularies.
(ii) A CFL is profligate if it is the image of a profligate language under some homomorphism.

[OPEN PROBLEM: Is profligacy decidable for an arbitrary CFL? I conjecture that it is not, but I have not been able to prove this.]

Clearly, only an infinite CFL can be profligate, and clearly the most commonly cited infinite CFL's are not profligate. For instance, $\{a^n b^n | n \geq 0\}$ is not profligate, because it has two terminal symbols but there is a grammar for it that has only one nonterminal symbol, namely S. (The rules are: {S --> aSb, S --> e}.) However, profligate CFL's do exist. There are even regular languages that are profligate: a simple example (due to Christopher Culy) is $(a^* + b^*)$.

More interesting is the fact that some string contrasting languages as defined above are profligate. Consider the string contrasting language over the vocabulary {a, b, c} where $A = (a + b)^*$. A string xcy in $(a + b)^*c(a + b)^*$ will be in this language if any one of the following is met:

(a) x is longer than y;
(b) x is shorter than y;
(c) x is the same length as y but there is an i such that the ith symbol of x is distinct from the ith symbol of y.

The interesting condition here is (c). The grammar has to generate, for all i and for all pairs <u, v> of symbols in the terminal vocabulary, all those strings in $(a + b)^*c(a + b)^*$ such that the ith symbol is u and the ith symbol after c is v. There is no bound on i, so recursion has to be involved. But it must be recursion through a category that preserves a record of which symbol is crucially going to be deposited at the ith position in the terminal string and mismatched with a distinct symbol in the second half. A CF-PSG that does this can be constructed (see Pullum and Gazdar 1982, 478, for a grammar for a very similar language). But such a grammar has to use recursive nonterminals, one for each terminal, to carry down information about the symbol to be deposited at a certain point in the string. In the language just given there are only two relevant terminal symbols, but if there were a thousand symbols that could appear in the x and y strings, then the vocabulary of recursive nonterminals would have to be increased in proportion. (The second clause in the definition of profligacy makes it irrelevant whether there are other terminals in the language, like c in the language cited, that do not have to participate in the recursive mechanisms just referred to.)

For a profligate CFL, the argument that a CF-PSG is a cumbersome and inelegant form of grammar might well have to be accepted. A CF-PSG offers, in some cases at least, an appallingly inelegant hypothesis as to the proper description of such a language, and would be rejected by any linguist or programmer. The discovery that some human language is profligate would therefore provide (for the first time, I claim) real grounds for a rejection of CF-PSG's on the basis of strong generative capacity (considerations of what structural descriptions are assigned to strings) as opposed to weak (what language is generated).

However, no human language has been shown to be a profligate CFL. There is one relevant argument in the literature, found in Chomsky (1963). The argument is based on the nonidentity of constituents allegedly required in comparative clause constructions like (4).

(4) She is more competent as [a designer of programming languages] than he is as [a designer of microchips].

Chomsky took sentences like (5) to be ungrammatical, and thus assumed that the nonidentity between the bracketed phrases in the previous example had to be guaranteed by the grammar.

(5) She is more competent as [a designer of programming languages] than he is as [a designer of programming languages].

Chomsky took this as an argument for non-CF-ness in English, since he thought all string contrasting languages were non-CF (see Chomsky 1963, 378-379),

but it can be reinterpreted as an attempt to show that English is (at least) profligate. (It could even be reconstituted as a formally valid argument that English was non-CF if supplemented by a demonstration that the class of phrases from which the bracketed sequences are drawn is not only infinite but non-regular; cf. Zwicky and Sadock.) However, the argument clearly collapses on empirical grounds. As pointed out by Pullum and Gazdar (1982, 476-477), even Chomsky now agrees that strings like (5) are grammatical (though they need a contrastive context and the appropriate intonation to make them readily acceptable to informants). Hence these examples do not show that there is a homomorphism mapping English onto some profligate string contrasting language.

The interesting thing about this, if it is correct, is that it suggests that human languages not only never demand the syntactic string comparison required by string matching languages, they never call for syntactic string comparision over infinite sets of strings at all, whether for symbol-by-symbol checking of identity (which typically makes the language non-CF) or for specifying a mismatch between symbols (which may not make the language non-CF, but typically makes it profligate).

There is an important point about profligacy that I should make at this point. My claim that human languages are non-profligate entails that each human language has at least one CF-PSG in which the nonterminal vocabulary has cardinality strictly less than the terminal vocabulary, but not that the best grammar to implement for it will necessarily meet this condition. The point is important, because the phrase structure grammars employed in natural language processing generally have complex nonterminals consisting of sizeable feature bundles. It is not uncommon for a large natural language processing system to employ thirty or forty binary features (or a rough equivalent in terms of multi-valued features), i.e. about as many features as are employed for phonological description by Chomsky and Halle (1968). The GPSG system described in Gawron et al. (1982) has employed features on this sort of scale at all points in its development, for example. Thirty or forty binary features yields between a billion and a trillion logically distinguishable nonterminals (if all values for each feature are compatible with all combinations of values for all other features). Because economical techniques for rapid checking of relevant feature values are built into the parsers normally used for such grammars, the size of the potentially available nonterminal vocabulary is not a practical concern. In principle, if the goal of capturing generalizations and reducing the size of the grammar formulation were put aside, the nonterminal vocabulary could be vastly reduced by replacing rule schemata by long lists of distinct rules expanding the same nonterminal.

Naturally, no claim has been made here that profligate CFL's are computationally intractable. No CFL's are intractable in the theoretical sense, and intractability in practice is so closely tied to details of particular machines and programming environments as to be pointless to talk about in terms divorced from actual measurements of size for grammars, vocabularies, and address spaces. I have been concerned only to point out that there is an interesting proper subset of the infinite CFL's within which the human languages seem to fall.

One further thing may be worth pointing out. The kind of string contrasting languages I have been concerned with above are strictly nondeterministic. The deterministic CFL's (DCFL's) are closed under complementation. But the complement of

(6) $\{\underline{x}\underline{c}\underline{y} \mid \underline{x} \text{ and } \underline{y} \text{ are in } (\underline{a} + \underline{b})^* \text{ and } \underline{x} \neq \underline{y}\}$

in $(\underline{a} + \underline{b})^*\underline{c}(\underline{a} + \underline{b})^*$ is (7a), identical to (7b), a string matching language.

(7)a. $\{\underline{x}\underline{c}\underline{y} \mid \underline{x} \text{ and } \underline{y} \text{ are in } (\underline{a} + \underline{b})^* \text{ and } \underline{x} = \underline{y}\}$

b. $\{\underline{x}\underline{c}\underline{x} \mid \underline{x} \text{ is in } (\underline{a} + \underline{b})^*\}$

If (7a) [=(7b)] is non-CF and is the complement of (6), then (6) is not a DCFL.

[OPEN PROBLEM: Are there any nonregular profligate DCFL's?]

2. STRONG GENERATIVE CAPACITY

I now turn to a claim involving strong generative capacity (SGC). In addition to claiming that human languages are non-profligate CFL's, I want to suggest that every human language has a linguistically adequate grammar possessing the Exhaustive Constant Partial Ordering (ECPO) property of Gazdar and Pullum (1981). A grammar has this property if there is a single partial ordering of the nonterminal vocabulary which no right hand side of any rule violates. The ECPO CF-PSG's are a nonempty proper subset of the CF-PSG's. The claim that human languages always have ECPO CF-PSG's is a claim about the strong generative capacity that an appropriate theory of human language should have---one of the first such claims to have been seriously advanced, in fact. It does not affect weak generative capacity; Shieber (1983a) proves that every CFL has an ECPO grammar. It is always possible to construct an ECPO grammar for any CFL if one is willing to pay the price of inventing new nonterminals ad hoc to construct it. The content of the claim lies in the fact that linguists demand independent motivation for the nonterminals they postulate, so that the possibility of creating new ones just to guarantee ECPO-ness is not always a reasonable one.

[OPEN PROBLEM: Could there be a non-profligate CFL which had $\#(\underline{N}) < \#\underline{T}$ (i.e. nonterminal vocabulary strictly smaller than terminal vocabulary) for at least one of its non-ECPO grammars, but whose ECPO grammars always had $\#(\underline{N}) > \#(\underline{T})$?]

When the linguist's criteria of evaluation are kept in mind, it is fairly clear what sort of facts in a human language would convince linguists to abandon the ECPO claim. For example, if English had PP - S' order in verb phrases (explain to him that he'll have to leave) but had S' - PP order in adjectives (so that lucky for us we found you had the form lucky we found you for us), the grammar of English would not have the ECPO property. But such facts appear not to turn up in the languages we know about.

The ECPO claim has interesting consequences relating to patterns of constituent order and how these can be described in a fully general way. If a grammar has the ECPO property, it can be stated in what Gazdar and Pullum call ID/LP format, and this renders numerous significant generalizations elegantly capturable. There are also some potentially interesting implications for parsing, studied by Shieber (1983a), who shows that a modified Earley algorithm can be used to parse ID/LP format grammars directly.

One putative challenge to any claim that CF-PSG's can be strongly adequate descriptions for human languages comes from Dutch and has been discussed recently by Bresnan, Kaplan, Peters, and Zaenen (1982). Dutch has constructions like

(7) dat Jan Piet Marie zag leren zwemmen
that Jan Piet Marie saw teach swim
`that Jan saw Piet teach Marie to swim'

These seem to involve crossing dependencies over a domain of potentially arbitrary length, a configuration that is syntactically not expressible by a CF-PSG. In the special case where the dependency involves stringwise identity, a language with this sort of structure reduces to something like {xx|x is in A*}, a string matching language. However, analysis reveals that, as Bresnan et al. accept, the actual dependencies in Dutch are not syntactic. Grammaticality of a string like (7) is not in general affected by interchanging the NP's with one another, since it does not matter to the ith verb what the ith NP might be. What is crucial is that (in cases with simple transitive verbs, as above) the ith predicate (verb) takes the interpretation of the i-1th noun phrase as its argument. Strictly, this does not bear on the issue of SGC in any way that can be explicated without making reference to semantics. What is really at issue is whether a CF-PSG can assign syntactic structures to sentences of Dutch in a way that supports semantic interpretation.

Certain recent work within the framework of generalized phrase structure grammar suggests to me that there is a very strong probability of the answer being yes. One interesting development is to be found in Culy (forthcoming), where it is shown that it is possible for a CFL-inducing syntax in ID/LP format to assign a "flat" constituent structure to strings like Piet Marie zag leren zwemmen (`saw Piet teach Marie to swim'), and assign them the correct semantics.

Ivan Sag, in unpublished work, has developed a different account, in which strings like zag leren zwemmen (`saw teach to swim') are treated as compound verbs whose semantics is only satisfied if they are provided with the appropriate number of NP sisters. Whereas Culy has the syntax determine the relative numbers of NP's and verbs, Sag is exploring the assumption that this is unnecessary, since the semantic interpretation procedure can carry this descriptive burden. Under this view too, there is nothing about the syntax of Dutch that makes it non-CF, and there is not necessarily anything in the grammar that makes it non-ECPO.

Henry Thompson also discusses the Dutch problem from the GPSG standpoint (in this volume).

One other interesting line of work being pursued (at Stanford, like the work of Culy and of Sag) is due to Carl Pollard (Pollard, forthcoming, provides an introduction). Pollard has developed a generalization of context-free grammar which is defined not on trees but on "headed strings", i.e. strings with a mark indicating that one distinguished element of the string is the "head", and which combines constituents not only by concatenation but also by "head wrap". This operation is analogous to Emmon Bach's notion "right (or left) wrap" but not equivalent to it. It involves wrapping a constituent A around a constituent B so that the head is to the left (or right) of B and the rest of A is to the right (or left) of B. Pollard has shown that this provides for an elegant syntactic treatment of the Dutch facts. I mention his work because I want to return to make a point about it in the immediately following section.

3. TIME COMPLEXITY OF RECOGNITION

The time complexity of the recognition problem (TCR) for human languages is like WGC questions in being decried as irrelevant by some linguists, but again, it is hardly one that serious computational approaches can legitimately ignore. Gazdar (1981) has recently reminded the linguistic community of this, and has been answered at great length by Berwick and Weinberg (1982). Gazdar noted that if transformational grammars (TG's) were stripped of all their transformations, they became CFL-inducing, which meant that the series of works showing CFL's to have sub-cubic recognition times became relevant to them. Berwick and Weinberg's paper represents a concerted effort to discredit any such suggestion by insisting that (a) it isn't only the CFL's that have low polynomial recognition time results, and (b) it isn't clear that any asymptotic recognition time results have practical implications for human language use (or for computer modelling of it).

Both points should be quite uncontroversial, of course, and it is only by dint of inaccurate attribution that Berwick and Weinberg manage to suggest that Gazdar denies them. However, the two points simply do not add up to a reason for not being concerned with TCR results. Perfectly straightforward considerations of theoretical restrictiveness dictate that if the languages recognizable in polynomial time are a proper subset of those recognizable in exponential time (or whatever), it is desirable to explore the hypothesis that the human languages fall within the former class rather than just the latter.

Certainly, it is not just CFL's that have been shown to be efficiently recognizable in deterministic time on a Turing machine. Not only every context-free grammar but also every context-sensitive grammar that can actually be exhibited generates a language that can be recognized in deterministic linear time on a two-tape Turing machine. It is certainly not the case that all the context-sensitive languages are linearly recognizable; it can be shown (in a highly indirect way) that there must be some that are not. But all the examples ever constructed generate linearly recognizable languages. And it is still unknown whether there are CFL's not linearly recognizable.

It is therefore not at all necessary that a human language should be a CFL in order to be efficiently recognizable. But the claims about recognizability of CFL's do not stop at saying that by

good fortune there happens to be a fast recognition algorithm for each member of the class of CFL's. The claim, rather, is that there is a single, universal algorithm that works for every member of the class and has a low deterministic polynomial time complexity. That is what cannot be said of the context-sensitive languages.

Nonetheless, there are well-understood classes of grammars and automata for which it can be said. For example, Pollard, in the course of the work mentioned above, has shown that if one or other of left head wrap and right head wrap is permitted in the theory of generalized context-free grammar, recognizability in deterministic time $n5$ is guaranteed, and if both left head wrap and right head wrap are allowed in grammars (with individual grammars free to have either or both), then in the general case the upper bound for recognition time is $n7$. These are, while not sub-cubic, still low deterministic polynomial time bounds. Pollard's system contrasts in this regard with the lexical-functional grammar advocated by Bresnan et al., which is currently conjectured to have an NP-complete recognition problem.

I remain cautious about welcoming the move that Pollard makes because as yet his non-CFL-inducing syntactic theory does not provide an explanation for the fact that human languages always seem to turn out to be CFL's. It should be pointed out, however, that it is true of every grammatical theory that not every grammar defined as possible is held to be likely to turn up in practice, so it is not inconceivable that the grammars of human languages might fall within the CFL-inducing proper subset of Pollard-style head grammars.

Of course, another possibility is that it might turn out that some human language ultimately provides evidence of non-CF-ness, and thus of a need for mechanisms at least as powerful as Pollard's. Bresnan et al. mention at the end of their paper on Dutch a set of potential candidates: the so called "free word order" or "nonconfigurational" languages, particularly Australian languages like Dyirbal and Walbiri, which can allegedly distribute elements of a phrase at random throughout a sentence in almost any order. I have certain doubts about the interpretation of the empirical material on these languages, but I shall not pursue that here. I want instead to show that, counter to the naive intuition that wild word order would necessarily lead to gross parsing complexity, even rampantly free word order in a language does not necessarily indicate a parsing problem that exhibits itself in TCR terms.

Let us call transposition of adjacent terminal symbols scrambling, and let us refer to the closure of a language L under scrambling as the scramble of L. The scramble of a CFL (even a regular one) can be non-CF. For example, the scramble of the regular language $(abc)^*$ is non-CF, although $(abc)^*$ itself is regular. (Of course, the scramble of a CFL is not always non-CF. The scramble of $a^*b^*c^*$ is $(a, b, c)^*$, and both are regular, hence CF.) Suppose for the sake of discussion that there is a human language that is closed under scrambling (or has an appropriately extractable infinite subset that is). The example just cited, the scramble of $(abc)^*$, is a fairly clear case of the sort of thing that might be modeled in a human language that was closed under scrambling. Imagine, for example, the case of a language in which each transitive clause had a verb (a), a nominative noun phrase (b), and an accusative noun phrase (c), and free word order permitted the a, b, and c from any number of clauses to occur interspersed in any order throughout the sentence. If we denote the number of x's in a string Z by $Nx(Z)$, we can say that the scramble of $(abc)^*$ is (8).

$$(8)\ \{x \mid x \text{ is in } (a, b, c)^* \text{ and } Na(x) = Nb(x) = Nc(x)\}$$

Attention was first drawn to this sort of language by Bach (1981), and I shall therefore call it a Bach language. What TCR properties does a Bach language have? The one in (8), at least, can be shown to be recognizable in linear time. The proof is rather trivial, since it is just a corollary of a previously known result. Cook (1971) shows that any language that is recognized by a two-way deterministic pushdown stack automaton (2DPDA) is recognizable in linear time on a Turing machine. In the Appendix, I give an informal description of a 2DPDA that will recognize the language in (8). Given this, the proof that (8) is linearly recognizable is trivial.

Thus even if my WGC and SGC conjectures were falsified by discoveries about free word order languages (which I consider that they have not been), there would still be no ground for tolerating theories of grammar and parsing that fail to impose a linear time bound on recognition. And recent work of Shieber (1983b) shows that there are interesting avenues in natural language parsing to be explored using deterministic context-free parsers that do work in linear time.

In the light of the above remarks, some of the points made by Berwick and Weinberg look rather peculiar. For example, Berwick and Weinberg argue at length that things are really so complicated in practical implementations that a cubic bound on recognition time might not make much difference; for short sentences a theory that only guarantees an exponential time bound might do just as well. This is, to begin with, a very odd response to be made by defenders of TG when confronted by a theoretically restrictive claim. If someone made the theoretical claim that some problem had the time complexity of the Travelling Salesman problem, and was met by the response that real-life travelling salesmen do not visit very many cities before returning to head office, I think theoretical computer scientists would have a right to be amused. Likewise, it is funny to see practical implementation considerations brought to bear in defending TG against the phrase structure backlash, when (a) no formalized version of modern TG exists, let alone being available for implementation, and (b) large phrase structure grammars are being implemented on computers and shown to run very fast (see e.g. Slocum 1983, who reports an all-paths, bottom-up parser actually running in linear time using a CF-PSG with 400 rules and 10,000 lexical entries).

Berwick and Weinberg seem to imply that data permitting a comparison of CF-PSG with TG are available. This is quite untrue, as far as I know. I therefore find it nothing short of astonishing to find Chomsky (1981, 234), taking a very similar position, affirming that because the size of the

grammar is a constant factor in TCR calculations, and possibly a large one,

> The real empirical content of existing results... may well be that grammars are preferred if they are not too complex in their rule structure. If parsability is a factor in language evolution, we would expect it to prefer `short grammars'---such as transformational grammars based on the projection principle or the binding theory...

TG's based on the "projection principle" and the "binding theory" have yet to be formulated with sufficient explicitness for it to be determined whether they have a rule structure at all, let alone a simple one, and the existence of parsing algorithms for them, of any sort whatever, has not been demonstrated.

The real reason to reject a cubic recognition-time guarantee as a goal to be attained by syntactic theory construction is not that the quest is pointless, but rather that it is not nearly ambitious enough a goal. Anyone who settles for a cubic TCR bound may be settling for a theory a lot laxer than it could be. (This accusation would be levellable equally at TG, lexical-functional grammar, Pollard's generalized context-free grammar, and generalized phrase structure grammar as currently conceived.) Closer to what is called for would be a theory that defines human grammars as some proper subset of the ECPO CF-PSG's that generate infinite, nonprofligate, linear-time recognizable languages. Just as the description of ALGOL-60 in BNF formalism had a galvanizing effect on theoretical computer science (Ginsburg 1980, 6-7), precise specification of a theory of this sort might sharpen quite considerably our view of the computational issues involved in natural language processing. And it would simultaneously be of considerable linguistic interest, at least for those who accept that we need a sharper theory of natural language than the vaguely-outlined decorative notations for Turing machines that are so often taken for theories in linguistics.

ACKNOWLEDGEMENT

I thank Chris Culy, Carl Pollard, Stuart Shieber, Tom Wasow, and Arnold Zwicky for useful conversations and helpful comments. The research reported here was in no way supported by Hewlett-Packard.

References

Aho, A. V. and J. D. Ullmann (1977). Principles of Compiler Design. Addison-Wesley.

Bach, E. (1981). Discontinuous constituents in generalized categorial grammars. NELS 11, 1-12.

Berwick, R., and A. Weinberg (1982). Parsing efficiency and the evaluation of grammatical theories. LI 13.165-191.

Bresnan, J. W.; R. M. Kaplan; S. Peters; and A. Zaenen (1982). Cross-serial dependencies in Dutch. LI 13.613-635.

Chomsky, N. (1963). Formal properties of grammars. In R. D. Luce, R. R. Bush, and E. Galanter, eds., Handbook of Mathematical Psychology II. John Wiley.

Chomsky, N. (1981). Knowledge of language: its elements and origins. Phil. Trans. of the Royal Soc. of Lond. B 295, 223-234.

Cook, S. A. (1971). Linear time simulation of deterministic two-way pushdown automata. Proceedings of the 1971 IFIP Conference, 75-80. North-Holland.

Culy, C. D. (forthcoming). An extension of phrase structure rules and an application to natural language. Stanford MA thesis. of Linguistics, Stanford University.

Dowty, D. R.; R. Wall; and P. S. Peters (1980). Introduction to Montague Semantics. D. Reidel.

Gawron, J. M., et al. (1982). The GPSG linguistic system. Proc. 20th Ann. Meeting of ACL 74-81.

Gazdar, G. (1981). Unbounded dependencies and coordinate structure. LI 12.155-184.

Gazdar, G. and G. K. Pullum (1981). Subcategorization, constituent order, and the notion `head'. In M. Moortgat, H. v. d. Hulst, and T. Hoekstra (eds.), The Scope of Lexical Rules, 107-123. Foris.

Ginsburg, S. (1980). Methods for specifying formal languages---past-present-future. In R. V. Book, ed., Formal Language Theory: Perspectives and Open Problems, 1-47. Academic Press.

Pollard, C. J. (forthcoming). Generalized context-free grammars, head grammars, and natural language.

Pullum, G. K. (1982). Free word order and phrase structure rules. NELS 12, 209-220.

Pullum, G. K. and Gazdar, G. (1982). Natural languages and context-free languages. Ling. and Phil. 4.471-504.

Shieber, S. M. (1983a). Direct parsing of ID/LP grammars. Unpublished, SRI, Menlo Park, CA.

Shieber, S. M. (1983b). Sentence disambiguation by a shift-reduce parsing technique. In this volume.

Slocum, J. (1983). A status report on the LRC machine translation system. Conf. on Applied Nat. Lang. Proc. 166-173. ACL, Menlo Park, CA.

Zwicky, A. M. and J. M. Sadock (forthcoming). A note on xy languages. Submitted to Ling. and Phil.

Appendix: a 2DPDA that recognizes a Bach language

The language $\{\underline{x} | \underline{x}$ is in $(\underline{a} + \underline{b} + \underline{c})^*$ and $\underline{N}a(\underline{x}) = \underline{N}b(\underline{x}) = \underline{N}c(\underline{x})\}$ is accepted by a 2DPDA with a single symbol $\underline{z}$ in its stack vocabulary, $\{\underline{a}, \underline{b}, \underline{c}\}$ as input vocabulary, four states, and the following instruction set. State 1: move rightward, reading $\underline{a}$'s, $\underline{b}$'s, and $\underline{c}$'s, and adding a $\underline{z}$ to the stack each time $\underline{a}$ appears on the input tape. On encountering right end marker in state 1, go to state 2. State 2: move left, popping a $\underline{z}$ each time a $\underline{b}$ appears. On reaching left end marker in state 2 with empty stack (which will mean $\underline{N}a(\underline{x}) = \underline{N}b(\underline{x})$), go to state 3. State 3: move right, pushing a $\underline{z}$ on the stack for every $\underline{a}$ encountered. On reaching right end marker in state 3, go to state 4. State 4: move left, popping a $\underline{z}$ for each $\underline{c}$ encountered. On reaching left end marker in state 4 with empty stack (which will mean $\underline{N}a(w) = \underline{N}c(w)$), accept.