Robert C. Berwick
Amy S. Weinberg

Parsing Efficiency, Computational Complexity, and the Evaluation of Grammatical Theories

1. Introduction: Grammars and Parsers

Modern linguistics has quite generally been taken to encompass the study of *what* constitutes knowledge of language and *how* that knowledge is put to use:

> The fundamental fact that must be faced in any investigation of language and linguistic behavior is the following: a native speaker of a language has the ability to comprehend an immense number of sentences that he had never previously heard and to produce, on the appropriate occasion, novel utterances that are similarly understandable to other native speakers. The basic questions that must be asked are the following:
>
> 1. What is the precise nature of this ability?
> 2. How is it put to use?
> 3. How does it arise in the individual?
>
> Chomsky and Miller (1963, 271)

It has also been quite widely assumed that the answer to the first question would go a long way toward providing a firm scientific basis on which to answer questions 2 and 3. The methodological import of this position is plain enough. Assuming that language use is, in some sense, an "implementation" of the system of linguistic knowledge, and assuming that the theory of language learning specifies how that system of knowledge can be acquired, it makes little sense to attempt to characterize either implementation or learning procedure before understanding, at least in part, just what that knowledge is. One would even expect (as has turned out to be the case) that a partial answer to question 1 would provide a great deal of insight into the answers to questions 2 and 3. This methodological slant should not, of course, be taken as implying that the investi-

We would like to thank Ed Barton, Noam Chomsky, Jerry Fodor, Gerald Gazdar, Norbert Hornstein, Thomas Wasow, and two *Linguistic Inquiry* referees for stimulating discussion and guidance during the writing of this article. Berwick is supported by the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's research is provided in part by the Office of Naval Research under Office of Naval Research contract N00014-80-C-0505.

Linguistic Inquiry, Volume 13, Number 2, Spring 1982
0024-3892/82/020165/27 $02.50/0

gation of grammar should have little contact with theories of language use or language acquisition, or, worse yet, that a complete understanding of language ends with the study of grammar. It simply claims that a proper way to *begin* the study of language is to start with a characterization of what that knowledge is—in short, with a theory of grammar.

Researchers working within the framework of generative grammar have taken the answer to question 1 to consist in a specification of the *class of possible grammars*. A member of the class of possible grammars is taken to be a formal characterization of a person's linguistic knowledge (or competence). Membership in the class of possible grammars is limited to just those characterizations that can be embedded in a theory capable of solving the "logical problem of language acquisition". The solution to this latter problem consists of a theory that explains how children acquire their language in a short period of time on the basis of radically degenerate and deficient data.[1] It has been intermittently proposed that this research scenario should be inverted: that one can gain insight into the nature of the system of knowledge that makes up the language faculty by considering *how* that knowledge is put to use (or acquired) in real time. In particular, since different grammatical theories are associated with different parsing models, some of which will be *plausible* and others less so, it is argued that one can exploit a theory of language use to constrain the class of possible grammars by insisting that a grammar is "possible" just in case it has an associated *plausible parsing model*.

For instance, one could argue that the class of natural languages (more precisely, grammars for those languages) be constrained by insisting that parsers associated with putatively possible grammars reproduce the detailed time complexity of human sentence processing; that is, sentences that are complex under some measure of psychological resource complexity (e.g. take a long time for people to analyze) are correspondingly complex for the processing model (e.g. take a long time for the model to analyze). A strengthened version of this condition might require that the model preserve an ordinal ranking of all sentences under the psychological complexity metric.

In this article we focus on proposals that suggest that one can constrain the class of possible grammars by imposing just such a cognitive fidelity requirement on the class of possible grammars, a criterion of *efficient parsability*. It is generally assumed that people can process sentences quite rapidly. To impose a condition of efficient parsability, then, is to claim that sentence processing models should reproduce this aspect of human behavior. This is the requirement advanced, for example, by Marcus (1980, 240–241):

> But there is another fact about language behavior that is only slightly less marvelous: that language works at all. It is far from apparent how the mind, given only the speech waveform, or even a string of written words, can reconstruct linguistic structure in something like real time. From this, it seems reasonable to assume that language must be constrained in ways which make it amenable to efficient generation and recognition.

[1] The "logical problem of language acquisition" should not be confused with questions about the actual time course of the acquisition process itself. See Chomsky (1981) and Hornstein and Lightfoot (1981) for discussion.

Couched at this informal and general level, such a quasi-functional view seems untendentious: if the "language faculty" is even roughly analogous to other organs of the body (like the heart), then we might reasonably expect, just as in the case of other systems of the body, that it has been "well designed" according to some as yet undetermined criteria of efficiency. This scenario clearly takes for granted the usual backdrop of natural selection. Since one of the evolutionary "design criteria" could well have been ease of language processing, it is certainly conceivable that efficient parsability has played a role in the shaping of the language faculty.

It is another question entirely, however, to take the position that considerations of parsing efficiency allow us to restrict the class of possible grammars to just those capable of generating certain *mathematically* defined classes of languages, because these languages, and no others, can meet the demand of efficient parsability. Specifically, it might be claimed that we should deliberately restrict our study to phrase structure grammars that can generate only context-free languages. This is because there are known efficient parsing algorithms to recognize (and parse) *any* language that is context-free; in contrast, there is no way to guarantee that *all* context-sensitive languages can be as efficiently parsed as the context-free languages, and of course broader classes of languages beyond the context-sensitive need not even have algorithmic recognition procedures.[2]

Such a line of reasoning appears to provide a simple a priori way to reject the theory of transformational grammar (TG) as a "psychologically realistic" account of the language faculty. Since otherwise unrestricted TGs can generate all the recursively enumerable languages (Peters and Ritchie (1973b)), and since (modestly restricted) TGs generate languages that cannot be parsed in less than exponential (2^n) time (Rounds (1975)), we can apparently conclude that there are theories of transformational grammar that generate languages for which we have no known efficient general parsing algorithms. Therefore, some researchers conclude, a theory of grammar that happens to generate only languages for which there *are* known efficient parsing algorithms has an important advantage over the theory of transformational grammar. In this view, for instance, if one restricts attention to the study of systems that generate only context-free languages (perhaps for independently motivated reasons, e.g. the usual linguistic reasons), then an important side benefit accrues because the entire class of context-free languages is already known to have "efficient" parsing algorithms. This is the view that Gazdar, for one, has advocated (1981, 155):

> Suppose, in fact, that the permitted class of generative grammars constituted a subset of those phrase structure grammars capable only of generating context-free languages. . . . we would have the beginnings of an explanation for the obvious, but largely ignored, fact that humans process the utterances they hear very rapidly. Sentences of a context-free language are provably parsable in a time which is proportional to the cube of the length of the sentence

[2] Readers who are unfamiliar with the characterization of classes of languages with respect to weak generative capacity known as the *Chomsky hierarchy* are referred to standard textbooks, especially Hopcroft and Ullman (1979) or Lewis and Papadimitriou (1980).

or less (Younger (1967), Earley (1970)). But no such restrictive result holds for the recursive or recursively enumerable sets potentially generable by grammars which include a transformational component.

The form of Gazdar's argument is simple enough:

(1) People parse sentences rapidly.
(2) The sentences of any context-free language can be parsed rapidly.
(3) Gazdar's phrase structure grammars generate only context-free languages.
(4) Not all languages generated by transformational grammars can be parsed rapidly.

Conclusion: The theory of transformational grammar cannot provide an explanation of how people parse sentences; in contrast, a theory that can (weakly) generate only context-free languages can provide such an explanation.[3]

But is this line of argument valid? We believe not. In the first part of this article we show that when the relevant formal language theory results are set in their proper real-world context, they do not choose between phrase structure and transformational grammars in the manner that Gazdar suggests. According to the argument above, what the property of context-freeness buys is a guarantee that TG cannot meet, that of efficient processability. But the identification of all and only the context-free languages as the "efficiently processable" languages is misleading. On the one hand, strict context-sensitivity is not an absolute barrier to efficiency in the manner implied by the argument above, since many strictly context-sensitive languages can also be efficiently analyzed. On the other hand, natural languages are patently a *restricted* subset of some class in the Chomsky generative hierarchy (the obviously unnatural languages must be excluded, no matter whether the attainable natural languages turn out to be context-free or not). Therefore, even if a context-free hypothesis is adopted, additional constraints must still be imposed, beyond mere context-freeness, in order to characterize all and only the

[3] We should be very clear that an argument about the computational benefits accruing from a restriction to context-freeness need not be the only reason, or even the primary reason, for preferring grammars that generate only context-free languages. Gazdar (1979; 1981; forthcoming) has argued that his nontransformational theory of grammar is superior primarily on the standard linguistic grounds, namely, that it accounts for linguistic generalizations better than the theory of transformational grammar. (However, see Williams (1981) for a partial reply to these arguments.) In a personal communication, Gazdar has informed one of us that the original motivation for the theory was a desire to eliminate the transformational component of TG theory entirely. Only afterwards was it realized that this move, since it permitted the generation of only context-free languages, might also have computational import. In this context, it is interesting to note (see Chomsky (1981)) that the generative grammars proposed in Chomsky (1951) were in fact systems that used the indexed phrase structure mechanisms described by Harman (1963) and Gazdar (1981). In this article we will focus attention on just the question of efficient parsability, leaving other questions aside.

From here on we shall use the term *phrase structure grammar* to mean a phrase structure grammar as described in Gazdar (1979; 1981) instead of the formal language theoretic sense of 'unrestricted rewriting system' (Type 0 grammar). The sort of phrase structure grammar advanced by Gazdar is based on formal work by Peters and Ritchie (1973a) and Joshi and Levy (1977), who show that these types of grammars are more faithful to the linguistic notion of "immediate constituent analysis" than are unrestricted phrase structure grammars.

natural languages. Since additional characterizing constraints must be investigated even in the context-free case, it would seem just as legitimate to look at those strictly context-sensitive (or even strictly Type 0) languages that are efficiently processable, given additional constraints. Since there are such languages and candidate additional constraints, the actual mathematical results do not preclude the possibility of either TG or phrase structure grammars being able to generate languages for which there are efficient parsing algorithms.[4]

In the second part of the article we consider more carefully the application of general mathematical results to a cognitive domain. We will see that biologically relevant parsing efficiency need not be primarily determined by general, mathematically defined measures of efficiency. Therefore, although mathematical efficiency measures may apply in an abstract, formal sense to rank (context-free) phrase structure grammars as "better" than transformational grammars, this ranking may be relevant only in formal theory, not in biological practice. The commonly used mathematical measures of efficiency—including the one cited by Gazdar—by and large abstract away from the structural features of parsing algorithms that may actually dominate the efficiency of a procedure in the biologically relevant sense. In particular, we will see how the size of a grammar (as embedded in a parsing procedure) can contribute to the efficiency of a parsing algorithm, and how, because grammar size shrinks with the move to more powerful descriptive formalisms, there is a possible trade-off between parsing efficiency and descriptive apparatus.

Our investigation of the use of "computational complexity" arguments to choose among alternative grammars also calls into question more generally the applicability of general mathematical results in a narrow cognitive domain. We will see that since parsing efficiency clearly depends upon the representational format chosen for computation, it may well be that narrow, highly-tailored representations for the *particular* grammatical formats associated with natural languages may allow a correspondingly particular and nongeneral algorithm to be quite efficient, as opposed to whatever general-purpose parsing algorithm one might propose. It seems likely that some of the computational work formerly done by the parser could be shouldered by the narrowness of the restricted set of representations under consideration. This is in fact the experience of those who work with programming languages: specific grammars admit specialized recognition procedures.

As will become clear, we find arguments such as the one sketched above to be symptomatic of an all-too-common confusion about the role of mathematics in linguistic theory. For instance, it is worth noting that the Peters and Ritchie result that an unrestricted transformational theory can generate all recursively enumerable sets has been used by some researchers as an argument that transformational grammars cannot be "psychologically real". The reason: since some recursively enumerable sets do not even have algorithmic parsing procedures, it must be the case that unrestricted transforma-

[4] See footnote 10 and Berwick (forthcoming).

tional grammars can generate languages for which no parsing procedure even exists, let alone an efficient one. The context-free language/efficient parsability argument is in effect merely a subrecursive analogue of this more general argument.[5]

In the broadest sense then, the aim of this article is to shed some light on just what the proper application of mathematical theory should be to an ultimately biological (and empirical) science. In this attempt at illumination, we shall touch upon a variety of related questions that have from time to time been raised in contemporary discussions of linguistic theory: Just what is the status of so-called "functional explanations" in linguistic theory? Can arguments grounded on notions of parsing efficiency provide telling restrictions on the form of our linguistic theories?

While our aim is to dispel certain technical confusions about the applicability of mathematical results to the domain of linguistics, we should stress that we are *not* implying that mathematical argument is of no value in the study of language. On the contrary, when the results of mathematical analysis are evaluated in the proper empirical context, they can provide (and have provided) insights of enormous depth. The point of the article, then, is not to furnish a simple, sweeping conclusion that all purely a priori arguments about language based on mathematical results are invalid, or that thinking about parsing efficiency is a worthless enterprise. There is no difficulty with admitting additional *valid* sources of evidence bearing on theories of language, be it from the domain of mathematics, reaction time experiments, or observations of child development. The problem is that such arguments seem to be far more difficult to make properly, at least given our current understanding.[6] In short, the moral of this article is that mathematical insights culled from the study of formal languages must be tempered with a sensitivity for the biological and empirical situation to which they are applied. In this we can do no better than to recall Kripke's (1976, 416) observation regarding the role of mathematics and formalization in philosophical thinking:

> Logical investigations can obviously be a useful tool for philosophy. They must, however, be informed by a sensitivity to the philosophical significance of the formalism and by a generous admixture of common sense, as well as a thorough understanding of both the basic concepts and of the technical details of the formal material used. It should not be supposed that the formalism can grind out philosophical results in a manner beyond the capacity of ordinary philosophical reasoning. There is no mathematical substitute for philosophy.

2. Parsing Efficiency and the Use of Mathematical Results

2.1. Context-free Languages and Efficient Parsability

To begin our study of mathematical arguments and their bearing on parsing efficiency and linguistic theory, we must first review the basic mathematical results about general

[5] See Lapointe (1977) for discussion of the significance of the Peters and Ritchie results; Matthews (1979) and Chomsky (1980) for a summary. See Berwick and Weinberg (forthcoming b) for a discussion of the Peters and Ritchie results within the current Government-Binding theory (Chomsky (1981)).

[6] For additional discussion of the difficulties, see Berwick and Weinberg (forthcoming a).

context-free parsing that have been most used to bolster various arguments about the difficulty of parsing natural languages.

The efficiency of general context-free parsing algorithms is most often couched as some function of the size (or length) of the input sentences that the parsing algorithm must analyze. Why is the efficiency of an algorithm expressed in this way? The intuition behind this approach is that as the sentences input to some parsing procedure grow longer and longer, the amount of computational work that must be done to analyze the sentences should likewise grow, while holding fixed the algorithm that is used. Thus, the *processing complexity* of algorithm i given a sentence n words long is typically denoted as $f_i(|n|)$. If computational resources are measured in terms of time, then the function f_i is simply a formula that, given the length of the sentence to be processed (measured in number of words), tells us how long algorithm i will take to finish its work. For example, if $f_i = n^2$ and the measure of computational work is time, then a sentence 10 words long will take four times longer to process than a sentence 5 words long—doubling the sentence length will quadruple the amount of computational effort that must be expended.

As implied in the preceding paragraph, computational work itself can obviously be measured in several ways, the most natural (and well-known) being the number of "time steps" taken by a procedure or the (maximum) amount of space used during the course of analyzing a sentence of a given length. Finally, it should be apparent that because what counts as a unit time step or a unit of space can vary from one computational model to another, the exact complexity of sentence processing might vary not only from algorithm to algorithm, but also from machine to machine, depending upon what brand of computer one adopts as a model for computation.

It should be no surprise, then, that one desirable feature of a mathematical theory of computation should be the ability to prove results that are invariant with respect to the model adopted for computation. Present-day computational complexity theory achieves this aim by focusing on functional rates of growth that are "far enough apart" to be unaffected by changes in reference machine. If one function can be computed faster on algorithm A than algorithm B when using, say, a multi-tape Turing machine as a reference model, and if that difference in speed is large enough, then the superiority of algorithm A will be preserved when it is switched to another (serial) computational reference machine, say, a random access machine.[7]

[7] A random access machine, or RAM, differs from a Turing machine in its ability to store numbers in any one of a finite number of *registers*, to which it has immediate access. This would seem to be a clear computational advantage, since such a machine can retrieve or modify these numbers in a constant amount of time. In contrast, a Turing machine must laboriously scan a storage tape in a strict left-to-right or right-to-left fashion in order to get to a particular symbol it must retrieve. Given this difference in power, it is an interesting fact that the *class* of functions that can be computed in time less than or equal to n^j for some integer j on a random access machine is exactly equal to the class that can be so computed on a Turing machine. (This class is that set of functions that can be computed in "polynomial time" on a deterministic Turing machine. Context-free language recognition, because it can be done in cubic time or less on a RAM, is in this class.)

Thus, moving to a RAM buys no more power in terms of the class of functions that can be computed in polynomial time. This result makes precise the sense of "far enough apart" alluded to in the text. If algorithm A runs in polynomial time (say, n^3) and algorithm B in exponential time (2^n), then A will be superior to B on any "reasonable" model of a serial computer, be it Turing machine or RAM.

The price paid for this ability to state complexity results independently of machines is the usual price of abstraction: the complexity metric may be a dull knife, unable to draw the requisite distinctions in the cognitive domain. We shall see that this is at the root of many of the difficulties surrounding the use of computational complexity results in linguistic theory. However, there does not seem to be any way around this part of the problem; if one is aiming at theorems that remain theorems even when the underlying machine is subject to gross changes, then one must be willing to forego theorems that can be proved only with respect to particular brands of computer, including, perhaps, the human "computational machinery".

What about the particular case of parsing context-free languages? An amount of time proportional to roughly the cube of the number of input words is known to be sufficient to parse any context-free language (on a computer model like a random access machine) (Earley (1968; 1970)).[8] Note that cubic time is known to be *sufficient* but is not known to be a *necessary* bound on the amount of time required. Indeed, all known context-free languages are recognizable in only an amount of time that is proportional to simply the number of words in the input string (i.e. a linear function of the input length, or simply $k \times n$, where k = some constant and n = the length of the input sentence in words).[9] Further, even if the cubic bound were shown to be necessary for *some* context-free language, it does not then follow that *all* context-free languages would take that much time to parse; only some "hardest" language need take that much time. Similarly, it is known that an amount of time proportional to an exponential function of the length of the input string, k^n, is sufficient to parse any context-sensitive language, but this too is an upper bound; many context-sensitive languages do not require that much time for parsing.[10]

[8] See the previous footnote for a brief description of a random access machine. This result is also proved, for example, in Hopcroft and Ullman (1969) or Harrison (1978, 430–437). Slightly "better" functional speed-ups are possible, reducing the n^3 time to nearly $n^{2.5}$. (The best result as of 1980 was $n^{2.52\cdots}$.)

[9] Recall that we are assuming all other potential sources of variation—e.g. the grammar—to be constant.

[10] The two examples of strictly context-sensitive (non-context-free) languages most often cited in the literature—$a^n b^n c^n$ and ww—are both recognizable in faster than linear time: whether or not a given string (sentence) is in either of these languages can be determined by using an algorithm that takes only a bounded number of steps (fixed in advance) between the reading of each symbol of the input sentence, that is, in real time. (See Rosenberg (1967) and Galil (1978).) However, it is important to keep in mind the distinction between *recognizing* a language (= determining simply whether or not a string is in the language generated by *some* grammar for that language) and *parsing* a language (= finding the derivation tree by which the sentence can be generated with respect to some *particular* grammar). In other words, a parser has to do more than a recognizer—it has to recover the correct labeled bracketings for sentences. A recognizer need only determine whether or not the string is in the language generated by a grammar, and may use whatever means at its disposal for this task. The notion of recognition is weaker, because it can be readily shown that there are some languages that can be efficiently recognized but not necessarily efficiently parsed *according to the rules of a given grammar*. For example, consider the following grammar: S → A; A → aAa; A → a. This grammar generates a language consisting of strings of odd numbers of *a*s. This language is clearly a finite-state language (see, for instance, Hopcroft and Ullman (1979)), and therefore it can be efficiently "processed" if the only demand is recognition. However, if the requirement is to recover the labeled bracketings that the above grammar would assign to such strings, then a deterministic push-down automaton *incorporating this particular grammar* will not suffice. (The example is from Knuth (1965); a grammar that can be so incorporated into a deterministic push-down automaton is called an *LR(k) grammar*.) Some other grammar *can* be used by a deterministic push-down automaton to parse the strings of this language (that is, such a machine can assign labeled bracketings to the strings of this language according to this new grammar (Knuth (1965))). But this other grammar is not structurally identical to the first—it does not preserve the "tree shapes" of the original

It is important to keep in mind just what these sorts of results show when applied to the case of natural languages. In its usual mathematical formulation, the "efficiency rating" of a given class of languages is determined by the "worst case" language for that class, that is, the language in the class that requires the most time to parse. Formally, if we let P_i be the parsing time with respect to language i, index i ranging over some class of languages L, then the parsing complexity of the class L—call this property P—is $max(P_i)$. For example, if we say (counterfactually) that the class of context-sensitive languages is of exponential time complexity, this means that *at least one* context-sensitive language requires exponential time; many of the other (strictly) context-sensitive languages in the class might require much less time (and indeed this is the case, as noted in footnote 10). This is the reason for the "sufficiency vs. necessity" distinction noted above. If such "worst case" theorems about mathematically defined classes of languages are to be applied directly to the class of possible natural languages (NL), then we must assume that the particular language used to demonstrate property P is itself in NL. This requirement is a sensible one to enforce, since, after all, we are clearly aiming to use property P as a (partial) characterization for what it means to be a natural language, and the proposed property would not be a very appropriate one if no natural language possessed it. A characterization based on parsing efficiency obviously is successful insofar as it helps us to characterize all and only the natural languages.

With this background in mind, consider again the argument to restrict the study of language to the study of rule systems that generate only context-free languages, because efficient general parsing methods exist for *any* context-free language. On this view, the property of "efficient parsability" (EP) that all context-free languages enjoy is a *partial* characterization of what it means to be a natural language. Languages that do not have property EP are simply not natural languages. A Chomsky hierarchy diagram of this situation is given in figure 1. The class NL has been identified with the class EP, and, since all members of the class of context-free languages (CFL) are also efficiently parsable under our current definition of "efficient", CFL is identified with EP as well. (Depending upon whether additional characterization constraints are imposed on the class NL, NL might be a *proper* subset of the CFL class, a matter to which we return shortly; we also ignore the question of just where the class NL is bounded from below.)

---

grammar. This being so, then if, say, semantic interpretation "runs off of" the bracketing provided by Grammar 1, the parse with respect to the Grammar 2 might not preserve the semantic properties of the first grammar. It may however still be possible to find an "efficient" procedure to translate between the parse trees provided by Grammar 2 and those of Grammar 1, thus preserving the semantics of the first grammar. (Familiar examples include the "readjustment rule" discussed by Chomsky and Miller (1963) and Langendoen (1975) for translating between center-embedded and right- or left-recursive trees, or Kuno's Harvard Syntactic Analyzer (1966) that translated context-free grammars into a non-left-recursive form so that they could be efficiently parsed top-down.) Formal research into this possibility goes under the heading of the theory of *covering grammars*. See Nijholt (1980) and Berwick and Weinberg (forthcoming a) for a more extensive discussion of covering grammars and their implications for the study of the relationship between grammars and parsers.

Given the distinction between recognition and parsing, the relevant formal result for the purposes of the discussion in the text is that Knuth's LR(k) condition can be imposed on grammars for non-context-free languages (e.g. strictly context-sensitive and even strictly Type 0 languages) so as to obtain parsing procedures that execute quite efficiently. (Specifically, in time proportional to the lengths of derivations of sentences; in the $a^n b^n c^n$ case, the time is at worst quadratic in the length of the input.) See Walters (1971) for these results, and Berwick (forthcoming).

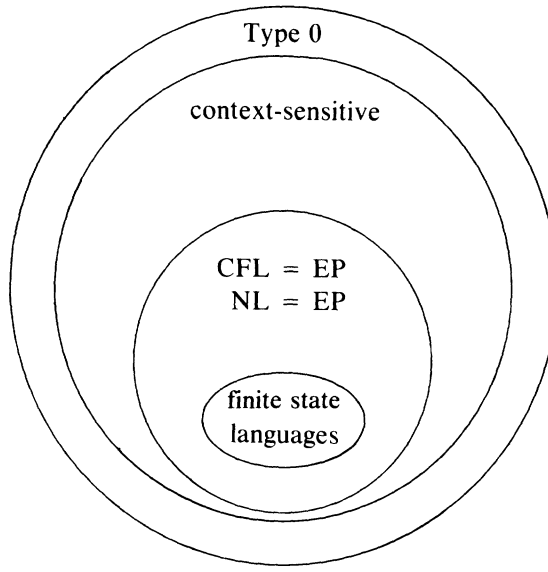

Figure 1

Strictly interpreted, though, this particular "state-of-the-world" diagram is misleading. "EP" is *not* a property unique to the context-free languages, since many strictly context-sensitive languages share the property of efficient parsability. So, *given that our sole criterion is for the moment efficient parsability,* we should include the strictly context-sensitive languages with property EP in our diagram, expanding the class NL beyond the strictly CFL boundary:

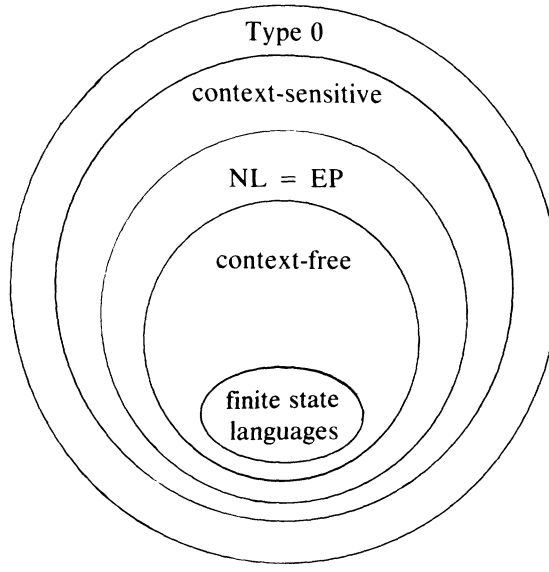

Figure 2

Pursuing the parsability characterization, recall that the parsability theorems also tell us that any context-free language is efficiently parsable. Hence, all the *un*natural languages that are context-free are also in the class EP, no matter how bizarre. On this account, mirror-image languages (generating only palindromes, like *abbaabba*) are also candidate natural languages, because they too are context-free and hence easily analyzed. Since these sets of strings are apparently not natural languages, it is clear that some context-free languages must be ruled out as natural languages based on criteria other than that of efficient parsability.[11] Thus, we must alter our diagram once more, redrawing the class NL so that it excludes some members of the class of context-free languages, and yet runs outside that class. Let us also assume that natural languages exceed the weak generative capacity of finite state (right- or left-linear) grammars. Our hierarchy diagram now looks like figure 3 (p. 176).[12]

What has the efficient parsability criterion bought us in this case, then? We can say for certain only that the class NL *cuts across* the context-sensitive and context-free language classes in some as yet undetermined fashion. But this is precisely what has generally been observed since the earliest mathematical work on the subject: the class

[11] For example, we might insist that the theory prohibit rules of the form S → ASA, S → BSB, S → empty, perhaps for reasons that can be independently maintained under some version of X-bar theory. Then at least the obvious context-free grammar for generating the palindrome languages would not be admissible. This example incidentally demonstrates that, if one takes seriously the view that what matters about grammars is their strong generative capacity (the structural descriptions, or labeled bracketings, that they can produce), then what is important for cognition is not so much the recognition time of a language (a process that may, but need not, ignore a particular grammar that weakly generates the language so as to use another grammar that is more amenable to efficient recognition) but rather *parsability with respect to a particular grammar of interest* that generates that language. For instance, as footnote 10 points out, it may be irrelevant that the string a^n for odd n can be recognized by a finite state device (hence in real time) if the underlying grammar is, for independent reasons, known to be of the form S → aSa, S → a. In short, it is the notion of *grammar*, in the sense of a system that pairs surface strings with labeled bracketings, that is crucial for parsing; the notion of *language*, in the sense of some set of strings, is derivative and not of interest independent of the grammar.

In one sense it is easy to see how this focus on the complexity classes for string recognition (and a corresponding attention to the notion of language as opposed to grammar) may have arisen. In the formal study of recognition complexity, the underlying set of possible grammars that generate a language of interest can ordinarily be freely varied so as to achieve the fastest possible recognition time for that language. But presumably the same freedom is *not* available in the study of natural languages. Rather, the situation is reversed: it is the grammar that is subject to constraints, and parsing efficiency must be evaluated with respect to that grammar, whatever its constraints turn out to be. These constraints might include: restrictions drawn from X-bar theory; compatibility with logical form or semantic interpretation rules (cf. the discussion of covering grammars in footnote 10) so as to provide the "right" input for semantic interpretation rules; a demand that grammars be projectible given the "right" conditions of exposure to data, and so forth. Of course, the set of surface strings the grammar generates does matter to parsing complexity, but the notion of grammar as (string, bracketing) pairs subsumes the contribution that the set of strings makes to the computational issues at hand. For further discussion of the grammar–parser relationship, see Berwick and Weinberg (forthcoming a).

[12] Note that Gazdar (1981) also assumes the class NL to be a proper subset of the class of context-free languages, as evidenced by the quotation at the beginning of this article. The intent of his restriction is presumably to rule out the "unnatural" context-free languages, perhaps using the criteria suggested above. According to Gazdar's argument, a powerful reason that the restriction to just the CFL languages pays off is that of parsability. However, as we have just seen, since we must in any event look at a restricted class of languages, one not coinciding with any of the Chomsky hierarchy classes, it makes little sense to exclude in an a priori fashion just those non-context-free languages that are also efficiently parsable, simply because they do not happen to meet some other mathematical condition.

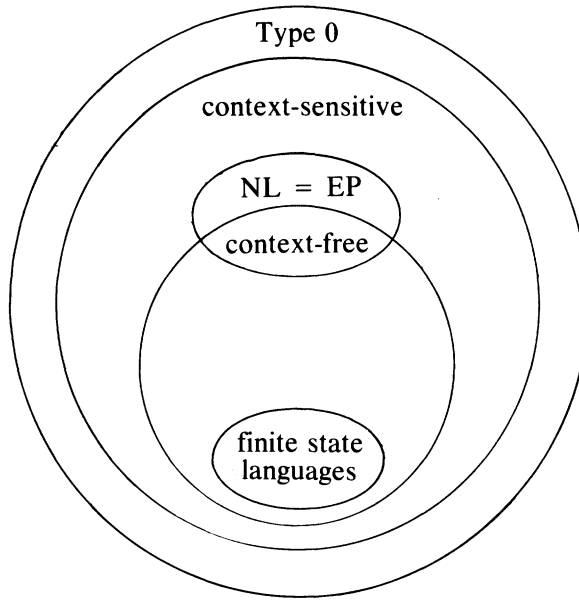

Figure 3

of natural languages is suspected to be some subset of the class of context-sensitive languages, including, perhaps, some non-context-free languages. Even this much is not certain, however; it is an open question whether there are natural languages that are, for example, nonrecursive or even nonrecursively enumerable (see Hintikka (1974) and Chomsky (1980) for discussion).[13] In this case, then, the imposition of the parsability criterion has told us nothing new at all; if our goal is to obtain as narrow a characterization of the class of natural languages as possible, then requiring that all natural languages possess property EP does not advance us beyond what has been suspected since the early 1960s.[14]

[13] See Postal (1964) for putative examples of strictly context-sensitive constructions in natural languages, and Gazdar (1979) for contrary evidence; the matter still seems unresolved. Even in the case of Type 0 languages, it is possible to add "locality restrictions" of the sort observed in current transformational theory and obtain a language that is efficiently parsable. Figure 3 would have to be correspondingly modified so as to show the "NL" class cutting across the Type 0 boundary. Such languages need not be context-sensitive, yet they may still be efficiently parsable. (Since these languages are algorithmically parsable, they obviously must be recursive, however.) We conjecture that the "locality principles" of current transformational theory conspire to guarantee that TG languages are processable by a bounded lookahead machine of the type designed by Marcus (1980). For example, it can be shown that a Marcus-type machine can parse $a^n b^n c^n$ in at worst quadratic time. Note that these locality principles do not entail a restriction to strictly context-free languages. See Berwick (forthcoming) for details. See Matthews (1979) and Chomsky (1980) for discussion of the issue of recursiveness vs. nonrecursiveness.

[14] It might be argued that the banality of the conclusions that can be drawn from this formulation of the efficient parsability criterion follows from the weakness of the definition of "efficient". Perhaps we could provide stronger constraints on parsing time, and these would prove to be more telling ones. Suppose, for example, that we restricted the notion of "efficiently parsable" even further, demanding that parsing time be a *linear* function of the length of input strings. But this will leave us in roughly the same position as before,

Summarizing the discussion, the problem with identifying context-free languages with the class of natural languages is that this characterization is at once too broad and too narrow. It is too broad in that it includes too many languages: not all context-free languages are natural languages. But it is also too narrow: if the efficient parsability argument is *only* that natural languages be efficiently parsable, then there would seem to be no reason to exclude (by fiat) that subset of strictly context-sensitive languages whose members are also efficiently parsable. In short, the fault lines of "efficient processing" do not fall neatly into place along the context-free/context-sensitive/Type 0 language boundaries.

> . . . the characterization of families of languages by means of resource bounds for parsing or recognition will not necessarily yield known families specified by other means.
>
> R. Book (1973, 15)

2.2. *The Relevance of Computational Complexity to Linguistic Study*

The preceding discussion raises the first important point about the use of mathematical results in cognitive science. What the previous arguments have shown is that one cannot simply assume that a class of languages derived via a consideration of purely mathematical properties will correlate in a neat one-for-one fashion with the class of natural languages; a mathematically relevant class need not be coextensive with a cognitively relevant one. In the case above, the mathematical property of context-freeness could not be shown to be coextensive with the (assumed) cognitively relevant property of efficient parsability. We might summarize this problem as follows: *mathematical relevance need not imply cognitive relevance.*

A further question of cognitive relevance arises with the mathematical results because of the way in which results in the complexity literature are couched. The typical measure of complexity used in the theorems most often cited to back up efficient parsability arguments is the notion of *asymptotic complexity,* and it is not clear whether

---

since some strictly context-sensitive languages as well as some obviously unnatural languages are recognizable in linear time or less.

It is true that stronger recognition time constraints in conjunction with restrictions on the assumed underlying reference machine can be quite potent in eliminating certain classes of languages. For example, if we impose a linear time recognition constraint (recognition can take only time $k\,n$, where n = the length of the input sentence) and in addition limit the machine to only one tape, then the string sets so recognizable are regular (from Hennie (1965)). Furthermore, there are some languages that are known to be recognizable in real time on one kind of reference machine (a random access machine) that are not recognizable in real time on a Turing machine. Not surprisingly, if one adopts this finer-grained complexity analysis, one can distinguish between various brands of machines. The problem then becomes one of justifying the machine restriction and the grain size of the complexity analysis. What evidence can be adduced that people are one-tape Turing machines, as opposed to random access machines, or two-tape Turing machines, or even k-tape Turing machines? What evidence is there that people process sentences in real time, as opposed to, say, linear time?

In short, given this lack of understanding of constraints on the human "computational machinery", it seems that even under the most stringent restrictions of time efficiency it still remains true that the non-context-free language constructs generally hypothesized to be operative in natural language are recognizable (see footnote 10). Thus, even under quite restricted senses of "efficient", it appears that an a priori identification of the natural languages to some subset of the context-free languages cannot be maintained.

asymptotic complexity is a biologically relevant measure of computational complexity. Let us run through an example of the use of asymptotic complexity measures to see just what is at stake here. Recall that any context-free language can be recognized in time proportional to n^3, where n is the length of the input sentence in words. (One might want to include grammatical formatives in this count as well.) The "exact" recognition complexity would therefore be k times n^3, where k is a constant of proportionality. Suppose we wish to compare the time complexity for this algorithm against some other procedure of time complexity, say, $k^* f(n)$. For instance, $f(n)$ might be n^2. As n approaches infinity the ratio of the two complexities $k\ n^3/k^*\ f(n)$ will in the limit be dominated by the functional terms alone, and hence could be expressed more simply as just $n^3/f(n)$.[15] This is because the constant factors, though possibly large, are fixed. In contrast, the two functional terms—n^3 and $f(n)$—get larger as the input sentence length, n, increases. Eventually, no matter how large the constant terms were to begin with, the $n^3/f(n)$ factor will be many thousands of times larger (or smaller) than the constants, and thus the constants will play no role in the comparison of one complexity formula against another. Hence the term *asymptotic complexity* for this kind of evaluation measure of computational efficiency.

Evidently if asymptotic complexity is used, the constant terms in front of functional forms may be dropped for comparative purposes: we say that an algorithm runs in time n^3 or in linear time, without specifying the constant of proportionality; conversely, whenever such terminology is used, it has been tacitly assumed (unless stated otherwise) that the notion of asymptotic complexity is the relevant one.

Why would anyone adopt such a rough measure, one that can only distinguish between gross *functional* form differences? The reason is that the constants in front of the functional forms—the k and k^* in the example above—are parameters that are independent of the length of the input sentence but highly dependent on such "details" as the size of the grammar (total number of symbols required to write down the grammar); the representation of the grammar as a data structure (as a list, an array, a special look-up table); how rules are accessed and manipulated by the control structure of the parsing procedure; and the primitive operations available in the instruction repertoire of the assumed underlying machine. In short, the constants in front of the functional forms depend upon all those idiosyncratic "details of implementation" that vary from particular machine to particular machine. The use of an asymptotic measure is intended to deal with precisely this problem; by comparing procedures only in the limit of input lengths, we have abstracted away from such details of implementation.

From one standpoint then, the use of asymptotic complexity measures is widely considered to be an appropriate solution to the problem of how to deal with variation in computational models. A mathematical theory of computation would not be of much use if its results could be invalidated simply by purchasing someone else's computer.

[15] Except, of course, in the case where $f(n) = n^3$, where the ratio of the functional forms is 1 for all values of n. In this case, the constant terms will still predominate.

Invariance over changes in computational model is a desirable property of the theory of computational complexity, just because there is large variation in the choice of reference model computers.

Turning now to the domain of cognitive science, it is much less clear that asymptotic measures are equally appropriate. The problem is that in the case of human sentence processing we are studying one *particular* machine (though which one we do not know), and if we assume that the "design" of this machine has been optimized at all, then it has been optimized *with respect to that machine,* i.e. to the particular case of whatever computational machinery we do possess, and not necessarily with respect to some abstract mathematical measure of complexity that considers all possible machines of a given type.[16] In particular, consider the claim that an algorithm that runs in time n^3 is "better" than one that runs in time 2^n—this being, roughly, one of the senses of "more efficient" that an advocate of context-free parsing would rely upon. This comparison uses the notion of asymptotic complexity, and therefore, the domination of n^3 over 2^n is guaranteed *only asymptotically,* in the limit as the length of sentences input to either procedure approaches infinity. But if sentences long enough to ensure the domination effect never arise in actual, biologically relevant sentence processing, then the theoretical difference in parsing times may simply never amount to a practical difference. All those features ignored by adopting an asymptotic complexity measure may be precisely those that are most important for the range of problems that the organism must actually solve.

The difference could not be more plain. Asymptotic measures (including the usual worst case analyses cited in the literature) ignore the range of input sentence lengths and the constant factors in front of the functional forms that specify the computational complexity of an algorithm. In contrast, cognitive measures must focus on the particular range of input sentence lengths that is actually encountered in biological practice, for the "constant" values in front of the functional forms are proxies for the mental representations that parsing algorithms presumably are to use.[17] Note that these detailed

[16] There is no reason to suppose that parsing has been "optimized" according to *our* sense of machine design, of course; worse yet, evolution is known to be opportunistic, not optimizing. This is not just idle speculation: as researchers in evolutionary biology know, there are many examples of evolutionary "designs" that are quite inefficient by certain engineering standards but nonetheless survive because there is now no way to "rechannel" whole enzymatic systems into new ways of doing things. For instance, the photosynthetic machinery of plants is apparently ill-designed because the oxygen generated as a by-product acts as a competitor for the enzyme sites used in the fixing of carbon dioxide. Wheat would grow 20% more if the oxygen content of the air was 2% instead of 20% (Moore (1981)). Certain plants have evolved clever ways to side-step this defect by getting rid of the excess oxygen, but there is apparently no way at this late date to redirect the enzymatic pathways to a completely different system. The reason for the difficulty is apparently that the photosynthetic system evolved hundreds of millions of years ago, when the oxygen content of the atmosphere was in the 2% range. It is important to keep in mind, whenever casual evolutionary arguments are offered as "functional explanations" for one or another aspect of some cognitive faculty (or, for that matter, any biological competence), that the systems of an organism cannot be evaluated in isolation from one another: for example, it cannot be assumed without additional argument that the syntactic parsing machinery (if such exists) has been "optimized" independently of other cognitive subsystems or even independently of the entire organism of which it is a part.

[17] It might of course still turn out that asymptotic measures are appropriate. But this cannot be determined in advance of empirical investigation.

problems are not those that are encountered when we consider the efficiency of algorithms at the most abstract level; they arise only when we start to address the question of how a grammar might be actually put to use in the human language faculty—when we start to consider the dimensions of *empirical significance,* to use a term of Chomsky's. We shall call the first problem—that of determining whether asymptotic theoretical complexity differences occur in practice—the *Relevant Range* problem; the second we call the *Implementation* problem.

Let us consider first the Relevant Range question and its quasi-biological import, ignoring for the moment the question of the constant values in front of the functional forms in asymptotic complexity measures. In particular, let us assume the constants in front of the functional forms to be equal, say, 1. Then the value of an exponential form like 2^n would not begin to exceed a polynomial form like n^3 until n is greater than 9. In other words, given the operative assumptions, only sentences 10 or more words long would serve to distinguish between a parsing method that runs in time 2^n and one that runs in cubic time.[18] In other words, given the assumptions above, an argument based on algorithmic superiority is only valid if we add the assumptions that: (1) sentences of the break-point length or greater actually occur in practice; (2) it actually matters that one procedure can parse a single sentence 11 words long in half the time of another—presumably for reasons of expressive power; and (3) the language faculty has been "shaped" by natural selection primarily on the basis of the selectional advantage conferred by more efficient sentence processing (leaving aside the question of whether or not this is indeed the primary "role" of the language faculty). However, it is actually difficult to see under what conditions this alleged parsing advantage could arise in practice. Not only are we forced to envisage a case where the speedier parsing of a long sentence matters, and matters in some selectional sense, but also this difficult-to-parse sentence can have no two-sentence expressive substitute (for otherwise, it would come under the functional umbrella of the "slower" exponential procedure as well).[19]

[18] For example, with $n = 9$, $n^3 = 729$, but $2^n = 512$, so the exponential form is still "better"; whereas with $n = 11$, $n^3 = 1331$, but $2^n = 2048$, and the exponential form takes almost double the time of the polynomial. It must be stressed that we do not mean to imply that the rough trade-off described above accurately depicts what is the case in human sentence processing. Rather, the trade-off scenario is meant to be illustrative, showing how the range of sentence lengths that the cognitive machinery actually deals with is what is crucial to the practical *functional* evaluation of an algorithm, and not necessarily its asymptotic behavior. The trade-off discussed above is unrealistic because, among other things, if the measure of length includes grammatical formatives, then the "break-point" where an exponential time method would begin to take longer than a cubic time method might be different. As we shall see below, whether anything of functional import hinges on the inclusion of formatives in the length count depends in part on the size of the constants in front of the functional forms, and since this cannot be determined in advance of a detailed formulation of the complexity of human sentence processing, it would seem premature to advance any detailed argument along these lines.

[19] It might be an interesting exercise to examine the actual range of input sentence lengths in spoken (and even written language); one's initial impression, in fact, is that the bulk of sentences are shorter than the 10–12 word length break-point that conceivably separates cubic from exponential time. However, this simple break-point is almost certainly not correct, since it ignores the effect of constant multipliers in front of the relevant functional forms.

Moreover, it is well known that it is not so much length as it is structural factors such as degree of nesting (or more potently, degree of center-embedding) that contribute to sentence processing difficulties (Chomsky (1965)).

Thus, the distinction between, say, cubic time and exponential time procedures is possibly of no import in biological practice, depending upon the range of sentence lengths that actually mattered in the evolutionary "design" of language. This is a potential outcome even in the restricted case where (as we assumed) the procedures to be compared had time complexities of n^3 and 2^n—that is, identical constant terms of "1" entering into the calculation of their execution times. If we were to weaken this assumption to include a broader class of cases where the constant factors in front of the functional forms like n^3 or 2^n can vary radically, then it is obvious that the range of input sentence lengths over which an exponential time bound may actually be of practical superiority to a cubic time bound could be vastly greater. For instance, suppose an exponential time algorithm executes in exactly 2^n time steps while a competing cubic time algorithm is known to take 1000 n^3 steps to do the same processing job. Then the exponential algorithm will be superior to the cubic one for sentences 20 words or fewer in length.[20]

Evidently the constant factors that are quite properly ignored in asymptotic complexity analyses may actually be crucial to the analysis of complexity in a cognitive setting.[21] As a case study of this possibility, let us examine more carefully the complexity of general context-free parsing. Recall that for any context-free language the time to parse a sentence of length n is $k\,n^3$. The constant k, as mentioned, is a function of many other factors, including the *size* of the grammar, $|G|$; thus the "true" complexity is something like $k^*\ f(|G|)n^3$. Clearly, if the size of the grammar is very large compared to the typical range of input lengths, then it is the grammar size that dominates the overall complexity of the procedure.[22]

[20] In general, if the constant factor for the cubic method is c times larger than the constant factor for the exponential method, then one can expect the exponential method's superiority range to be extended by $\log_2 c$ words ($\log_2$ 1000 is approximately 10).

Presumably, part of the job of the cognitive psychologist is to try to find out whether people use cubic time or exponential time algorithms. What would seem to distinguish between these alternatives is which algorithm supports the right counterfactuals—that is, the ability to formulate statements such as, "If we increase sentence length by, say, one word at the point where exponential methods begin to take longer than cubic methods—does the time for analysis rise rapidly (thereby lending credence to the internalization of an exponential time method) or does it increase only modestly (a cubic time method)?" The problem is that the crucial test points may be well outside the range of psycholinguistic access: if the relevant (simple) sentences are, say, thirty or more words long, then other cognitive factors like attention span, memory, and the like may intervene so as to render such test cases problematic. On the other hand, it may well be that suitable test sentences can be constructed. This is a matter that can only be settled by empirical work.

[21] There is another familiar result in complexity theory that might be interpreted to mean that constant factors "don't matter" in the analysis of algorithms, and hence that grammar size may be safely ignored in the determination of parsing efficiency (see Hopcroft and Ullman (1979)). This is the theorem that states that any algorithm that runs in time $k\ f(n)$ can be recoded so as to make the constant k as close to 1 as desired ($f(n)$ must be a linear time function or larger). Since grammar size $|G|$ is just another "constant", this result would seem to indicate that one can recode an algorithm and eliminate the effect of grammar size.

However, a more careful analysis of the implicit assumptions of this theorem shows that it is actually inappropriate to invoke it for the analysis presented in the text. The reason is that the theorem demands an ability to manipulate what are assumed to be the underlying "primitive operations" of the reference computer model. Intuitively, what this amounts to is that instead of being able to examine and execute *one* rule per time step or *one* input token, by recoding k symbols into new "complex symbols" one is now permitted to examine and execute C rules or read C tokens at a time, where C is a constant that depends on the constant k (incorporating grammar size). Thus, one must change the basic "unit operations" of the finite state control

To say anything more specific, we must talk about a specific algorithm. Consider the "standard" cubic-time context-free recognition algorithm, Earley's algorithm (1968; 1970).[23] Earley shows that the time his method takes on an input sentence of length n is $k\,|G|^2 n^3$—proportional, that is, to the square of the size of the grammar and the cube of the length of the input sentence.[24] Roughly speaking, the bigger the grammar, the more time the algorithm spends running through its list of potential recognition rules, deciding which one is applicable next. Which factor, grammar size or sentence length, dominates the time complexity for processing sentences? The outcome of the analysis clearly depends upon the relative size of the grammar compared to the range of sentence lengths. If the grammar is of a size adequate to describe natural language, then we might expect there to be many hundreds of rules; however, the sentences input to the recognizer will almost invariably be at the most 20 words long. (For example, if $|G| = 500$ and $n = 10$, then the recognition time complexity according to the Earley algorithm is 2.5×10^8). Neglecting the constant k, in a logarithmic scale the grammar size $|G|$ contributes about two-thirds to the total complexity product, with input sentence length supplying the remainder.[25] In short, grammar size can dominate processing complexity for a "relevant" grammar size and a relevant range of input sentence lengths.

---

of the Turing machine. This alteration of underlying primitive operations is permissible in the case of Turing machines since we can program them at our whim. It is much less clear what one is allowed to do in the cognitive analogue. As pointed out above, it is probably the case that the underlying computational machinery is more or less fixed (though we do not know what the "primitive operations" are). Consequently, it would not seem valid to allow recoding of the sort required by the linear speedup theorem. The conservative approach is to assume that the constants matter.

As a final postscript on this issue, it is worthwhile to point out that the trade-off between constant factors and asymptotic complexity arises even in the realm of computer science: it is widely known that the asymptotically "fastest" context-free recognition algorithm involves such large constant factors that it is impractical for actual use (this is Valiant's method; see Harrison (1978)).

[22] This fact has been noted by many researchers in computer science; for an excellent analysis of the issues, see Pratt (1975), as well as Graham, Harrison, and Ruzzo (1980).

[23] Earley's algorithm is based on the "tabular" parsing methods of Cocke and Schwartz (1970), Kasami (1965), and Younger (1967). Numerous variants of Earley's method have been proposed. These include chart parsing (Kay (1967)) and well-formed substring table methods such as the one in Kaplan (1973).

[24] It is interesting to compare this complexity result with that of the older Cocke–Kasami–Younger (CKY) algorithm (Younger (1967)) (see also Hopcroft and Ullman (1979)). The CKY algorithm runs in time that is proportional to the size of the grammar (*not* its square) and the cube of the length of the input sentences. Why then isn't the CKY method superior to the Earley algorithm? The reason is that the CKY method works only on grammars in Chomsky normal form, i.e. grammars that produce only binary branching trees except for preterminals, with rules of the form $A \rightarrow BC$ or $A \rightarrow a$. It is easy to show (as observed in Ruzzo (1978)) that the Earley algorithm is essentially a method for converting an arbitrary context-free grammar into Chomsky normal form. The conversion involves "splitting" the righthand sides of rules of the form $A \rightarrow BC \ldots D$ into binary branching form by introducing new nonterminal names and expansion rules that incorporate "complex symbols" formed by all the possible ways of dividing a righthand side up into two sets of nonterminals; for example, $A \rightarrow BCD$ becomes $A \rightarrow .(BCD)$, $A \rightarrow (B).(CD)$, $A \rightarrow (BC).(D)$, $A \rightarrow (BCD)$. In the worst case this expansion squares the size of the grammar. On the other hand, the best-known algorithm for converting an arbitrary grammar into Chomsky normal form in the worst case also squares the size of the grammar. Thus, given an arbitrary context-free grammar and considering only the factors of grammar size and input length, the CKY method and the Earley method are on an equal footing.

[25] Input sentence length will not dominate the complexity equation until n = approximately 63 ($63^3 = 250047$).

We might also consider whether directly reducing the size of the grammar has a telling impact on the overall efficiency of an algorithm. The potential advantages of a succinct representation is a familiar theme for the linguist (more compact grammars are generally assumed to capture generalizations better than their bloated relatives, and are often taken to be more easily learnable as well). From the preceding paragraph we see that the linguist's intuition also has some computational support: if grammar size can be reduced "easily enough", then this kind of reduction may be more advantageous than a reduction on the exponent of input sentence length from cubic to quadratic. As an illustrative example, consider a case where the size of the grammar $|G| = 500$, and the maximum length sentence analyzed is 10 tokens long. Now suppose that one is faced with a new alternative grammar, grammar B, that is a bit more than triple the size of the old grammar ($|G_A| = 1600$), but runs in only quadratic (n^2) time in the length of input sentences. In this situation, the first grammar will take time $k \times 2.50 \times 10^8$ to process an input sentence of length 10, whereas grammar B, although processed by an *asymptotically* faster algorithm than grammar A, will take time $k \times 2.56 \times 10^8$. Given this particular set of assumptions about the range of input sentence lengths and algorithmic complexity functions, the more succinct grammar is more efficiently processed.[26]

Clearly, an exact trade-off between grammar size and exponent on input sentence length cannot be calculated without a specific set of grammars in hand. Moreover, it would be a pointless exercise to show that succinctness is a potential advantage if one cannot reduce grammar size at all. In this regard it is important to observe that it appears generally true that as one moves from weaker to more powerful rule systems, one can express languages more succinctly. This informal suspicion also has some formal mathematical backing. Meyer and Fischer (1971) have shown that as one moves up in expressive power from deterministic finite-state automata, to nondeterministic finite-state automata, to push-down automata (context-free languages), to context-sensitive languages and beyond, there are always languages lower down in the hierarchy that can be expressed more succinctly via the more powerful rule systems higher in the hier-

[26] If the relevant range of input sentence lengths is 10 or less, then even a reduction from a cubic to a linear time function of n will not necessarily outweigh the gains of succinctness. For example, suppose that we can obtain a linear algorithm at the price of expanding the number of grammar rules to 6000 or so (a twelvefold increase). Then the complexity of the new grammar with the new, "faster" algorithm and a maximum input sentence length of 10 will be 3.6×10^8, whereas under the old "slower" algorithm it was only 2.5×10^8.

More generally, assuming the Earley algorithm as a basic functional form, if G_0 = the size of one grammar, and G_1 = the size of another grammar, then efficiency gains from succinctness outweigh gains from a reduction of the exponent on input sentence length from n^j to n^k if $G_1/G_0 >$ square root (n^j/n^k). In the example in the main text, with $j = 3$ and $k = 2$, this will occur when the ratio of grammar sizes exceeds the square root of n; for $n = 10$, this is approximately 3.16, and $1600/500 = 3.2 > 3.16$.

We should emphasize that these examples are meant to be purely illustrative in nature. However, order of magnitude blow-ups in grammar size of the sort described here are not atypical. As we shall see below, even exponential expansions in grammar size are theoretically possible if one opts for a "weaker" formalism instead of a stronger one.

archy.[27] For example, the gain in economy of push-down automata over finite automata for describing finite (and trivially regular) sets can be exponential; the gain in using deterministic push-down automata to describe infinite regular sets can be even better; and, perhaps surprisingly, the amount of "compaction" achieved by using a *nondeterministic* push-down machine to describe an infinite regular set can be arbitrarily large. (Succinctness gains can also be unbounded as one moves from context-free to context-sensitive languages, and from context-sensitive languages to arbitrary recursive languages.)

Thus, although a set of strings may be perfectly well *describable* (in the weak generative sense) by a system of low expressive power, it may actually be advantageous in terms of parsing efficiency to capture the structure of that set by a more powerful formalism. The reason is simply that if in one formalism parsing time is some linear function of the length of the input and the size of the grammar (i.e. is proportional to $k \times |G|n$), and if one can move to, say, a context-free formalism and reduce the size of the grammar exponentially, then the price of using the n^3 context-free parsing algorithm could be well worth it: a reduction in the size of the grammar could more than make up for the increase due to the exponent change from n to n^3.[28] Note that this advantage of succinctness is quite different from the usual linguistic claim that a more compact grammar is more easily learnable; we are claiming that it is *possible* that a more compact grammar, expressed by a more powerful formal system, is more efficiently processed as well.[29]

[27] On the other hand, it can also be shown that there exist languages for which there is no gain in succinctness by moving to a more powerful descriptive formalism. The question of interest for linguistics is whether the use of more "powerful" descriptive machinery, e.g. transformational grammar, permits more succinct descriptions of natural language. This was one of the arguments made for transformational grammar in *Syntactic Structures*.

In this regard, it is interesting to note that the Meyer–Fischer results apparently *can* be applied in certain linguistically relevant cases. For example, Meyer and Fischer show that a deterministic push-down automaton with n states and s push-down symbols can be exponentially smaller for representing certain finite (hence trivially regular) languages than any finite-state automaton that generates the same language. It can be easily demonstrated that this potential exponential compaction is actually achieved in the case of languages that have self-embedding constructions of *finite* depth: as is well known, a finite automaton *can* generate languages with a fixed, finite bound on allowable self-embeddings (Chomsky and Miller (1963)); however, this equivalent finite-state automaton will have an exponentially larger number of states than the minimal push-down automaton for the same language. Since this case actually arises in the study of grammars for English, one can conclude that the Meyer–Fischer succinctness results apply to linguistically relevant examples.

[28] Here we assume on-line recognition, so that at least n time steps are required merely to read an input of length n. On this assumption, recognition cannot take less than time n.

[29] More pertinent succinctness results have been obtained. Joshi, Levy, and Yueh (1980) show that there are context-free languages whose context-sensitive phrase structure grammars are more compact, by any factor k that one desires, than any equivalent context-free grammar for those languages. They further show that this sort of context-sensitive phrase structure grammar can be incorporated into the Earley parsing framework, though with a modified polynomial bound, thus demonstrating that substantial efficiency gains are in some cases possible by moving from a context-free phrase structure system to context-sensitive phrase structure rules. (This result is actually a particular subcase of the more general Meyer–Fischer theorems.)

It remains to be seen whether the formal devices that Gazdar introduces (slashed categories as complex symbols and metarules) actually lead to efficiency gains for parsing; this would depend upon just how much larger a slashed category grammar is than an equivalent TG, the computational bookkeeping required, and so forth. The one worked-out example of such a context-free grammar the authors are familiar with is the slashed

Analyzing the complexity of a parsing procedure as some joint function of grammar size and sentence length is but the first crude step in a more detailed comparison of parsing procedures. The unanalyzed grammar size term itself covers a multitude of implementation details. Besides reducing the sheer size of the grammar, one could also try to discover alternative representational formats for the grammar rules that are more easily coupled to the demands of parsing routines; perhaps one could reduce the exponent on the grammar size contribution from quadratic to linear. The success of this effort in turn depends upon both the exact form of the grammar and available representational formats—what primitive operations and structures the brain actually has available (or can "quickly" simulate). This is an important correlative (and almost inevitable) effect of moving away from abstract analysis and toward more fine-grained efficiency analysis: our comparisons become more attuned to the relevant cognitive details, but we lose the ability to say that our comparisons will remain fixed over all possible variations in primitive machine operations. If the human cognitive machinery does not have the requisite unitary operating characteristics that we have assumed for our low-level, detailed efficiency analysis, then the comparison is simply beside the point. For instance, if we have concluded that algorithm A is faster than algorithm B on the assumption that the primitive instruction set of the underlying computational machinery includes a unit operation to multiply two numbers together, but the actual machinery provided includes only addition operations, then it may well be that the superiority of algorithm A is merely academic. Similarly, detailed complexity analyses of variations in one or another parsing algorithm that are based upon differences in the microstructure of *computer* organization seem problematic without independent verification that the relevant differences in computer organization are reflected in the cognitive domain. A careful complexity analysis must ride a thin line between over-abstraction and over-specialization.

The fact that one can arrange for the "internalization" of grammar rules in a wide variety of formats—as matrices, linked lists, or more complicated arrangements—would be of little interest for efficiency analysis if it were true that modifications of this kind had little impact on overall parsing efficiency. As it turns out, however, variation in "data structures" can have a significant practical effect on the efficiency of an algorithm. Quite often, merely changing the way in which rules are accessed can make order-of-magnitude differences in algorithmic efficiency. To take a concrete example, Ruzzo and his colleagues (Ruzzo (1978), Graham, Harrison, and Ruzzo (1980)) have shown that the Earley algorithm can be made more efficient by a combination of actual changes to the algorithm plus a palette of "implementation" techniques. These include shifts to alternative representational formats for storing grammar rules (and how those rules are "looked up" as the parse progresses); alternative primitive machine operations (whether

---

category grammar of Harman (1963). Harman claimed that this grammar generated the same language as the TG of *Syntactic Structures*, using about the same number of rules. An up-to-date comparison might yield different results. Recall that the TG of *Syntactic Structures* had separate rules for each superficially "different" transformation; in the current Government–Binding theory there is only one movement rule. In any case, it does not immediately follow that conversion to the slashed category notation is computationally advantageous.

parallel operations are available for certain tests); and whether "preanalysis" of the grammar is permitted (so as to compute in advance commonly used derivation steps). As Graham, Harrison, and Ruzzo point out, these changes can speed up the same algorithm ten times or more, and may well dominate the algorithm's execution time for practical-sized inputs. Thus, "implementation details", far from being safely ignored, may actually be crucial in the *practical* evaluation of an algorithm's complexity. Importantly, Graham, Harrison, and Ruzzo point out that some of these format "tricks" are not available if one uses the original Earley algorithm.[30]

Since different representational formats can make for quite significant differences in parsing efficiency in the case of context-free parsing, it seems reasonable to conclude that the proper *practical* evaluation of an algorithm is a sophisticated task. It requires careful attention to alternative data structures and the underlying organization of the computer that has been assumed. In the cognitive domain the task is even more difficult, since the attendant computational assumptions are more likely to be lacking independent support. For example, if the primitive parallel operations demanded by the most efficient of the Graham, Harrison, and Ruzzo techniques have no analogue in cognitive machinery, then we cannot exploit the efficiency gains of this method.[31] In short, we again discover that we must have a *theory* of implementation and some specific knowledge of the computational capabilities of the brain.

A stronger case for a particular algorithm's superiority could be made if we were able to show that its efficiency was preserved for input sizes of practical interest over many (in the best case all) conceivable implementations. Then we might be more confident that, no matter what particular "implementation" the brain had picked, our algorithm would still be superior (though it of course still does not necessarily follow that the brain would pick that particular algorithm). It is this property of invariance over implementation that lends at least some credibility to the distinction between procedures that run in some *polynomial* function of the length of their inputs (such as n^3, as in the

[30] Some storage representations admit efficient "preprocessing" operations on the (presumably now fixed) grammar that are simply not available in other formats. For instance, given a (fixed) grammar, we might compute in advance of the parse of any input sentence a table that tells us which nonterminals can be derived from other nonterminals (for example, in the grammar $S \Rightarrow AB$; $A \Rightarrow C$, we would store the "lemma" that S can derive A, B, or C; and that A can derive C). These finite-step "lemmas" can possibly shorten parsing work later on, if they can be integrated in an efficient manner into the parsing algorithm as a whole. It turns out (although we cannot demonstrate this fact here) that trade-offs between preprocessing operations, alternative grammar representations, and different parsing algorithm organizations can be quite subtle.

To get some idea of the subtlety of "real" implementations, see the variants proposed by Graham, Harrison, and Ruzzo (1980, 437, 440, 441, 442 (figures 4, 6, 7, and 8 of the original article)). Each variant provides a slightly different way of stating which rules are related to which other rules (given a particular input sentence), and of organizing the actual parse of a sentence. Without grasping in detail just what these different formats are supposed to do, one can still note that they entail quite different ways of storing and manipulating rules of the grammar, and hence make quite different assumptions about just what "primitive operations" are available in the cognitive machinery.

[31] Since we are looking only at parsing efficiency, let us say that there is a "cognitive analogue" of the parallel operations if the cognitive machinery can simulate the requisite operations while preserving the "unit time" execution character of the parallel operations. That is to say, the simulation must be "fast enough". For further discussion of the impact of parallel computational machinery on the usual sort of serial machine complexity analyses relevant to linguistics, see Berwick and Weinberg (forthcoming a).

case of context-free recognition) versus those that take some *exponential* amount of time. It is easily shown that all "natural" models of serial computation—Turing machines, random access machines, and others—can simulate each other in polynomial time.[32] Therefore, we know that procedures that take polynomial time under some particular model of computation (say, a Turing machine) will still take polynomial time under any other natural model (though the exponent may be larger or smaller). For instance, Earley's algorithm takes n^3 time on a random access machine, but n^4 time on a Turing machine (under the straightforward simulation of one machine by the other). The important point is that the *asymptotic* superiority of polynomial over exponential algorithms will be maintained across all "natural" machine implementations.

Unfortunately, as we have seen, this invariance is again an asymptotic property, and hence its relevance for cognitive science is not clear. It seems just as likely that it is the constant multipliers that dominate the usual execution time of parsing algorithms for natural language. Consequently, we suggest that the proper analysis of parsing algorithms will have to wait upon the as yet undeveloped theory of implementation or perhaps even some hard (but still abstract) information about the computational abilities of the language faculty/brain (cf. the case of Marr's research on visual processing cited in Marr and Poggio (1979)).

3. Conclusions

In this light, let us summarize the examination of the efficient parsability assumptions. The view that context-free languages form a privileged class because of the claim of parsing efficiency requires both a *functional* and a *mathematical* argument. The mathematical argument that has appeared in the literature rests on the direct use of computational complexity results without, apparently, either a proper consideration of the domain of application of the relevant theorems or a proper evaluation of the efficiency condition in a realistic biological setting. In short, there is a distinction to be drawn between *relevant cognitive complexity* and the *mathematical complexity* of a language. As noted above, it is relevant cognitive complexity that is actually of interest to linguists and psychologists. However, this measure must apparently be couched at a level of detail that incorporates many implementation factors that the usual theory of mathematical complexity abstracts away from. In particular, to determine relevant cognitive complexity one must determine the range of inputs that will be encountered, the size of the grammar, its internal representation, and the basic architecture of the machine actually instantiating a parsing algorithm that makes use of the grammar. As we have tried to show in this article and in Berwick and Weinberg (forthcoming a), there seems to be so much possible variation in these implementation details alone—at least given our current inability to posit constraints on the architectural features of the human cognitive machinery—that the same grammar can be incorporated into algorithms with

[32] That is, one brand of machine—say, a Turing machine—can simulate what another brand of machine does, while using, say, n^3 "extra" time to do the simulating. (See Machtey and Young (1978).)

vastly different time complexities. Furthermore, a language that is quite "high up" in the Chomsky hierarchy—e.g. a strictly context-sensitive language—may in fact be parsed more rapidly than languages lower down in the hierarchy—e.g. faster than some context-free languages—if the gain in succinctness is enough to offset the possible increase in parsing time.

There is, of course, nothing in principle that prevents the theory of language use from serving as a domain of evidence constraining the class of possible grammars. The preceding discussion suggests, however, that parsing efficiency criteria as typically defined in a mathematical sense, and in particular the Chomsky hierarchy, are not much of a criterion at all: the entire class of context-free languages, plus many other languages that are strictly context-sensitive, are efficiently parsable. Indeed, it seems likely that when the factors influencing the implementation of algorithms are taken into consideration (including the possibility of parallel hardware), and when one realizes that it is only of interest to consider sentences of "practical" size, then it actually seems likely that *almost any language is efficiently parsable*. Consequently, attention to the criterion of efficient parsability alone, at least in the mathematical sense as it is usually developed, can do little to advance us toward our goal of constraining the class of possible grammars.[33] This apparent fact was noted in Chomsky (1965, 62), where the difference between mathematical and biological relevance is also pointed out:

> Thus one can construct hierarchies of weak and strong generative capacity, but it is important to bear in mind that these hierarchies do *not* necessarily correspond to what is probably the empirically most significant dimension of increasing power of linguistic theory. This dimension is presumably to be defined in terms of the scattering in value of grammars compatible with fixed data. Along this empirically significant dimension, we should like to accept the least "powerful" theory that is empirically adequate. It might conceivably turn out that this theory is extremely powerful (perhaps even universal, that is, equivalent in generative capacity to the theory of Turing Machines) along the dimension of weak generative capacity, and even along the dimension of strong generative capacity. It will not necessarily follow that it is very powerful (and hence to be discounted) in the dimension which is ultimately of real empirical significance. . . . It is important to realize that the questions presently being studied are primarily determined by feasibility of mathematical study, and it is important not to confuse this with the question of empirical significance.

It seems not unreasonable to suppose that the domain of "empirical significance" might require finding criteria above and beyond that of efficient parsability that would serve to restrict the class of possible grammars. One possible criterion derives from research into the class of grammars that are *learnable*, a class that is extremely small (see Chomsky

[33] We leave open the question of whether resource complexity more generally is an appropriate measure for the "finite-sized" problems that the brain computes. It may well be that the alternative measures of *program size* or *Kolmogorov* complexity are more suitable (see Gewirtz (1974)). The theory of program size complexity attempts to find the shortest program it takes to compute some function, irrespective of the amount of time or space it may take to do the actual computation. There are obvious connections between the notions of *shortest program* and *simplest theory*, as classically described by Goodman and others and used by Chomsky, that deserve further study.

(1981, chapter 1) for discussion). We might restrict the class of possible parsers to those that incorporate more or less directly grammars drawn from the class of learnable grammars. Our task would then be to show that this class of parsers met the demand of efficient parsability.[34] As this would take us far beyond the scope of this article, we will leave it as a topic for future research.

References

Berwick, R. (forthcoming) *Locality Principles and the Acquisition of Syntactic Knowledge,* Doctoral dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, Massachusetts.

Berwick, R. and A. Weinberg (forthcoming a) *Language Use and Language Acquisition: The Implications of Grammatical Theory,* MIT Press, Cambridge, Massachusetts.

Berwick, R. and A. Weinberg (forthcoming b) *Generative Capacity and Government–Binding Theory.*

Book, R. (1973) "Topics in Formal Language Theory," in A. Aho, ed., *Currents in the Theory of Computing,* Prentice-Hall, Englewood Cliffs, New Jersey.

Chomsky, N. (1951) *Morphophonemics of Modern Hebrew,* M.S. dissertation, University of Pennsylvania.

Chomsky, N. (1965) *Aspects of the Theory of Syntax,* MIT Press, Cambridge, Massachusetts.

Chomsky, N. (1980) *Rules and Representations,* Columbia University Press, New York.

Chomsky, N. (1981) *Lectures on Government and Binding,* Foris, Dordrecht.

Chomsky, N. and G. Miller (1963) "Finitary Models of Language Users," in R. D. Luce, R. R. Bush, and E. Galanter, eds., *Handbook of Mathematical Psychology,* vol. 2, Wiley, New York.

Cocke, J. and J. Schwartz (1970) *Programming Languages and Their Compilers,* Courant Institute, New York University, New York.

Earley, J. (1968) *An Efficient Context-free Parsing Algorithm,* Doctoral dissertation, Carnegie-Mellon University, Department of Computer Science, Pittsburgh, Pennsylvania.

Earley, J. (1970) "An Efficient Context-free Parsing Algorithm," *Communications of the Association for Computing Machinery* 14, 453–460.

Ford, M., J. Bresnan, and R. Kaplan (1981) "Toward a Theory of Lexico-Syntactic Interpretations in Sentence Processing," MIT Cognitive Science Center Occasional Paper, Cambridge, Massachusetts.

Galil, Z. (1978) "Palindrome Recognition in Real-time," *Journal of Computer and System Science* 16, 140–157.

Gazdar, G. (1979) *English as a Context-free Language,* unpublished report, University of Sussex.

Gazdar, G. (1981) "Unbounded Dependencies and Coordinate Structure," *Linguistic Inquiry* 12, 155–184.

[34] Much recent work has gone into showing that the human parsing system in fact uses principles borrowed rather directly from linguistic competence theories. In fact, much of the work previously thought to be accomplished by independent parsing strategies seems more adequately handled by grammatically based devices. (See, for example, Marcus (1980); Ford, Bresnan, and Kaplan (1981); Berwick (forthcoming).)

It should be clear, however, that the degree of correspondence allowed between grammar and parser can affect the amount of constraint that this approach imposes on the class of possible parsers. We should note that the grammar–parser relation is not an all-or-none choice between a one-to-one grammar–parser correspondence and no grammar–parser correspondence. For a discussion of several intermediate grammar–parser correspondence proposals and their impact on the interpretation of psycholinguistic results, see Berwick and Weinberg (forthcoming a).

Gazdar, G. (forthcoming) "Unbounded Dependencies," in P. Jacobson and G. Pullum, eds., *The Nature of Syntactic Representation,* Reidel, Dordrecht.

Gewirtz, W. (1974) *Investigations in the Theory of Descriptive Complexity,* Courant Computer Science Report #5, New York.

Graham, S., M. Harrison, and W. Ruzzo (1980) "An Improved Context-free Recognizer," *ACM Transactions on Programming Languages and Systems* 2, 415–462.

Harman, G. (1963) "Generative Grammar without Transformation Rules: A Defense of Phrase Structure," *Language* 39, 597–616.

Harrison, M. (1978) *Introduction to Formal Language Theory,* Addison-Wesley, Reading, Massachusetts.

Hennie, F. (1965) "One-tape, Off-line Turing Machine Computations," *Information and Control* 8, 553–578.

Hintikka, J. (1974) "Quantifiers vs. Quantification Theory," *Linguistic Inquiry* 5, 153–177.

Hopcroft, J. and J. Ullman (1969) *Formal Languages and Their Relation to Automata,* Addison-Wesley, Reading, Massachusetts.

Hopcroft, J. and J. Ullman (1979) *Introduction to Automata Theory, Languages, and Computation,* Addison-Wesley, Reading, Massachusetts.

Hornstein, N. and D. Lightfoot (1981) "Introduction," in N. Hornstein and D. Lightfoot, eds., *Explanation in Linguistics,* Longmans, London.

Joshi, A. and L. Levy (1977) "Constraints on Local Transformations," *SIAM Journal of Computing* 6, 272–284.

Joshi, A., L. Levy, and K. Yueh (1980) "Local Constraints in Programming Languages, Part I: Syntax," *Journal of Theoretical Computer Science* 12, 265–290.

Kaplan, R. (1973) "A General Syntactic Processor," in R. Rustin, ed., *Natural Language Processing,* Courant Science Symposium, Algorithmic Press, New York.

Kasami, T. (1965) *An Efficient Recognition and Syntax Algorithm for Context-free Languages,* Air Force Cambridge Research Laboratory report #AF–CRL–65–758, Bedford, Massachusetts.

Kay, M. (1967) *Experiments with a Powerful Parser,* Rand Corporation RM–5452–PR, Santa Monica, California.

Knuth, D. (1965) "On the Translation of Languages from Left to Right," *Information and Control* 8, 607–639.

Kripke, S. (1976) "Is There a Problem with Substitutional Quantification?" in J. McDowell and G. Evans, eds., *Truth and Meaning,* Oxford University Press, Cambridge.

Kuno, S. (1966) "The Augmented Predictive Analyzer for Context-free Languages—Its Relative Efficiency," *Communications of the Association for Computing Machinery* 9, 810–823.

Langendoen, D. T. (1975) "Finite-State Parsing of Phrase-Structure Languages and the Status of Readjustment Rules in Grammar," *Linguistic Inquiry* 6, 533–554.

Lapointe, S. (1977) "Recursiveness and Deletion," *Linguistic Analysis* 3, 227–265.

Lewis, H. and C. Papadimitriou (1980) *Elements of the Theory of Computation,* Prentice-Hall, Englewood Cliffs, New Jersey.

Machtey, M. and P. Young (1978) *An Introduction to the General Theory of Algorithms,* North-Holland, New York.

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

Marr, D. and T. Poggio (1979) "A Theory of Human Stereo Vision," *Proceedings of the Royal Society of London B* 204, 301–328.

Matthews, R. (1979) "Are the Grammatical Sentences of a Language a Recursive Set?" *Synthese* 40, 209–223.

Meyer, A. and M. Fischer (1971) "Economy of Description by Automata, Grammars, and Formal Systems," *Proceedings of the 12th Annual ACM Symposium on Switching and Automata Theory,* 185–194.

Moore, P. (1981) "The Varied Ways Plants Tap the Sun," *New Scientist,* February 12.

Nijholt, A. (1980) *Context-free Grammars: Covers, Normal Forms, and Parsing,* Springer-Verlag, New York.

Peters, S. and R. Ritchie (1973a) "Context-sensitive Immediate Constituent Analysis—Context-free Languages Revisited," *Mathematical Systems Theory* 6, 324–333.

Peters, S. and R. Ritchie (1973b) "On the Generative Power of Transformational Grammars," *Information Sciences* 6, 49–83.

Postal, P. (1964) *Constituent Structure,* Mouton, The Hague. Distributed by the Indiana University Linguistics Club, Bloomington, Indiana.

Pratt, V. (1975) "LINGOL—a Progress Report," *Proceedings of the 4th International Joint Conference on Artificial Intelligence,* 422–428.

Rosenberg, A. (1967) "Real-time Definable Languages," *Journal of the Association for Computing Machinery* 14, 645–662.

Rounds, W. (1975) "A Grammatical Characterization of the Exponential Time Languages," *Proceedings of the 16th Annual Symposium on the Foundations of Computer Science,* 135–143.

Ruzzo, W. (1978) *General Context-free Parsing,* Doctoral dissertation, University of California at Berkeley, Department of Computer Science.

Walters, D. (1971) "Deterministic Context-sensitive Languages," *Proceedings of the 12th Annual ACM Symposium on Switching and Automata Theory,* 133–148.

Williams, E. (1981) "Transformationless Grammar," *Linguistic Inquiry* 12, 645–653.

Younger, D. H. (1967) "Recognition and Parsing of Context-free Languages in Time n^3," *Information and Control* 10, 189–208.

(Berwick)
Artificial Intelligence Laboratory
NE43–814
MIT
Cambridge, Massachusetts 02139

(Weinberg)
128 West 73 Street, #A
New York City, New York 10023