The Linguistic Connection

Martin Kay

Xerox Palo Alto Research Center

and

CSLI, Stanford

Set theory is the sole foundation of the whole edifice of mathematics, or so I have been given to understand. Sequences are constructed from ordered pairs pairs in a fairly obvious way, and ordered pairs result from an artifice that can only cause the lay observer to put his hand on his wallet. In computing, on the other hand, sequences have always been treated as primitive. Sets are represented by an arbitrary permutation of their members. They are sets, and not sequences, only in as much as the algorithms that operate on them are expected to produce equivalent results regardless of the permutation chosen. Now, I take it that an important effect of connectionism will be to bring computing more into line with mathematics by giving first-class status to sets.

This is doubtless good news for the mathematician, for the theoretical computer scientist, and possibly for many others. But, in the linguist, it seems, on the face of it, to be cause for modified rapture, at best. Language is probably the most striking example of a two dimensional object in this three-dimensional world. Its most obvious property is its sequentiality. But, this has been said before, for example by those who offered computational linguists string processing languages like COMIT and SNOBOL as the tools most obviously fitting their needs, and sophisticated people are no longer beguiled by the argument. According to the more enlightened view, sequentiality is a very superficial property of language. By applying the rules of syntax, we are able to uncover the richer, more revealing, multidimensional structures that lie behind it, and which are closer to the essence of language. In fact, there has been much interest in recent times in languages that are alleged to have a much more set-like character than English has, in that many of the permutations of a sentence often constitute a logical, or semantic, equivalence class.

Linguists have organized their subject in various ways, a familiar one being by level of abstraction. Phonetics is about the sounds people make when they talk and what they do to make them. Some would have it that there is so little abstraction in phonetics that it should not properly count as part of linguistics at all. Phonology talks about how the raw material of phonetics is organized into basic symbols of the most basic kind, about allowable sequences of these symbols, and about the way in which the phonetic forms of particular basic symbols are conditioned by the

environment. This is all unrepentantly sequential, except when the discussion turns to such things as intonation and stress.

Morphology and lexicology are more abstract in the simple sense that they take the organization that phonology imposes on the primary material as primitive, and impose a further level of organization on that. Morphology is about how lexical items, themselves represented by sequences of phonologically defined units, are arranged to make words. It is mostly a matter of sequence, but morphology is sometimes conspicuously "nonconcatenative", to use the word that McCarthy (1979, 1981) coined in connection with semitic languages. However, though morphology is sometimes not simply a matter of just which sequences of morphemes do make up words, and with what properties, it is inescapably a matter of how the phonetic or phonological material supplied by morphemes is arranged into a sequence so as to form a word.

The next level of abstraction is syntax, the way in which words are collected to form sentences. Just about all of the multifarious formal theories of grammar that have been proposed have been particularly strong in the facilities they provided for describing the proper ordering of words in a sentence, though it is widely recognized that there may be some ethnocentrism in this, for formal linguists have been overwhelmingly speakers of languages where word order plays a predominant role. But it was not so in traditional informal grammar, which took Latin as as a model for all languages. Many formalists are now in search of less strongly sequential paradigms as they attempt to account for so called *free word order* and *nonconfigurational* languages.

By the time we reach the next level of abstraction, that of semantics, essentially no reflection of the ordering of the initial phonetic material remains. But, by this time, it is also possible to claim that the territory that falls most clearly within the purview of linguists has already been traversed. Linguistics makes contact with the real world at two points: the sounds that people utter and the meanings that are associated with them—phonetics and semantics. At all of the intervening levels of abstraction, the reflexes of the temporal ordering of the sounds is usually strongly in evidence.

If the picture I have painted of language is substantially correct, and if I have not misunderstood the nature of the connectionist revolution in computing too grossly, it seems that we may have to conclude that the human linguistic faculty, if not human intelligence at large, have more in common with the von Neumann machine than with the connection machine and that my colleagues, and I will regretfully not be part of this new adventure. But now, let us see if we cannot find another side to the coin.

For all that language gives up its sequentiality grudgingly and emerges into the brighter set-theoretic world only as its identity is confounded with that of intelligence at large, it nevertheless remains remarkably *context-free*. I say "remarkably" because we know from mathematics that context free languages are a subset—a small subset in some sense—of the theoretically possible languages. What this means for language users and computational linguists is that one can analyze any part of the sequence of phonemes, morphemes, words or whatever, with the expectation that, if the analysis of the whole string incorporates an analysis of that part, then the analysis one has already made will fill the requirement. Given that the subject and the object of the sentence do not overlap, the analysis of each of them can proceed in parallel. This is the property of language that makes chart parsers an attractive option.

Chart parsers in general, and so-called *active* chart parsers in particular, are fundamentally exercises in parallel computing. If, along with the chart, there is usually a second data structure called the *agenda*, it is simply to facilitate the simulation of this parallel architecture on sequential machines. But what is going on in chart parsing is much better understood if one thinks of each vertex in the chart as an autonomous device responsible for delivering all phrases that begin with the word at that vertex. The process of finding these phrases is dependent on similar work going on at other vertices only to the extent that, when phrases are delivered at other vertices, it may become possible to recognize others that begin here. But the relationships are *intrinsic*, to use the term in the linguist's special sense. In other words, these relationships are not dictated by some general algorithm or external set of principles, but by the obvious requirement that computations that require a particular piece of information cannot be completed until that information is available.

Some twenty years ago, when local networks were a relatively new thing, I harnessed the nocturnal energies of several machines at the Xerox Palo Alto Research Center to just such a task, more for fun than enlightenment. Of course, it worked. Furthermore, if the speed had been multiplied by a substantial factor, it would have been quite fast. The idea behind what I did was simple and obvious. An active edge consisted of a message from one machine to another asking for any phrases with a certain description that appeared at that other vertex. An inactive edge was a phrase already found that enabled a given machine to answer such requests. Each machine kept old requests against the possibility of finding phrases later with which to amplify its answer to previous requests. Each machine also had a complete copy of the grammar so that there could be no contention over access to it.

So, if the sentence to be analyzed was "Brutus killed Caesar", three machines would have been assigned and the talk on the net might have been somewhat like this:

1. a. From *Brutus* to *killed*: need a singular, 3rd. person VP.
   b. From *killed* to *Caesar*: need a NP.
2. From *Caesar* to *killed*: herewith one NP, namely "Caesar", ending where the sentence ends.
3. From *killed* to *Brutus*: herewith one VP, namely "V(killed) NP(Caesar)", ending where the sentence ends.

The *Brutus* machine is now in a position to deliver an analysis of the whole string. The ordering of the work into these three stages is intrinsic. In particular the *killed* machine cannot honor the request for a VP until information about the NP to its right is in. However, *killed* does not wait to be asked for a VP to send out its request for a NP. Each machine goes to work building whatever it can in a bottom up manner, just in case it may prove useful. So, if there had been a fourth machine to the right of 'Caesar', then 'Caesar' would have asked it for VP's in the hope of building sentences with them, even though no request for sentences was destined to reach it from the left.

This approach to syntactic analysis falls down because of a property of languages that I have not mentioned so far, namely that they all assiduously avoid center embedding in favor of strongly left- or right-branching structures. It is easy to see that, if syntactic structures were more or less well balanced trees, the time that my parallel device would require to find a single analysis of a sentence of n words would be of order $\log(n)$. But, if the most richly developed part of each subtree is almost always on its righthand side, as in English, then the intrinsic ordering of the processes will be such as to make this scheme essentially similar to standard sequential ones. If the language is predominently right recursive, then it will rarely be possible for a machine to finish its work before all, or almost all, the machines to its right. The situation is no better for left-recursive languages.

One further remark may be in order before I leave the question of chart parsing. Chart parsers exploit the grossly context free nature of natural languages in a dynamic programming scheme that avoids, to a large extent, doing the same computation more that once. The chart is a record of the computations that have been attempted, and the results they produced, together with an index that makes it easy to check the record before repeating the work. It does a great deal to speed the business of finding all the structures that a grammar allows for a given sentence. But it is just as bad as a psychological model as it is good as a computational

technique. If we had charts in our head, we would never be led down the garden path, and we should have no difficulty in reanalyzing the early part of a long sentence, possibly several times, to reconcile it with what occurred much later. But we do not seem to be able to do this. The evidence, such as it is, is all to the effect that linguistic problems are solved on the basis of very local information and that it procedes much faster than even the chart model suggests. The connection machine may be able to provide a model that accounts for some greater speed, but locality and sequentiality remain.

They may be reason to suspect that the most obviously linguistic aspects of language processing—those that concern phonology, morphology, and syntax—are even more sequential even than the best known linguistic theories make them seem. It has often been pointed out that intonation and speech rhythm betray an organization of utterances into phrases of a different kind than emerges from considerations of syntax and semantics. It turns out that it is more natural to pause at some points in an utterance than at others, but these places are not always at syntactic boundaries. So we may have to countenance two different phrasings. Indeed, we may have to go further, because it has also been claimed that there is an *informational*, or *functional*, organization to discourse which does not respect the boundaries of either of the other two that I have mentioned. In Prague, this is known as the *functional setence perspective* and it has to do with the differential treatment that a speaker gives to information he supposes his interlocutor to know already, as against the information that he is explicitly offering as new. These things are poorly understood, but the claim of those who do battle with them is that they are based on essentially sequential, local patterns in the text.

So far, my attempt to find another side to the coin has failed. Furthermore, those who know me well may be beginning to suspect that I am talking against myself because I have for a long time been singing the song of monotonicity in linguistics, calling for the banishment of all that is essentially procedural. Many current linguistic theories attract attention largely for having abandoned *derivations* in favor of systems of *constraints*. Examples are Lexical Functional Grammar, Generalized Phrase Structure Grammar and its derivatives, and my own Functional Unification Grammar. Government Binding theory seems to me to be moving fast in the same direction and I suspect that it would be profitable to formulate a notational variant of it in which such procedural notions as "move-α" give way to a static system of constraints.

There are at least two advantages that constraint based systems like these have over derivation based systems, such as transformational grammar, at least in its older forms. The first is that it achieves a cleaner and more thoroughgoing separation of competence from performance, and the other is that it gives first-class

status to methods of computing with only partial information. The second of these has also been touted as onr of the strengths of the connectionist apporach. The situation with the first is less clear. Consider the case of what I will refer to generically as *unification grammar*.

The view of scientific philosophy that prevails among linguists focuses a lot of attention on a Utopian situation in which they are called upon to chose between two descriptively perfect grammars. They prepare themselves for this challenge by setting up metrics that will be ready for immediate application when this need arrises. I take it that, *ceteris paribus*, a competence grammar will be preferred if it more readily if it more readily supports some plausible theory of competence. In the long run, it will be even more to be preferred if it supports the *right* theory of competence. Now, a competence grammar that is based on a calculus in which operations have to be carried out in a specific, very carefully orchestrated way, is less likely to have this property than one in which no reliance is placed on carefully ordered sequences of operations. One might counter that the carefully ordered sequence could be just the one that people in fact follow so that the competence grammar could could go into immediate service as a performance grammar without substantial change. But this is clearly a forlorn hope if only because the sequence of opertations that a speaker and a hearer must perform are unlikely to be ordered in the same way. The constraint based systems of grammar that have been proposed, on the other hand, are hospitable to a variety of different processing strategies. The freedom that this gives to performance theorists also extends to computational linguists with more practical aims in mind; they are much freer to bring their ingenuity and expertise as computer scientists to bear.

The constraint based grammars that are now in vogue are based on unification and, to a lesser extent, on related operations, such as generaliztion and subsumption. These are logical operations, in a strong sense of the word, as evidenced by the fact that unification is also a basic operation of logic programming in general, and Prolog in particular. Logic programming, and computation with constraint-based grammars rests heavily on implementing the notion of a *logical variable*, as opposed what programmers have usually called "variables", and which are really names of storage locations. The values of logical variables, unlike the contents of storage locations, do not change over time, at least on one path through a nondeterministic process. Unification is an operation as a result of which it sometimes comes to light that sets of two or more variables refer to the same thing. Henceforward, any constraints imposed on the value of one of the variables, as a result of its appearance in one expression, must be consonant with those imposed upon other membrs of the set thorugh their appearance in other expressions. If these conditions are violated at the outset, the unification operation does not go through. I do not know what would

be involved in modeling this situation in something like the connection machine, but such uninformed speculations as I have allowed myself on the subject, together with occasional remarks from others who know better than I, suggest that this is not entirely in the connectionist spirit.

The skepticism I have expressed here on the matter of connectionism in linguistics is based to some extent on facts and to some extent on speculation on matters where I believe the evidence to be inconclusive. If I were wrong about most of the matters in the second class, it may be that the role of connectionism in linguistics could be very great. It seems to me that our way is clear. Arguments along the lines of those I have outlined should not be used against the attempt to apply connectionist ideas to linguistics. Quite the contrary. Connectionism hsould be pursued vigoroulsy in the hope that, if nothing else, it will shed light on these areas of uncertainty, most of which have resisted attack for far too long a time.

Bibliography

McCarthy, J J. (1979). *Formal problems in Semitic Phonology and Morphology.* Doctoral Dissertation, MIT, Cambridge Massachussetts.

McCarthy, J J. (1981). "A Prosodic Theory of Nonconcatenative Morphology". *Linguistic Inquiry*, 12.3.