SYNTACTIC PROCESSING AND FUNCTIONAL SENTENCE PERSPECTIVE

Martin Kay
XEROX PARC

This paper contains some ideas that nave occurred to me in the course of some preliminary work on the notion of reversible grammar. In order to make it possible to generate and analyze sentences with the same grammar, represented in the same way, I was led to consider restrictions on the expressive power of the formalism that would be acceptible only if the structures of sentences contained more information than American linguisits have usually been prepared to admit. I hope to convey to you some of my surprise and delight in finding that certain linguists of the Prague school argue for the representation of this same information on altogether different grounds.

1. REVERSIBLE GRAMMARS

For me, a grammar is not so much a set of well-formedness conditions on strings of words as a relation between strings of words (sentences) and structures. Usually it is constructive in the sense that there is a device that interprets it to convert either structures into sentences (a generator) or sentences into structures (a parser). A grammar for which both a generator and a parser can be found is what I call reversible and, of course, what I am interested in is not so much particular reversible grammars as a formalism which guarantees reversibility for any grammar that adheres to it. Context-free grammars are clearly reversible in this sense and transformational grammars are clearly not. Augmented Transition Networks (ATNs) are reversible provided some quite reasonable restrictions are placed on the operations that can be performed on registers. This is not to say that it is a trivial matter to obtain a generator from an arbitrary ATN grammar.

It goes without saying that the composition of a generator and a parser interpreting the same reversible grammar on the same sentence or structure does not, in general, perform the identity transformation. Frequently, sentences are ambiguous and structures often correspond to sets of more than one grammatical paraphrase. Parsing followed by generation of each member of the resulting set of structures will therefore yield the grammatical paraphrases of the original under all grammatical interpretations.

The practical advantages of a reversible grammar are obvious. A computer program that engages in any kind of conversation must both parse and generate and it would be economical to base both processes on the same grammar. But, to me, the theoretical appeal is stronger. It is plausible that we have something in our heads that fills the function I am ascribing to grammar, though I am not insensitive to the claims of those who deny this. But it is altogether implausible that we have two such things, one for parsing and one for generation, essentially unrelated to one another.

A left-to-right generator is one that deposits words into the output string one after another, in left-to-right sequence. A left-to-right parser is one that examines the words of the input string one after another, from left to right. A left-to-right reversible grammar is one for which there is a left-to-right generator and a left-to-right parser. Once again, it is clear that context-free grammars, in the usual notation, meet the requirements. ATN grammars probably do not. They certainly do not if we exclude the possibility of entirely reworking the grammar and presenting it in an entirely new form to the generator. The kind of grammar I have in mind would require no such reworking. Intuitively, the notion is a simple one. It is very like an ATN in that it analyses sentences by moving through a network, examining the input string at each transition and, if the current symbol meets the conditions specified for the transition, assigning it to a register. The generator would also move through the network, making exactly the same transitions, but depositing the contents of registers into the string at each step.

2. THE PROCESSOR

Generators and parsers for the kind of reversible grammar I have in mind could be implemented in a great variety of ways. One of the simplest I know would be to use a version of the General Syntactic Processor (GSP). GSP contains:

(1) a grammar in the form of a transition network, that is, a directed graph in which the permissible transitions between states are represented by arcs, each labeled with a more or less complicated set of actions which determine the applicability of the arc and cause side effects, such as the assignment of values to registers,
(2) an agenda of tasks to be carried out,
(3) a chart, that is, a directed graph consisting of vertices and edges which represents the sentence being analyzed or generated together with its component parts--phrases, intermediate derivations, or whatever--which, together with the agenda, completely encapsulates the state of the entire processor at any given point in its operation,
(4) a set of scheduling rules whose job is to determine the order in which the tasks on the agenda will be carried out, and
(5) the interpreter itself.

Edges in the chart are either complete or incomplete. Complete edges represent

completely specified words or phrases and, if there is a path through the chart from one edge to another, it is because the first precedes the second in temporal, or left-to-right, sequence. If there is no path from one to another, then they belong to alternative hypotheses about the structure of of the sentence. So, the sentence "they are flying planes" has, let us say, two analyses, each consisting of a noun phrase followed by a verb phrase. But there is no path between the phrases in one analysis and those in the other. The verb phrase in one analysis consists of the verb "are" followed by the noun phrase "flying planes", which are therefore adjacent on the same path, but there is no path from either of them to the verb phrase they make up because this is an alternative analysis of the same set of words.

An incomplete edge represents part of a phrase together with an indication of what would have to be added to complete it. For example, an incomplete noun phrase might include the string "the big black" plus an indication that a following noun, possibly preceded by some more adjectives, would complete it. A special kind of incomplete edge is an empty edge. Empty edges are successors of themselves. In other words, they are always incident from and to the same vertex, reflecting the fact that they represent no part of the sentence, but merely the potential for finding some structural component. The specification of how an incomplete edge can be completed takes the form of one or more arcs in the grammar, each paired with a direction--left or right. If the direction is right, then completion can be achieved by following sequences of arcs incident from the given state; if it is left, then arcs incident to the state must be followed.

The interpreter uses the scheduling rules to chose an item on the agenda which it then carries out. If all the tasks that were ever put on the agenda in the course of generation or analysis were carried out in an arbitrary order, then all results that the grammar allowed would be obtained sooner or later. The scheduling rules formalize strategies of one kind and another. They are presumably designed so as to shorten the time required to reach a result which is, in some sense, acceptable, at which time the remaining entries on the agenda can simply be abandoned.

The typical task is an attempt to apply an arc from the grammar to an edge in the chart. If the arc applies successfully, some new material will be added to the chart. In generation, the new material typically consists of one or two new edges constituting a sequence with the same end points as those of the initial edge. Usually, no more than one of the newly introduced edges will be incomplete. Thus, there might be a task in which an arc was applied to the noun-phrase edge representing "the big black dog" and which resulted in the complete article "the" and the incomplete noun phrase "big black dog". In parsing, the task specifies one or two edges, one of which is incomplete. The idea is to attempt to take the complete edge at least one step nearer to completion by incorporating the other edge. If only one edge is specified, then the new edge will have the same end points as the original, but will presumably be differently labeled.

Within this version of GSP, top-to-bottom, left-to-right parsing, in the manner of an ATN parser, proceeds broadly as follows:

1. Whenever, as the result of introducing a new edge into the chart, a new sequence consisting of an incomplete edge followed by a complete one comes into existance, put a new task on the agenda for each of the "category" arcs named on the incomplete edge. When one of these tasks is executed, the arc will be applied to the complete e e giving rise, if successful, to a new edge, complete or incomplete.
2. Whenever a new incomplete edge is introduced that names a "Pop" or "Jump" arc, create tasks that will cause these to be carried out.
3. Place an empty sentence edge before the first word of the sentence.

The process starts with step 3, which immediately causes a sequence of instances of steps 1 and 2.

An incomplete edge represents a stack frame in the ATN processor. It is labeled with a set of registers and it spans a portion of the chart representing the part of the string so far analyzed. "Category" arcs are applied to an incomplete edge and a complete edge immediately to its right. If successful, the result is a new incomplete edge. "Pop" arcs produce a complete edge, in exchange for an incomplete one. "Jump" arcs produce an incomplete edge in exchange for an incomplete edge, the differences being in the arcs that specify how to proceed towards completion, and possibly in the label.

It turns out that the mechanism of recursive calls that "Push" and "Pop" arcs provide for is embraced by the devices already described. Suppose that sentences are to be analyzed as consisting of a noun phrase followed by a verb phrase and that a noun phrase, say "the big black dog" has somehow been recognized at the beginning of the sentence. This means that there will be a complete edge representing this noun phrase and an incomplete sentence edge which has the same end points with an arc specifying that a verb phrase with a singular, third-person verb, is to follow. Since the grammar contains a subnetwork giving the structure of verb phrases, an empty edge labeled with the category "verb phrase" is introduced following that incomplete sentence provided there is not one already there. In due course, this will presumably cause a complete verb phrase to appear. The general principle is this: whenever an incomplete edge specifies a "category" arc for which there is a corresponding subnetwork, an empty edge is created following that one for each of the

initial arcs in the subnetwork in the hope that this will lead to the creation of a new complete edge that the "category" arc can be successfully applied to.

3. THE USE OF REGISTERS

The principal problem with this simple plan, when applied to reversible grammars, is that the registers cannot be guaranteed to have the necessary contents at the time required. One of the strengths of the ATN formalism is that it allows the parser to "change its mind". The canonical example is the passive construction. The first verb phrase in the sentence is assigned to the subject register. But when a passive verb--part of the verb "be" and a transitive past participle--has been encountered, the contents of the subject register are simply transferred to the object register. If a "by" phrase follows, its object will later go into the subject register. In this way, a great deal of backing up is avoided.

In generating a passive sentence, it is clear that the first step cannot be to deposit the contents of the subject register in the first position. An alternative might be to decide which register to use in filling the first position by examining a "voice" register, using the object instead of the subject register if its value is "passive". But this would require us to assign a value to the voice register in parsing before the relevant evidence is in. It would work only if the contents of the voice register were changed at the same time as the passive verb was recognized and the contents of the subject register were moved to the object register. It could indeed be made to work, but the solution is unsatisfactory because it does not reflect any general principle that carries over to other cases. More important, it violates a princple that must be regarded as fundamental for the achievement of reversibility in general, namely that each elementary operation that an arc in the grammar can specify must have two systematically related interpretations, for use in generation and parsing respectively.

Another solution would be to assign the first noun phrase to a neutral register when it is encountered in parsing, and only to copy it into the subject or object registers when it was finally established which one it belonged in. This neutral register would have to be reflected directly in the structure assigned to the sentence because it would be from there that the first noun phrase in the sentence would have to be taken by the generator. One advantage of this scheme is that a passive marker would no longer be required in the structure of passive sentences. Instead, the voice of a sentence would be determined by the generator on the basis of which register--subject or object--had the same contents as the special neutral register. The general principle behind this strategy is that the contents of a register are never changed in the course of either generation or parsing. This is the solution I advocate.

A name is needed for the neutral register, and <u>topic</u>, or <u>theme</u>, suggest themselves immediately. But consider the case of cleft sentences like "It was Brutus that killed Caesar" and "It was Caesar that Brutus killed" and assume, for the sake of the argument, that these are to be handled as main clauses and not by the relative-clause mechanism. Once again, the underlying grammatical function of the first noun phrase is not known when the parser first encounters it. The problem can be solved by the same device, but of all the names one might choose for the neutral register, "topic" is least appropriate in this instance. Something like <u>focus</u> or <u>comment</u> would be more to the point. What, then, of datives? Consider "He gave Fido a bone" and "He gave Fido to Mary". The problem here is the noun phrase following the verb. In neither case can we convincingly argue that it is either the topic or the focus of the sentence.

4. FUNCTIONAL SENTENCE PERSPECTIVE

The most satisfying solution to these problems is to be found in the work of the Prague school of linguists, particularly Mathesius, Firbas, Danes, and Sgall. The basic notion is that of the <u>Functional Sentence Perspective</u> according to which topic and focus are two regions in the scale of <u>communicative dynamism</u> along which each of the major constituents of a sentence are ordered. In the unmarked case, each succeding constituent in the surface string has a higher degree of communicative dynamism. The point on the scale at which one passes from topic to focus may or may not be marked. In speech, special stress can be used to mark any element as the focus; in writing, several devices like clefting fill the same role.

Communicative dynamism correlates with a number of other notions that are more familiar in this part of the world. Elements that are low on this scale are the ones that are more <u>contextually bound</u>, which is to say that they involve presuppositions about the preceding text. In "It was Brutus that killed Caesar", "that killed Caesar" is the topic and it clearly involves the presupposition that someone killed Caesar. In an unmarked sentence, like "Brutus killed Caesar", it is not clear whether the dividing line between topic and comment falls before or after the verb; there are nevertheless three degrees of communicative dynamism involved.

According to this view, the difference between "He gave Fido to Mary" and "He gave Mary Fido" is not in what is topic and what is focus but simply in the positions that "Mary" and "Fido" occupy on the scale of communicative dynamism. Consider the sentences:

the reward to Bill.

(2) John did all the work, but they gave Bill the reward.

(3) They were so impressed with the work that they gave Bill a reward.

(4) They were so impressed with the work that they gave a reward to Bill.

I claim that (2) and (4) are less natural than (1) and (3) when read with even intonation. Sentence (5), with underlining for stress, is, of course, quite natural, and (6) is questionable.

(5) John did all the work, but they gave Bill the reward.

(6) They were so impressed with the work that they gave a reward to Bill.

The claim is simply that the last item carries the greatest communicative load, represents the most novel component of the sentence.

This is consistent with the observation that dative movement is at best awkward when the direct object is a pronoun, as in

(7) I gave him it.

and it becomes more awkward when the indirect object is more ponderous, as in

(8) I gave the man you said you had seen it.

In fact, it is consistent with the observation that ponderous constituents tend to be deferred, using such devices as extraposition. It is in the nature of pronouns that they are contextually bound, and the complexity of large constituents presumably comes directly from the fact that they tend to convey new information.

What this suggests is a formalism in which the structure of a phrase is a list of attributes named for grammatical functions, whose values are words or other phrases. They are ordered so as to show there positions on the scale of communicative dynamism and there is provision for a marker to be introduced into the list explicitly separating the topic from the focus. Considering only the sentence level, and simplifying greatly, this would give examples like the following, using "/" as the marker:

[Subject:John Verb:gave Dir-obj:(the candy) Indir-obj:Mary] => "John gave the candy to Mary"

[Indir-obj:Mary Verb:gave Dir-obj:(the candy) Subject:John] => "Mary was given the candy by John"

[Verb:gave Dir-obj:(the candy) Indir-obj:Mary / Subject:John] => "It was John that gave Mary the candy" or "John gave Mary the candy"

[Subject:John Verb:gave Dir-obj:(the candy) / Indir-obj:Mary] => "It was Mary that John gave the candy to"

Verb:gave Indir-obj:Mary] => "What John did with the candy was give it to Mary"

The implications for reversible syntactic processing seem to be as follows: The familiar set of registers, named for the most part for the names of grammatical functions, are supplemented by three others called topic, focus and, say, marker. Marker will have a value only when the sentence is marked in the sense I have been using. Topic and focus will contain ordered lists of elements. The structure of a passive sentence, for example, will be recognizable by the fact that it is unmarked and has a patient (dative, or whatever) as the first item on its topic list. The parser will place the first noun phrase in a "standard" sentence on this list and only copy it into some other register later. The generator will unload the first item into the string and decide later what form of verb to produce.

The ill-formedness of the ideas I have tried to present here is clear for all to see. I have so far acquired only the most tenuous grasp of what the Czech linguists are doing and, while I should publicly thank Petr Sgall for his patience in explaining it to me, it is only right that I should also appologise for the egregious errors I have doubtless been guilty of. But, whatever errors of detail there may be, one important point will, I hope, remain. The notions of topic and focus are clearly well motivated in theoretical linguistics, and the richer notion of functional sentence perspective probably is also. I have been led to these same notions for purely technical reasons arising out of my desire to build a reversible syntactic processor.

REFERENCES

Firbas, J. "On Defining the Theme in Functional Sentence Analysis", Travaux Linguistiques de Prague, Vol. 1, pp 267-280, 1964.

Kaplan, Ronald M. A General Syntactic Processor in Randall Rustin(ed.) "Natural Language Processing", New York, Algorithmics Press, 1973.

Mathesius, V. "Zur Satzperspektive in modernen English", Archiv fuer das Studium der neueren Sprachen und Literaturen, Vol. 155, pp. 202-210, 1929.