COLING 82, J. Horecký (ed.)
North-Holland Publishing Company

INCREMENTAL SENTENCE GENERATION: IMPLICATIONS FOR THE STRUCTURE OF A SYNTACTIC PROCESSOR

Gerard Kempen & Edward Hoenkamp

Department of Psychology
University of Nijmegen, The Netherlands

Human speakers often produce sentences incrementally. They can start speaking having in mind only a fragmentary idea of what they want to say, and while saying this they refine the contents underlying subsequent parts of the utterance. This capability imposes a number of constraints on the design of a syntactic processor. This paper explores these constraints and evaluates some recent computational sentence generators from the perspective of incremental production.

An important characteristic of spontaneous speech is that overt pronunciation of a sentence can be initiated before the speaker has completely worked out the conceptual content he is going to express in that sentence. Apparently, the speaker is able to build up a syntactically coherent utterance out of a series of syntactic fragments each rendering a new part of the conceptual content. This incremental, piecemeal mode of sentence generation imposes some important constraints on the design of possible mechanisms for building syntactic structures.

CONSTRAINTS ON INCREMENTAL SYNTACTIC PROCESSORS

1. Lexically driven syntactic processing. The first constraint derives from the fact that it is conceptual structures which serve as input to the tree formation process. A good strategy for translating these meanings into language begins by looking up words covering them ("lexicalization"). Subsequently, the processor attemps to build a syntactic framework which accommodates all words while respecting their syntactic properties (e.g. word class). In case of success, the result is a syntactic tree with lexical items as terminal nodes. In case of failure, one or more words are replaced by other lexical material which expresses the same meaning but whose syntactic properties are more favorable. The point we want to make here is that it is the syntactic properties and peculiarities of lexical items which guide the tree formation process.

In short, syntactic processing is lexically driven. This feature requires special rules not found in current linguistic grammars where it is common practice to set up a linguistic framework (e.g., by applying phrase-structure rules) without reference to syntactic properties of lexical items [1]. Adopting this practice would presuppose that syntactic trees are directly computable from the shape of conceptual structures, that is, without the intermediation of lexical items. This supposition is valid only for conceptual structures which are virtually isomorphic with syntactic trees. Most probably, such an isomorphism does not hold for the structures delivered by the conceptualization system in human speakers.

2. Hierarchy and order of constituents computed by different components. The second constraint hinges upon the independence between the order of conceptual fragments coming in and the order of the corresponding syntactic fragments. With the possible exception of languages with extremely flexible word order, grammar rules do not always permit a new syntactic fragment to be simply appended to the right-hand side of the current tree. Other spatial arrangements of the new fragment with respect to the current syntactic tree are possible, depending on the word order rules in the grammar. Sometimes these rules even ask for the presence of other elements between the current tree and a newly computed syntactic fragment. A clear example is provided by the position of verbs in main clauses of Dutch and German. Subject noun phrases and adverbial phrases cannot follow each other at the beginning of a main clause. The finite main verb or auxiliary is always in between: either NP-V-AP or AP-V-NP but not NP-AP-V or AP-NP-V. Grammars which use some version of the traditional phrase-structure rules do not keep word order apart from phrase membership (more precisely, constituent hierarchy from constituent order). For example, consider the following rules which express the above word order contingencies:

S ----> NP+V+AP
S ----> AP+V+NP

Now suppose that the syntactic processor is working on a conceptual fragment which lexicalizes into a verb, and applies the first rule which says, among other things, that the verb needs an NP at its left-hand side. In the meantime a new conceptual fragment has come in which receives the syntactic shape of an AP. The first rule does have an AP slot, but not to the left of the verb. This implies the syntactic processor has to wait for a third conceptual fragment which can be worded in the form of an NP. At that point the syntactic processor can deliver its first output: an NP-V-AP utterance. The waiting time, that is, the period between onset of (conceptual) input and onset of (syntactic) output, would have been shorter, had the syntactic processor picked the second phrase-structure rule. Then, output could already have begun after the second conceptual fragment ("AP-V...") and closed off grammatically with "...NP". Because the order of conceptual fragments is unknown in advance, the syntactic processor can never be sure of having made the best choice between rules. This problem does not arise in a rule system which allows word order to be computed independently of phrase membership. We conclude, therefore, that in an incremental syntactic processor it is desirable to have seperate components for tree (or rather "mobile") formation and for word order.

3. Explicit computation of grammatical (functional) relationships. Traditional phrase-structure rules allow grammatical relationships (subject, direct object, nominal modifier, etc.) to be inferred from configurations of categorial nodes in the syntactic tree. This is not true of tree formation rules which leave left-to-right order of constituents undefined. If such a system contained a rule

VP ----> V-NP-NP

it would be impossible to determine which of the NPs served the function of direct object. Functional information of this kind is needed by the word order component (or, in languages with free word order, by the morphological case component).

An additional motivation for direct computation of functional syntactic relationships is provided by the lexicalization process. Constraint #1 makes the prescription that it is choice of lexical items which guide the formation of syntactic trees rather than vice versa. Many lexical items require a specific syntactic environment, a typical example being verbs like give which cause the formation a VP with two NPs, for direct and indirect object respectively. In conjunction with customary phrase-structure rules, this property of give could be expressed in terms of a desired configuration of categorial nodes, e.g. (VP(NP___)(NP___)). This option is not available in a tree formation system which generates "mobiles". Here, the lexical entry for give should explicitly reference direct and indirect objects as desired constituents.

4. Simultaneous construction of parallel branches of syntactic trees. Constraint #2 entails a work scheduling problem: if more than one branch is to descend from a given node, in what order should they be constructed by the syntactic processor? The standard solution, i.e. to develop constituents in their order of appearance in surface structure, is no longer applicable since left-to-right order is undefined at this stage [2]. In a lexically driven syntactic processor, the most efficient solution is a priority scheme based on the order of arrival of lexical items (cf. Constraint #1). This order, in turn, is the combined result of the order in which conceptual fragments become available and the manner of operation of the lexicalization process, and need not corrrespond at all to their surface structure order. For example, the verb-second rule of German and Dutch applies irrespective of whether the verb comes into play earlier or later than other lexical materials of the main clause.

When computing a branch connecting a lexical item to the current syntactic tree, the processor has to take into account the functional relations this item maintains with other lexical items put forward by the lexicalization process (cf. Constraint #3). For example, the noun designated as subject of a verb will receive a different place in the syntactic tree than the object noun. For the rest, there are no cross-branch computational dependencies forcing a systematic order upon the construction of branches of syntactic trees. This statement is supported by the success most grammar types obtain by having context-free rules generate deep structure trees (and sometimes even surface structure trees). This implies we can trust the above priority scheme (simply follow order of arrival of lexical items) even though it does not bring any syntactic expertise to bear. Now suppose the syntactic processor is capable of a certain amount of parallel processing. We are then permitted to assume that lexical items are attached to the tree simultaneously (again respecting their order of arrival) rather than sequentially. As a matter of fact, this is what human speakers seem to do, as witnessed by certain speech error phenomena (Garrett, 1975) and by reaction times to initiate sentences (Kempen & Huijbers, in press).

5. Operations on syntactic trees subject to locality constraints. In an incremental syntactic processor, the application of tree formation and word order rules will often yield "narrow" trees dominating small sentence fragments. Now suppose that some trees have to undergo certain obligatory operations (e.g., transformations to be executed or anaphoric relationships to be established) and that such operations are triggered as soon as the tree matches a specific pattern. One can imagine "horizontal" (left-to-right) trigger patterns spanning a number parallel branches, "vertical" (top-down) patterns

specifying some configuration of dominating and dominated nodes on one branch, or "mixed" patterns. The pattern that triggers passivization is an example of a horizontal one involving several parallel branches (object NP, passive marker, main verb, and optional subject NP). Wh-fronting and Raising transformations are triggered by vertical or nearly vertical patterns. Incremental production favors vertical trigger patterns because they are more easily satisfied by narrow (partial) syntactic trees corresponding to fragmentary conceptual inputs. Horizontal patterns can only be matched by "wider" syntactic trees which correspond to more elaborate conceptual structures. This latter requirement, however, runs counter to the very idea of incremental sentence production.

The interesting point is that conditions on transformations and other linguistic rules can usually be expressed in terms of (nearly) vertical node configurations. A clear example is provided by Koster's (1978) Locality Principles where a central role is played by dominance and command relationships between nodes. In the context of a computer implementation of our Procedural Grammar (Kempen & Hoenkamp, 1981), which was specially designed for the purpose of incremental production, we have attempted to make more precise the parallelism between vertical trigger patterns and locality constraints on transformations (see Hoenkamp, 1982, for a more formal approach).

6. Lexical transformations. By constructing a partial syntactic tree the syntactic processor commits itself to a limited range of continuation alternatives. The lexicalization process should be sensitive to such syntactic commitments by making available lexical materials which are compatible with what has been said in earlier parts of the sentence. Take the example of a concept which, after having been expressed as a subject NP, turns out to be the patient of an action, as specified in a subsequent conceptual fragment. A typical lexical realization of the action might be an active verb which prescribes the patient to be rendered as the object NP. However, this would entail incompatibility with the current syntactic tree. The solution is provided by a lexical passive transformation that, among other things, alters the pairings of conceptual cases with syntactic functions (causing patient to go with subject). The transformed lexical item is then successfully attached to the current tree. By attuning lexical items to the exigencies of incomplete syntactic trees, the lexicalization component greatly enhances the left-to-right planning capabilities of the syntactic processor [3].

7. Sentence production in two stages. In the foregoing we have not yet touched upon the issue of where and when inflectional computations are carried out. The obvious placement of an inflectional component -- somewhere at the end of the tree formation process -- leads to an interesting problem. In many languages, including English, German and Dutch, clitics (monosyllabic function words) are optionally contracted with preceding lexical items. Some examples are John is ill --> John's ill; will not --> won't; (Ger.) unter dem Turm --> unterm Turm ("under the tower"); von dem --> vom ("of the ..."). Clitic contraction implies merging the lexical items of two adjacent branches of a syntactic tree into a single word.

In the context of Constraint #4 we have seen that the most efficient order of constructing branches of a syntactic tree simply copies the order of arrival of lexical items. This, in turn, implies that clitic contraction cannot be performed by the tree formation components (including the word order component): there is always a chance that a

later lexical item gets hold of a place inbetween the clitic and its predecessor. For instance, John be ill might be expanded into John will be ill, or von dem into von all dem ("of all the ..."). Therefore, clitic contraction must take place after tree formation, that is, after the moment the syntactic processor decides that the current tree (possibly an incomplete one) is a suitable linguistic formulation of the conceptual input received so far. It follows there is a subsequent stage of processing which takes care of clitic contraction, and maybe of other aspects of the morphological shape of words. This latter addition is plausible from the point of view of efficiency. It does not make sense to have the tree formation components engage in detailed inflectional computations if some of these are undone later (namely, the computations that are superseded by clitic contraction).

It is a remarkable fact that speech error data have given rise to a two-stage sentence production model with a similar division of labor between stages: roughly, syntactic tree formation versus inflectional morphology (Garrett, 1975; Kempen & Huijbers, in press). These data also suggest that the second processing stage deals with the terminal nodes of a (possibly incomplete) syntactic tree in their left-to-right order.

INCREMENTAL SENTENCE PRODUCTION IN MODELS OF THE SPEAKER

It will come as no surprise to the reader that the only computational model of sentence production which, in the authors opinion, satisfies all or most of the above constaints, is the one developed by the authors themselves (Kempen & Hoenkamp, 1981). We know of one other computational sentence generator whose design was explicitly concerned with incremental production. It was written by McDonald (1980, 1982) and embodies a broad range of syntactic constructions. However, this model fails to distinguish hierarchical from word order rules and, consequently, violates Constraint #3. We cannot judge whether removal of this shortcoming will necessitate drastic changes to the rest of the program.

The type of grammar embodied by the Kempen & Hoenkamp model (Procedural Grammar) is similar to Lexical Functional Grammar (Kaplan & Bresnan, 1982; see also Bresnan, 1981). The main difference concerns the attitude towards transformations. In Lexical Functional Grammar, surface trees are base-generated and no transformational component is needed. If Kaplan & Bresnan motivate their rejection of a transformational component on psychological grounds [4], we disagree. Neither incremental production nor any other known fact about human sentence production processes argues for complete banishment of transformational operations on syntactic trees.

Procedural Grammar is unique in its ability to deal effectively with conceptual inputs which may change on line. A conceptual structure which is altered after it has been expressed linguistically causes the processor to backtrack and to make "repairs".

Acknowledgements. The work reported in this paper was supported by a grant from ZWO, the Netherlands Organization for Pure Scientific Research. We are indebted to Patrick Hudson for his valuable comments.

NOTES

[1] Categorial grammars form an exception here. However, a processor based on this grammar type violates Constraints #2 and #3: categorial rules presuppose left-to-right order of lexical items, and make no use of functional syntactic relations.

[2] This solution is the one that has been adopted of old, from Yngve (1960), via ATN-based generators (e.g. Simmons & Slocum, 1972; Anderson, 1976) to McDonald (1980).

[3] Lexical transformations may involve other types of alterations as well, e.g., derivational morphological operations and insertion of function words. Actually, the addition of function words and inflections (or, rather, inflectional presciptions) is another general possibility for the lexicalization component to accommodate a lexical item to properties of the current syntactic tree. (Inflectional prescriptions are executed during a subsequent processing stage; see Constraint #7.)

[4] Bock (1982, p. 28) opts for Gazdar's (1981) context-free grammars because they are "much more compatible with on-line processing models than transformational grammars".

REFERENCES

Anderson, J. Language, memory, and thought. Hillsdale, N.J.: Erlbaum, 1976.

Bock, J.K. Toward a cognitive psychology of syntax: information processing contributions to sentence formulation. Psychological Review, 1982, 1, 1-47.

Bresnan, J. An approach to Universal Grammar and the mental representation of language. Cognition, 1981, 10, 39-52.

Garrett, M. The analysis of sentence production. In: G. Bower (ed.), The psychology of learning and motivation, Vol. 9. New York: Academic Press, 1975.

Gazdar, G. Unbounded dependencies and coordinate structure. Linguistic Inquiry, 1981, 12, 155-184.

Hoenkamp, E. Aspecten van een computermodel van de spreker. Ph.D. Dissertation, University of Nijmegen, 1982 (in prep.).

Kaplan, R.M. & Bresnan, J. Lexical-Functional Grammar: a formal system for grammatical representation. To appear in: Bresnan, J. (ed.), The mental representation of grammatical relations. Cambridge, Mass.: MIT Press, 1982.

Kempen, G. & Hoenkamp, E. A procedural grammar for sentence production. Report 81 FU 03. Department of Psychology, University of Nijmegen, 1981.

Kempen, G. & Huijbers, P. The lexicalization process in sentence production and naming: indirect election of words. Cognition, in press.

Koster, J. Locality principles in syntax. Dordrecht: Foris, 1978.

McDonald, D. Natural language production as a process of decision-making under constraints. Ph.D. Dissertation, MIT, 1980.

McDonald, D. Natural language generation as a computational problem: an introduction. To appear in: Brady (ed.), Computational theories of discourse. Cambridge, Mass.: MIT, 1982.

Simmons, R. & Slocum, J. Generating English discourse from semantic networks. Communications of the ACM, 1972, 15, 891-905.

Yngve, V. A model and a hypothesis for language structure. Proc. Amer. Phil. Soc., 1960, 104, 444-466.