A FRAMEWORK FOR INCREMENTAL SYNTACTIC TREE FORMATION

Gerard Kempen

Experimental Psychology Unit, University of Nijmegen,
Montessorilaan 3, 6525 HR Nijmegen,The Netherlands.

ABSTRACT.

A syntactic tree formation system is described for use in incremental sentence generators, i.e. generators which allow for parallel planning of conceptual content and linguistic expression. The system achieves full generality in the sense that (1) all forms of incremental production (upward expansions, downward expansions, and insertions) are covered, and (2) branches of a syntactic tree are grown in a maximally independent way (3) while at the same time grammatically of the utterance as a whole can be maintained (aside from 'syntactic deadlocks'). The tree formation system is called Incremental Grammar (1G). Two novel ideas are the use of node-arc-node segments as elementary building blocks (e.g S-subject-NP, S-object-S, NP-head-N, PP-object-NP), and the introduction of feature matrices associated with segments rather than with nodes.

1. INTRODUCTION

The problem of *incremental* sentence generation has begun to attract the attention of AI researchers, computational linguists and psycholinguists (Joshi, 1987; MacDonald & Pustejovsky, 1985; Kempen, 1978). In psycholinguistics, incremental generation accounts for the observation that speakers plan their utterances partly from left to right and sometimes 'talk themselves into a corner' (syntactic deadlock, self-corrections). In the context of interactive AI systems, the need for an incremental generation strategy arises whenever those systems attempt to spare the user long pauses between the successive utterances of a natural-language interface. In combination with synthetic speech, an incremental generator may produce a very natural output.

Although it is gradually being recognized that incremental generation imposes special requirements upon syntactic mechanisms, no satisfactory framework for incremental syntactic tree formation has emerged. Linguistic formalisms which have addressed the issue more or less explicitly, are Functional Unification Grammar (Appelt, 1985) and Tree Adjoining Grammar (Joshi, 1987). However, as pointed out by De Smedt & Kempen (1987), they can handle only certain types of'incrementation' and do not achieve a full solution.

In Section 2, I put forward three basic demands to be made upon truly incremental tree formation systems. A framework satisfying these requirements is described in Section 3. Finally, Section 4 is devoted to a comparison with existing syntactic formalisms and to some evaluative remarks.

2. INCREMENTAL GENERATION: IMPLICATIONS FOR TREE FORMATION

De Smedt & Kempen (1987) distinguish three types of incrementation: upward expansion, downward expansion, and insertion. A combined illustration is provided by utterance (1).

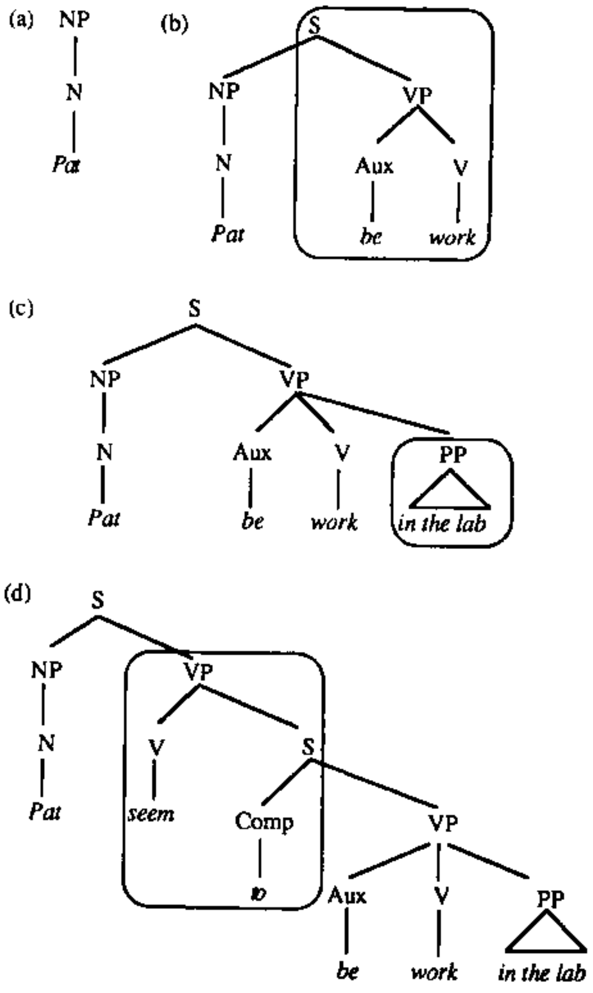


Figure 1. Incremental generation of sentence (1).

(1) Pat / was working / in the lab / seemed to be working in the lab

The increments (separated by slashes) are also shown in the boxed parts of Figure 1. Figure lb is an upward expansion of la because the original root node (NP) has become the daughter of a new parent (S). Downward expansion is exemplified by Figure lc: the increment is attached as a subtree to an already existing node. Figure ld shows a case of insertion: the increment is added inbetween two existing nodes. (One may view insertion as a combination of upward and downward expansion.) All three varieties of incrementation should be within reach of an incremental tree formation *system* This *is* the first requirement.

The second requirement reflects the observation, as witnessed by (1), that increments may be very small, sometimes no longer than a single word. Therefore the grammar should enable growing individual branches without imposing unnecessary constraints upon the simultaneous growth of other branches. In other words, the grammar needs a 'vertical' orientation: Establishing connections between mother and daughter nodes, between daughter and grand-daughter, etc. (or vice-versa) should be its primary concern. Existing grammar formalisms tend to have a 'horizontal' orientation: Their rules typically specify *combinations* of daughters for a given mother node, that is, they emphasize sisterhood. Phrase structure rules are the best example. An incremental generator, on the other hand, needs a type of rule which stresses mother-daughter relationships (allowing for *inferences* concerning sisterhood). From the point of view of incremental generation, the main advantage of 'vertical' rules is that the development of two or more sister branches need not be initiated at the same point of time.

The third requirement addresses maintenance of grammatical coherence in the course of utterance realization. The chronological order in which the various constituents are attached to the syntactic tree clearly need not be identical to their left-to-right order in the resulting utterance. This applies in particular to languages with rigid word order patterns such as English, Dutch and French. If the generator would overtly realize constituents immediately after their being attached to the tree, massive ungrammaticality would ensue. An example is provided by the finite (main or auxiliary) verb in main clauses of Dutch and German. In many circumstances, this constituent occupies an obligatory 'second position*. Now suppose the subject and direct object constituents are attached to the S-node earlier than the finite verb (subject at position 1, direct object at 3). Overtly realizing subject and direct object before attachment of the finite verb would lead to a sequence of constituents which will remain ungrammatical, irrespective of how it is completed by further constituents. What is needed, apparently, is a device which prevents the utterance realization process to skip over obligatory constituents with reserved positions.

3. SYNTACTIC TREE FORMATION IN INCREMENTAL GRAMMAR

The grammatical framework developed below is called Incremental Grammar (IG). It may be viewed as abstracted from Kempen & Hoenkamp's (1987) Incremental Procedural Grammar framework (IPG) by leaving out psycholinguistic processing (i.e. 'procedural') aspects. Two important innovations are reported here for the first time: the introduction of node-arc-node segments as elementary building blocks, and a more systematic treatment of features and feature transport than was given in IPG. The latter includes the idea of associating feature matrices not only with nodes but also with the larger segments.

3.1 Tree Structure

IG trees consist of labeled nodes and arcs. Names of syntactic categories serve as node labels; arcs are labeled by syntactic functions. A simple example is given in Figure 2.

The elementary building blocks are node-arc-node triplets called segments. Table 1 lists 19 *segment* types needed in many natural language grammars. The mother node of a segment ('root') is a *phrasal* category: Sentence (clause), Noun Phrase, Prepositional Phrase or Adjectival/Adverbial Phrase. The daughter node (' foot') is either a phrasal or a lexical category (Verb, Noun, Pronoun, Cardinal Number, Article, Coordinating Conjunction, Subordinating Conjunction, etc.). The arc labels have been selected from a small set of syntactic functions such as Head, Subject, Direct Object, Indirect Object, Modifier, Quantifier, etc.

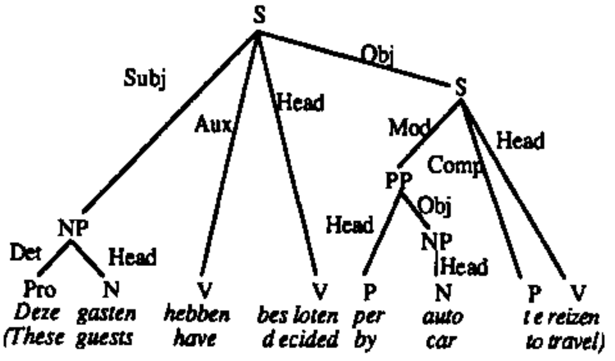

S
Subj
Aux
Head
Obj
S
Mod
Head
Comp
PP
Obj
Head
NP
Head
NP
Det
Head
Pro N V V P N P V
Deze gasten hebben bes loten per auto t e reizen
(These guests have d ecided by car to travel)

Figure 2. Syntactic tree corresponding to the Dutch equivalent of *These guests have decided to travel by car.*

The members of a coordination are dominated by an arc labeled 'Conjunct', and the coordinating conjunction by 'Sequencer'[1] (cf. Figure 3).

There are three *composition* operations: concatenation, insertion, and furcation. They all involve merging identically labeled nodes from two segments. In case of *concatenation,* the root node of one segment is merged with the foot of the other one. Strings of one or more concatenated segments are branches; they, too, have a foot and a root. *Furcation* is a merge of two root nodes. In case of *insertion,* one node of a segment/branch is replaced by a segment/branch whose root and foot labels are identical (to each other and to the replaced node).

The composition operations are illustrated by the incremental construction of Dutch sentence (3a) out of the six segments listed in (3b).

```
(3a)    Ik  / wil  /  geen  appels / eten
        I      want    no    apples   eat
(3b)
   A.   NP—Head—N (Ik)
   B.   S—Subj—NP
   C.   S—Head—V (wil)
   D.             Det—Pro (geen)
                  /
                 S—Obj—NP
                  \
                  Head—N (appels)
   E.   S—Obj—S
   F.   S—Head—V (eten)
```

The first word of sentence (3a) can be realized after concatenating segments A and B. (I assume that nominative case is selected only after the NP has been assigned the role of Subject.) The second word follows after furcating C with the Subject branch. Furcation of the resulting structure with Object branch D gives the third increment. Then segment E is inserted into D at root node S (effectively 'lowering' the Object NP) and segment F furcated with the embedded S. The result is depicted in Figure 4.

Table 1. Important types of segments. Alternative node labels are separated by slashes.

ROOT	ARC	FOOT	ROOT	ARC	FOOT
S	Head	V	PP	Head	P
S	Subj	NP/S	PP	Obj	NP/S
S	Obj	NP/S	PP	Mod	AP/PP
S	IObj	NP			
S	Pred	NP/AP/PP/S	AP	Head	A
S	Mod	NP/AP/PP/S	AP	Mod	AP/PP
S	Aux	V			
S	Comp	SC/P	NP/AP/PP/S	Seq	CC
			NP/AP/PP/S	Conj	same as root
NP	Head	NP/Pro/S			
NP	Det	Art/Pro			
NP	Mod	AP/PP/S			
NP	Quant	Card			

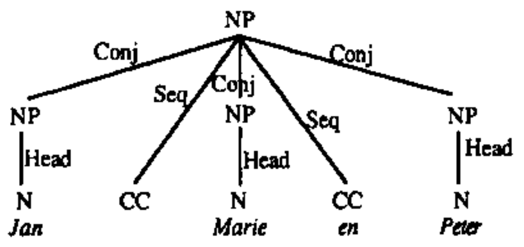


Figure 3. Syntactic tree corresponding to the Dutch equivalent of *John, Mary and Peter.*

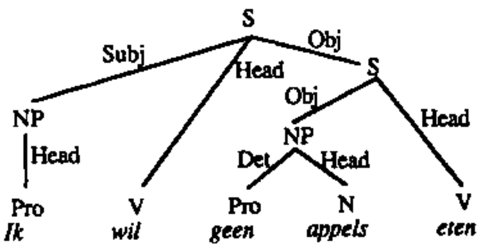


Figure 4. Syntactic tree corresponding to sentence (3a).

The example shows that all three kinds of incrementation distinguished in Section 2 are within reach of IG: upward expansion (A+B), downward expansion (B+C, C+D), and insertion (D+E). Moreover, the nature of the segments and the composition operations defined over them lends the grammar a vertical rather than a horizontal orientation.

3.2 The Lexicon

The lexical entries in an IG lexicon are arranged in the form of a hierarchy of objects, each object representing a *segment* or segment type. Figure 5 illustrates a small portion of the IG lexicon for Dutch. The lines represent *inheritance* links between objects: a lower segment possesses all properties of its parent(s). except those which are explicitly overwritten. (De Smedt, 1984 advocates the utilization of object-oriented programming techniques for the representation of lexical knowledge, e.g.. for the purpose of default reasoning.) The entry for the Dutch verb *willen (to want)* is shown in simplified form at the bottom of Figure 5. The inheritance links dominating the object *'willen-vtrb* 'indicate that *willen* is an instance of an S-Head-V segment. The expression *furcate(...)* says that this segment is forked with a S-Subj-NP segment and either an S-Obj-NP or an S-Obj-S segment (cf. (3bD) and (3bE)). This exemplifies how *subcategorization* restrictions on verbs can be stated very easily. Notice also that the inheritance hierarchy minimizes redundancy. For instance, the fact that segments have a foot, an arc and a root need be mentioned only once, namely, at the topmost member of the hierarchy.

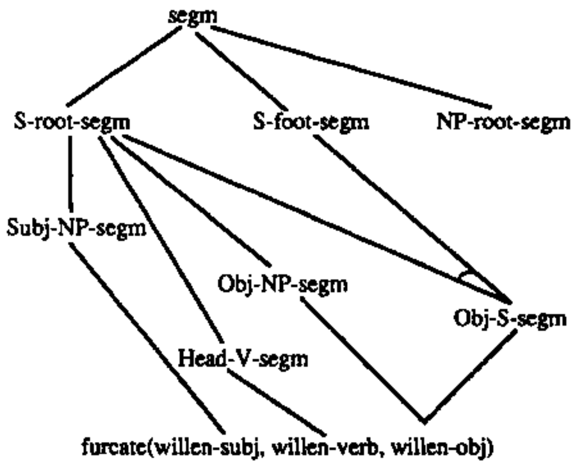
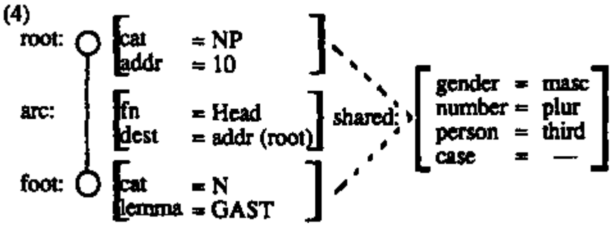


Figure 5. A small portion of an IG Lexicon.

3.3 Features and Feature Transport

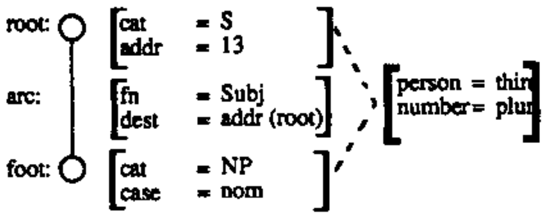
The structure of a segment is more complex than discussed so far. In fact, it is a set of features which is partioned into subsets associated with root, arc and foot. The subsets are conveniently depicted as *feature matrices.* For instance, a more detailed notation for one of the two NP-Head-N segments in Figure 2 is shown in (4).

(4)

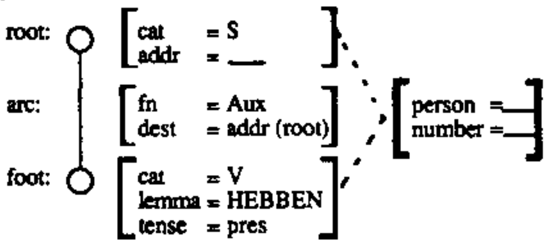


The feature matrix at the right lists features which are *shared* by root and foot. The values of 'cat' (= category) and 'fn' (= function) are labels serving to identify segment type (e.g. NP-Head-N). Undefined values are indicated by *underscores* ('J).The 'addr' (= address) and 'dest' (= destination) attributes defined for root and arc, respectively, refer to an aspect of word order computation which I will discuss below in Section 3.4. The 'lemma' attribute of the segment's foot takes as its value a pointer to a morpho-phonological specification (Kempen & Huijbers, 1983).

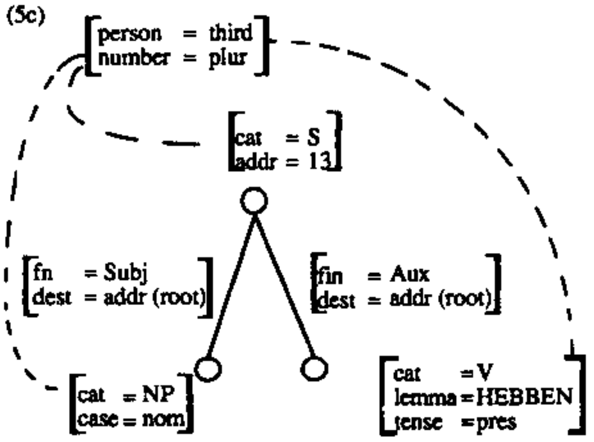
(5a)



(5b)



(5c)



The introduction of feature matrices necessitates reconsidering the operation of node merging discussed above. I define it here as the unification (in a sense similar to Kay, 1985) of the associated feature matrices, including the shared component. For example, furcation of segments (5a) and (5b), both belonging to the tree in Figure 2, yields (5c).*

Unification is an effective mechanism for distributing feature information over syntactic constituents. The burden put upon it is considerably smaller, though, than in Functional Unification Grammar because it operates in conjunction with the tree formation mechanism discussed in Section 3.2.

3.4 Functional and Positional Trees

In the foregoing I have paid no attention to word order. In fact, the trees discussed so far contain no information on this score. For this reason I call them *functional trees.* Word order is computed in the course of a *mapping* from functional into *positional trees.* It proceeds as follows.

To each segment of a functional tree, a value is assigned for two attributes: *destination* (point of attachment, 'address') and *precedence* (serial position amidst segments attached to the same node).

A destination evaluates to an address, i.e., a number associated with the root node of the current segment or one of its ancestors. The default case is 'dest = addr (root)'. Under certain conditions (e.g. related to WH-constituents) the destination value is computed by evaluating special functions imported from the lexicon.

The precedence value of a segment is a sequence of one or more rank numbers. For instance, the four segments attached to the top S-node in Figure 2 might be assigned precedence values 1, 2.1, 2.2 and 3. (A partial set of precedence rules for Dutch, which are mostly applicable to German as well, is worked out and justified in Kempen & Hoenkamp, 1987.) The positional tree belonging to a functional tree is assembled by attaching all segments to their destination node and ordering them from left to right according to their precedence values. (Branches which, in the course of the mapping process, have lost their lexical segment, are pruned away at the lowest furcation point.) The hierarchical structure of a positional tree is often identical to that of its functional counterpart (in Figure 2, for example), but sometimes the positional tree is flatter. A case in point is Dutch sentence (3a) whose functional tree is depicted in Figure 4. This effect — the IG equivalent of Clause Union — comes about as follows.

The lexical entry for the Dutch verb *willen (to want,* cf. Figure 6) lists a special rule which causes the value of the address feature of its object complement S-node to be overwritten by 'addr (root)' in case it is a non-finite clause. The two segments dominating the object NP *appels* now look basically as in (6).

In the course of determining a destination ('addr (root)') for the lower segment, the embedded S is accessed. Since the address listed there evaluates to the topmost S, the lower segment is attached to the top-S rather than to the embedded S. The S-Head-V segment dominating the infinitive verb *eten (to eat)* undergoes the same fate. Left without offspring, the upper S-Obj-S segment is pruned. A flat positional tree is the result, depicted in Figure 6. (Numerical arc labels are rank numbers referring to serial positions within phrases.)

* Notice that the value 'undefined' of a feature does not prevent successful unification. For example, unification of [person = _] and [person = third] yields [person = third].

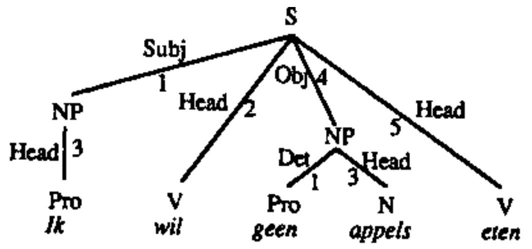
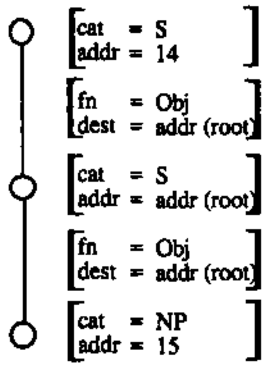


Figure 6. Positional tree corresponding to sentence (3a). The corresponding functional tree is shown in Figure 4.

(6)



3.5 Utterance Realization

When the positional tree dominating a new increment is finished and values for all necessary features have been computed, it is passed on to the *Utterance Realization* module. Here, various morphological and phonological operations take place whose nature I cannot go into (see Kempen & Hoenkamp, 1987; Van Wijk & Kempen, 1987). The important point, in the present context, is that the positional tree spanning a new increment (or a full sentence) is processed in depth-first, left-to-right manner. The output of the realization module is a phonological representation (of a complete or fragmentary sentence) containing all information needed for further phonetic processing.

I owe a solution to the problem, raised at the end of Section 2, of obligatory constituents which have not yet been attached to the tree. How can the Utterance Realization module be induced to halt at the position reserved for such missing constituents? The danger of skipping a reserved position is lurking whenever constituents further down to the right have already been attached. It can be warded off effectively by the following convention. A dummy head segment is obligatorily connected to every phrasal node of the functional tree, until a regular head has been selected. The foot node of a dummy head segment has an empty feature matrix, but otherwise receives the same treatment as a regular one. So it will show up in the positional tree at the reserved position. When running into the defective feature matrix, the Utterance Realization module will come to a halt**.

** Under certain special circumstances, phrases are permitted to remain without a head. Gapping is a case in point (e.g. *Robin wrote a book and John a paper,* where a S-Head-V segment is missing in the second conjunct). For a detailed treatment of conjunction reduction, see Pijls & Kempen, 1986.

4. EVALUATION, RELATED WORK, CONCLUSION

I have described a framework for syntactic tree formation which meets the basic requirements of incremental sentence generation. To my knowledge, no satisfactory alternative is available to date. Other grammatical formalisms reported in the literature fall short in several respects.

Transformational Grammar does not generate anything smaller than full sentences. All base-generated structures have to pass the transformational component of the grammar, and transformations are defined for sentences only. Formalisms which use some form of phrase structure rules (not only Transformational Grammar but also Lexical-Functional Grammar and Generalized Phrase Structure Grammar) are biased towards downward expansion. Upward expansion and insertion are impossible without additional machinery. In addition, as already observed in Section 1, these rules have a horizontal rather than a vertical orientation. Functional Unification Grammar seems biased to downward expansion, too. Categorial Grammar, on the other hand, only allows for upward expansion. Tree Adjoining Grammar can handle insertion very well, and is probably unbiased. However, the elementary trees serving as building blocks are roughly the size of a deep clause, so it is questionable whether TAG could handle smaller increments.

These brief assessments are concerned with *current* versions of the grammar formalisms, which of course were designed for other purposes. How easily they could be tailored to the demands of incremental production without losing sight of their original goals, is difficult for me to judge. Moreover, it remains to be seen whether IG can develop into a viable syntactic formalism living up to the standards of present-day linguistic research. One who wishes to evaluate IG from this point of view be referred to Kempen & Hoenkamp (1987) and Pijls & Kempen (1986). Using the more complicated IPG framework, these authors present detailed treatments of various complicated constructions of Dutch, including interrogatives (WH-movement), object complements (cross-serial dependencies), and coordination (gapping, forward and backward conjunction reduction). The essence of their conclusions generalizes to the IG framework, as interested readers may judge for themselves. These papers also explain how semantic/conceptual structures computed by an AI system can be mapped into IG trees for the purpose of overt expression by artificial language generators.

ACKNOWLEDGEMENT

I owe considerable debt to the members of the Language Technology Project of the University of Nijmegen with whom I had numerous discussions on matters of syntax: Fieny Pijls, Bart Geurs, and in particular to Koenraad De Smedt.

REFERENCES

Appelt, D. (1983) *Planning English sentences.* Cambridge: Cambridge University Press, 1985.

De Smedt, K. (1984) Using object-oriented knowledge-representation techniques in morphology and syntax programming. In: *Proceedings of ECAI '84.* Pisa, Italy.

De Smedt, K. & G. Kempen (1987) Incremental sentence production, self-correction, and coordination. In: G. Kempen (ed.) *Natural language generation: new results in Artificial Intelligence, Psychology, and Linguistics.* Dordrecht/Boston: Kluwer Academic Publishers, 1987.

Joshi, A.K. (1987) Tree Adjoining Grammar — Relevance to generation. In: G. Kempen (ed.) *Natural language generation: new results in Artificial Intelligence, Psychology, and Linguistics.* Dordrecht/Boston: Kluwer Academic Publishers, 1987.

Kay, M. (1985) Parsing in functional unification grammar. In: D.R. Dowty, L. Karttunen & A.M. Zwicky (eds.) *Natural language parsing.* Cambridge: Cambridge University Press, 1985.

Kempen, G. (1978) Sentence construction by a psychologically plausible formulator. In: R. Campbell & P. Smith, (eds.) *Recent advances in the psychology of language (Vol. 2: formal and experimental approaches).* New York: Plenum Press.

Kempen, G. & Hoenkamp, E. (1987) An incremental procedural grammar for sentence formulation. *Cognitive Science, 11(2).*

Kempen, G. & P. Huijbers (1983) The lexicalization process in sentence production and naming: indirect election of words. *Cognition, 14,* 185-209.

McDonald, D. & Pustejovsky, J. (1985) TAG'S as a grammatical formalism for generation. In: *Proceedings of the 23rd annual meeting of the Association for Computational Linguistics.* Chicago, Illinois.

Pijls, J. & G. Kempen (1986) Een psycholinguistisch model voor grammatische samentrekking. *DeNieuwe Taalgids, 79,* 217-234. (English translation available: A psycholinguistic model of grammatical conjunction reduction.)

Van Wijk, C. & Kempen, G. (1987) A dual system for producing self-repairs in spontaneous speech: evidence from experimentally elicited corrections. *Cognitive Psychology, 19(2).*