VOL. 2 NO. 3 CC-AI

IPG: A COGNITIVE SCIENCE APPROACH TO SENTENCE GENERATION+

Michele Drolet
Applied Epistemology Lab
University of Ghent
Blandijnberg 2
B-9000 Ghent, Belgium

Gerard Kempen
Psychological Laboratory
University of Nijmegen
Postbus 9104
6500 HE Nijmegen, The Netherlands

ABSTRACT
This article describes a language production theory called Incremental Procedural Grammar. IPG exemplifies the AI paradigm as applied within a psycho-linguistic context.

1. Introduction: The AI Paradigm in Cognitive Psychology

The ideas and techniques made possible by the building of artificial cognitive systems have strongly influenced the formation of theories about natural (human) cognitive systems, giving them a new look. One can speak of a cognitive revolution in the human sciences. At the same time a stream of ideas moves in the opposite direction. Psychological and linguistic knowledge serve the informatici who are building those cognitive systems called expert systems and dialogue systems.

Cognitive science concerns itself with cognitive systems (knowledge representation and manipulation). Knowledge here means information couched within a system of richly structured, heterogeneous objects. These objects are elements of knowledge or concepts which interrelated in complex ways. Investigations in cognitive science consist largely in the mapping of the contents of human knowledge. Some examples:
- language behaviour (linguistic and psycholinguistic);
- medical diagnoses (expert systems);
- naming of colours, plants etc. in diverse cultures (anthropology);
- diagnosis of systematic mistakes in calculating by children learning arithmetic;
- understanding temporal and spatial concepts (diverse scientific areas).

Mapping a knowledge domain in terms of its objects and their interrelations is usually termed the representation of that domain. Knowledge here is to be differentiated from the information found in databases, which is understood as being a collection of objects with a relatively simple, homogeneous structure.

A cognitive system must be able efficiently to handle, to manipulate, the knowledge bases available to it; it must have the means for interpreting incoming data in order to reason with knowledge already present; it must be able to communicate knowledge outward.

Some investigators in cognitive psychology spend much of their time attempting to write computer programs which simulate human behaviour. This can be termed the synthetic style of theory formation. The research method of others is more analytical. On the basis of data and observations a list is made of those independent variables and their interrelations which determine behaviour. The AI paradigm is a kind of mix of the synthetic and the analytic styles. Empirical methods complement the use of AI techniques in developing cognitive models as in Figure 1.

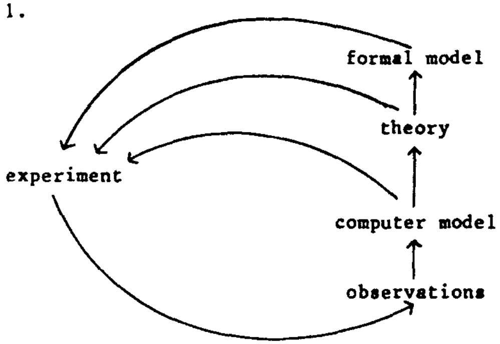


Figure 1: The AI Paradigm

A computer model is designed and implemented, a description is made of the model, i.e. a set of propositions that specify the working of the model (and if the theory holds, the working of the cognitive system being investigated). The description is then converted into some logical formalism for the sake of explicitness and precision (formal model). Hypotheses which are derived from the formal model can be empirically tested. Test results form new observations which may or may not fit into the theory. The computer model and the theory stages can also suggest hypotheses for experimental testing.

The AI paradigm, including as it does the building of a computer model, will advance the quality and the tempo of theory formation. Computer simulations enhance the exchange among large numbers of richly structured and heterogeneous components and make possible the spotting of inconsistencies and incomplete lines of reasoning. In recent years the AI paradigm has been successfully applied to psychology of language production.

2. Speaking as a Cognitive Process

Speaking is a cognitive process which entails the articulated expressing of observations, thoughts and feelings. It consists of three groups of processes which are responsible respectively for the conceptual and semantic content, the syntactic-morphological form, and the phonetic sound of language utterances. Kempen (1977a, 1977b) has termed these processes conceptualization, formulation, and articulation.

2.1 The Classical Concept

Thirty years ago, after centuries-long speculation by philosophers, rhetoricians, linguists, psychologists and neurologists, the idea of grammar-as-machine was given form in Noam Chomsky's Syntactic Structure (1957). Chomsky's work and the Transformational Generative Grammars (TGG's) which followed were based on what can be termed the classical view of the language production process:

- Speaking is considered as a sequentially organized process, i.e. first the meaning content is completely specified, then the content is converted into a linguistic structure, followed by phonetic realization (pronunciation).
- Sentence building is a centralized process, i.e. language utterances are produced by a single central processor which has full oversight over the building process and is in full command of all steps to be taken.
- The sentence-building process is syntactically guided. The rules of grammar are supposed to work from top to bottom. They decide the form for the sentence before all necessary words are fitted in. Word choice therefore is made the stepchild in the process.
- Syntactic rules will produce complete sentences. Applying the rules readily makes for a language utterance, delivered as a whole, which precisely expresses the intention of the speaker. No account is taken of the possibility that the speaker may, during the formulating process, add new elements to his intention, nor is it possible to generate an incomplete structure which on the basis of later-added meaning-content can be continued.

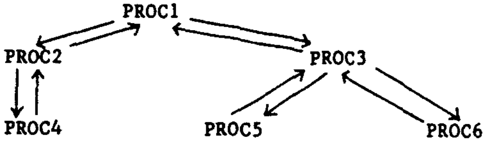
2.2 The Modern Concept

The modern view of speech production pretty well does away with the classical assumption mentioned above.

- Conceptualizing, formulating and articulating are running as parallel subprocesses. Most speakers have experienced situations in which they have begun to speak at a moment when the intended meaning of their utterance was yet vague in their minds. This implies that conceptualizing and articulating at least are running in parallel (Kempen 1977a, 1977b). A fragment of the conceptual content which the conceptualizer has delivered is handled without delay by the

formulator. This tries to transpose the fragmentary meaning into a fragmentary utterance which fits well with the foregoing fragments. If this works, then the articulator can carry on with pronouncing the new fragment. In the meantime the conceptualizer and the formulator continue on their work with new conceptual and syntactical fragments. This manner of speech production is termed incremental.
- Sentence building as a decentralized process. In an extensive empirical study of sentence production, Kempen and Huijbers (1983) could verify the hypothesis that the speaker is simultaneously busy with the building of several sentence parts. If this is indeed so, i.e. if sentence parts are built in parallel or simultaneously, even those far apart, then this undermines the theory of the central processor, since by definition it can be busy on only one sentence part at a time and must plan in which sequence the different sentence parts will be worked on. There is an explanation which suits the conclusion of simultaneity. It is necessary that tree diagrams showing the structure of sentences are interpreted in an unorthodox manner.

Computer scientists use tree diagrams to show the flow of control among procedures in a computer program. For example, procedure PROC1 has two subroutines PROC2 and PROC3 (see below). PROC2 calls on PROC4 and PROC3 calls on PROC5 and PROC6. Higher procedures trigger the working of lower ones, which when finished return control to the higher one.



Now suppose that the subprocedures work independently, e.g. that PROC3 in fulfilling its task is not dependent on the fulfilling by PROC2 of its task, and vice versa. Suppose also that subprocedures are triggered simultaneously by the higher procedure. This is the heart of the sentence construction system developed by Kempen and Hoenkamp (1984) under the name of Incremental Procedural Grammar (IPG). Figure 2 shows the hierarchy of higher and lower procedures which taken together deliver the following sentence:

Tonnie wil een cake bakken.
Tony wants to bake a cake.

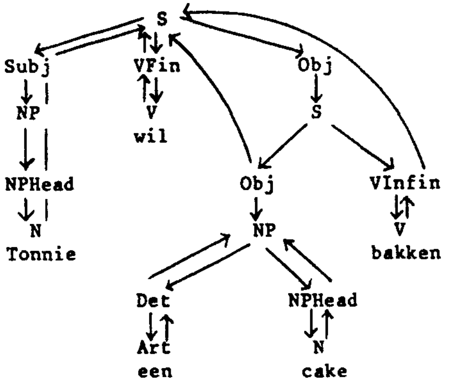


Figure 2: Hierarchy of syntactic procedures according to Kempen and Hoenkamp's IPG (1984). Note that control is not always returned to the calling procedure.

The modern architecture of speaking sketched here poses the experimental psycho-linguist with a difficult problem. The fact that during speaking many subprocesses are running in parallel and occur simultaneously makes it unfeasible to use the usual experimental techniques to measure separate process durations. For example, the cognitive psychologist who wants to test a language production model experimentally nearly loses his most important instrument, viz. the measurement of reaction times.

3. Incremental Procedural Grammar

The cognitive processes underlying sentence production are categorized under the headings of content, form and sound. One group of activities is concerned with planning the conceptual (semantic) content for language utterances. To-be-verbalized conceptual structures are selected in such a way as to be comprehensible. A conceptual structure is linearized by splitting it up into a sequence of messages expressible in a complete or partial sentence. This is conceptualizing. A second group of processes takes care of translating meaning content into sentence form, i.e. formulation. Finally, syntactic and morphological structures built by the formulator system are handed over to the mechanisms of speech for overt articulation (Fromkin, 1971; Kempen, 1977; Levelt, 1981).

We are concerned only with sentence formulation. We suppose a sentence construction device, termed Incremental Procedural Grammar (IPG) which aims at both psychological and linguistic plausibility. By psychological plausibility we mean that the device may be said to simulate human sentence production. The goal of linguistic plausibility implies that we try to incorporate into the device grammatical (syntactic, lexical, morphological) rules which a linguist would not qualify as ad hoc. In particular the device should incorporate an optimal solution to what has become one of the central issues in the theory of syntax: conditions or constraints on the application of rules.

3.1 Psychological Constraints

One property of the human sentence production system is its high level of output fluency. The primary factor conducive to fluency derives from the temporal alignment of the three subprocesses of speaking: conceptualizing, formulating and articulating. The traditional view, implicitly held by many students of sentence production, is that they are ordered strictly serially in time. First the conceptual content, next the syntactic structure for the whole utterance, and finally the phonetic realization of the structure. This serial model is empirically wrong (cf. Goldman-Eisler, 1968) and contradicted by the following introspective observation.

Speakers can initiate overt speech production after having worked out only a fragment of the conceptual content of the resulting utterance. They can also take up the thread of a broken-off sentence spoken by someone else and bring it to a syntactically impeccable end. We hold that the three subprocesses run in parallel. Sentence production occurs in fragments, and the order of conceptual fragments does not always correspond to the order of utterance fragments. We make the assumption that the conceptualizer system has no syntactic knowledge. The order in which it delivers its conceptual fragments will in principle be uncorrelated with the order of the corresponding utterance fragments in the spoken sentence. In reality the correlation will be positive, however, because the formulator will try to match them.

The mode of sentence production intended here we shall term incremental or piecemeal. Its usefulness undoubtedly relates to the efficient management it enables of the processing capacities of working memory and other mental machinery involved in formulating and articulating.

This analysis imposes important constraints on the shape of possible mechanisms for building syntactic structures. (See also Kempen & Hoenkamp, 1982). Let us start out from the customary assumption that syntactic structures can be represented by tree-shaped diagrams where nodes stand for constituents. The first constraint derives from the fact that conceptual structures serve as input to the tree formation process. Much attention has been given by linguists to the problem of mapping from syntactic structures into logical form. Conversely, mapping from logical into sentence form receives little attention.

The same happens in the field of artificial intelligence wherein language parsing and understanding are intensely studied yet language generation has received systematic interest only in the last few years (Mann, 1982). The approach we have taken consists in designing a tree-formation module which is sensitive to

- properties of the input conceptual structure representing the to-be-expressed meaning, and
- properties of the lexical items rendering this meaning.

We have concluded that the tree formation component is both conceptually and lexically guided.

Whether the IPG model leads to parsable and learnable grammars remains to be investigated.

We have assumed that the order of conceptual fragments delivered by the conceptualizer does not depend on the order of the corresponding syntactic fragments. We conclude that in an incremental sentence formulator it is desirable to have separate components for tree formation (or rather mobile formation) and for word order.

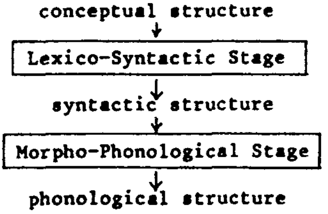
The last constraint is correct word order and/or correct morphological case. An obvious possibility is to introduce functional notions, committing ourselves to syntactic structures similar to Functional Grammar (Dik, 1978), Lexical-Functional Grammar (Bresnan, 1982) and Relational Grammar (Cole & Sadock, 1977).

3.2 Sentence Formulation in Two Stages

Garrett (1975, 1980) has developed a two-stage model of the sentence formulation process. Garret hypothesized that word exchanges "represent interactions of elements at a level of processing for which functional relations are the determinant of 'computational simultaneity'", whereas combined-form exchanges "represent interactions at a level of processing for which the serial order of the elements of an intended utterance is the determinant of computational simultaneity" (1975, p. 154).

Next was the postulating of two successive processing stages, called Functional and Positional respectively. During the first stage, the syntactic skeleton for an utterance is constructed specifying hierarchical and functional relationships among constituents. The syntactic skeleton does not contain any closed-class lexical material (function words, inflectional morphemes) and word order is open. The Functional Stage works on all constituents more or less simultaneously. The Positional Stage assigns the constituents a left-to-right order and enriches them with closed-class items, traversing the sequence of constituents from left to right.

In the IPG model we have adopted the essentials of Garrett's proposal. The only deviation concerns the stage which is responsible for inserting function words (i.e. those closed-class items which have word status) and for computing word order. We have allotted these (syntactically interrelated) tasks to the first, functional stage rather than to the second, positional stage. Our reason derives, among other things, from the observation that exchanged words often carry along dependent function words.



The foregoing summarizes the resulting make-up of the sentence formulation process, using our own terminology.

3.3 Rules and Mechanisms of Incremental Procedural Grammar

What follows is a description of the basic machinery employed by IPG in constructing sentences that express a speaker's intention. The lexico-syntactic stage gets most attention; the morpho-phonological stage is briefly discussed at the end.(1)

3.3.1 Preliminaries. In the linguistic and psycholinguistic literature it is commonly agreed that the notions of "syntax" and "syntactic processor" should be kept carefully apart. The difference is usually construed as an instance of the prototypical "database" versus "processor" distinction. The database contains rules of syntax which the syntactic processor can access and utilize for the purpose of computing correct sentence forms. The distinction has been invoked in attempts to explain why linguistic operations as defined in existing grammar types have been unsuccessful in accounting for language performance data. It encourages linguists to claim that their grammatical models only concern the database (knowledge of the language). Psychologists can use it as an excuse to concentrate on processing issues and lose interest in grammar. The drawback is that we are left with two disparate partial theories of the human language faculty the relationship of which is not easy to understand.

We take a different perspective. Our model integrates assumptions about data (rules of syntax) with assumptions about the processor manipulating the data. This combined strategy gives the advantage of making it possible to account for linguistic phenomena not in terms of grammar rules, but rather in terms of the structure and functioning of the syntactic processor. Any model which articulates - preferably empirically grounded - assumptions about both the format of grammar rules and the structure and functioning of the syntactic processor we call a procedural grammar.

3.3.2 Syntactic Procedures. Traditional models contain a centrally controlled processor which grows syntactic trees in a depth-first, left-to-right manner, at every node consulting the rules of the database. However this processing schedule entails temporal properties which are at odds with the speech error phenomena discovered by Garrett (1975). He explains word exchanges as being exemplified in terms of computational simultaneity, e.g. between direct and indirect object phrases, or between the verbs in two successive coordinate clauses. Production models which operate left-to-right certainly do not process such constituents simultaneously since the interchanged words may be at considerable distance from each other in the utterance. Another model, operating breadth-first and left-to-right probably fares somewhat better (see Kempen, 1978), but the ultimate solution clearly requires machinery for growing branches of a syntactic tree in parallel.

The basic step towards a mechanism for parallel branch construction is to view symbols such as NP, N, SUBJECT, OBJECT etc. not as passive structural elements but as active procedures or modules. Each procedure is an expert specialized in assembling one type of syntactic constituent. Like procedures or routines in ordinary computer programs, syntactic procedures are permitted to call on each other as subprocedures (subroutines). Procedure S, for instance, may decide to delegate portions of its sentence formation job to SUBJECT and OBJECT as subprocedures. OBJECT need not necessarily wait for SUBJECT to finish: they can get started simultaneously and run in parallel. They are free to call further subprocedures, a typical candidate being NP. Thus a hierarchy of procedure calls arises which is conveniently (and conventionally) depicted as a tree.

Before explaining what syntactic procedures do we must distinguish between two groups: categorial procedures (CPROCs) and functional procedures (FPROCs). CPROCs are capable of building structures of various syntactic shapes (NP, S, PP etc.); FPROCs take care of the grammatical (functional) relations between such structures (e.g. subject, object, modifier).

Below we list the most important procedures along with indications of the constituents they deliver.

Categorial Procedures (CPROCs)

S	clause
NP	noun phrase
PP	prepositional phrase
AP	adjectival or adverbial phrases
V	main verb
Aux	auxiliary verb
N	noun
A	adjective or adverb
P	preposition
Art	article
Conj	subordinating conjunction

Functional Procedures (FPROCs)

VFin	finite verb
VInfin	infinitive verb
Subj	subject
Obj	object
IObj	indirect object
SMod	sentence modifier
Comp	complementizer
NPHead	head of noun phrase
NMod	noun phrase modifier
Det	determiner
PPHead	head of prepositional phrase
PObj	prepositional object
PMod	prepositional phrase modifier
APHead	head of adjectival or adverbial phrase
AMod	modifier in adjectival or adverbial phrase

Categorial procedures come in two varieties: phrasal CPROCs and lexical CPROCs. The latter correspond to the traditional parts of speech (V, Aux, N, A etc.), the former to major phrase types as commonly distinguished in current linguistic practice: S, NP, PP and AP. Note below how the functional and categorial procedures can be grouped into four non-overlapping families (phrase types). The rows contain listings of (a) phrasal CPROCs, (b) functional procedures and (c) lexical CPROCs.

	clauses	noun phrases	prepositional phrases	adjectival or adverbial phrases
(a)	S	NP	PP	AP
(b)	Subj, Obj, VFin, VInfin, IObj, SMod, Comp	NPHead, NMod, Det	PPHead, PMod, PObj	APHead, AMod
(c)	V, Aux, Conj	N, Art	P	A

3.3.3 Lexicalization. Some of the characteristics of the lexicalization system employed by speakers in naming and sentence production are:

- Words belonging to an overt naming or sentence production response come about as the resultants of two lexical selection processes connected in series. The *first one yields abstract* pre-phonological items (L1-items or lemmata), the second one adds their phonological shapes (L2-items or lexemes) (see Kempen & Huijbers, 1983).
- The selection of several L1-items for a multi-word utterance, sentential or otherwise, can take place simultaneously.
- A monitoring process watches the output of L1-lexicalization to check if it is in keeping with prevailing constraints upon utterance format. Time taken for monitoring depends on the probability of erroneous outputs from L1-lexicalization, the seriousness of the consequences of overt errors etc.
- Retrieval of that L2-item which corresponds with a given L1-item waits until the L1-item has been checked by the monitor and all other L1-items needed for the utterance under construction have become available.

This set of operating characteristics differs from Seymour's (1979) model of object naming, wherein the processing stage is devoted to the elaboration of a perceptual-semantic code and immediately followed by the retrieval of a phonologically specified lexical item. We prefer a model which assumes parallelism of Seymour's perceptual-semantic coding and our L1-lexicalization. The lexical processing units are able to watch and respond to the evolving perceptual-semantic code while it is still being elaborated. The new assumption we are forced to make is that there are lexical processing units corresponding to our pre-phonological L1-items. One must be prepared to redefine the perceptual-semantic stage as a combination of perceptual-semantic coding and retrieval of pre-phonological lexical items.

As for the conceptual structures serving as input to IPG's tree formation component, we use an informal case-frame notation similar to what one,tends to find in the literature on semantic representation. Such structures contain slots or regions the contents of which are accessible through path functions. We assume a lexicalization system the task of which consists in inspecting conceptual structures (often using path functions) and looking up in the mental lexicon words or expressions rendering the speaker's intention. It is the lexicalizer which starts up the tree formation process. After that, the conceptual structures only play a minor role, namely, when it comes to inflectional computations and to the insertion of function words.

The standard format of a syntactic procedure call is

PROC(cp, <synspec>)

where	PROC	is the name of a categorial or functional procedure;
	cp	("conceptual pointer") is a variable or an expression evaluating to a conceptual structure; and
	<synspec>	("syntactic specification") is a list of zero or more calls to special functions which influence the shape of the constituent that PROC will build.

The first actions taken by a procedure are those of lexicalizing. The retrieved lexical entries are procedural in nature, i.e. they consist of a list of one or more procedure calls, and are denoted by the term lemma. The second argument to procedure calls in lexical entries is a synspec list. Functions there take as their arguments pointers to lexemes. Lexemes are lexical entries which specify phonological shapes for words.

We define successful lexicalization as the retrieval of exactly one lemma covering at least part of the to-be-expressed meaning. (When fewer or more lemmata turn up, hesitations or speech errors such as word-blending might ensue.) Any non-covered fragments of the conceptual structure are assigned to modifier procedures. To this purpose the four phrasal CPROCs S, NP, PP and AP have at their disposal the procedures SMod, NMod, PMod and AMod respectively.

3.3.4 Appointment rules and functorization rules. The construction of procedure call hierarchies is governed by a set of appointment rules. They specify the possible shapes of such hierarchies by telling each procedure call contained in a retrieved lemma which role it is going to play within the context of the lexicalizing procedure. The general format of appointment rules is the following:

PROC1, PROC2, <cond1, cond2, ..., condn> ---> PROC3>>PROC2
(The symbol >> means "is parent of".)

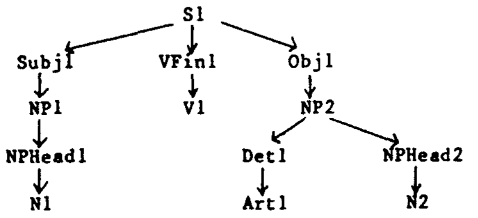
An important further issue is how function words (articles, prepositions, auxiliaries etc.) come into play. Their presence in an utterance is chiefly motivated on syntactic grounds, so they cannot be supposed to originate simply from lexicalization. The same conclusion follows from the well-known linguistic fact that function words are often in complementary distribution with inflections. For instance, in English as well as in Dutch, the present and past tenses of verbs are indicated by inflectional morphemes, whereas the future requires an auxiliary, a morpheme with word status. A convenient term covering both groups of syntactic morphemes is functor. We propose the term functorization to denote the process of inserting functors.

Functorization is best characterized as refining the set of procedure calls contained by a lemma. This may happen in two ways, corresponding to the distinction between inflections and function words. The refinement either affects the synspec list of a procedure call by inserting a new function there, or it supplements the current set of subprocedure calls with an additional member. In the case of the former, the synspec function will influence the inflectional shape of the resulting constituent; in the latter, a separate function word will emerge.

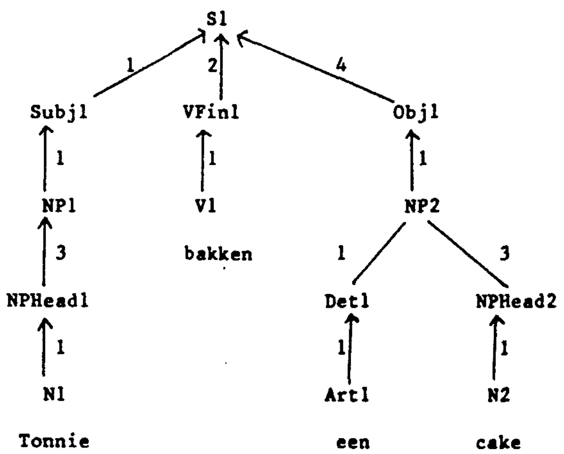
Functorization must take place prior to application of any appointment rules because it sometimes leads to additional subprocedure calls which need to be assigned a role within the lexicalizing procedure.

Although functorization clearly is a different kind of process than lexicalization, the division of labour between them - activation of function words and of content words respectively - is less clear. Both English and Dutch have many words which according to their grammatical class are to be regarded as function words but the meaning of which is so salient that they could justifiably be labelled content word. An example is the preposition "without" ("zonder" in Dutch). The converse case occurs as well: words the grammamtical class of which grants them the status of content word, although they are interchangeable with a function word in many syntactic contexts, e.g. the Dutch adjective "zeker" (English: "certain"). Such observations on the vagueness of the boundary between function and content words lead to devising normal lexical entries (i.e. lemmata) for prepositions like "without" on the one hand and functorization rules which lead to inserting adjectives like "certain" on the other. The proposed mixed treatment of prepositions corresponds to the distinction between those which are clitics (of, by, on, in) and those which aren't (without, under, after).

3.3.5 Combining and communicating subtrees. Apart from assembling a list of subprocedure calls and putting all of them to work simultaneously, a syntactic procedure also has the duty of processing the subtrees they return as their values. Top procedure S1 in the example receives values representing the subject, finite verb, and object constituents.



How does S1 combine these subtrees into a single grammatical clause? A procedure creates a data structure, called a holder, containing a sequence of numbered positions P1, P2, ... Pn. Each of these slots can serve as a receptacle for subtrees delivered by a subprocedure. Most types of holder have just one slot. Only holders created by procedures S, NP, PP and AP (the four phrasal CPROCs) contain more than one slot, namely 6, 4, and 3 and 2 respectively. Upon receipt of a value (subtree) computed by one of its daughters, a procedure deposits it in a holder slot. (This operation is the IPG version of what is usually called node attachment.) These slots are chosen on the basis of a set of Word Order Rules which we explain now in terms of the value return hierarchy shown hereafter:



The upward arrows denote the operation of a returning value. Their numerical labels refer to slots.

In the foregoing we have taken for granted that the output value delivered by a procedure consists of the holder created by that procedure together with its contents. It is the lowest (innermost, deepest) subprocedure which is first to deliver its output value. In our procedure call hierarchies this is always a lexical procedure. It delivers its one-slot holder after filling it with a pointer to a lexeme. The destination selected by all lexical procedures is the parent. Actually, all categorial procedures return their value to their parent. This is not true of functional procedures.

We can now explain the IPG equivalent of movement transformations. This is a mechanism which causes procedure call hierarchies to build differently shaped value return hierarchies. The resulting syntactic trees are less deep than the procedure call hierarchies which put them together. The mechanism is essentially a set of rules whereby FPROCs choose a destination other than their parent (usually located higher up in the procedure call hierarchy). When computing destinations for their output values, FPROCs utilize the following system for referencing holders created by other procedures.

Immediately upon being called, syntactic procedures (both functional and categorial) declare a variable the name of which consists of the character string "var" prefixed with the procedure's own name. For instance, the variables declared by S, NP and V are s-var, np-var and v-var respectively. The value assigned to such a variable is the name of an instantiated procedure (e.g. S1, NP2). The destination rules used by FPROCs are phrased in terms of such variables. For example, Obj seeks s-var as its destination. This means it climbs the procedure call hierarchy until it hits upon an occurrence of s-var. Obj then ascertains the name of the instantiated procedure bound to that variable and sends its value to that address.

Below is a summary of the destination rules discussed thus far.

SOURCE	DESTINATION
CPROC	Parent procedure
FPROC	Instantiated procedure bound to:
of S-family:	s-var
of NP-family:	np-var
of PP-family:	pp-var
of AP-family:	ap-var

Under the influence of lexical information, s-var is sometimes given a different value than the name of the S instantiation which declared the variable.

The main activities performed by syntactic procedures are listed here:

A. Declare and initialize variables.
B. Create a holder.
C. Evaluate cp and synspec arguments.
D. Lexicalize cp.
E. Apply functorization rules.
F. Apply appointment rules.
G. Run subprocedures in parallel.
H. Apply word order rules to received subtrees.
I. Apply destination rules.
J. Return holder with contents to destination.
K. Exit.

Terminal (lexical) procedures corresponding to single words skip steps D through I.

3.3.6 The morpho-phonological stage. The output value computed by a terminal procedure contains a lexical pointer which serves to locate a lexeme in the mental lexicon. A lexeme is a phonological specification of a to-be-uttered word. The final shape of the word awaits the application of inflection rules and various sound rules which belong to the domain of articulation. The morpho-phonological stage converts syntactic trees delivered by the lexico-syntactic stage (more precisely, trees returned by the top member of a procedure call hierarchy) into phonological structures.

Syntactic procedures compute all information needed by rules of inflection. Some of the relevant computations are within the context of functorization rules.

3.3.7 Coordinate structures. We shall very briefly outline our treatment of coordination and two related phenomena: conjunction reduction and gapping. One of our assumptions about the shape of conceptual structures underlying coordinate structures is that logical conjunction is expressed by the presence of AND, OR, BUT etc. in between conjuncts, i.e. conjoined concepts or conceptual structures. Many concepts mentioned in conjoined structures are repetitions of a concept which figured in an earlier conjunct.

What happens when the cp argument of a syntactic procedure is a conjunction of two or more conceptual structures? The basic idea behind the IPG approach to coordination is that of iteration. Above we summarized as a sequence of steps the activities of syntactic procedures. That sequence is repeated below, however we have added provisions for dealing with conjoined conceptual structures as cp value. Note that step D attempts to lexicalize the various conjuncts of a cp one by one, and that step K instructs the procedure to resume step D so long as any conjuncts must be lexicalized. Thus an iterative loop is created spanning steps D through K. For each conjunct, the loop is traversed exactly once.

A. Declare and initialize variables.
B. Create a holder.
C. Evaluate cp and synspec arguments.
D. Lexicalize (the next conjunct of) cp.
E. Apply functorization rules.
F. Apply appointment rules.
G. Run subprocedures in parallel.
H. Apply word order rules to received subtrees.
I. Apply destination rules.
J. Return holder with contents to destination.
K. Exit if cp has been lexicalized exhaustively; otherwise go to D.

3.4 Incremental Sentence Production

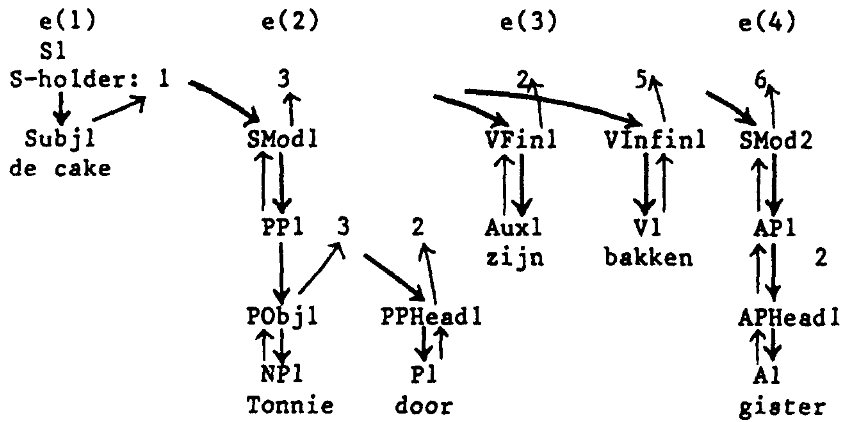
We shall now concentrate on psychological issues. How can the formulator build sentences which dovetail into the evolving conceptual structures delivered by the conceptualizer? The conceptualizer delivers the conceptual structure for a sentence as a cumulative sequence of expansions e(1), e(2), ... e(n). Each expansion e(1) is a proper subset of its successor e(i+1). The computational principle we employ is iteration. Into every syntactic procedure we build an iterative loop spanning steps G through K, much like our treatment of coordination. The incrementation loop is nested within the coordination loop. During each new iteration of the loop, the next expansion in the sequence is processed. One integrated utterance should result which is syntactically coherent as a whole. The syntactic shape of the integrated utterance is dependent on the order in which the various parts of the conceptual structure are expanded, i.e. from their conceptualization order. We have assumed a "first in, first out" schedule which, within the limits of grammaticality, attempts to assign to new parts of the utterance a position as much to the left as is possible. We shall clarify the incrementation in terms of the following sentence.

De cake ... is door Tonnie gebakken ... gister.
The cake has-been by Tony baked yesterday.

Imagine that the conceptualizer delivers the meaning underlying this sentence as a sequence of four expansions:

e(1) the cake
e(2) the cake, Tony
e(3) Tony having baked the cake
e(4) Tony having baked the cake yesterday

Figure 3 shows how the hierarchy of procedure calls grows in response to the meaning expansions. Procedure S1 goes through four iterations; the corresponding lists of subprocedure calls are given in Table 1.



De cake...................is door Tonnie gebakken.gister

Figure 3: How the hierarchy of procedure calls grows in response to meaning expansions. Procedure S1 goes through four iterations; the corresponding lists of subprocedure calls are given in Table 1.

Table 1
Lists of subprocedure calls composed during the incremental production of the sentence "Gister...bakte Tonnie...een cake (Yesterday Tony baked a cake). Cp1, cp2 and cp3 refer to the meanings underlying "cake", "Tony" and "yesterday" respectively. Arrow --> indicates which procedures are actually run. See also Figure 3.

```
e(1)  old:     ---

      new:  -->Subj(cp1, <>)

e(2)  old:     Subj(cp1, <>)

      new:  -->SMod(cp2, <>)

e(3) after first lexicalization attempt:

      old:    +Subj(Path(actor...), <>)

      new:     VFin(nil, <V(nil, <Lex(bakken)>)>)

               Obj(Path(product...), <>)

     after second lexicalization attempt:

      old:     Subj(Path(product...), <>)

           -->+SMod(Path(actor...), <P(nil, <Lex(door)>)>)

      new:  -->VFin(nil, <Aux(nil, <Lex(worden)>)>)

            -->VInfin(nil, <V(nil, <Lex(gebakken)>)>)

e(4)  old:     see the list after second lexicalization of e(3)

      new:  -->SMod(cp3, <>)
```

Iteration 1. After having lexicalized and applied appointment rules to noun lemma cake, S1 assigns it the role of syntactic subject. Subj deposits its value into slot P1 of S-holder and exits. The contents of P1 are passed down to the morpho-phonological stage and pronounced as de cake.

Iteration 2. The lexicalization within S1 selects the noun cake and deals with the meaning increment by handing it over to SMod, where appointment rules force the noun lemma Tonnie into the role of prepositional object, with the preposition left undecided. The new contents of S-holder's P3 slot cannot be processed by the morpho-phonological stage yet.

Iteration 3. Lexicalization within S1 during its third iteration yields the active verb lemma bakken. However the path function associated with the Subj call in this lemma evaluates to Tony, i.e. the content of the actor region of e(2), and is not coreferential with cp1 (see Table 1). Here a compatibility check must be carried out. The notion of compatibility is defined thusly:

Procedure call PROC2(cp2, <synspec2>) is compatible with
procedure call PROC1(cp1, <synspec1>) iff
  (a) PROC2 and PROC1 are identical procedure names, and
  (b) cp2 and synspec2 are identical to or expansions of
      cp1 and synspec1 respectively.
(Anything non-Nil is considered an expansion of Nil.)

The compatibility check discloses that the new Subj call is incompatible with the call to Subj in iteration 2 (+-sign in Table 1). Another problem concerns the SMod call which has no counterpart in the current list. A second consultation of the lexicon yields the passive lemma bakken which is less incompatible:

```
VInfin(nil, <V(nil, <Lex(gebakken)>)>)
Aux(nil, <Lex(worden)>)
Sujb(Path(...), < >)
SMod(Path(...), <P(nil, <Lex(door)>)>)
```

The path functions associated with Subj and SMod single out the product and the actor of the baking event respectively. These happen to coincide with the contents expressed by Subj and SMod during the first two iterations. The synspec arguments presenting no compatibility problems, the lemma is accepted.

Iteration 4. S1 adds a second call to SMod with the new temporal information as to-be-expressed meaning content. This SMod retrieves the adverb gister (*English: yesterday*) and attempts to deposit its output value into P3 of S-holder. After e(3) the morpho-phonological stage has got as far as position P5 of S-holder: the VInfin participle gebakken has already been pronounced. Rather than dropping the adverb at P3, SMod now selects P6 - a possibility having low priority. Position P6 is still open, that is, no output values deposited there have yet been processed by the morpho-phonological stage. We assume that *syntactic procedures try to avoid incursions into positions* within a holder which have already undergone morpho-phonological processing.

A problem with incremental sentence production is that the slots of holders are not filled in an orderly left-to-right fashion. Moreover, slots often remain empty during the construction of a sentence. A device is needed for marking the slots which are going to be occupied by obligatory constituents. One could launch obligatory procedures as soon as they are dictated by the syntactic constellation. We propose the convention that a procedure which is running with Nil arguments also delivers Nil as its output value and deposits it at the standard destination. This symbol is interpreted by the morpho-phonological stage as a halt signal. Later on, such a dummy obligatory procedure will be replaced by a new instantiation with adequate cp and synspec arguments. It computes a non-empty holder as its output value overwriting the Nil symbol. This is a tentative solution and ad hoc. It must be said though that forward syntactic inferencing is probably of greater variety than this.

3.5 Repairs and Ellipses

A speaker who decides to repair part of the utterance he is pronouncing often backtracks to the beginning of the last constituent, thus restoring the integrity of an interrupted syntactic unit. Levelt (1983) proposes the following well-formedness rule for repairs (here in simplified form):

A repair <A,C> is well-formed if there is a string B such that the string <AB and C> is well-formed, where B is a completion of the constituent directly dominating the last element of A.

IPG accounts for the well-formedness rule in a very straightforward manner. It assigns the duty of carrying out self-corrections to the mechanism which is also responsible for computing the shape of coordinate structures.(2)

3.6 Further Psychological Issues

3.6.1 Speech errors. A lemma exchange will indirectly cause an exchange of dependent function words as well. Exchanged lexemes on the other hand cannot carry along dependent function elements simply because their dependence is not specified at the morpho-phonological level. IPG is therefore able to handle two classes of Garrett's speech errors: word exchanges and combined-form exchanges.

3.6.2 Clitic omission in agrammatism. Kean (1977, 1979) has observed that agrammatic patients tend to leave out those words and morphemes characterized as belonging to the class of clitics, i.e. inflections, articles, pronouns, auxiliaries, subordinating conjunctions and small, monosyllabic prepositions.

Kolk, van Grunsven & Keyser (in press) argue that agrammatical speech is caused by a simplified conceptual input which is detailed enough to enable a patient to find the communicatively important content words (through lexicalization) but lacks information triggering the insertion of clitics. The necessity of maintaining the complex computational environment presupposed by correct application of functorization rules is obviated while communication loss is minimized. IPG appears compatible with both the observation and the theory.

3.6.3 Speech formulae. Fluency profits from the ease with which speakers avail themselves of all sorts of idiomatic expressions which may range over a fair number of words. It should be unproblematic for the formulator to look up and retrieve such speech formulae from the lexicon and to fit them into the grammatical structure it is working on. Lexical entries which correspond to idiomatic expressions spanning more than a single word have no special status in IPG.

3.6.4 Formulating as automatic activity. The numerous syntactic computations which are carried out during language production hardly require conscious attention on the part of the speaker. Whole sentences may spring to mind. This tallies with the idea, embodied in IPG, of sentence formulation by a team of syntactic experts rather than by a single processor. It also helps to understand that sometimes several formulations of the same conceptual structure seem to be developing simultaneously.

4. One Device for Parsing and Formulating?

Psychologists and linguists should be pleased to have one grammar which could be utilized for both sentence production and sentence perception. Theoretical proposals for a grammar which can do both jobs are conspicuously absent from the psycholinguistic literature, and discussions of the attainability of such a grammar tend to end with discouraging conclusions (Fodor, Bever & Garrett, 1974). Unificational Grammar (Kay, 1984) is the first linguistic formalism which is truly bidirectional. However, psycholinguistically desirable features are lacking in it.

Without claiming to have a workable plan, we wish to draw attention to the fact that, from the point of view of IPG, syntactic parsing (as part of the language perception process) is remarkably similar to syntactic formulating (as part of the language production process). Parsing and formulating are both lexically driven, i.e. operate on the basis of syntactic information stored with individual words of the lexicon. Both processes use that information for the purpose of constructing a syntax tree with these words as terminal elements. They both have facilities for growing syntax trees from left to right. The parser needs them for attaching new words to the current syntax tree, the formulator for computing a continuation (incremental production). The origin of the words is different of course: they stem from speech recognition in case of parsing but from lexicalization in the case of formulating. We hope that exploring these unexpected parallels will stimulate the study of both human language perception and language production, and bring us to the attractive situation of having one device which is a syntactic parser and a syntactic formulator at the same time.

+In this paper an overview of Kempen's psycho-linguistic work is presented by Drolet. The text is based primarily on Kempen, 1981, 1984 and Kempen and Hoenkamp, forthcoming.

Notes

1. In various other papers we have worked out further details on the basis of new experimental psycholinguistic evidence (Kempen & Huijbers, 1983; van Wijk & Kempen, 1984b). We have also considered the problem of how intonation contours become woven into an utterance. Van Wijk & Kempen (1984a) argue that the morpho-phonological stage has an important role to play there and describe the computational system they developed for automatically generating Dutch intonation contours for syntactic structures delivered by the lexico-syntactic stage.

2. Apparently there exist (conceptualizing & monitoring?) processes having the authority to interrupt ongoing speech production activity at any point in time (cf. van Wijk & Kempen, 1984b, for some supporting experimental evidence).

References

Bresnan, J., ed. (1982). The Mental Representation of Grammatical Relations. Cambridge: MIT Press.

Cole, P., Sadock, J., eds. (1977). "Grammatical relations". In Syntax and Semantics, Vol. 8, New York: Academic Press.

Dik, S.C. (1978). Functional Grammar, Amsterdam: North-Holland.

Fodor, J., Bever, T., Garrett, M. (1974). The Psychology of Language, New York: McGraw-Hill.

Fromkin, V. (1971). "The non-anomalous nature of anomalous utterances". In Language, 47, pp. 27-52.

Garrett, M. (1975). "The analysis of sentence production". In G. Bower, ed., The Psychology of Learning and Motivation, Vol. 9, New York: Academic Press.

Garrett, M. (1980). "Levels of processing in sentence production". In Butterworth, B., ed., Language Production (Vol. 1 Speech and Talk), New York: Academic Press.

Goldman-Eisler, F. (1968). Psycholinguistics: Experiments in Spontaneous Speech, New York: Academic Press.

Kay, M. (1984). "Functional unification grammar: A formalism for machine translation". In Proceedings of COLING84, Stanford.

Kean, M-L. (1977). "The linguistic interpretation of aphasic syndromes: Agrammatism in Broca's aphasia, an example". In Cognition, 5, pp. 9-46.

Kean, M-L. (1979). "Agrammatism: A phonological deficit?" In Cognition, 7, pp. 69-83.

Kempen, G. (1977a). "Conceptualizing and formulating in sentence production". In S. Rosenberg, ed., Sentence Production: Developments in Research and Theory, Hillsdale, N.J.: Lawrence Erlbaum Associates.

Kempen, G. (1977b). Onder Woorden Brengen. Psychologische Aspekten van Expressief Taalgebruik. (Inaugurale rede.) Groningen: Wolters-Noordhoff.

Kempen, G. (1978). "Sentence construction by a psychologically plausible formulator". In R.N. Campbell, P.T. Smith, eds., Recent Advances in the Psychology of Language. Formal and Experimental Approaches, New York: Plenum Press.

Kempen, G. (1981). "De architektuur van het spreken". In Tijdschrift voor Taal- en Tekstwetenschap, 1981, 1, pp. 110-123.

Kempen, G. (1984). "Inleiding". In Kempen, G. and Sprangers, C., eds., Kennis, Mens en Computers, Lisse: Swets & Zeiglinger.

Kempen, G., Hoenkamp, P. (1982). "Incremental sentence production: Implications for the structure of a syntactic processor". In Proceedings of the Ninth International Conference on Computational Linguistics, Prague.

Kempen, G. Hoenkamp, P. (forthcoming). "An Incremental Procedural Grammar for Sentence Formulation".

Kempen, G., Huijbers, P. (1983). "The lexicalization process in sentence production and naming: Indirect election of words". In Cognition, 14.

Kolk, H., van Grunsven, H., Keyser, A. (in press). "On parallelism in agrammatism: A case study". In M-L. Kean, ed., Agrammatism, New York: Academic Press.

Levelt, W.J.M. (1983). "Monitoring and self-repair in speech". In Cognition, 14, pp. 41-104.

Mann, W. (1982). "Text generation". In American Journal of Computational Linguistics, 8, pp. 62-69.

Seymour, P.H.K. (1979). Human Visual Cognition, London: Collier MacMillan.

van Wijk, C., Kempen, G. (1984a). "From sentence structure to intonation contour: An algorithm for computing intonation contours on the basis of sentence accents and syntactic structure". In B.S. Mueller, ed., Sprachsynthese, Hildesheim: Olms.

van Wijk, C., Kempen, G. (1984b). "A dual system for producing self-repairs in spontaneous speech. Evidence from experimentally elicited corrections", in press.