SENTENCE CONSTRUCTION

BY A PSYCHOLOGICALLY PLAUSIBLE FORMULATOR

GERARD KEMPEN

University of Nijmegen

Natural language production comprises a variety of processes that may be grouped under two headings. The conceptualization processes select a conceptual theme for expression. They decide which parts of the theme must be actually communicated to the hearer and which can be left unexpressed: the latter are already present in the hearer's memory or can be inferred by him from what the speaker said. And the conceptual content selected for expression must be organized into a linear sequence of messages so that each is expressible as a complete or partial sentence. The psychological mechanism that accomplishes these tasks I will call the conceptualizer. The second main mechanism is the formulator which maps each input conceptual message into a natural language utterance. Formulating consists of two main processes:

(1) lexical search, for locating and retrieving from memory language elements which express the conceptual information, and

(2) sentence construction, i.e. assembling a partial or complete sentence from language elements.

It is only the latter aspect of language production this paper is concerned with. I will try to work the various empirical data that are known about human sentence production into a blueprint of a possible sentence construction procedure. The data I have in mind are observations on speech errors (Fromkin, 1973; Garrett, 1975), hesitation and pausing phenomena,(Fodor, Bever and Garrett, 1974), experimental results on semantically constrained sentence production and reproduction (several chapters in Rosenberg, 1976),

and some general properties of the human cognitive system. I offer the blueprint as a point of departure for more detailed and more formal theorizing on human sentence production and as a source of meaningful hypotheses for psychological experimentation.

Section I reviews the main empirical facts the sentence construction system tries to encompass. Section II outlines the system itself. Section III explores some linguistic consequences of the model. Finally, Section IV contains a few detailed generation examples.

I. Empirical observations on sentence construction

I.1 Heavy reliance on multiword units

As building blocks to construct sentences from, the formulator uses not only single lexemes but also multiword units which span several lexemes and/or "slots" to be filled by l xemes. Becker (1975) argues a similar point. Left-to-right order of the elements of a multiword unit is more or less fixed. I quote the following observations in support:

(i) Phraseology linked to stereotype situations. People speaking in a stereotype situation often have available sentence schemes which they know will enable them to express what is on their mind. Especially if rapid speech is required, such schemes will actually be put to use (example radio reporters doing running commentaries of soccer matches or horse races). Although there is little empirical data to either underpin or undermine this claim, some informal observations by this author support it strongly (Kempen, 1976b).

(ii) Syntactic retrieval plans. In a series of experiments on paraphrastic reproduction of sentences, I demonstrated the existence of retrieval mechanisms which look up a specific pattern of conceptual information in memory and directly express it in the form of a specific syntactic frame. For details see Kempen (1976a,b) and Levelt & Kempen (1975).

(iii) Speech errors. Garrett (1975), in the most extensive and detailed study of speech errors to date, suggests that people sometimes use syntactic frames consisting of functors (articles, prepositions, inflection morphs, etc.) in left-to-right order, with slots to be filled by content words. Occasionally, content words are put into a wrong slot and speech errors like I'm not in the READ for MOODing and she's already TRUNKed two PACKS result (inter-changed

pieces in capital). But Garrett never observed errors like The boyING shouts disturbed us with functors interchanged.

I.2 A clause-like unit as the largest unit of planning

The units by which sentence construction proceeds vary from, say, words to clauses. That is, the segments of speech that get attached to the output string are neither very small units like single phonemes nor very large ones like convoluted sentences. (I'm only concerned with spontaneous speech production, not with writing).

The largest unit, and also a very predominant one, is often thought to be the surface clause or the phonemic clause. Boundaries between such clauses frequently attract pauses and hesitations. Boomer (1965), however found that the highest proportion of pauses occurred after the first word of a phonemic clause. Disregarding this exception, the other positions showed a gradual decrease towards the end of a clause.

Many first words of clauses must have been subordinate or coordinate conjunctions. If so, Boomer's data suggest that decisions regarding conjunctions are rather independent of decisions regarding the other lexical material (verb, noun phrases, etc.) of a phonemic clause. Two possibilities come to mind. First, the conjunction may have been selected at a very early stage of the formulation process. E.g. a speaker who wants to express a causal relation between two events may very early on decide to construct an utterance of the form "EVENT2 because EVENT1". After verbalization of EVENT2 he only needs to put the word because that is waiting in some buffer store, into the output stream. After because a pause may develop depending on how much time it takes to verbalize EVENT1. The second possibility is perhaps more interesting: the conjunction initializing a certain clause may be syntactically required by the verb of another clause. For instance, the _if in I don't know if John is in is dependent on the verb know of the main clause, and can be uttered even before the formulation process for the subordinate clause has begun. The line of reasoning of this paragraph.

(i) I can't help you because *1* I don't know if *2* John is in.

would point at positions *1* and *2* of sentence (i) as likely places for pausing. The segment between *1* and *2*, that just misses being a phonemic clause, I will call a verb dependency construction (VDC). A VDC contains a verb as the "head" or "governor" and all the phrases that are dependent on it. In sentence (i), if, don't, and I are dependents in the VDC which has know as its governor. Two other VDCs in (i) are I can't help you and John is in.

My motivation for introducing notions of syntactic dependency here is, first of all, that they nicely represent the relations between "predicting" and "predicted" sentence elements, even if the predictions surpass clause boundaries. Other empirical arguments for dependency grammar are reviewed in sections 1.3 and III.2.

Raising a clause-like unit such as the VDC to the status of the largest unit of planning is also supported by an observation on the maximum size of idiomatic expressions. Except for proverbs, there doesn't seem to exist any idiom or phraseology that is substantially longer than a single clause. All idiomatic expressions which allow for some variation (word order, slots to be filled) observe this upper boundary. Proverbs sometimes spanning several clauses, are no counterexamples since they are totally fossilized and don't need a "formulator" at all. To put it differently, non-fossilized phraseology never consists of, for instance, two successive half clauses or two complete clauses. If an expression spans more than one clause, then only one clause is variable and all the others are fossilized. A Dutch example is: "NP LACHEN als een boer die kiespijn heeft" (NP LAUGH as a farmer who has toothache). This limitation on the size of syntactic constructions suggests that the formulator never works on more than one clause at the same time.

To be sure, non-fossilized idioms sometimes do violate clause boundaries, but apparently only with conjunctions just as VDCs do. Examples are: My name is not ... or ..., NP TAKE it that —

I.3 Speech errors which are exchanges within dependency levels

One category of speech errors in the collection studied by Garrett (1975) are "word exchanges": two complete words which occupy non-adjacent positions end up at each other's place. Examples (2) through (13) are all the word exchange errors (interchanged words in capital) Garrett lists in his paper.

(2) I have to fill up the GAS with CAR.

(3) Prior to the operation they had to shave all the HEAD off my HAIR.

(A) She donated a LIBRARY to the BOOK.

(5) Older men CHOOSE to TEND younger wives.

(6) ...which was parallel TO a certain sense, IN an experience..

(7) Every time I put one of these buttons OFF, another one comes ON.

(8) She SINGS everything she WRITES.

(9) — read the newspapers, WATCH the radio, and LISTEN to T.V.

(10) Slips and kids - I've got BOTH of ENOUGH.

(11) I broke a DINGHY in the STAY yesterday.

(12) Although MURDER is a form of SUICIDE, ...

(13) I've got to go home and give my BATH a hot BACK.

In his total corpus there are 97 such cases. Garrett remarks not only that the interchanged words usually belong to the same word class but also that they "come from corresponding positions within their respective structures ... The parallelism of structure is most strikingly evident for the word exchanges that cross clause boundaries, but even the within-clause exchanges show a strong correspondence, usually involving two similarly placed words from distinct phrases. These phrases are quite often, for example, the noun phrases (NPs) of direct and indirect objects, or the NPs from a direct object and an adverbial phrase, or from successive adverbial phrases" (p.155-156).

I would like to add one further constraint that seems operative in such word exchanges. From examples (2) - (13) and some further statistics provided by Garrett one can conclude that the overwhelming majority of the exchanged words belong to the same syntactic dependency level. This is true of 8 out of the 12 examples listed above (5, 8, 10 and 12 seem to be exceptions).

These observations and those of the previous Section provide some empirical basis - although I admit it is very slender - for setting up the sentence construction process as one which roughly proceeds dependency level by dependency level.

I.4 Parts of the syntactic form of an utterance may be given before the formulation process starts.

In certain circumstances the formulator is not completely free in determining the syntactic shape of an utterance. Stylistics includes phenomena such as these: certain themes/contents and certain audiences prefer certain syntactic forms; the syntax of individual sentences is partly controlled by their position in the total text. Apparently, some mechanism prior to the formulator (perhaps the conceptualizer) biases him towards certain syntactic forms.

Another instance of limited "freedom of expression" is provided by situations of repetitive speech. A radio reporter who has to read out a series of sports results is tempted to use the same syntactic scheme for several successive scores. Elsewhere I have discussed this point in detail (Kempen, 1976b).

I.5 Preferred word orders

Recent experiments by Ertel (1976), Osgood & Bock (1976) and Jarvella (1976) have uncovered some of the rules underlying preferred or "neutral" word order in spontaneous speech. For instance, speakers have a tendency to express ego-related, vivid and salient concepts early in the sentence. Such tendencies importantly determine the selection the speaker makes from the total set of paraphrases he might use to express the content he has in mind.

In terms of standard transformational grammar, some such paraphrases have a longer transformational history than others. For instance, passive sentences are supposed to be more complex than actives, and subject complement constructions with extra-positioned (trailing) subject (It amazed me that he went) have a longer derivation than their counterparts with subjects in initial position. However, the available evidence disconfirms the hypothesis that differences in derivational complexity will show up in actual human sentence production. The experimental study of James, Thompson & Baldwin (1973) renders very implausible the hypothesis that passives are more difficult to produce than actives (except, perhaps, for length). Jarvella (1976) compares ease of production of subject complement sentences with the that-clause in leading vs. trailing positions. He concludes "there was no real indication that subject complements were effortfully postponed".

I.6 Very limited working memory

Humans have a small working memory and a huge, easily accessible and very flexible long-term memory (LTM). The opposite is true of modern computers. Large amounts of data (inputs, intermediate results of computations, etc.) can be very quickly stored without "rehearsal" and don't get lost as a function of time or of new data coming in. On the other hand, LTM lookup of some little piece of data in a large computerized data base is very cumbersome. Sentence generators built by both transformational and computational linguists tend to require a large working memory for keeping intermediate results (typically, tree-like structures). And this memory can only be cleared at a very late stage, if not after completion, of the generation process. No part of the content of the working memory may be released earlier, e.g. put into an output channel for pronunciation since there is always the chance

for a later "transformation" to be dependent on it or to change left-to-right word order.

Thus, in order to ease the burden put onto a working memory by the sentence generation process, it seems wise to first decide upon the left-to-right order of constituents so that speaking can start relatively early and need not wait till all details of the total utterances have been computed.

This line of reasoning, however, also applies to the level of the conceptualizer. Since the conceptual messages it composes have to fit in a small working memory it would be efficient if it could pass partial results down to the formulator for quick translation into natural language. Since most people would agree they often start talking before they have completely worked out what they want to communicate, I will allow for conceptual messages that are fed into the formulator in bits and pieces. And the formuIator must be enabled to start working on parts available instead of having to wait until the complete message is in.

Such a system has an interesting consequence as regards naturalness of word orders (cf. previous Section). The speech segments the various conceptual pieces translate into will show an order correlating positively with the order in which these pieces were sent out by the conceptualizer. (Syntactic constraints on word order will often prevent the correlation from being perfect). Following a suggestion by Osgood & Bock (1976), we might hypothesize that the order in which concepts are processed by the conceptualizer is determined by saliency, vividness, ego-relatedness, etc. Also, the linguistic notions of topic and comment may be related to order of conceptualization (topic first, comment later). Consequently, a formulator which is able to process fragmentary conceptual messages in their order of arrival, will spontaneously show natural or preferred word order and won't need any special machinery for computing them. In other words, the problem of natural word order should be studied at the level of the conceptualizer, not the formulator.

II. Outline of the sentence construction procedure

II.1 A constructional lexicon

In Section I.1, the importance of syntactic frames, sentence schemes, standard (canned) phrases and the like was discussed. Here I will introduce the notion of a syntactic construction - a notion that I think encompasses most multiword units occurring in natural language, and one which has proved useful in accounting for the results of some experiments on sentence (re-)production (Kempen, 1976a,b). A syntactic construction is a pair consisting

of

(1) a pattern of conceptual information, and

(2) a sequence of lexemes and/or syntactic categories.

The latter I will call a syntactic frame, the former a conceptual pattern. The syntactic frame expresses the conceptual pattern.

For example, the syntactic frame "NP1 give NP2 to NP3" expresses a specific form of transfer of possession of NP2 between NP1, the actor, and NP3, the recipient.

The "passive" syntactic frame "NP1 be given NP2 by NP3" belongs to a separate syntactic construction whose conceptual pattern is identical to that of the active give-construct ion. The idiomatic expression "NP shoot Q bucks" is a syntactic frame expressing the number of dollars NP spends. So far, the examples all have open slots (indicated by capital letters) but there are also many constructions whose syntactic frame is completely closed and allows no variation at all, not even word order permutations (like as a matter of fact, other things being equal, proverbs). Parenthetically, the examples make it clear that I use the term syntactic frame in a broader sense than Garrett (1975) who only considered sequences of functors as the body of frames (e.g. The N is V-ing; cf. Section I.l.iii).

The lexicon contains syntactic constructions as lexical entries. An individual lexeme (single word) figures as a lexical entry only if it constitutes a syntactic frame on its own. (14) gives an idea of what the syntactic frame of a VDC lexical entry looks like, in LISP notation.

```
(14)    VDC: ('(ppl {Cat: NDC; Case:  Subj} ) (leave  {Cat:
               V} ) (pp2 {Cat:  NDC; Case:  Obj;  Status:
               Opt} ))
```

It is a list containing three sublists as top-level elements. Each of the sublists is a pair whose right-hand member is a list of attribute-value pairs {between square brackets}. The latter provide syntactic information for the procedures operating on lexical entries that have been retrieved from the lexicon. These right-hand members I will call synspecs (syntactic specifications). The left-hand member of the top-level sublists is either a single lexeme or a "pointer procedure" which computes a pointer to a field of the conceptual pattern that is being translated. For instance, ppl sets up a pointer to the actor field in the conceptual pattern. It is from this field that the lexical filler for the subject slot will be derived. Likewise, the value of pp2 will be a pointer to the location the actor travels away from. {Cat: NDC; Case: Subj}

means: the lexical realization must be a Noun Dependency Construction (or NP if you wish) in subject case. { Status: Opt} in the third sublist marks this NDC as optional (leave is a middle verb).

I will now give a more formal definition of a syntactic frame. It is a list of one or more pairs of the form "(pp synspec)" or "(l synspec)", where pp is a pointer procedure (returning a pointer to a field of a conceptual pattern); synspec is a list of attribute-value pairs (marking syntactic properties that have to show up in the utterances under construction); and l is a lexeme (which can be put into the output stream after the applicable morphological rules have worked on it). Furthermore, I propose the following conventions. If the left-hand member of a top-level sublist is a lexeme and the right-hand member is a single attribute-value pair {Cat: X}, where X is any part of speech, then the sentence construction procedure will assume this lexeme can be dumped into the output stream without any modification. E.g. (because { Cat: Conj}) means that because, a conjunction, doesn't need any morphological shaping up before it is pronounced. The part of speech attributes can also be used to decide which sublist contains the governor of a construction. Each syntactic frame in the lexicon is explicitly marked as a Dependency Construction of some sort: VDC, NDC, ConjDC, etc. The governor of the frame is the sublist which contains the corresponding "Cat:" mark. (This will work only if the frame contains exactly one such sublist. Nominal compounds like apartment building or graduation day which have two nouns in them could not be handled. Since the first noun cannot be separated from the second one and is not subject to morphological changes, I propose to treat these compounds as single nouns, as is done in German and Dutch).

II.2. Sentence Assembly

II.2.1. General Overview

The formulator starts constructing an utterance with two pieces of information:

(1) a conceptual pointer, and

(2) a synspec which enumerates zero or more syntactic properties of the to-be-constructed utterance.

Empirical arguments for (2) were given in Section I.4. I will first describe the workings of the proposed formulator if it operates on complete conceptual patterns. In Section II.2.3 the extra machinery for dealing with fragmentary conceptual patterns will be outlined.

As for terminology, the two main procedures the formulator uses I will call LEX (for lexicalization) and FIN (for finalization). LEX receives as input a formula, which is a pair of the form "(p synspec)" or "(l synspec)" where p is a pointer to a conceptual pattern, l a lexeme, and synspec as defined above.

The formulator passes the input, which is a formula, on to LEX which replaces it by another formula or by a list of formulae. To this output, LEX is applied again, that is, to each of the formulae, going from left to right. The result of this "pass" or "sweep" of the lexicalization procedure is, again, a new list of formulae. The formulator continues such lexicalization sweeps until all formulae in the list have the form "(l synspec)", i.e. until they all have lexemes in them and no pointers to conceptual patterns anymore. To this list, the formulator applies FIN which computes the final form of the lexemes. The left-to-right order of lexemes in the formula list corresponds to order of words in the final utterance.

Although this is not clear from the description just given, applying LEX this way enables growing a syntactic dependency tree from top to bottom, dependency level by dependency level. Consider the dependency tree in Fig. 1 which depicts dependency relations among the words of sentence (15).

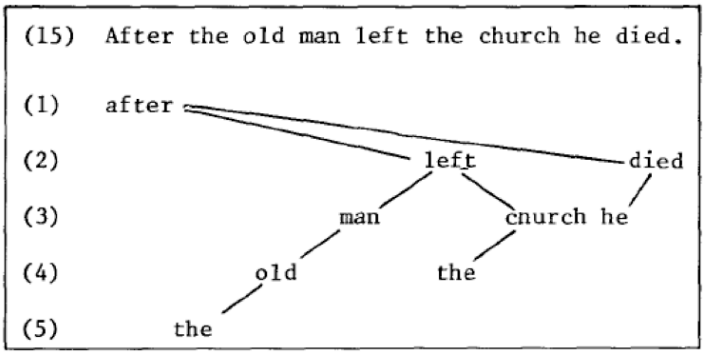

Fig. 1 Syntactic dependency tree for sentence (15).

Suppose the formulation process starts out with the formula (pi {Cat: S}. The first lexical frame LEX finds (N.B. this paper is not concerned with lexical search itself) is the ConjDC "after S S" which will replace the earlier formula:

((after {Cat: Conj}) (p2 {Cat: S}) (p3 {Cat: S})).

This list of three formulae, with the conjunction as governor, has two slots to be filled. Pointers p2 and p3 refer to fields of pi: the events between the temporal relation after is specified. These

pointers have been set by executing the pointer procedures (pp's) in the syntactic frame. Now the formulator notices that after, the leftmost lexeme in the list, has its final shape (cf. the end of section II.1) and can be pronounced. The remaining two-member formula list is then lexicalized with the VDC frames leave (see (14) and die: (((p 4{Cat: NDC; Case: Subj}) (leave {Cat: V} (p5 {Cat: NDC; Case: Obj})) ((p4 {Cat: NDC; Case: Subj}) (die {Cat: V}))).

The next lexicalization pass replaces the slots p4 and p5. During this pass, LEX notices that p4 occurs twice, inhibits lexical lookup for the second token and uses a pronoun instead.

This is a rough description of how lexicalization proceeds if the complete conceptual pattern is known to the formulator right from the start. After each lexicalization pass, the formulator checks if the leftmost top-level element of the formula list has been completely lexicalized. If so, this element is processed by FIN which computes the definitive shape of its lexemes, dumped into the output stream, and finally detached from the formula list. If the new leading top-level element has been completely lexicalized, too, FIN will work on it; if not, then the whole remaining formula list is subjected to a new lexicalization pass. FIN will be mainly a routine for handling VDCs, since top-level elements of the formula list are either VDCs or unchangeable words like conjunctions or sentence adverbs. The latter may be uttered as they are, without any further processing. So FIN's task is to shape up the constituents of VDCs in accordance with rules of tense, number, person, case, etc. (To this purpose FIN might use, among other things, a stock of syntactic frames in Garrett's sense (see Section I.1.iii); FIN will call itself recursively if a VDC contains another VDC.

This general setup of the formulator is consistent with the empirical observations reviewed in Section 1. Since the left-to-right order of formulae in the formula list is never changed, the formulator can release leading VDCs, conjunctions and adverbs very quickly and reclaim the freed working memory space (cf. Section 1.6). The observations on pausing in Section 1.2 can be accommodated too. For sentence (16), Boomer's rule would identify the transition between after and he as the place most likely to attract a pause.

(16) The man died after he left the old church.

This is also true of the proposed model: after FIN has worked on the first VDC the formulator can just read off the man died and after. But then he has to complete lexicalization of the second VDC and to finalize it before he left the old church can be said. (Lexicalization of the second VDC takes one more lexicalization

sweep than the first one, because of the modifier old.) Since lexicalization proceeds dependency level by dependency level, the type of speech errors discussed in Section 1.3 become understandable. E.g. in terms of sentence (9), the VDCs watch and listen to were interchanged during one lexicalization pass, or the radio and T.V. were during another. (Exactly how such interchanges come about, 1 don't know. Here I only appeal to some notion of temporal contiguity - "within the same pass..." - but other factors may be involved as well, e.g. similarity between interchanged elements).

II.2.2 Details of the lexicalization procedure

The complicated job LEX has to do for each formula may be divided into eight tasks. I will discuss them in turn, in the order they are carried out. Subtasks i through v are mainly concerned with finding adequate syntactic frames in the lexicon. By subtasks vi through viii, a selected syntactic frame will be trimmed down to the format needed for insertion in the formula list.

(i) Expanding synspec. The two sources of synspec we have considered up till now are the conceptualizer and the lexicon. Synspecs may be "incomplete", in the sense that syntax requires further specifications. For instance, the synspec {Cat: VDC} must be expanded so as to contain information about subcategories "main" vs. "subordinate" (at least in German and Dutch where they condition certain aspects of word order). If the lexicon or conceptualizer didn't specify which, the formulator must have a means of adding this information; for instance by assuming a default value, by looking at neighbouring formulae, or by inspecting the current conceptual pattern. If "Subcat: Main" is chosen, further information about Mode must be added: declarative, interrogative or imperative. The value of the Mode attribute can only be determined by inspecting the conceptual pattern. Similarly, a synspec {Cat: V} requires information about Tense, which can be derived from time information in the conceptual pattern.

LEX must have a set of rules defining completeness of synspecs, and mechanisms which execute the tasks illustrated by the examples.

(ii) Inspecting the conceptual pattern. The pointer in a formula points to a total conceptual pattern (i.e., the input pattern delivered by the conceptualizer) or to a part of it (e.g. the actor field of an event describing pattern). For simplicity, I call them both "conceptual patterns". LEX must know what kind of information to extract from the current conceptual pattern. For instance, if it has the

form "EVENT 1 cause EVENT 2" or "EVENT 1 time-relation EVENT 2", LEX should pick out the connector information and not, say, the actor of EVENT 2.

(iii) Lexical search. The information extracted from the conceptual pattern guides LEX through the lexicon when searching for an adequate syntactic frame. A good example of how to set up procedures for both (ii) and (iii) is provided by Goldman's (1975) generator.

(iv) Matching a candidate frame to synspec. Not every syntactic frame which expresses the current conceptual pattern can be inserted into the formula list. First, a candidate frame must be checked for compatibility with synspec. For example, if synspec is {Cat: S} , then a NDC frame wouldn't match, but a Conjunction-S-S frame would. Also, the synspec of a formula may impose certain word order restrictions upon the syntactic frame that will replace it. So LEX has to check if these restrictions can be met by a given candidate frame. What is needed of course, is a system of rules formalizing the notion of "matching" and "non-matching" frames.

(v) Checking for modifiers. Often, lexical search will not be able to locate a syntactic frame which expresses all aspects of the current conceptual pattern. In terms of sentence (15), suppose a speaker of English doesn't have a single syntactic frame expressing the conceptual pattern which underlies old man. Lexical search will suggest man for part of the conceptual pattern; which leaves the part underlying old unexpressed. I assume this second part win be lexicalized during LEX's next pass. As a reminder, LEX will tag the formula (man {Cat: N}) with a special synspec label: (man {Cat: N; NMOD: pi}). The attribute-value pair "NMOD: pi" says that the conceptual pattern pi will have to be lexicalized in the form of a noun modifier (e.g. an adjective or a relative clause). LEX will find this tag during the next pass and then come up with old as the translation of pi. LEX must be supposed to know special rules for modifier placement. (old {Cat: Adj.}) may simply be placed before the formula containing man, but especially in the case of verb modifiers LEX will need to consult more complex rules.

(vi) Executing lexical procedures. A syntactic frame may reference procedures that have to run before it is put into a formula list. As yet, the only type of lexical procedures we have seen are the pointer procedures (cf. Section II. 1), but other types may very well prove necessary).

(vii) Transforming syntactic frames. It is generally recognised that humans have a very limited working memory and a large and easily accessible long-term memory (cf. Section 1.6). In line with this, I assume that the lexicon, which is part of long-term memory, contains many ready-made syntactic constructions which in standard transformational grammar would be produced by applying transformations, i.e. by real-time computations in working memory. Examples are the passive constructions and subject complement constructions with extraposed subject. By assuming that these constructions as well as their transformationally less complex counterparts are both entries in the lexicon, we have an easy way of accounting for the experimental data mentioned in Section 1.5.

But I certainly do not hold the view that all transformations should be dealt with this way. Consider, for instance, interrogative sentences (yes-no questions) in Dutch and German. They differ from declarative sentences in that the subject follows the tensed verb; e.g. John saw Mary ——> Saw Mary John? The problem at issue is: does the lexicon contain a separate interrogative entry in addition to each declarative VDC entry, or are interrogative constructions computed from declarative entries? Another feature of German and Dutch is word order differences in subordinate and main clauses. E.g. "John fainted, for he SAW MARY" turns into "John fainted, because he MARY SAW". This example raises the same question: is the NP1-V-NP2 order in the lexicon, or NP1-NP2-V, or both? Whatever is the answer in these two concrete cases, I don't think we can do without a limited number of transformations which reorder or delete elements of syntactic constructions retrieved from the lexicon.

Pro-forms, too, entail changes of constituent orders. For instance, object NPs follow the verb in French, but precede it if they are pronouns. In many languages, interrogative pronouns occur in initial position, even if the standard position of the questioned NP is further down the sentence (e.g. John saw ?NP ——> What did John see?

Such transformations are applied to a syntactic frame before it is inserted in a formula list, for at that time the relevant synspec information is available. E.g. the Question transformation will be applied to a syntactic frame if synspec reads {Cat: VDC; Subcat: Main; Mode: Y/N-Question} . The transformed syntactic frame is then put into the formula list; whereafter synspec is lost. (The formulator doesn't keep a generation history of the

formula list, for reasons of efficient management of working memory).

At the present time I don't know which members of the set of linguistically defined transformations should be treated in terms of alterations to syntactic frames (like the Question example) and which members deserve separate lexical entries (like passive constructions). This problem may be experimentally investigated in experiments where subjects spontaneously construct sentences of a specific syntactic format, e.g. while describing a perceived or memorized event or picture.

(viii) Replacing the input formula. Finally, LEX replaces the input formula with the selected and possibly transformed syntactic frame.

II.2.3 Lexicalization of fragmentary conceptual patterns

The conceptualizer often delivers a conceptual pattern in bits and pieces, and the formulator must be able to immediately operate on such fragmentary information (cf.Section I.6). There are obviously many ways to divide a conceptual pattern into parts, and many different orders for feeding these parts into the formulator. I will outline here how the present model can handle an interesting subclass of all these cases, namely when a dependent (more precisely: the conceptual pattern underlying a dependent) arrives earlier than its governors, so that the natural top-down order of lexicalization cannot be followed.

By way of example, the conceptualizer delivers the nominal concept "Mary" to the formulator before embedding it in a conceptual pattern as the recipient of a transfer-of-possession action. The formulator prepares and utters "Mary..." without knowing what kind of VDC it will have to fit in. Then, after receiving the conceptual action, it is forced to look up a VDC which expresses the recipient in leading position. The passive give-frame "NP1 be given NP2 by NP3" will do, as well as the active get-frame "NP1 get NP2 from NP3", but neither the active give-frame nor "NP1 is given NP2 by NP3".

To permit the formulator to handle such cases it has to be extended along the following lines. The conceptualizer provides its output messages with a delimiter symbol, informing the formulator when messages start and finish. The input formula - which is either a complete or a fragmentary conceptual pattern - is passed along, not to LEX directly but to a monitor function MON. On the one hand, MON watches the lexicalization sweeps and prepares a "syntactic summary" for the utterance part LEX is constructing currently.

On the other hand, MON registers any new parts the conceptualizer adds to the current message. Suppose, at a given moment, LEX has finished its last pass for a fragmentary conceptual pattern and, in the meantime, the conceptualizer has added a new part to it. The latter implies there is a new input formula with a new pointer and a new synspec, MON will now

(1) append the syntactic summary for the last-produced utterance part to the synspec of the new input formula,

(2) add a tag "DONE" to the part of the conceptual pattern which has just been lexicalized, and

(3) register the role played by the DONE part in the new conceptual pattern.

As a result of this, LEX will receive a new input formula whose synspec tells it what kind of partial utterance the formulator has committed itself to already, and which part of the conceptual pattern need not be expressed anymore. This is enough information for LEX to construct a good continuation, if any (I), of the utterance.

In terms of the above example, if the synspec of the new input formula would simply have said "produce an S for this conceptual pattern", then MON would change it to "produce an S which expresses the recipient as an NDC in leading position" (assuming here that "NDC" is the syntactic summary for "Mary ...").

Finally, MON will hand over the modified input formula to LEX, monitor and summarize the lexicalization process and, if it sees no delimiter symbol, repeat its operation for still other fragments coming in. This facility for handling fragmentary conceptual patterns requires only one modification to LEX: rules for treating DONE parts of conceptual patterns.

III The formulator viewed from a linguistic point of view.

In this Section attention is shifted from psychological and computational to linguistic aspects of the proposed formulator.

(i) Except for cases discussed in Section II.2.3, lexicalization proceeds top-down (dependency level by dependency level, and from left to right within dependency levels). Is this regime compatible with the bottom-up principle of the transformational cycle, as discovered by transformational grammar?

A definite answer to this question cannot be given as long as many details of the model remain unspecified. But the examples I have worked out show that the proposed formulator is indeed able to handle some sentence types which need cyclically applied transformations in a standard transformational grammar. Consider sentence (17) (cf. Fodor et al., 1974, p. 121-131), whose deep structure contains a sentoid "doctor-examine-John" as part of the verb phrase of the matrix sentoid "Bill-persuade-John".

(17) Bill persuaded John to be examined by the doctor.

Two cyclical transformations are applied to the subordinate sentoid:

(1) passivization, resulting in "John-be-examined-by-the-doctor", and

(2) equi-NP-deletion, deleting the first NP of the subordinate sentoid ("John") which is referentially identical with the object of the matrix sentoid.

The proposed formulator can use synspecs to make such transformations superfluous. The active persuade-frame in the constructional lexicon looks (informally) like (18):

```
(18) VDC:  (  (ppl {Cat: NDC; Case: Subj} )
              (persuade {Cat: V} ) (pp2 {Cat:
              NDC; Case:  Obj} ) (to {Cat:
              Prep} )
              (pp3  {Cat:  VDC; Subcat: Infinitive-construction;
                   Detail: ppl of this VDC must deliver the same
                   value as pp2})  )
```

This frame is selected during one lexicalization sweep; during the next, candidate frames for the subordinate VDC are matched against the synspec following pp3. The active examine-frame wouldn't do because the value of its pointer procedure ppl (the concept "Doctor-such-and- such" in the field referenced by PP3) is not the same as the value computed by pp2 of the persuade-frame (the concept "John"). But the passive examine-frame would match. The attribute-value pair "Subcat: infinitive-construction" has two consequences as regards the final shape of the subordinate VDC:

(1) it influences later lexicalization sweeps in such a way that no verb tensing will occur, and

(2) it will delete the subject NDC.

The general idea seems to be:

(1) to make the synspecs for subordinate VDCs maximally specific so that they are only matched by frames which approximate the required syntactic shape as closely as possible:

(2) if any transformations to a selected frame are still needed, to execute them before inserting the frame into the formula list. The effect of doing this will be similar to the effect of cyclically applied transformations, but the computational processes are very different.

(ii) My chief motivation for using syntactic dependency as generative mechanism is computational efficiency. The only psychological evidence consists of the observations discussed in sections I.2 and I.3. Linguistic evidence, be it of an indirect nature, is provided by the study of linguistic intuitions. Levelt (1974) summarizes several studies of so-called cohesion judgments. They strongly favour dependency grammar over other grammar types (e.g. constituent structure grammars). Schils (1975) has confirmed this finding.

Levelt remarks that dependency trees do not represent the difference between endocentric and exocentric constructions. Interestingly, the difference J^s brought out by the lexicalization procedure. Exocentric constructions (like verb with subject, object, etc.) are the ones that are retrieved as a whole from the constructional lexicon. Endocentric constructions result from modifiers (see Section II.2.2 (v)): LEX notices it can only find a syntactic frame which expresses part of the current conceptual pattern, marks this frame with MOD, and lexicalizes the remainder of the conceptual pattern at a later stage.

IV Two generation examples

(i) In Section II.2.1, the first lexicalization steps for sentence (15) were discussed. Here I will follow the remaining passes. The formula list after LEX's second pass looks like (19).

```
(19)    ( ((p4  {Cat: NDC; Case:  Subj} )
            (leave {Cat: V; Tense: Past} ) (p5
            {Cat: NDC;  Case: Obj} )) ((p4
            {Cat: NDC;  Case:  Subj}) (die {Cat:
            V;   Tense: Past} ))  )
```

It differs from the formula list given in Section II.2.1 in that it contains tense properties for the verbs. These were added by LEX because, in English, verbs dominated by S (finite verbs) have obligatory tense markers. This is done by the procedure which

expands synspecs (Section II.2.2.(i)).

The third lexicalization pass works on the three noun dependency constructions. The synspecs of (20) lists four properties:

(20) (man {Cat: N; Case: Subj; Number: Sing; Mod: p6})

Of these, the first two were simply copied from (19) , the other two were added by the synspec expanding routine (Section II.2.2(i)) and the modifier checking routine (Section II.2.2(v)) respectively. The last NDC becomes (21) , with the appropriate Personal Pronoun instead of the noun.

(21) (he {Cat: PP; Case: Subj; Number: Sing})

The fourth lexicalization pass adds one level of modifiers to the lexeme string man-leave-church-he-die. (20) is changed to (22) by looking up a frame for (part of) p6 and consulting rules for placement of noun modifiers.

```
(22)  (( old {Cat: Adj})
       ( man {Cat: N; Case: Subj; Number: Sing; Mod:p8}))
```

The fifth pass only leaves p8 to operate on. The article is inserted before old. Others might prefer to determine the article during the same pass as the governor noun. Speech errors like (2) and (4) which have the articles at the correct place even though the nouns have been interchanged, might be taken as evidence for that alternative. Here I have strictly followed the dependency hierarchy. The end result is formula list (23) to which FIN is applied.

```
(23)  ((((the {Cat: Art})
          (old {Cat: Adj})
          (man {Cat: N; Case; Subj; Number: Sing})))
        (leave{Cat: V; Tense: Past}) ((the {Cat:
        Art})
         (church {Cat: N; Case: Obj; Number: Sing})))
        ((he {Cat: PP; Case: Subj; Number: Sing})
         (die {Cat: V; Tense: Past} ))  )
```

The first top-level element is processed, resulting in the utterance the old man left the church. Finally, FIN treats the remaining formula list (((he {...}) (die{...}))).

(ii) the second example has to do with fragmentary conceptual patterns (Section II.2.3). I will demonstrate how sentence (24) is produced

(24) The old man left the church and then he died.

if the conceptual pattern underlying (15) comes in in two fragments: first EVENT1, then" ... time-relation EVENT2".

The input formula for the first fragment is, I assume, (pl{Cat:S}). The translation into English proceeds in exactly the same way as the first VDC of (15). The syntactic summary prepared by MON reads simply "Cat:S". Assuming that the input formula for the complete conceptual pattern is (p2 {Cat: S}), MON changes it to (25).

(25) (p2 {Cat: S; Order: {EVENT1 {Cat: S} rest{ } })

MON has also figured out that pi plays the conceptual role of EVENT1 in the event sequence delivered by the conceptualizer. The notation between curly brackets specifies order and form of expression of the various conceptual parts: first EVENT1 is an S (which has been DONE already), then the rest in any form LEX wishes (this synspec is empty). This order prescription excludes "After S S" as a matching frame, but frame (26) is alright.

(26) ConjDC: ((ppl {Cat: S})
(and {Cat: Conj})
(then {Cat: Adv})
(p_P2 {Cat: S}))

Procedures ppl and pp2 set pointers to EVENT1 and EVENT 2 respectively. During his first pass, LEX will simply detach the first top-level element, that has been expressed already, and select the die-frame for EVENT2. Subsequently, FIN will put and and then into the output stream. Then the one remaining VDC is lexicalized and finalized.

Notes

This paper was written when the author was a Postdoctoral Researcher at the Department of Computer Science of Yale University (1975-76). His stay there was made possible by a grant from the Netherlands Organization for the Advancement of Pure Research. (ZWO).

I'm indebted to Dick Proudfoot, Department of Computer Science, Yale University for setting up a tentative computer implementation of the model outlined in this paper and for commenting on earlier drafts of the paper.

References

Becker, J. (1975). The phrasal lexicon. In R.C. Schank & B. Nash-Webber (eds.) Theoretical issues in natural language processing. Cambridge, Mass.;MIT.

Boomer, D.S. (1965). Hesitation and grammatical encoding. Language and Speech, £, 148-158.

Ertel, S. (1976). Where do the subjects of sentences come from? In Rosenberg (1976).

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

Fromkin, V.A. (1973) (ed.). Speech Errors as Linguistic Evidence. The Hague: Mouton.

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

Goldman, N. (1976) Conceptual generation. In R.C. Schank Conceptual information processing. Amsterdam: North Holland.

James, C.T., Thompson, J.G., & Baldwin, J.M. (1973). The reconstructive process in sentence memory. Journal of Verbal Learning and Verbal Behaviour, 12, 51-63.

Jarvella, R. (1976). From verbs to sentences: some experimental studies of predication. In Rosenberg (1976).

Kempen, G. (1976a). Syntactic constructions as retrieval plans. British Journal of Psychology, 67, 149-160.

Kempen, G. (1976b). On conceptualizing and formulating in sentence production. In Rosenberg (1976).

Levelt, W.J.M. (1974). Formal Grammars in Linguistics and Psycholinguistics. The Hague: Mouton.

Levelt, W.J.M. & Kempen, G. (1975). Semantic and syntactic aspects of remembering sentences. In R.A. Kennedy & A.L. Wilkes (eds.) Studies in long-term memory. New York: Wiley.

Osgood, C.E. & Bock, J.K. (1976). Salience and sentencing: some production principles. In Rosenberg (1976)

Rosenberg, S. (1976).(ed.) Sentence production: developments in research and theory. Hillsdale, N.J.: Erlbaum.

Schils, E. (1975). Internal Report, Department of Psychology, University of Nijmegen, The Netherlands.