MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

Working Paper 193

April 1979
revised July 1979

Steps Toward a Psycholinguistic Model of Language Production

David D. McDonald

ABSTRACT

This paper discusses what it would mean to have a psychological model of the language production process: what such a model would have to account for, what it would use as evidence. It outlines and motivates one particular model including: presumptions about the input to the process, a characterization of language production as a process of selection under constraint, and the principle stipulations of the model. This paper is an introduction, which is largely nontechnical and uses only simple examples. A detailed presentation of the architecture of the model, its grammar, and its interface to the speaker will be forthcoming in other papers.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defence under Office of Naval Research contract N00014-75-C-0643.

Table of Contents

1. Is a model possible?

There is common agreement on what a model of natural language production would be a model of. We produce an utterance because we have something to say. That "something" is a mental object which we can call the speaker's *message*. It is represented in non-linguistic terms, presumably the same ones that the speaker uses for thinking in general. Language production is the operation that takes a message and produces—"says"—the corresponding utterance in whatever natural language the speaker is using.

This is the view one finds in the review of the psycholinguistics literature given in [Fodor, Bever, and Garrett 1974]. It is the departure point for the psycholinguistic models of [Garrett 1975,1976] and [Kempen 1977]. And it the basis of virtually all of the production literature in artificial intelligence, e.g. [Slochum 1973; Perrault & Cohen 1977; McDonald 1978b].

However, no one is now in a position to develop a model of the *actual* human production facility—nor is the situation likely to change soon. The problem is that nothing is known about the nature of human messages. Any hypothesized language production process will be intimately dependent in its internal structure on the character of its input[1]. Consequently, before the operations, control processes, constraints, etc. of production can be developed, we must be reasonably certain of the details of the message representation. But, as is well known, the nature of the mental representation used by people for cognitive processing is an almost total mystery.

1.1 A synthetic model

On the other hand, if there were available a detailed and independently motivated representation in which messages could be couched—the mentalese of some other sort of cognitive processor, then the enterprise of designing and investigating the consequences of particular language production models could go forward.

Non-human "cognitive processors" with detailed and accessible internal representations are available; these are, of course, the computer programs developed by the artificial intelligence community. I am thinking particularly of the "advisor" programs such as [Genesereth 1978; Goldstein 1978; Swartout 1977]. These programs are specialists or adjuncts to specialists in a single domain such as digitalis therapy or symbolic mathematical analysis, and are intended to be able to advise their novice human users—explaining how

particular instructions work, pointing out misconceptions, and so on, with all of their interactions (ultimately) being carried out in English.

This paper describes a theoretical model for language production that has grown out of my research on a linguistic *production component* for these advisor programs. The model is synthetically (rather than experimentally) derived. It posits representations and operations that for the most part are beyond the ability of presently conceived psycholinguistic experiments to affirm or deny. At a minimum, it can provide a rich, descriptive metaphor. Possibly, it will suggest explanations for observed phenomena and motivate new experimental designs.

What makes the model more than just an experiment in procedurally encoding a grammar is the fact that the implementation adheres to certain psychologically interesting assumptions about how the processing is done. It takes seriously the notion that utterances are produced incrementally, from left to right; all actions are accountable, contributing to the output; and there are motivated limitations on the kinds of actions which are permitted, limitations which, it would appear, also hold for humans.

These limitations raise the possibility of an explanatory theory of those linguistic phenomena, based on processing restrictions during production. Such a theory will not be proposed in this paper because too little empirical research has been done and too small a sample of English grammar[2] has actually been implemented. An indication of the sort of phenomena that a theory of production could be used to explain will be given in [McDonald in preparation b].

The first section of this paper gives the background assumptions of the model—what it is and is not responsible for. The second section considers language production as a computational problem, developing a picture of production as decision-making under constraint. The final two sections introduce the model itself.

2. What has to be accounted for?

A psychological model of natural language production must be more than just the addition of an algorithm to a grammar (e.g. transformational generative grammars "produce" sentences but we do not consider them plausible models). Language production is a *process*, and therefore the model must provide explanatory answers to the questions one asks about processes: what are its subprocesses, how are they coordinated, what are the limitations on data access, and so forth. Equally important is the fact that language production necessarily occurs in a *context*, whereas whether or not a sentence is grammatical is taken to be true regardless of context. The model must characterize this context in so far as it bears on the process.

All of these considerations affect the nature of the model's representation of English grammar. In particular, the phenomenon of *linguistic creativity* receives a particular interpretation. What it is that people know about their language that allows them to freely understand and produce totally new sentences? Or, to put it another way, how is an infinite language represented by finite resources? All theories of language, regardless of their focus, must provide an explanation of this phenomenon. For a production model, this becomes the "*selection problem*":

> *How can a speaker, in a specific context and with a specific message, select a specific, corresponding text from the infinite set of texts that make up his language.*

From this point of view, the creativity of the total system is founded not in the language facility but in the message originator—the cognitive facilities of the speaker. He says new things because he has new things to say. The structure of the model and its representation for the grammar must support this. The grammar must be organized in such a way that it can take a potentially infinite set of pre-existing[3] messages into their corresponding texts[4]. Actually, as we will see, in this model the grammar does not directly control the processing but rather governs what is possible, i.e. it is a representation of linguistic competence.

2.1 Assumptions about the input to the process

Without being more specific, I will take a *message* to be a description of what the speaker wants to say, formally, an expression in some internal language. This makes *production* the problem of finding a natural language

text which meets such a description.

The idea of starting from a description draws on the common intuition that there is some "distance" between wanting to say something and having the linguistic means to express it. That is, rather than the tokens of a message being inevitably associated with fixed English phrases, the association is mediated by a context-sensitive process which can take other phenomena into account. This process interprets the description in terms of the linguistic means available at that moment and the speaker's perception of their effect on the audience. The correspondence between description and text is not inevitable but flexible, e.g. even if it takes only one word to evoke some concept for the listening specialist but several paragraphs for the layman, the same idea is being conveyed at the message level.

Since our natural language is a system of symbols which already have interpretations within the community, messages involving ideas that are new to us or to our audience can be difficult to express. There is no guarantee that the first phrasing we come up with will say "what we really meant", especially if we have little experience with the topic. There is also no guarantee that all of our message can be expressed—the language often does not provide enough flexibility. Thus when the "sub-descriptions" of a message are not mutually realizable, production comes to involve notions like preference and compromise. This is part of what makes the process non-trivial.

Given the complete lack of direct evidence, conjectures about message representation tend to be based on arguments about the usefulness or efficiency of the proposed scheme in ordinary research, e.g. predicate logic for the logician or semantic nets for the computer scientist. In their various scientific domains, the established representations are quite rich, and furthermore have differing, subtle implications for the capabilities of the systems using them. (See for example [Woods 1975].)

To actually develop an operating English-speaking A.I. program, it was necessary to stipulate a particular input representation. Otherwise there would have been no consistent guide for the extremely detailed design that is required. However, in developing a theoretical model, one wants to abstract away the details of any implementation. There is no need here to settle the question of what is the "right" representation. There is no need to fix the details of the actual syntax, the grain of the conceptualizations, or how concepts are implemented. We can settle for a very broad characterization as long as the model makes clear at what point the details of any particular

representation would make a difference, and how far the effects of those differences would propagate.

This model depends on two related assumptions about messages.

(1) Messages are compound structures — relationally structured amalgams of smaller structures, which are themselves recursively composed of other structures down to the level of unanalyzed tokens.

(2) Speakers have a finite, augmentable pool of tokens from which messages are assembled.

It is difficult to imagine how these assumptions could fail to be true since they appear to be an inevitable consequence of the creative nature of natural languages. If messages were unable to share subunits, i.e. if they were each single, non-decomposable tokens, then there would have to be a new token for each new utterance—an extremely implausible situation.

Given these assumptions, a natural working representation for messages is as a list of predicate-argument relations. No more detail than this is needed, since all that the model production facility needs to know is the compositional structure and the identity of the tokens used, i.e. the names of the composing relations and of the atomic arguments. As a trivial example, consider the situation of making a telephone call where someone answers other than the person you wanted to talk to. Your goal at that moment is to say something that will make them bring the person you want to the phone. A minimal description of this message (assuming that you wanted to talk to "Joe") would be:

bring-to-the-phone(Joe)

It consists of two tokens in a hierarchical relationship. The first names the goal and the second names the particular person to be "brought to the phone". In this example and those that follow, English names are used for expository convenience. They are not themselves intended to be interpretable—"#$%&" could have been use instead. All that is necessary here is that identical message tokens be identifiable as such.

2.2 Interpreting messages

With the composition operator, the possible message tokens constitutes a message representation language. The production component needs a way to interpret this language, i.e. a general way to read messages as descriptions of what the speaker wants in the final text. In this model, the interpretation is

provided by a translating *dictionary*.

To a first approximation, finding an English text to fit a message will entail finding a suitable "sub-text" for each of the tokens from which the message is composed, and assembling them into the full text. Therefore, the production facility needs to know what would be a good text for each possible token it might meet. This information is given in the dictionary. The dictionary has an entry for each token or token type in the message language, which records what the speaker knows about how to describe that token.

The speech act **bring-to-the-phone()** has a very steriotypic realization in American English, presumably because of the large role the telephone plays in our lives. Consequently, its entry will contain largely idioms: "*is X there?*", "*can I speak to ?*", "*X please.*". Also in the entry is a description of the pragmatic and intentional conditions when they can properly be used, e.g. that "*X please*" is something you might say to a telephone operator but not to your mother. This example will be developed further in section 4.1.

Less conventional speech acts of course require more planning for their realization and the speaker cannot just "pick an idiom off the shelf". However, the concern of this paper is with the production process rather than the planning process, and accordingly I will use only simple examples where planning is not required.

2.3 Theory of production vs. theory of use

It is important here to demarcate what is to be covered by a theory of language production and what would be part of a theory of language use. A theory of use specifies which linguistic device(s) will and will not achieve a speaker's goals. For example, it explains why asking for "*Joe please*" would offend your mother. Starting from the fact that it is the conventional way to address telephone operators, a short deduction reveals that to use that phrase with your mother would be to implicitly place her in the role of phone operator, which would be offensive. Accordingly, the entry for bring-to-the-phone() would be sensitive to the goal of not giving offense and would inhibit the use of that phrase whenever talking to someone who would not like to be treated like a phone operator. The bulk of any dictionary entry will make use of information of that sort to determine associations between the speaker's goals and descriptions and the phrases that could be used to convey them.

But it is one thing to observe these associations and list them in a dictionary and another entirely to explain them — to give a theory of the relationships between texts and the situational contexts in which they occur. This would require prior theories of human social interaction, motivation, cultural cliches, etc.. It is unclear where to draw the line short of formalizing all of human experience.

Developing a general theory of language use is not now a sensible project. In fact, it has been suggested that it is genetically impossible[5]. But the computer programs that have been built employing this production model embody defacto theories of language use in their dictionaries—"micro-theories" which are good for some specialized, small domain. Those theories are only applicable for those particular computer programs with their particular resources and style of reasoning.

Fortunately, a theory of language production does not require a prior theory of language use. Theories of linguistic competence have been developed by factoring out the influence of situation. Similarly, for a production theory it is sufficient to specify how they two types of theory interact. Briefly, in this model the grammar specifies what choices of phrasing are possible in an utterance, the theory of use specifies which choices should be made in specific instances, and the theory of production determines when the choices can be made and what evidence will be available.

2.4 The Domain of the Model

The character and time-course of the human production process varies tremendously according to the circumstances, e.g. from speaking to an old friend to writing a difficult essay. The variation involves speed, whether and how previously produced text can be edited, whether the decision making is conscious, and the kind of errors that are made.

Rather than capture the entire range of variations within a single, uniform model, I have developed a model of what I take to be the *basic* language production process, with the variations coming about through interactions between the basic process and other processes such as the speaker's language comprehension process ("listening to oneself") or a planning process with a source of external memory (e.g. pencil and paper).

It is suggestive to identify this basic process with the subjective phenomenon described immediately below. However, introspection and existing psycholinguistic data do not even begin to provide sufficient data to

determine the details of any processing model—they must be completed using computational and linguistic criteria. Therefore this identification is made only as a way to focus one's attention when assessing the model; there is not yet the evidence available to go any further.

In everyday conversation, we operate in "immediate mode"—we speak without rehearsing what we will say and with only a rough sketch of what we will say next. Similarly, when writing a paper or reflecting on what to say, it is a common experience for phrases or even multi-sentence texts to "spring to mind" already in their final form as though we were actually hearing them, and without our having made any conscious effort to form them from their constituent parts. If, upon reflection, we decide a text will not do, we do not consciously break it down and fashion the changes, but rather, another somewhat modified text appears in our mind with the modifications we want.

I have taken this immediate mode of human production as a subjective guideline for what the capabilities of the model should be. This is a post-hoc judgment. The computer program design that this model originated in was not intended to match any particular human behavior but just to provide the needed kinds of decision-making with a suitable environment in the abstract. It developed that the features of the design which made it good for decision-making lead to *inherent limitations* on the amount and kinds of information available from moment to moment.

These limitations parallel those of people in immediate mode, e.g. words can not be taken back; there is only a limited lookahead[6]; and the program, like a person, sometimes talks itself into a corner. This coincidence of abilities, as coarse as it is, is intuitive evidence that this synthetically derived model is a plausible one for the human production process and not merely for language production in the abstract.

2.5 What level of processing to describe

In a process model the actions that the model defines are taken to be actions that *actually happen* in the system being modeled. However there is a potential ambiguity. Any sophisticated process can be described with equal legitimacy at different levels of increasing detail: input/output relations, interactions between the major modules, the actions of an abstract machine embedded in the process, etc., on down to the level of the primitive actions of the machine on which the process is implemented—machine language for a computer; perhaps axonal pulse trains for us.

Which level is appropriate for this model and what should the model take as its primitive actions? For example, is the fixed phrase "*can X come to the phone?*" stored in the vocabulary as one item and instantiated with one action, or is only a plan or recipe for the phrase stored that is then assembled by a set of actions: "make an interrogative clause; use the modal *can*; etc.". The principles for deciding this question are given in the next section.

3. Texts as sets of choices

A very convenient level at which to describe the production process is the level of the *linguistic choices* that the speaker makes. When a speaker selects a text, he is making a set of choices—choices of wording, of phrase structure, of morphological markings, etc. Some choices are disjoint, e.g. a clause may be either passive or active. Some choices are free within a limit, such as the number of modifiers in a noun phrase. In the course of the production process, the speaker's choices combine to select a single[7], unique text from the possible texts in the language. This view of language in terms of choices provided has been extensively developed by Halliday [1970].

In the model, each choice that the grammar provides is understood to signify the program of actions that will be performed to add that choice to the growing text. But the model itself refers only to the choices and never to the finer actions that they imply. In abstracting a program of actions into one unit action (the choice) one is saying (1) that there can not be any significant interaction between the actions that comprise the various choices—the execution of any one choice is computationally independent of execution of any others; and (2) that the model makes no claims about their internal structure.

To be sure, in the operational text-producing computer program an internal structure for choices is worked out in detail. However, when it now comes to propose a model of human processing, the implementation detail must be abstracted away. We should look for the correspondence between model and reality at only this level of possible linguistic choices and how they are controlled.

Just what actions are taken to be unit actions thus becomes a matter of what choices we believe the speaker makes. If we take "*can (Joe) come to the phone?*" to be an idiom then it should be modeled as the result of a single

choice. If instead we see it as the result of several choices—perhaps the speaker is only learning the language and hasn't yet acquired the idiom—then we would (presumably) analyze the phrase as the result of a set of choices: the choice of an indirect speech act, and the choice of modal word and idiomatic verb phrase.

3.1 The grammar specifies the legal choices

The representational power of English is restricted, i.e. the actual language is only a small subset of the powerset of the possible choices, since at every point in the production process, just what choices are still free is contingent upon the choices that have already been made. For example, root transformations are prohibited in subordinate contexts; verbs in non-predicate verb phrases may no longer show their association with their subject NP through person or number agreement; modals force infinitives, and so on.

It is exactly this contingency information that a grammar—seen now from the point of view of production—must represent, i.e.

> *The grammar is a description of which sets of choices from the total set constitute texts in the language.*

The form of the grammar is determined by how that information can be best represented. The grammar is not the production process itself. The process draws on the grammar, but the course of the process in not given in the grammar.

3.2 Decision making

It follows from the definition of a text as a set of choices that those choices, now understood as the actions of a production component, are just the actions needed to produce that text. Thus language production can be seen as *a process of making choices*, where, when each choice is made, its associated actions are performed and the corresponding part of the final text produced.

That there are choices implies that there must be a decision-maker(s), something(s) that controls what choices are made. Two sources are available for decision-making: the message and the grammar. Each corresponds to a qualitatively different kind of choice.

The message is responsible for the choices that convey information. These *free choices* constitute the bulk of those that must be made: word choices, and choices of syntactic and intonational structure. The grammar is responsible for the choices that insure that the text will be well-formed: agreement, constituent order, morphological markings, etc.. These are *fixed choices* because the need for them is determined by what free choices have been made. They are "choices" only from the point of view of the grammar as a whole. Potentially, a clause can be marked past or present tense—this is the choice. But in particular contexts, for example those where the sequence-of-tenses rule applies, the choice is fixed. To keep the decision-making metaphor, we can say that in such cases *the grammar decides* what the choice will be, and that choices made by the grammar take precedence over choices by the message.)

3.3 Control structure

The control structure of a process determines the order in which its actions are performed. It is a central and often neglected part of any process model. In simple computer programs, action order is fixed by the order of the statements in the listing of the program. Their control structure is trivial: follow the listing as given. More sophisticated programs, just like people, determine what they will do on a context sensitive basis. Their control structures may be complex functions involving representations of the current context and internal scripts.

Control structure is particularly important in a process model because all of the actions that the model posits are hypothesized to be psychologically real. Since a model's control structure determines how its actions are structured, when they may apply, how far their effects may propagate, and similar properties, it is a prime source of empirical predictions.

A case in point, several earlier researchers based the control of their production components on a traversal of the grammar [Slochum 1973; Davey 1974; Wong 1975]. Wong, for example, represented the grammar as an augmented transition network, where the conditions on the arcs were predicates testing conditions in the message. In order to process a message, his control structure dictated that the arcs of the grammar were to be scanned in order until some arc's conditions were satisfied by (part of) the message. That arc would be taken, possible producing some output as a side-effect, and the scanning would continue on the arcs of the next state. If

we were to adopt Wong's design as a model of human production, his identification of the control structure with the ATN grammar would predict that we should find a correlation between processing load and the current branching factor in the ATN grammar.

The control structure for the present model will be discussed in section 4.3. At this point, I want only to lay the groundwork for the next two sections.

The message-based decision process is compositional. It follows the recursive structure of the message, realizing relations in terms of the realizations of their arguments. The grammar-based process acts as required to insure that the choices made by the other process are grammatical.

The two decision-making sources are modeled as distinct processes—they do not need to consult each other to make their decisions. At the same time, the two processes are very intertwined in their effects. A choice made for some relation high in the hierarchical structure of the message may create a grammatical context in which all of the possible choices for some lower relation are rules out. Conversely, a narrow range of choices for a lower relation will force the choices made for those above it to be compatible.

Because of interactions like these, the effectiveness of the production component as a whole will depend critically on the order in which decisions are made. If, for example, all the message-based decisions were made before any of the grammatical ones, it would turn out that many of those decisions would have to be recanted because of incompatibilities uncovered during the process of making the grammar-based decisions.

3.4 Stipulations

A priori, there are many conceivable control structures for the language production process, i.e. many different orders in which the decisions required for a given text could be made. Even if we eliminate those which appear implausible computationally, a large number of candidate control structures still remain which we cannot chose between for lack of direct empirical evidence.

The control structure adopted for this model was designed in response to the two stipulations discussed below. These stipulations are empirical hypothesis about the nature of language production. Their inclusion in the model strengthens it, since without them, there are too many degrees of freedom available in the design of the control structure—no predictive theory

deriving from the structure of the process would be possible.

Decisions are indelible.

Once a linguistic decision has been made, it cannot be retracted—it has been written with "indelible ink". An obvious instance of indelibility is the simple fact that a word can not be taken back once it has been spoken. However, the stipulation is intended to have a stronger force. It requires that *every* choice made during the production process, at whatever level, cannot be changed once it have been made—choices must be made correctly the first time. (This same stipulation has been adopted as an integral part of a theory of natural language parsing developed by Marcus [1977].)

Two points of production "strategy" follow immediately from this stipulation. First is the importance of *planning*. If choices cannot be casually taken back, then one must determine, before making a choice, whether or not it will have the desired consequences. Second is *the principle of least commitment* [Marr 1976]. It is critical that when a choice is made it specify no more detail than can actually be justified by the evidence at hand, as unsubstantiated details in choices will likely as not turn out to be inappropriate and will have to be retracted. Decisions can be made only when the evidence they require is available.

Texts are produced incrementally and in their natural order.

The point of stipulating an "incremental" process is to disallow extensive buffering. People are taken not to develop the entire text unconsciously before they begin to speak. The words of a text are to be uttered while the process in in progress, in the "left to right" order in which they are ordinarily spoken. This stipulation has the effect of limiting the workspace available to the production process, constraining the scope of its decisions.

By insisting that the process be designed to be indelible and incremental, one gets a model where every action adds to the text. If instead, choices could be invisibly recanted at any time while the text was in progress. then effective choices could no longer be told apart from ineffective, temporary choices. There would be no way to distinguish the effects of decisions that were pre-planned from those that were the result of editing a structure when it was later found to cause difficulties.

3.5 Dependencies between decisions

The effect of stipulating incremental processing and indelibility is to say that, in error-free speech, no individual decision is made until after all of the choices upon which it depends have been made. It follows that if we determine the networks of dependency within the message and grammar, we will have simultaniously determined the correct control path through the choice set of the target text, namely to trace out that network: begin with those decisions that do not depend on any of the others; then make the choices that depended only on those just made, and so on. The dependency network is the control structure of the model.

There appear to be two kinds of dependencies[8].

(1) Dependencies due to a message element's hierarchical position in its message.

Natural language phrases are morphologically specialized according to their grammatical role, which in turn is dictated by thematic relations. What thematic relations an element participates in depends on what message-level relations it is a member of. For example, suppose there was an element which denoted an action and was always realized by a verb phrase. Depending on the role that the action played in the message, this verb phrase might act as a description, as part of a propositional argument, or as a subject or predicate. After starting from a common set of choices, e.g. the choice of verb and complement, each role is expressed in a text by a different combination of refinements to the verb and auxiliaries and relative position.

By the assumption of indelibility, a message element cannot be realized until its thematic role is known for if it were acted on prematurely and later its true role turned out to be different than assumed, all the morphological and positional specialization performed in realizing the phrase the first time would have to be changed. Because thematic role is a linguistic choice and not something that can be deduced from a message by itself, it follows that the correct realization of each message element is dependent upon all the message elements which refer to it. The realization choices made for those elements will fix its thematic role.

(2) Dependencies due to position in the text sequence.

By the time that the realizing phrase for any part of the message is spoken or printed, the audience will have heard and begun to react to all the earlier text. This gives rise to a body of grammatical and rhetorical conventions

which must be followed or at least appreciated. The use of pronouns, of deletions, of the extent to which an object has to be described, and of how the audience makes inferences, are all dependent on ordering relations.

"Mutual" dependencies, that is, relational structures in messages which are general networks rather than just trees, will not be discussed in this paper. They do occur in the messages developed for computer programs and, at the moment, are dealt with through an initial global planning state which produces a structure that has no mutual dependencies, at which point the "regular" process described below takes over. There is reason to believe that the mutual dependencies actually only because of a deficiency in the existing formalization and that given a model of "rhetorical strategies" now under development, they will reduce to the regular case.

4. The Model I: control structure

Language production is modeled here as a process of decision-making under constraint: the speaker must decide what would be a suitable natural language rendering for his message, and is constrained in his choices by the conventions of his language's grammar. This viewpoint is reflected directly in the structure of the production process. It affects the choice of the primitive actions, how are they structured, what data they act upon, and how it is represented. In particular, it affects the representation of the grammar, since the grammar's form must fit its function (particular to the production process) of acting to constrain decision-making.

Thus far in this paper I have (1) characterized the input to the process as a relational structure over a fixed vocabulary denoting concepts in the speaker's mind; (2) argued that the "grain size" of the production process, i.e. the number of decisions that must be made in order to realize a speaker's message in a natural language, corresponds to the number of elements in the hierarchical structure of the input relation, or, to put it another way, language production is a compositional process to which each message element contributes a "subtext"; (3) hypothesized that the process can be decomposed into two distinct subprocesses: (a) determining the informational content a text should express, and (b) determining what form the text should have so as to conform to the conventions of, e.g. English grammar; and finally (4)

stipulated that the control structure of the process must be such that the effects of actions are indelible and the text is produced incrementally in its natural order. These stipulations lead to a network of dependencies between the elements of the message which dictates the order in which they can be expressed.

Two things in particular remain to be discussed. First is the question of control structure—how can this network of dependencies between decisions be elicited, represented, and used for control? Second is the all-important question of representing the grammar—how is it to fulfill its role of constraining the decision-making process? As we will see, the two questions have a common answer.

4.1 Realization choices

Intuitively, what is chosen is an English phrase. The question is: how is the phrase represented; what must be represented given the needs of the production process?

The idioms stored in the example dictionary entry for bring-to-the-phone(Joe) are a good place to begin. Notice that the phrases will typically include variable constituents (e.g. the "X" in "*is* X *there?*"). This variablization corresponds to that of the message element relations for which the phrases are plausible realization choices. Consequently, choices appear in entries as *functions.* They are given names (for the benefit of the human linguist and to provide a mechanism for reference and annotation) and formal parameters, e.g. is-x-there(x).

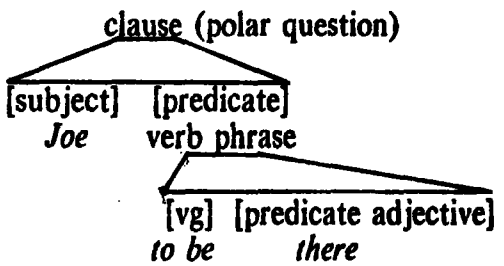
Making a decision for a message element involves first consulting the usage conditions in that element's dictionary entry in order to determine the choice, then evaluating that choice (seen as a function) substituting the appropriate values for its parameters (e.g. Joe for X). The value returned by the choice is the realizing phrase.

Language production is done incrementally and in real time. This fact has a considerable forcing effect on the design of the model. Consider that a choice is made in an instant, yet the phrase that is chosen necessarily takes some time to utter. There must, therefore, be some kind of buffering within the production process—some representation of what it is that has been decided upon which can be referred to in order to control subsequent processing.

It is not sufficient merely to record an ordered list of the words in the phrase. For one thing, the identity of some of those words is not yet known since at the time of the decision, embedded message elements (e.g. Joe) have not yet have been realized. (It might need to come out as "*the guy who runs the lathe*".) Furthermore, there are always grammatical interactions to be considered, both within the phrase and between elements of the phrase any embedded message elements. For example, if the phrase realizing X turned out to be plural, then the form of the verb *to be* in *is* X *there* would have to accommodate it.

> *The model records as the value of a choice a representation of the immediate constituent structure of the selected phrase, with the specific parameter message "sub-" elements (if any) incorporated as constituents.*

So for example, after evaluating the choice named (arbitrarily) choice1(x) and substituting the message element Joe for x, the working buffer of the process would have the tree below added to it. (The internal structure of choices is discussed in section 5.4)



This is a syntactic structure, given in tree form. It has two nodes, one of category clause (with the additional feature "polar question") and one of category verb phrase. The clause has two constituents: the first is the embedded argument Joe, the second the verb phrase. The constituents of the verb phrase are a verb (not yet specified for tense, person, or number) and the word *there*. Any differences between this grammatical analysis and any others the reader may be familiar with are not important at this point with one exception: the naming of constituent locations (i.e. "[subject]", "[predicate]", etc.).

It is important to note that what a choice specifies is a *surface structure*, i.e. the final form that the linguistic structure will take and the one from which any phonetic realization of the choice would be based. At the same time, the structure is minimally specified. Only those details which are in fact decidable at the time the choice is made are included. Notice also that at this point in the process, the correct, inverted auxiliary configuration for

expressing a polar question is not apparent. This is done to capture the grammatical generalizations about the common form of declaratives and interrogratives. The correct structure will be constructed later (as the phrase is actually being spoken) by a grammatical process triggered by that feature.

4.2 Progressive refinement

At the level being modeled, the basic processing step is to take an element of the message, decide what its realization should be by consulting its dictionary entry, and then evaluate that choice and add the resulting constituent structure to the growing text. At least one such step is required for every element in the message; the question is how they are to be ordered. The stipulations require that the ordering reflect the decision-dependencies of the message. If that is done, the indelibility of the process is guaranteed because all of the decisions that could affect an individual message element will have been already made when the time comes for it to be realized and can thus be taken into account.

> *This requirement can be met very neatly by using the constituent structure selected with each decision as the control structure of the process.*

To demonstrate how this is done, let me use a slightly larger message as an example. Suppose that the "mentalese" of our example speaker was the predicate calculus, and that he wanted to relate the following proposition:

man(Socrates) -> mortal(Socrates)

In order to deal with this message, the production dictionary for this speaker must have an entry for implication, one for predication, and one each for the two predicates and one constant. (A fairly complete generation dictionary for the predicate calculus is described in [McDonald 1978a].)

Recall that two types of decision dependencies are important in this model: (1) hierarchical position within the relational structure of the message, and (2) position in the left to right sequence of the final output text. The hierarchical structure of this example is fixed by the precedence rules of the predicate calculus.

Left-to-right sequence in the text is not fixed until at least one decision has been made, introducing some linguistic structure to the message. However, as soon as that decision is made, the fact that decisions always result in a *surface structure* description, plus the stipulation of indelibility, guarantee that the left-to-right relations within the realized linguistic structure

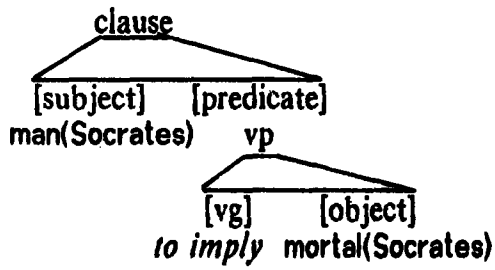
will not change, and can be relied upon as a description of the linear dependencies among the message elements embedded within it.

Thus the first step in the control program is to realize the hierarchically dominant element of the message, in this case the implication.

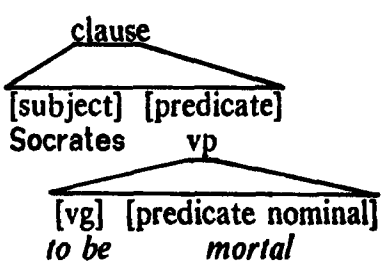
An implication is a relation with two arguments, taken over without modification into the selected phrase. Until the implication is realized, the textual order of its arguments is undefined. After realization, their order is defined by their position in the phrases's constituent structure. For an implication, A->B, possible choices of realization include:

choice1 -> "*If* A, *then* B"
choice2 -> "A *implies* B"
or choice3 -> "A B*'s*"

Let us say, for the sake of this example, that the surrounding context is such that the usage conditions of the entry will elect choice2. Then the selected surface structure will be as below. This structure will control the order in which the rest of this message is realized.



From this point on, the process proceeds by *progressive refinement.* The established linguistic structure is not changed (it's indelible), but details are added as appropriate. Eventually each of the embedded message elements will be replaced by its realizing phrase at the same point in the structure. For example, the instance of mortal(Socrates) which was the original complement constituent will be replaced, realizing the predication and predicate as an English phrase and embedding the constant Socrates.



The refinement process terminates when all of the embedded message elements have been realized.

4.3 Control by interpreting the generated surface structure

The control structure of the production process is the surface level constituent structure that is selected by the entries for the elements into which the input message decomposes. The individual contributions for each elements are knit into the whole by direct replacement. Once the initial constituent structure is created (call it *the tree*) it determines all the further actions which the process must perform and determines their ordering.

To extract this information, the tree must be *interpreted.* A simple, fixed recursive algorithm is used for this purpose. I will refer to it as the process *controller.* The controller scans the tree, beginning at its root node, following a depth-first pattern, top-down and left-to-right.

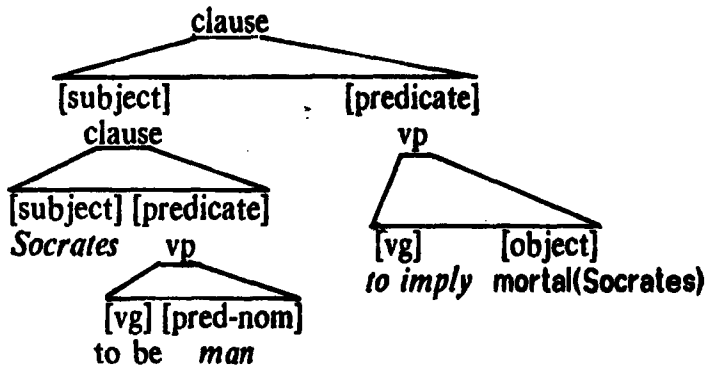
The following is a sketch of the algorithm.

At each node of the tree,
examine each of its constituents in turn,
beginning at the left and moving right.

For each constituent, depending on the kind of object it is, do:
(1) if empty: nothing
(2) if a word: apply morphological processing as appropriate
and utter the result
(3) if another node: call the node-examination recursively
(4) if a message element:
(a) find and construct a realization for it through
pronominalization or by consulting its entry
(b) replace the element in the tree with this new phrase
(c) call the constituent-examination recursively

For this example, the controller would begin with the root node of the tree—the clause on page 21—and proceed to its first constituent. As this constituent, man(Socrates), is a message element, the fourth case of the controller applies and the dictionary entry for predication is consulted. Let us say that this entry selects and returns a copular clause with the constant Socrates as its subject and the predicate name, man, as its nominal complement. This phrase now replaces man(Socrates) as the subject clause of the main clause. The controller now recurses to scan the new subject, goes to its first constituent, Socrates, and consults its entry. Let us say the entry returns the word, *Socrates.* Then the controller replaces the constant with the

new word. Now that a constituent contains a word, the controller takes its second case, and passes the word, *Socrates*, to a morphological process which, of course, does nothing to the word since it is a proper noun in a nominal context. Then the word is uttered and the controller proceeds to the next slot. The state of the process is now as shown below.



Several non-trivial fixed choices must be made in the course of realizing this message in order that the end product is grammatical. In particular, the two embedded clauses (corresponding to the two predications) have to be subordinated: "*Socrates being a man implies his being mortal*". Their parallel structure is the result of a left-right dependency (a usage convention) which calls for the same subordinating transformation in both cases. Unfortunately there is not enough space here to demonstrate how this can be done, although the extended example at the end of this paper may help.

4.4 Consequences of the control structure

This assertion that the actions of the production process are defined and organized by an incrementally developed, constituent structure tree is central to the model. (Whereas exactly how the tree is grammatically annotated is not a central point.)

> *This choice of representation solves the problem of how to indelibly assemble the text for a whole message from the texts of its parts. It does this by fixing the order in which to incorporate individual elements (the "parts") in such a way that this order captures the dependencies that are operating.*

Notice that by using this representation with its one-pass controller the production process is *guaranteed* to be indelible. The only way that the modeled process will actually utter a word is for the controller to "find" that word at a leaf of the tree; at which point the word is uttered immediately.

The controller traverses the tree only once, thus each word-bearing leaf is reached only once.

Were we to allow the actions taken by choices to modify parts of the tree "behind" the controller, it would still make no difference. By the same definitions and attendant theorems that we assure ourselves that a depth-first, tree-traversing algorithm makes only one pass through a tree, we are assured that it does not pass through any part of the tree more than once. Thus even if choices had the ability to modify earlier parts of the tree, these modifications would, so to speak, never see the light of day.

The question arises, what about modifications to the tree "in front of" the controller. The answer is that such modifications will go through—the controller will not treat "modified" constituent structure any differently than "original" constituent structure. Indeed, this is exactly the behavior we want if the model is to accommodate our psychological intuitions.

For example, there is the obvious fact that people can stop in mid-utterance—because they have forgotten what they intended to say; because they no longer need to convey some information; because they find they have said something they didn't want to—they made a mistake, etc.. People can change their minds and finish with something different that what they had planned when they started. Correspondingly in the model, the planned but not yet acted-upon sections of the tree in front of the controller are not engraved in stone and may be revised for good cause.

Of course, these editing phenomena are not accounted for by a production model in isolation. The story that one would like to tell about them will involve some sort of "overseer" component, which is, in effect, listening to what the production component is producing. This overseer would have the ability to edit portions of the tree in advance of the controller's position. However, it is unclear what the limitations of such an overseer should be, especially since its judgments will be of the same kind as those of the process which assembles messages in the first place. This places at least the control over editing outside the domain intended for this production model.

One editing phenomena that probably does fall within the scope of the model is *heavy noun phrase shift.*

1) I dismissed as unworkable the suggestion that the meeting should be held at eight.
2) I dismissed the suggestion that the meeting should be held at eight as unworkable.

Many people find that the first of these two sentences is easier to understand because the idiom "*dismiss as unworkable*" is not broken up by the long "heavy" noun phrase. Ross [1967] said that it had undergone a transformation moving the noun phrase *the suggestion...* from its "original" position just after the verb (as in sentence two) to the end of the clause.

For a processing model, what we want to know is when is the decision to use such a transformation made. Though at this time we have little more to go on than intuition, it seems likely that the need for a heavy-phrase shift does not become apparent until the site of the phrase is actually reached. One imagines that in the course of deciding what text should go into the noun phrase, the speaker realizes that its size will be excessive and, since there is a grammatical alternative available where the size will not be a problem, he chooses to delay the expression of the phrase until the end of the clause.

A transformation like this fits naturally into the model. The size of the planned phrase would be noticed while the entry constructing it was being interpreted but before the phrase had been attached to the tree. The choice to position the phrase at the end of the clause would be implemented by editing that position in the tree ahead of the controller.

Heavy noun phrase shift, under this analysis, is an instance of a reordering transformation the need for which could not be noticed until the realization of the text embedding it was well advanced, i.e. all of the tree behind the controller at the point when this "last-minute" transformation is selected will have been processed.

> *If the assumptions of this model are correct, no last-minute transformations will be found which move constituents leftward in the text, since to do so would violate indelibility.*

The apparent exceptions to this prediction are very revealing. Certain usually optional transformations become obligatory if the second of the two constituents involved is realized as a pronoun.

The dragon picked up the poor knight and ate him.
The dragon picked the poor knight up and ate him.
The dragon picked him up and ate him.
** The dragon picked up him and ate him.

These are examples of "particle movement", the same phenomena occurs with "dative shift". If, as seems likely, this obligatory particle movement is a

"last-minute" transformation and if the unmoved form is the basic one, then this is, on the face of it, a case of a constituent moving leftward behind the controller—a violation of the indelibility stipulation.

In order to retain the indelibility stipulation we have to posit a *look ahead* mechanism in the controller which would be triggered in the grammatical contexts where such transformations are possible. In those cases where the transformation was not chosen for stylistic reasons, this mechanism would have to look ahead of the first constituent (the controller is presumably just at the position of that constituent) to the second and determine if it was going to be pronominalized.

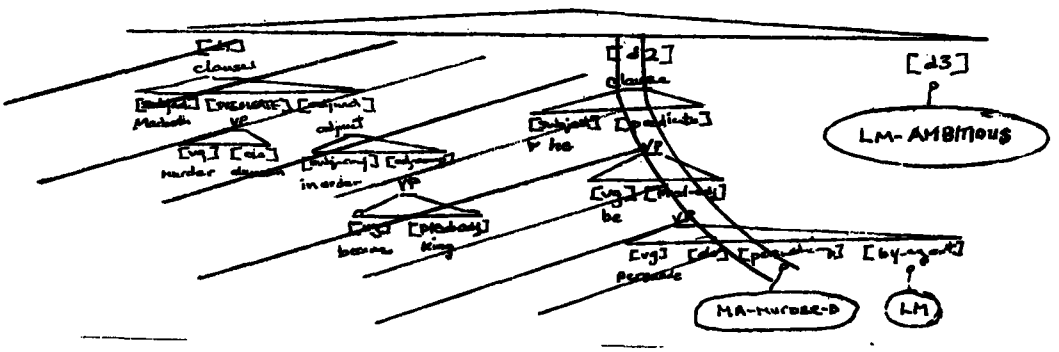
Notice that the look ahead is only required to be one constituent ahead. If an arbitrary look ahead had been needed, it would have amounted to totally discarding the incremental production stipulation since, in practice, the only difference between looking ahead and operating normally is that in look ahead "the printer is turned off"—the notion of making only one pass through the tree would become meaningless. But a look ahead of only one (and then only in very specific circumstances) is a different matter. In the case of checking for obligatory particle movement, all that is required is that after the controller has realized the particle (i.e. the word *up*) but before it has attached it to the tree a check be made to determine whether or not the direct object (the knight) will be realized as a pronoun (a relatively simple check to make compared to, say, determining whether the adjective *poor* will be used). If a pronoun is going to be used, the controller holds the particle in a buffer while it attaches the pronoun and then attaches the particle immediately afterwards.

The incremental production stipulation will permit look ahead of one constituent because the decisions which went into the construction of the buffered item (i.e. the particle) could not be effected should the transformation be required—the choice of particle is not different if the direct object precedes it instead of following it. On the other hand, if we imagined a look ahead process general enough to find the sites of origin of (supposed) left-moving last-minute transformations that moved across more than one constituent in front of the controller, we could not, in general, guarantee that the realization decisions for the intervening constituents would not be effected by the new position of the moved item, e.g. new pronominalizations might be possible.

4.5 Information limitations

The choice of control structure for this model will, as the model is developed, lead to further predictions deriving from the effects of *information limitations* on the speaker's decision-making process.

The source of information limitations is illustrated schematically in the diagram below. This diagram represents the computational state of the production process— the state of the tree—at a point part way through the realization of a message. The broad line running from the top node to one of the leaves represents the present position of the controller. At the syntactic nodes and constituent positions which the line runs through are "active" and contribute to the grammatical description of the *current position* of the controller. The region of the tree to the left with slanted lines through it is where the controller has already been; the clear region to the right of the line is where the controller has yet to go.



It is this asymmetry of the tree which leads to the limitations. Decision-making requires information. As we will see in the section of representing the grammar, the position of the controller makes an enormous difference in the quantity and kind of information that can be extracted from the tree. If some needed or desirable information is not available when the time to make a decision arises and the decision can not be postponed, then the decision may not be a good one—it may cause mistakes later on.

For example, in the micro-theories of language use that I have studied [McDonald 1978a], a very important kind of information that is needed is information on the kind of linguistic realization a given message element will have: will it be a clause, a noun phrase, an adverb, will it be long or short, will it use quantifiers or conjunctions whose scopes might be missinterpreted, and so on. However, the extent to which this information is available in the state of the generation process varies dramatically depending on the position of the message element in question with respect to the controller.

There is no limit to the amount and kind of linguistic information that could be learned about an element that has already been realized—one that the controller has already scanned—since all of that information can be "read out" directly from the tree behind the controller. But elements that the controller has not yet reached are largely mysteries. All that can be know about them linguistically in advance of their realization by the controller has to be learned deductively from consideration of the realized portion of the tree above them and from general properties deduced from the structure of their entries.

The point is that the incremental, left-to-right development of the text results in a natural sort of "blinders". It seems very likely that such blinders are implicated in the hesitation and restart phenomena that are so common in natural speech. For example, the phrase below was heard during a lecture on how children learn to understand stories.

> ...we'll look at some children where it appears that some of these plans have become talkable about.

This is a case where the speaker has "talked himself into a corner". An adjectival context was set up ("..*these plan have become* ____") even though the description that the speaker wants to use could not be expressed as a legitimate adjective. I would speculate that the speaker intended to focus on the "plans" but neglected to first check that the relationship (i.e. "the children can talk about these plans") could in fact be fluidly expressed as an adjective phrase. Presumably the failure to check was due to time limitations since he was speaking off the cuff.

However, one cannot formulate precise predictions about this kind of phenomena without having an independently-justified theory of grammar in the production process. Without an independent justification of the grammatical analyses there are too many degrees of freedom available in the analysis of limitations. Such a theory will not be presented in this paper, though the next section will sketch how one would be organized.

For the time being, it is clear that the model does place restrictions on the availability of information during the generation process which, if the assumptions of the model are correct, must have repercussions on a person's decision-making and planning and which, given a developed production grammar, will yield predictions about possible speech patterns and likely places of error.

5. The Model 2: the representation of grammar

In this paper, the term "grammar" refers to a descriptive notion that we could term: "the grammatical facts of English", and not to the sort of thing one thinks of in the context of, for example, transformational grammar or systemic grammar. Such "grammars" consist of the language facts plus some particular, theoretically motivated choice of representation. In this paper, the choice of representation for the grammar *for use in production* is the question to be decided.

Determining what "the facts of English grammar" in fact are is not really a proper part of a theory of production. The definition of a natural language is not influenced by the details of the production process (except indirectly insofar as the nature of the process will make some kinds of constructions more difficult to execute than others). Consequently, the production model will "inherit" its specification of the language from a separate theory of grammatical competence. The question here is how should the information in a competence grammar be represented when used for production.

The model uses its grammar to do three kinds of things: (1) Define the set of possible choices—the possible constituent structure fragments and features from which the tree can be constructed. (2) Contribute to the computational state of the process a description of the current "grammatical context". This description is available to the usage-conditions on use in the dictionary for guiding decisions, and serves automatically to inhibit the selection of any choices that would be grammatically incompatible with the context at the time of the choice. (3) Realize the "fixed choices" in the text—textual details such as agreement, case markings, some function words, etc., whose presence is required in order to have a grammatical text, but which are entirely predictable from the free choices that have been made.

Two additional hypotheses are being explored in the representation that will be described. They are not central to the model, but if they are found to be feasible, they will be an interesting extension to its explanatory power. The first is a distinction, in the structure of the representation, between items derived from the grammar, i.e. function words and bound morphemes, and items that derive from the message, i.e. the content words and intermediate message elements The second hypothesis divides the "facts of English" into two classes: those that can be used by programs other than grammar routines to plan the utterance, and those which only come into play in the realization of the text. The first class is represented using declarative devices, the second

using procedures. These hypotheses will be elaborated somewhat in the extended example.

5.1 Grammatical context

The level of representation in this model is the level where choices are made. Accordingly, in order to insure grammatical output, what the production component must do is to insure that only grammatically legitimate choices are ever made.

The notion of a "legitimate choice" is a dynamic one. What choices are legitimate at a given point in the process depends upon what choices were made earlier. As a consequence, the structures which enforce grammaticality must also be dynamic and sensitive to the effects of individual choices. Furthermore, these "structures" should not encompass all of the facts of the grammar at one time, e.g. when a noun phrase is being constructed, the grammar of participles is not relevant; the rules of auxiliary construction are not relevant to relative clause construction; and so on. This is formalized in the concept of a *grammatical context.*

All decision-making, indeed, all actions of any sort take place only at the current position of the controller within the tree. Thus we can naturally define the grammatical context in terms of this position and make it available to the decision-making process. Just what this context should specify is, of course, a function of what criteria the decision-making is sensitive to. In the computer program, these criteria have included: what is the category of the immediately dominating node? are we in a subordinate context? what is the subject of the immediately dominating clause? and so on.

Consider the earlier example of the well-formed formula man(socrates). This expression can be realized in several different ways, depending on how it relates to the other logical expressions in the message and how they themselves were realized. Since it may be applied to arbitrarily many different arguments, the formula's predicate, man, has a dictionary entry of its own. The usage conditions in this entry are sensitive to the grammatical context of the formula's position within the tree.

This context does not come into being until the decision is made realizing the particular logical relation that the predicate is embedded in. The very same embedding relation, say:

all(x) man(x) implies mortal(x)

can create different grammatical contexts for man(x), depending, in this case, on the choice made for the realization of the universal quantifier[9]. e.g.

Everything that is a man is mortal
All men are mortal

In each case, the predicate man is embedded in a different grammatical position. Since entries are identified with unique message elements (or element types) rather than some notion of element plus context, the entry itself must include some kind of conditional test which, in this case, will distinguish "nominal" positions such as subject or object, from "clausal" positions such as relative clauses.

The grammatical context is constituted from the grammatical relations that can be computed from the contents of the tree with respect to the current position of the controller. As the tree is the result of composing the output of the choices made so far, it follows that grammatical context is ultimately defined by the annotation on that output: how the nodes are labeled; what features are defined; what constituent position are defined; and so on. This is the usual repertoire of descriptive devices that linguists use, though there are differences in exactly what the various annotations (e.g. subject) mean brought on by the special circumstances of production.

The form that the context takes is a model-maker's decision, rather than a linguistically principled one. As we will see, I have chosen to organize the representation of the grammatical context around a notion of *relative ease of access.*

5.2 Contextual relations are not all created equal

A linguist looking at a complete constituent structure tree can "read out" any of the relations represented in it with essentially equal facility. This is not true during production under the stipulations of this model, a consequence of the inherent information limitations brought on by the incremental, one-pass nature of the process. When we now consider a grammatically annotated tree, the variations on ease of access throughout the tree are even more striking.

The picture that emerges is that there are at least three grades of relative accessibility to information in the tree. That is to say that when a dictionary entry needs a certain piece of contextual information to make a decision, the ease with which that information can be supplied to it will vary according to the grade of accessibility involved. In a richer model which considered the effect of resource limitations on decision making, we would expect to find

decisional criteria that were difficult to access would tend to be avoided whenever the time available in which to make the decision was short.

The most accessible information in the tree is that which is directly in the purview of the controller. In the diagram on page 27, this is any information that can be read out from the narrow region of the tree dominating the controller—the "active" nodes. This immediate context is best described in deitic terms, for example specifying the "subject of the current clause", the "category of the immediately dominating node", the "name of the current constituent slot", and so on. Information in this region is taken to be precomputed, requiring little or no effort to access.

The next most accessible grade of linguistic information is the annotation on those portions of the tree that the controller as already passed through. Here arbitrary amounts of information are potentially available (assuming it is included with the annotations). For example, the entry for logical implication might want to know when was the last time in the discourse that the word *imply* was used and what the context was at the time, or it might want to know where and why one of its argument message elements has appeared before.

It remains to be determined just how extensive the annotation of earlier material must be. Must the entire tree structure be retained, or would it be adequate just to keep some sort of summary. In the implemented program, it has been necessary so far to retain only the names of the choices—the actual words and phrases were dispensible. In "*Socrates being a man implies his being mortal*" for example, the choice of subordinating technique for the second embedded clause is fixed to be the same as the choice made for the first clause (a stylistic convention). Because the condition is stated at the level of choice names, there is never any need to look back at the text and "deduce" what the choice had to be from the presence of the *-possessive* and *-ing* morphemes.

However to get this kind of information two prices must be paid. First there is the expense of remembering the annotated parts of the tree after the controller has passed through them. (Of course, references to "the" tree are not intended to preclude the structure from actually spanning many sentences of a discourse.) Language production surely involves some sort of short term, presentational buffer, albeit a specialized one, and its size will be limited.

Then there is a more subtle expense, brought on by the process of locating the information in the tree. Only the controller can provide immediate, deitic information. If there is to be a predicate that, say, yields a description of the

grammatical context relative to the last instance of *imply*, then that predicate must be prepared to do the work of actually scanning the tree in that region and deducing the relevant relations. This searching process can be made less expensive if additional burdens are taken on by the memory, e.g. maintaining indexes to important earlier items directly or compiling descriptions while the controller is present and storing them.

The least accessible linguistic information is that which hasn't been computed yet—the realization choices for the message elements ahead of the controller. At the same time, this kind of predictive information can be very valuable in a decision. For example, the precondition for realizing an implication as a single clause (i.e. "*All men are mortal*") is that the antecedent message element can be used as the grammatical subject of the consequent. (This is not always true, consider how "*if wishes were horses, then beggars would ride*" might derive from: horses(wishes) -> ride(beggars)).

Predicates like can-be-the-subject-of can be defined in terms of an examination of the dictionary entries for the message elements involved (see [McDonald 1978b] for one way that this could be done), but the expense is very high, since the tests that the entry would make must be simulated, and certain of the entry's tests may be unsimulatable. For example, the conditions for using pronouns depend critically on knowing just what other elements have been mentioned in the text intervening between that instance of the element and its antecedent. Should the element about which a prediction is desired be anything other than the very next element in front of the controller, then its conditions on pronominalization will be indeterminant—the intervening elements might turn out to introduce potentially confusing references into the text but there would be no way to know this unless their own realization was simulated as well. Ergo, this mode of production does not insure "perfect" texts.

5.3 Automatic actions by the grammar

The grammatical context has both passive and active aspects. The passive aspects have been discussed: the tree is annotated and a body of predicates are defined for use by the entry conditions that examine those annotations. The active aspects are of two types: directly constraining the choices of the entries and implementing the fixed choices of the text.

Rather than being self-contained processors, dictionary entries are simply data structures which must be interpreted in order to make a decision. This

indirection is useful because it facilitates a simple treatment of grammatical constraints using already developed devices.

Associated with the grammatical terms used to annotate the tree (e.g. "clause", "subject", "polar question", etc.) are sets of conditional tests—the same sort that dictionary entries contain to specify pragmatic conditions on use. As the controller passes through the tree, it not only realizes embedded message elements but maintains the current set of these grammatical conditions according to the categories and features of the nodes in the region of the tree that it is in. Every time it consults a dictionary entry to realize some element it adds these conditions to the conditions already with the entry and evaluates them together. The grammatical conditions act as filters. If true, they cause some entry choice or class of choices to be removed from consideration, leaving only grammatical choices for the entry's regular usage conditions to select from.

5.4 The structure of choices

So far I have spoken about realization choices as though they were black boxes. In fact they are modeled in terms of a two part data structure which is interpreted when the realization action is performed and which is also available for passive examination by other parts of the process for planning purposes.

Every choice has two parts: a phrase and a map. The *phrase* is a specification of a surface-level constituent structure, possibly with some of its leaves filled by words. Its other leaves will be empty. The *map* is a function from the formal parameters of the choice to empty leaves of the choice's phrase. When a choice is taken, its phrase is instantiated and the arguments of the relation are embedded as constituents of the phrase as specified by its map.

So, for example, the grammar would include the choice A-implies-C below. (Names are given to choices to uniquely identify them. They convey information only to the human model builder.)

```
A-implies-C
  formal parameters  (one, two)
```

```
phrase : (Clause ()  ;no specializing features
           [subject] nil  ;to be filled by the mapping
           [predicate] (Verb-phrase ()
                         [verb] imply
                         [object] nil  ;to be filled in

map ((one . (subject))
     (two . (predicate object)))
```

This breakdown of the natural language (phrasal) dictionary into a structural description and a mapping is very much like the breakdown used by Bresnan [1977] notational differences aside. It is very interesting that in her "Lexicalist" theory, there are (for the most part) no transformations, that is, no operations which move constituents from one position in the tree to another. Instead, there are redundancy rules in the lexicon which permit the same general relationships to be expressed while operationally building only "already transformed" structures directly from the dictionary entry.

The modeled production process handles "transformations" in exactly the same way. Its lexicalist treatment is a direct consequence of the indelibility stipulation, i.e. carrying out a structure moving operation in the manner of standard transformational-generative theory would necessitate changing prior decisions about the relative positions of constituents and this is not allowed.

In my model, as in Bresnan's, transformational alternatives to a given choice correspond to alternate mapping functions, positioning the argument message elements at different points in the phrase. (In my model, the alternate maps are paired with alternate phrases as well, since the full specification of position is part of the phrase rather than part of the map as in Bresnan's model.) There will be several examples of transformations in the extended example.

5.5 Fixed choices - extending grammatical structure

Every natural language requires the use of particular function words, suffixes, particles, etc., which are incidental to the message and contribute only to the well-formedness of the utterance qua utterance—they are, so to speak, part of the carrier wave and not part of the signal. I have referred to these phenomena collectively as *fixed choices*, because their presence or absence can be entirely predicted from the free choices selected.

Conceptually, fixed choices can be thought of as an extension of the constituent structure that is build by the free choices. The fixed choices that a given free choice calls for are specified implicitly with the phrase which that free choice uses. When a simple clause is selected, person and number agreement between its subject and verb is selected simultaniously. When a pronoun is inserted into an oblique context the fact that the pronoun will have to be oblique is also determined. No other decisions have to be made for these grammatical "details" to be included with the output text.

Fixed choices are particularly significant in the overall design of a processing model because of their implications for timing and for what the tree is to represent. Notice for example, that at the moment a clause is selected (example on page 21) the information on which the agreement choice would be based can not be determined because the [subject] constituent hasn't been realized yet. Even if it were available, the token on which the agreement would be expressed (the verb) will not actually be spoken for some time yet because of its distance from the current position of the controller. Furthermore, locally unpredictable events may occur which block its expression entirely (in the case on page 21, the verb will have to be made a participle). Thus the choice must be "stored" until the controller in the right position.

What solution to the "storage problem" turns out to ultimately be correct is not important to the viability of this model, only that some solution is possible. The representation implemented in the computer program that instantiates this model is hypothesized to be such a solution. Briefly, fixed choices are implemented by having the controller execute procedures associated with the various grammatical annotations as instances of those annotations are reached in the tree. So, for example, when the controller enters a constituent marked infinitive-complement it executes a procedure which causes the word *to* to be spoken directly, knowing that the first item to be reached within that constituent and spoken will be a verb in the infinitive. The extended example will discuss this a greater length.

6. A longer example

The purpose of this example is to illustrate the points I have made about the operation of this model. To this end, a domain was selected where the problem of finding phrasings for message elements was minimized, and instead the emphasis would be on the influence of grammatical context.

The example is taken from a merger of my English production component with Winston's story matching program [Winston 1979]. Winston's program takes simple descriptions of stories as its input, encodes them as items in a frames data base [Roberts & Goldstein 1977], and can then compare them using various metrics. For example, by considering facts like whether a king was murdered or whether someone got married, it finds that Shakespeare's "Macbeth" and "Hamlet" are much more alike than, say, "Macbeth" and "The Taming of the Shrew"

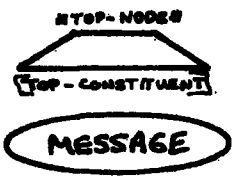
The actual "speaker" in this merger is a third program, still under development at this writing, which knows how to go about describing a story or a comparison between stories. This program uses the data structures and procedure traces of Winston's program as the "raw material" for its discourses, and uses the production component to coordinate the English realizations and to assume responsibility for grammaticality.

The message below will be used for this example. It is typical in its structure. It consists of a list of relations, collectively labeled "message" so that the production component can easily locate its dictionary entry. The relation names are part of a vocabulary both programs share (i.e. the speaker and the production component); the arguments are either references to story-program data structures or constants known to both programs.

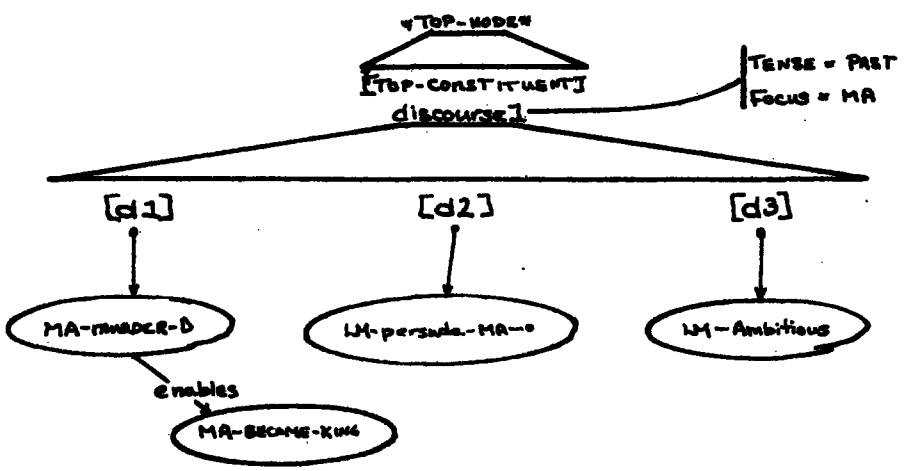
When the production component is finished with this message, it will have said:

Macbeth murdered Duncan in order to become king. He was persuaded to do it by Lady Macbeth, who was ambitious.

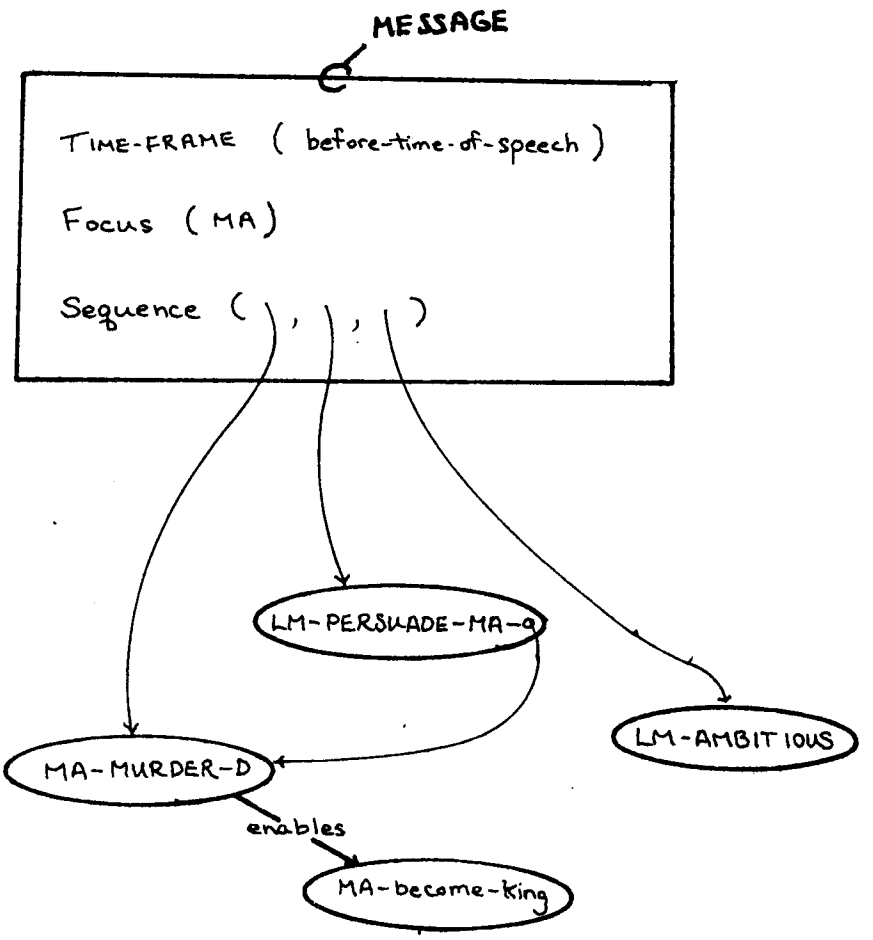
Receipt of this message activates the production component and causes the construction of the initial, trivial version of the tree:



The controller is then initialized to the top node and takes over the control of the process. Its first significant action is to use the dictionary entry for messages to determine the (immediate) realization of the input message (step four of the algorithm on page 24). That realization will then replace the message in the tree, making it the [top-constituent] as shown below.



What the message-entry did was to take the one substantive relation in the message, the sequence, and use it as the basis of the realizing phrase. The sequence relation specifies the order of presentation of its arguments and that they are to form a unit in the discourse. This is reflected in the choice to bridge the three arguments with a common node, using a category, "discourse", which does not presume upon what their own realizations will be. (A suitable base-rule for this category would be discourse -> sentence*.)



MA = "Macbeth"

LM = "Lady Macbeth"

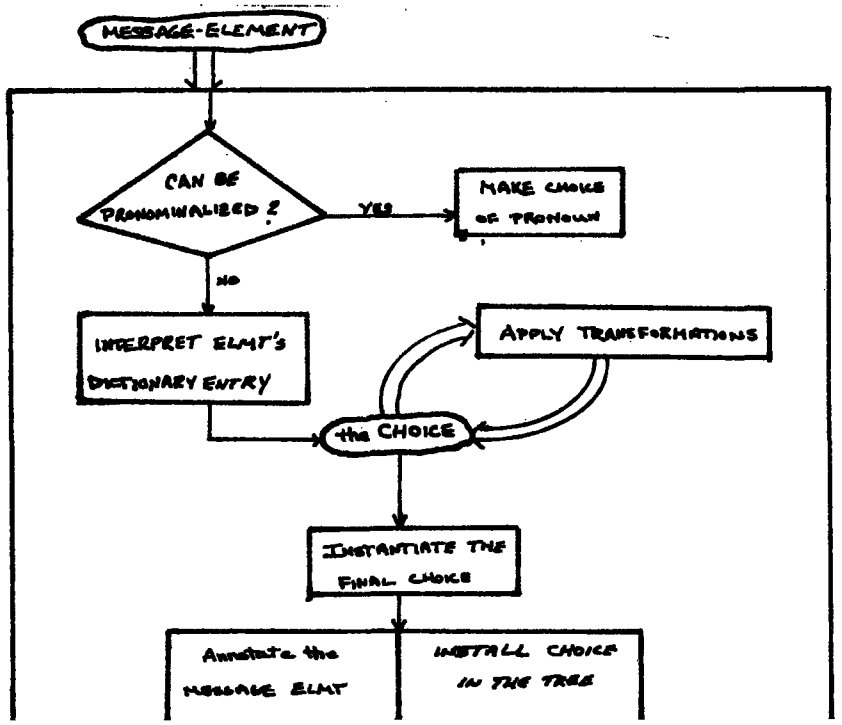
D = "Duncan"

The other two relations are effectively instructions to the production component: "during this utterance, the element in focus is to be MA"; "during this utterance, all events and states mentioned are to be referred to as happening before the time of this speech-act". Instructions like these will always be needed in a message because the data structures they reference do not contain any intrinsic values for such discourse variables, rather, they are imposed upon them by the speaker who composed the message. The very same data structure could be realized as "*Macbeth will murder Duncan*", "*who will Macbeth murder*", "*Duncan, who is murdered by Macbeth [in the 2d act]*', or "*Macbeth's murder of Duncan*", depending on the values of such variables.

The focus and time-frame instructions have their effect by influencing the realization of the "real" message elements. However, the actual performance of this influence can not take place until the controller reaches those element at their positions in the tree. The instructions must be "stored" somehow until they can be carried out. This is done by attaching properties to the discourse node, where they will be seen by the grammar routines or entry conditions that might be effected by them.

With the message realized, the controller recurses to traverse the result. The discourse node has no special grammatical actions associated with it, nor does the slot-name of its left-most constituent, [d1]. Thus the next refining action will be the realization of the first constituent, the story node MA-murder-D.

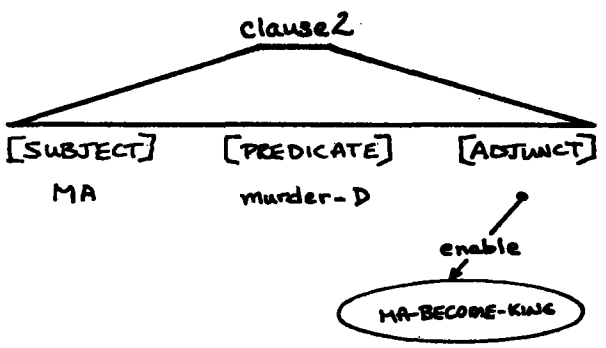
The realization procedure is initiated whenever the controller encounters a message element as a constituent. It follows this flowchart.



In this case of MA-murder-D, I will stipulate that it has not been mentioned before, thus ruling out pronominalization. Its entry is a very simple one, reflecting the uniform structure of items in the story-matcher's data base. The entry has three (sub-)message elements to distribute, MA, murder-D, and the link, enables-->MA-become-king. Its choice, common to the entries of all four of the story nodes in this message, is to have them mapped into the immediate constituents of a clause. This involves applying a common, named choice to these specific message elements, to yield a specific phrase-specification and map.

Every instance of a choice, because of its phrase structure and its parameters, is associated with a set of potentially applicable transformations. In this case, the presence of a [subject] constituent in the phrase draws transformations which, under certain conditions in subordinate contexts, would suppress its expression by pruning it away.

The transformations are governed by usage-conditions, just like the choices of an entry. These include grammatical conditions (such as apply in the adjunct to follow), and thematic conditions, such as the expression of focus. In the case of MA-murder-D, the fact that MA is in focus (rather than D) forces the use of the unmarked choice that we started with. and the choice passes to the procedure which constructs instantiated phrases from choices, yielding the clause below, which is knit into the tree in place of the message element.



The controller next recurses to traverse this new clause, clause2 Entering any clause causes the rebinding of a set of variables maintained by the controller, e.g. current-clause is bound to clause2; and current-subject to MA.

This production component, as a restriction on its power, does not include a general-purpose "tree-walking" ability in the grammar or in the predicates available to the dictionary entries, e.g. one cannot write a predicate that would examine the constituent two nodes up and one position back. Access

to any part of the tree away from the immediate position of the controller must be made via a specifically recorded pointer—such as these controller-based variables. For this reason, they assume a central place in the expression of grammatical generalizations.

Because clause2 is the first syntactic node on a path down from the routine node, a grammar routine runs and marks it as beginning a sentence. It sets a flag which will cause the next word printed to be capitalized.

Finished with those grammar routines, the controller moves to clause2's left-most constituent, tests the grammar routines associated with the slot name [subject], but none apply, and so proceeds to the realization of MA. In the story-matching domain, identifiers like MA, LM, or D are used as buckets where the characters' properties, including their English names, are accumulated. This makes the job of realizing those identifiers very simple: just retrieve their name and use it. An entry "schema" is included in the production dictionary for this purpose.

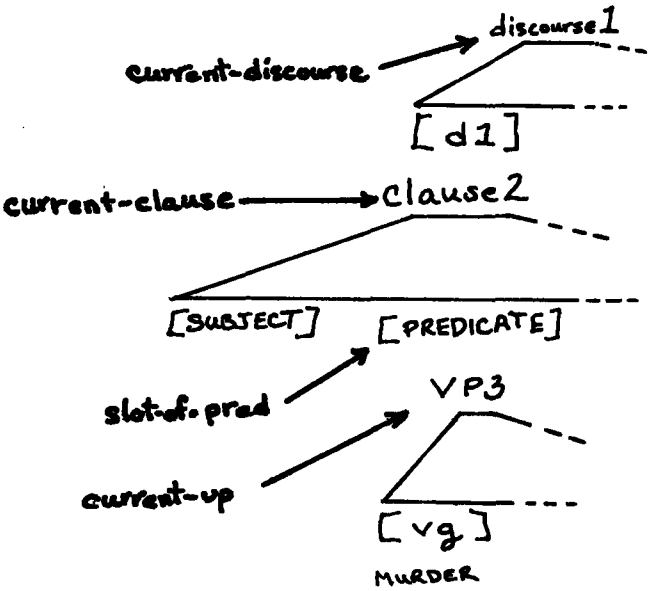
In this case, *Macbeth* is retrieved, and replaces MA as the subject. Recursing now, the controller meets it—the first word of this text, and sends it down a pipeline for morphological processing (irrelevant here, since it is a proper name) and then printing.

Moving on to the next constituent of clause2, [predicate], the grammar routine for verb phrase deletion is found not to apply (but see the next sentence) and instead its contents, murder-D are realized and the obvious verb phrase produced.

The left-most constituent of this VP is [vg], standing for "verb group", a term from systemic grammar; there is no distinct "aux" node. This choice of analysis derives from a much more basic choice about what "constituent slots" are and are not to contain, which was discussed briefly in section 5.5. Constituent slots are only to be occupied by items, words or message elements, whose source is the message. All function words, including the verbal auxiliaries, arrive in the output text as a "side-effects" of the passage of the controller through the tree—just like the indication a new sentence is starting. If this separation can be maintained in a fully-developed system, it will constitute an important, structural hypothesis to explain the commonly noted distinction between function words and substantives that is seen in speech-error data.

Associated with [vg] is a grammar routine whose function is to look up the tree and collect any stored properties that are relevant to the auxiliary structure at this level. It does not search exhaustively, but uses the variables

maintained by the controller as shown below. If there is more than one instance of a given property, the closest one is used. The properties will be found at the level where they were first noticed; this is the action that percolates them down to where they have their effect.



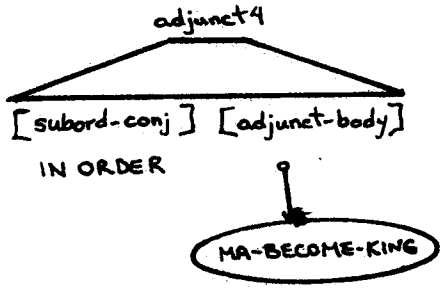
This [vg] routine is an example of procedurally encoded grammar in the sense of [Winograd 1973]. It represents the grammar of the auxiliary by being able to assemble one correctly formed when called upon in the appropriate circumstances. This is as opposed to the representation of possible constituent structures, which is declarative. The critical difference between these two modes of representation is intelligibility to other parts of the system. Possible constituent structures are easily read out and reasoned with, while the composition of the auxiliary is determined by a procedure and effectively impossible for another program to decipher. This difference amounts to a hypothesis about which parts of a grammar need to be "public" and which not.

The [vg] grammar routine is not entirely a black box. It assembles the properties it finds into a maximal constituent structure, e.g.

[TNS/MODAL/INF/PARTCP] [HAVE+EN] [BE+ING] [MVB]

This is returned to replace the bare verb as the [vg] constituent, and then traversed by the controller, who sends each item in turn to the morphological processor. This process is where the rest of the grammar of the auxiliary is located, as a procedural form of Chomsky's rule of "aux-hopping". This done to take advantage of the difference in "point of view" which makes this rule so much simpler to state in this process than in the tree-based process. The morphological processor receives each word of the text, one after the another. It does not need to traverse the empty constituents—the controller has done that for it. Thus, implemented as a simple finite state acceptor, it has a very clear notion of the "next word" relation which is central to the rule.

The direct object is processed just like the subject, leaving the controller at the adjunct. It too has a simple structure, and is realized as a specific subordinate conjunction followed by the body of the adjunct, still a message element. They are spanned by a node of type "adjunct", which is needed by the rules of this derivational system but which only serves to mark them as a unit.

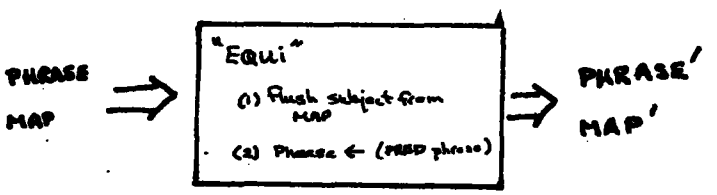


After the compound conjunction is printed, the entry for MA-become-king makes its choice:

```
(Clause-SUBJ-PRED                ⇒    PHRASE  (basic-clause)
    MA                                MAP     ((MA. (Subject))
    become-king)                               (become-king . (pred)))
```

However, this time there is an applicable transformation. The slot name [adjunct-body], which otherwise conveys no information except to a person, has two descriptive features: equi-context, and to/that. The first is a trigger for the transformation, the second will soon supply the complementizer. The transformation is a general one, firing also for participial-adjuncts, and the complements of *want*, and *persuade*. It therefore it must first check to determine whether this position is controlled by the current-subject or the current-do.

This transformation resembles "equal noun phrase deletion" in the transformational literature: the first NP constituent of the subordinate clause is compared with a NP constituent of the upper clause; and, if they are identical, the first is deleted [Burt 1971]. However, there are two important differences. The first is that it is message elements, pre-linguistic entities, that are being compared. This makes for an enormous simplification. The second is still more basic. The "transformation" applies before the subordinate phrase has even been constructed. Instead, the phrase and map of the selected choice are changed; then the phrase is instantiated directly as the correct form.



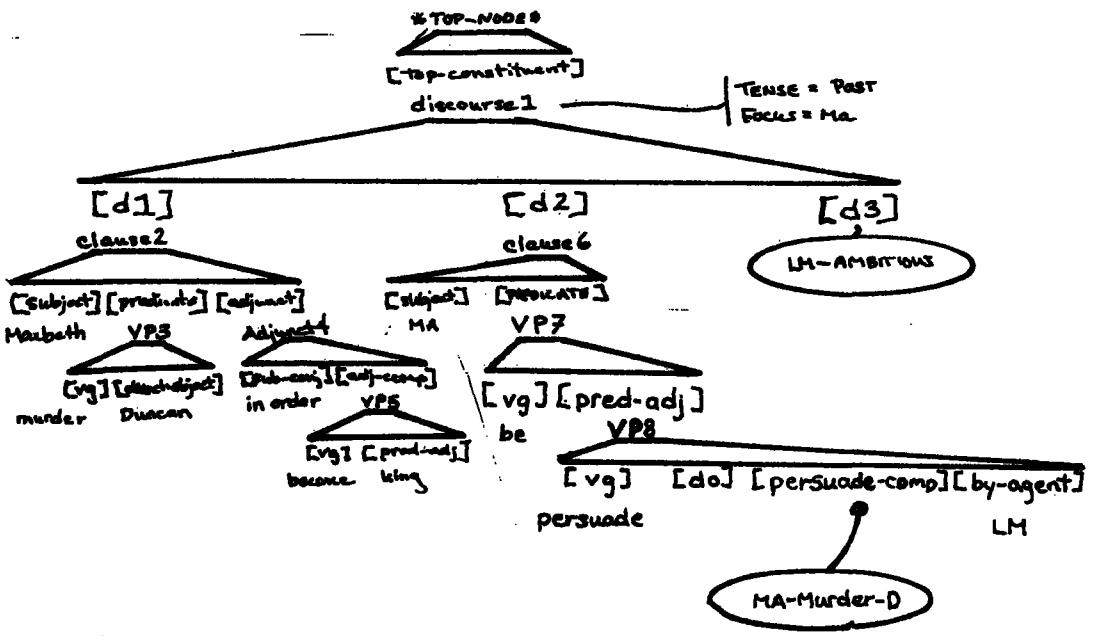
In this case some "pruning" occurs as well, as the deletion is performed by instantiating only the VP specification within the phrase. The grammar routine associated with the other slot feature, to/that, is sensitive to this distinction. It runs after the message element has been realized but before the controller goes through it, and adds the appropriate choice of complementizer directly to the output text, in this case, *to.*

The refinement of the rest of this sentence is undistinguished. The [vg] routine reacts to the name of the slot containing the current-vp and determines that tense can not be expressed.

On the way back up to the next discourse level slot, [d2], it is noticed that we are finished with the syntactic node which started the current sentence. This sets a flag to indicate that the sentence can be considered finished, but does not actually close the sentence and print the terminal punctuation, that will occur when the time comes to actually print the next word. The option has been retained to extend the sentence through (at least) the next node, which is what happens at [d3].

The contents of [d2] are, of course, the message element LM-persuade-MA-(ma-murder-d). Its entry's choice is again straight-forward: to use a clause with main verb *persuade*, and to distribute the three message elements, LM, MA, and (ma-murder-d) to [subject] [direct-obj] and [persuade-comp] respectively.

Because MA is marked as the focus of this discourse, a transformation fires to passivize the clause. A new constituent slot, [by-agent], is adjoined to the end of the verb phrase specification and LM redirected to it. MA is redirected to [subject], leaving the [direct-obj] slot vacant. The original predicate is mapped into the [predicate-adjective] of a new predicate created by an alteration to the choice's phrase specification. This is the tree after the transformed choice is interpreted.



The traversal of this clause is essentially like the previous one. The [persuade-comp] has the same features as the earlier [adjunct-body] and these will trigger the same processing, except, of course, that since the current-mvb is *persuade* and the constituent is in a complement rather than an adjunct position, the current-do is used as the controller.

The most important new thing is the first case of pronominalization. Recall that, as an intrinsic part of the realization procedure, every message element is checked for the possibility of using a pronoun for it. This check, like most, begins with a simple test that easy to apply, and only if it is met are the more "expensive" tests applied. MA is the first message element to pass through the realization procedure more than once, which is the simple test for applicability of pronominalization. (There is also an initial test for the structural conditions that permit "backwards pronominalization". See [McDonald 1978c].)

Testing for the acceptability of a pronoun involves preparing a description of the relationship between the position of this instance of the message element and the position of the last instance(s). A message element based, random access record is used to simplify the process. The record is added to automatically as the last step in the realization procedure, and makes it unnecessary to exhaustively scan back through the tree.

Once the description is prepared, a body of weighted heuristics are applied, and the decision determined from them. The fact that MA is marked as being in focus is an extremely strong vote for its pronominalization, though, in this case, since no other story characters have been mentioned up to this point, a prônoun would be chosen just on the strength of the initial test.

The next message element to pass the initial test is murder-D. Its first appearance was as the [predicate] constituent of clause2, now it is the [persuade-comp] of VP8. The reasons for "pronominalizing" a VP are no different (in the large) than those for an NP. The very same deliberation procedure, and only slightly different heuristics are applied. Of course, the form this "pronominalization" takes is very different as there are no pronouns for VP's. Instead, we must say something like "*do it*" or "*do so*", or else leave a gap. The choice is a matter of writing style and calls for a heuristic judgment. The present program selects "*do it*", because it uses the other forms only in explicitly conjoined contexts.

This "pro-form" is returned to fill the [persuade-comp] slot as the realization of murder-ma. The complementizer *to* is printed in response to the to/that feature on the slot, and then the "compound word" *do it*. The next constituent slot, [by-agent], has a grammar routine that prints the preposition *by* just before its contents (LM) are realized. This another example of the use of a procedural representation for grammatical facts, and it is a way to implement the structural distinction between grammatical morphemes and substantives.

The contents of the final discourse-level slot, [d3], are realized as a simple clause, following the model of the previous story nodes. But here, the special circumstance of being positioned at a "potential sentence end" triggers a grammar routine that operates after a message element has been realized but before it has been knit into the tree.

The routine begins its more "expensive" tests, and determines that the final NP of the previous sentence (1) included no postnominal qualifiers, and (2) originated from the same message element as will be the subject of the just constructed clause, namely LM. In this circumstance, the routine can

decide to adjoin the new clause directly to that last NP as a relative clause, postponing the end of the last sentence until (at least) the end of the new clause.

Decisions should be made when the information upon which they depend is most available. The ability to restructure the input "on the fly" serves this sort of purpose. The speaker program can assign sequence to the set of story-nodes it wants to mention without having to know anything about how they will be individually realized. General purpose heuristics can then be included in the production component to take advantage of "positional coincidences" like this one to smooth the flow of the text. On the other hand, the speaker could plan out all of the syntactic relations in detail. But to do this, it would need access to the possible outputs of the entries and their conditions and to the conditions on the transformations. It is not clear that a speaker program could perform this planning in any less time than it would take the production component to produce the text.

The analysis of relative clauses is very important in a system such as this one where sentences are "derived" from top to bottom and left to right in one pass. In a relative clause, as in a Wh-question, the "fronted" constituent, a relative pronoun, must have the same case and "animateness" as a constituent which would have appeared (but does not) at some indefinite depth within the clause (barring the effect of island constraints). To construct such a phrase correctly in one, depth-first pass, one must know, at the very beginning, (1) what the "relativized" message element is to be, and (2) what the grammatical context will be (i.e. nominal or oblique) at the point where that element would have appeared. The first is trivially acquired, it is, in this case, LM, the element being qualified by this new clause. The second can be extracted symbolically given the knowledge of which entry controls what that context will be (in this case the entry for LM-ambitious) and of what choice it will make (see [McDonald 1978b]). This small amount of lookahead is not unreasonable, because the speaker, in assembling the message, can easily accumulate the needed items. The computation is performed by a grammar routine associated with the constituent slot for postnominal modifiers.

Once the relative pronoun has been determined and printed (or not), the realization of LM-ambitious is carried out exactly as it would have been if this were a declarative sentence. The difference comes within the controller, where a flag has been set which causes a check, at each constituent slot, to compare the contents of the slot with the message element source of the NP being relativized (i.e. LM). When the first match is found, the message

element is replaced with a trace (a null word) which is bound to it. This trace is used by the number-concord routine and the pronominalization routine.

When the traversal of the relative clause is complete, the controller winds it way back up the tree to *top-node*, and the realization of the message is finished.

7. Conclusions

What kind of a process is language production? What kinds of computation are taking place? The model presented in this paper asserts that production is a constrained decision-making process.

Once this is established, it follows that a model must specify what the decisional units are, what information sources they draw upon, and how they are ordered. It must specify how the constraints arise and their effect upon the decision-making. This paper has argued that if we presume that a speaker's internal representation of his message is a compositional structure over a finite, predetermined vocabulary, then all of these specifications are related through a single linguistic structure: the surface-level, constituent tree which represents the state of the production process.

I see the most important contribution of this model to be its consideration of the effect of psychologically interesting criteria on the design of the production process. This model formalizes our common intuitions about incremental production, planning ahead, and the unconscious operation of the grammar. Earlier models which were developed to a comparable extent (i.e. which were sufficiently rich to specify a computer program) took none of these phenomena into account [Davey 1974; Wong 1975; Kempen 1977].

A synthetic model like this one is of use in a problem area like language production where direct experiments have been almost impossible, because it can supply specific, detailed hypotheses on which experiments may be based. In particular, the study of hesitations and planning units should benefit from predictions based the information limitations due to the model's one-pass control structure, and slip-of-the-tongue phenomena on division between grammar-based and message-based text elements.

8. Notes

(1) Effects of different message representations on production
Consider the fact that the internally represented relations corresponding to surface complement-taking verbs like *"want"* or *"promise"* may, *a priori*, be relations either between actors and propositions or between actors and "actions"—actor-less lambda abstractions. Certainly, computer programs have been written using both analyses with no ill effects. This variation would imply a sweeping difference in a production facility's representation of cyclic phenomena and construal relations, e.g. the use of propositions requires a "suppressing" process akin to **equi-np-deletion** while the use of subject-less actions would not—they would be realized directly as verb phrases.

(2) Only English has been considered in the development of this model. To build a production facility that is fluent and appropriate requires designers who are of native fluency in the language being used and I am only that fluent in English.

(3) Prior messages ? There is an implicit assumption being made here that the message exists in full before it is applied to the grammar. There is no evidence available that would prove this assumption. Indeed, the notion of temporal ordering is not truly sensible before there is a theory which specifies exactly how much information and of what sort a "message" contains. As it stands, the assumption that the message exists prior to any linguistic processing simplifies the process and is a convenient place to begin.

(4) Production of texts, not of speech. This model derives from research with computer programs whose output has been characters printed on a console rather than acoustic waveforms. When actual speech output is developed for these programs, it is expected that the new parts of the system will be organized along essentially the same lines used for the dictionary and syntactic grammar of the existing system, e.g. stress and intonation patterns will be named and added to the pool of possible choices and will be selected subject to contextual controls of the established sort. However, as I have

done no work with speech, I prefer to couch this model in terms of text production only.

(5) That suggestion was made by Chomsky in an unpublished talk, part of the "Technology and Culture Seminar" series given at MIT in 1974.

(6) Lookahead When I refer to lookahead limitations in people, I am considering phenomena like these: while writing the first draft of a paper you reach the end of a complex sentence and realize that the sentence would work better if it began with the phrase that you just ended it with; or, later in the paper after you make a particular point, you realize that for that point to be properly understood, you must go back to a much earlier paragraph and expand some definition. With unlimited lookahead capabilities, these problems would have been foreseen and handled correctly the first time.

(7) The choices select a single text It is not pretheoretically clear that a speaker's choices only determine one text—especially considering that typically only one or two of the several dozen choices involved may have been conscious. Perhaps "details" like the choice of subordination strategy or the position of an adverb are left "open" to be made at random. The result would be that a set of choices then determine an *equivalence class* of texts.

From the point of view of this model, how this question ultimately comes out will make no difference. The work of deciding those free choices that the speaker declined to make would be taken up by the grammar—in effect, they would become fixed choices with a random element added. However, as a matter of experimental approach, I have always designed rules on the assumption that the entire choice set was determinate. This is in part a hunch about the strong hypothesis, and in part a practical consideration—programs with random elements are notoriously difficult to debug.

(8) Grammatical dependancies The general constraints that have been developed in other frameworks, e.g. the A-over-A constraint or complex-NP

constraint are not automatically relevant to the production process. The facts which those dependencies encode must first be reinterpreted in terms of a grammar designed for production. Some may turn out not to be constraints on grammar but on messages or even to not need to be explicitly stated at all.

For example, consider the A-over-A constraint found in some versions of transformational generative grammar. It stipulates how the structural description of a transformation is allowed to match with constituents within the phrase marker when matching a category type of which there are two, and they are self-embedded within that phrase marker. Namely, it must bind to the outer, "higher", of the two candidates.

In the production model, this action (i.e. manipulating the dominating element before its subordinates) falls out automatically as a consequence of the standard pattern of control flow. No decisions can ever be made about the refinement of a sub-token within some relation already positioned in the tree until that relation is refined and the sub-token "exposed".

(9) Realizing the quantifier Given a relation and its arguments, one can do other things besides build a sub-tree and embed the arguments. In particular, to realize an operator such as a quantifier or negation the correct operation is often to just return the argument, having annotated it in such a way that when that argument is realized, e.g. the variable quantified over will be in focus (*"Everything that is a man is mortal"*). That is, the argument provides the body of the text and the operator is expressed within it as an adverb, a determiner, or a special ordering.

9. References

Bresnan J. (1977) "A Realistic Transformational Grammar", in Halle, Bresnan, & Miller eds. **Linguistic Theory and Psychological Reality**, MIT Press.

Burt M. (1971) **From Deep to Surface Structure** Harper & Row.

Davey A. (1974) **The Formalization of Discourse Production** Ph.D. Dissertation, University of Edinburgh.

Fodor J., Bever T., Garrett M. **The Psychology of Language** McGraw-Hill.

Garrett M. (1975) "The Analysis of Sentence Production" in Bever ed. **The Psychology of Learning and Motivation**, vol. 9, Academic Press.

_____ (1976) "Syntactic Processes in Sentence Production" in Walker and Wales eds. **New Approaches to Language Mechanisms**, North-Holland.

Genesereth M. (1978) **Automated Consultation for Complex Computer Systems** Ph.D. Dissertation, Harvard University.

Goldstein I. (1978) "Developing a Computational Representation for Problem Solving Skills", memo 495, MIT Artificial Intelligence Lab.

Halliday M.A.K. (1970) *Functional Diversity in Language As Seen From a Consideration of Modality and Mood in English* Foundations of Language 6, 322-361.

Kempen G. (1977) "Building a psychologically plausible sentence generator" presented at the Conference on Empirical and Methodological Foundations of Semantic Theories for Natural Language, Nijmegen, Netherlands.

Marcus M. (1977) **A Theory of Syntactic Recognition for Natural Language** Ph.D. Dissertation, Dept. of Electrical Engineering and Computer Science, MIT.

McDonald D.D. (1978a) "How MUMBLE Translated the Barber Proof" unpublished manuscript, MIT Artificial Intelligence Lab.

_____ (1978b) "A Simultaniously Procedural and Declarative Data Structure and Its Use in Natural Language Generation" in the proceedings of the Second National Conference of the CSCSI/SCEIPO, University of Toronto.

_____ (1978c) *Subsequent Reference: Syntactic and Rhetorical Constraints* MIT A.I. memo 500.

_____ (in preparation a) **Natural Language Production as a Process of Decision-making Under Constraint**, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT.

_____ (in preparation b) "Three Squibs on Language Production".

Perrault and Cohen (1977) *Planning Speech Acts* technical report, Dept. of Computer Science, University of Toronto.

Roberts B. & Goldstein (1977) *The FRL Manual* MIT A.I. memo 409.

Ross J. (1967) **Constraints on Variables in Syntax** reproduced by the Indiana University Linguistics Club.

Slochum J. (1973) *Question Answering via Canonical Verbs and Semantic Models* technical report NL 13, Dept. of Computer Sciences, University of Texas.

Swartout W. (1977) "A Digitalis Therapy Advisor With Explanations" TR-176, MIT Laboratory for Computer Science.

Winograd T. (1973) **Understanding Natural Languge** Academic Press.

Winston P. (1979) *Learning by Understanding Analogies* MIT A.I. memo 520.

Wong H. (1975) "Generating English Sentences from Semantic Structures" Technical Report 84, Dept. of Computer Science, University of Toronto.