Language Generation: Applications, Issues, and Approaches

KATHLEEN R. MCKEOWN

Invited Paper

As computer systems become more sophisticated they must be able to communicate their results successfully to their users. Natural language generation is the area of research concerned with developing methods that will allow a computer system to respond to its user in human language. In this paper, the need for natural language generation is first motivated by showing how it is used in several applications. Given that language generation is necessary for such systems, the paper also focuses on the issues that must be taken into account in developing a system that can generate language. Finally, techniques that have been used in two question-answering systems, the TEXT system [21] and TAILOR [22], are discussed.

I. Introduction

As computer systems become more sophisticated they must be able to effectively communicate their results to their users. Many potential users of complex computer systems are "naive" and "infrequent" users; that is, not only are they unfamiliar with the computer and the formal languages available to interact with it, but their planned use of the system is infrequent enough that it does not warrant the time needed to learn a formal language. Many potential users of database systems, information systems, and expert systems fall into this category. Thus much research in natural language has focused on developing facilities that allow querying of such systems in human language. But users can only successfully take advantage of this new world of information and tools if they understand the response they receive. This, then, implies not only capability on the part of the machine to accept instructions in everyday terms but also to reply in kind.

Natural language generation is the area of research concerned with developing methods that will allow a computer system to respond to its user in human language. In this paper, the need for natural language in several different applications is discussed first. Given that language generation is necessary for such systems, the paper focuses next on the issues that must be taken into account in developing a system that can generate language. In particular, how can a system decide what information to communicate, when to say what, and which words best express its intent? Finally, techniques that have been used in two question-answering systems, the TEXT system [21] and TAILOR [22], are presented.

Manuscript received September 13, 1985; revised February 10, 1986. Research in language generation at Columbia University is partially supported by ARPA under Grant N00039-84-C-0165.

The author is with the Department of Computer Science, Columbia University, New York, NY 10027, USA.

II. The Need for Language Generation

One main area where language generation has proved valuable is as part of interactive systems. These include question-answering systems, which allow a user to both ask a question and receive an answer in natural language, as well as systems that can provide explanations of their reasoning but offer the user only limited ways to request such explanations. Expert systems and computer-aided instruction systems fall into this last category.

A. Question-Answering Systems

For many years, the problem of question answering was seen as primarily a parsing problem. A user's question was translated into a formal query, whether a database query, a formal logic representation of the question, or a specialized AI language representation. The question was answered by doing a search of the underlying database or knowledge base, as specified by the formal query. The results of the search were simply presented to the user, using list or table format and sometimes embedded within a sentence.

There are many questions, however, that cannot be answered by a simple search of the underlying knowledge base. For example, it has been shown [15], [29] that many users of database systems, particularly naive and infrequent users, need to ask questions to familiarize themselves with the database before asking specific questions about its contents. Such users need to know what information is available in the database (e.g., "What kind of data do you have?"), what specific terms mean in the context of the database (e.g., "What is production cost?"), or what the differences are between different terms (e.g., "What's the difference between manufacturing and production cost?"

0018-9219/86/0700-0961$01.00 ©1986 IEEE

In spatial domains[1] that we have studied, questions that require description of a physical object also require more than a database search. Questions such as "Describe the disk drive?" or "What are the parts of a telescope?" can be answered either very succinctly or by providing a great deal of detail.

In general, high level questions that do not precisely characterize a required piece of information cannot be answered by a simple database retrieval. Such questions are often termed *meta-level* questions because the information required to answer them is not found in the database itself, but rather in a meta-level description of the database.[2] Meta-level questions would include at least the following classes of questions:

- requests for definitions
- requests for available information
- questions about the differences between objects
- requests for object descriptions.

Since meta-level questions have no corresponding formal query which can produce the content of the response, a language-generation component is required to determine the appropriate content, organization, and expression of the response. Note that natural language is particularly appropriate for answering such questions as they require definitions, descriptions, and longer textual sequences. Moreover, the knowledge base contains a large amount of information that could potentially be included as part of an answer. Exactly what information is appropriate for a response will vary from one situation to another, depending in some cases on how knowledgeable the user is about the domain.

B. Expert Systems

Communication with the user in expert systems has been needed primarily to explain the reasoning used by the system in producing its advice. Textual explanation has proved crucial to the success of expert systems for several reasons.

First, expert system users are often not computer scientists and would be unable to follow a formal representation of the system's reasoning. For example, users of medical expert systems are doctors and medical students. Natural language is a mode of communication that is familiar to users such as these who may not want to take the time to learn other modes of interaction.

While not experts in the programming methodology of expert systems, users are often experts in the domain of the system. Again, doctors fit this characterization. Their purpose in using the system is often for consultation: to gain advice on a case or to confirm their own diagnosis. In order to evaluate the advice provided and to determine whether to accept it or not, such users need to be able to understand both how and why the system came up with its advice.

Builders and maintainers of expert systems are now pointing out the value of textual explanation in identifying errors in the underlying inferencing process. Often a trace of the inference process itself can be so lengthy (for example in some systems [27], a single recommendation may invoke up to 15 000 individual production rules) that errors are difficult to detect. Often, a system is constructed incrementally by a number of different researchers who may not follow the conventions used previously. In such cases, explanation facilities have been shown (e.g., [12]) to point out even such simple discrepancies as errors due to round-off which had gone undetected.

C. Noninteractive Applications

Language generation is also used for noninteractive applications such as abstracting of technical texts and summarization of stories [14]. In these applications, the generation system uses an internal formal representation of the text or story and must select and abstract information to include in the summarization. As with the answering of high-level questions and generation of explanations, the process involves determining which information should be included as well as how to express it.

III. Problems in Language Generation

Given that there is a need for language generation in a variety of applications, what issues must a designer of a language generation system take into account? To get a feeling for what a language generation theory must handle, consider an example of the kind of text a system that generates definitions should be able to produce (see Fig. 1).

Flagey-Echezeaux *(France)* Important red wine township in the Cote de Nuits with two front-ranking vineyards, Echezeaux and Grands Echezeaux. The first produces a fine rich, round wine and the second, which is not a single vineyard but a group, is also capable of producing fine wines but, like other divided properties, the quality of its wine is variable. The lesser wines of Flagey-Echezeaux are entitled to the appelation Vosne-Romanee.

Fig. 1. Naturally occurring definition.

This text (taken from *The Hamlyn Pocket Dictionary of Wines* [23]) was written for the explicit discourse goal of *defining* Flagey-Echezeaux. It presents information relevant to that goal in a comprehensible organizational framework. What must a generation system take into account to generate a text such as this one, given a specific discourse goal? To illustrate these issues, we will consider how problems in language generation differ from those of language interpretation and show the range of choices a generation system must consider.

Although there is research that suggests that the same information can be used both for interpretation and generation of language (e.g., [11], [31], [32]), there are some important distinctions that can be made about the processes required for each task. Interpretation of natural language requires examination of the evidence provided by a particular text in order to determine the meaning of the text and intentions of the writer who produced it. It necessitates using that evidence to examine the limited set of options the system knows to be available to the writer to determine the option actually taken. For example, in interpreting the second sentence of Fig. 1, a system would use the evidence that "produce" occurs in the active form to determine that

[1]i.e., domains containing detailed information about physical objects: what they look like, their subparts, how subparts are geographically related to each other, etc.

[2]A meta-level description might list the objects found in the database, their defining characteristics, and database attributes. The database itself contains values of attributes.

"a rich round wine" is the object being produced and Echezeaux (to which "the first" refers, one of many problems for interpretation not discussed here) is the agent that does the producing.

While interpretation involves specification of how a speaker's options are limited at any given point (for example, by writing grammars), it does not require a formulation of reasons for selecting between those options.[3] Thus in interpreting sentence (2) of Fig. 1, a system does not consider *why* the writer used the active form as opposed to any of the other options available at that point.

In generation of natural language, however, this is exactly what is required. A generator must be able to construct the best utterance for a given situation by choosing between many possible options involving a wide range of knowledge sources. To produce the second sentence of the example, a generator must decide that although both the active and passive forms are possible (the passive would result in "a fine rich, round wine is produced by the first"), the active is better than the passive. Furthermore, the generator must have a principled reason for making that decision, which it can use in all similar cases. Where research on interpretation may describe limitations on options in order to more efficiently determine the option taken, research in generation must specify why one option is better than others in various situations.

The choices that a language generator must face include options regarding the content and textual shape of what is to be said and choices in the transformation of the message so determined into natural language. A language generation system must be able to decide *what* information to communicate, *when* to say what, and *which* words and syntactic structures best express its intent. In the last of these stages, local decisions such as syntactic and lexical choices are made, often using a grammar and dictionary to do so. It is in this stage that the active form would be selected for sentence (2) of the example in Fig. 1. Until recently, this has been considered the extent of language generation research. But determining what to say and how to put it together above the sentence level also introduce language issues that must be addressed by any speaker or writer of extended discourse. These three classes of decisions are all part of the language generation problem.

If connected text (and not simply single sentences) is to be generated, issues of discourse structure and discourse coherency are particularly important. Generation of text requires the ability to determine how to organize individual sentences. A writer does not randomly order the sentences in his text, but rather, plans an overall framework or outline, from which the individual sentences are produced. This is obvious in Fig. 1. The author has chosen an organizational framework that is appropriate for providing definitions. Here, he first identifies Flagey-Echezeaux by describing its superordinate ("important red wine township in the Cote de Nuits") and then introduces two of its constituents (Echezeaux and Grands Echezeaux). Next, characteristic descriptive information about each of these vineyards is presented in turn, and finally, the author presents additional information about Flagey-Echezeaux (the item being defined) in the last sentence.

[3]Note that as interpretation systems become more sophisticated, the analysis of reasoning behind the selection of a choice may be helpful in determining the goals of the speaker.

To generate texts that are well organized, an analysis of the kinds of structures that are appropriate for providing definitions and other kinds of texts is needed. In general, in any situation where text must be produced, we will call the purpose for producing the text the *discourse goal*. For example, the discourse goal for Fig. 1 is *define* since the author's purpose in writing the text is to provide a definition of Flagey-Echezeaux. Other discourse goals would include *describe*, *compare*, *support* (as in an argument), and so forth. The kind of structure that is appropriate for producing a text will vary depending on the discourse goal; while one type of structure may produce an effective definition, it may produce a very poor argument. Thus a first step in building a generation system is to analyze texts that were written for the same discourse goals for which the system will generate texts. In this way, structures that people successfully use for producing text will be identified. These structures are termed *discourse strategies*.

In addition to identifying such strategies through analysis, methods are needed for formalizing the results so that they can be used by a computational process. While the description given several paragraphs back of the structure used in Fig. 1 is adequate for us as readers to follow, it cannot be used as is by a computer. Instead it must be specified very precisely using a formal representation so that it can be embodied as part of a computer program.

A second main requirement for generated text is discourse coherency: the computational process must produce a text that is in some sense a unit. This means that only information that is relevant to the discourse goal is included and that each sentence must be semantically related to the previous text. In Fig. 1, only information supporting the definition of Flagey-Echezeaux is included in the text. This is due partly to the fact that the author only considers information that is related to Flagey-Echezeaux, but it is also due to the organizational strategy he has chosen. It dictates that information about each of the two constituents be included and not information about the Cote de Nuit, for example. Furthermore, each sentence relates to the previous sentences.

IV. A Model for Language Generation

In this section, a model for language generation that has been used successfully as part of the TEXT system, and later extended in TAILOR, is presented. In this model, processing is divided into two phases. In the first phase, embodied in the *strategic* component of the system, the content and order of the text are determined. All decisions about what to include in the text and when to include it are made. The output of the strategic component is an ordered message, which is passed to the *tactical* component. In this second phase, a grammar and dictionary are used to determine how to express the message in English. The actual words of the text are chosen and strung together as sentences.

This separation of conceptual and linguistic decisions allows focus on problems in one phase or the other. Much of previous work in language generation has focused on problems in the tactical component, under the assumption that some other part of the system would determine what to say. This has ranged from work on direct translation of an underlying formal representation [5], [26], the development of grammars and mechanisms for using those grammars to produce language [11], [17], [20], and the development and representation of criteria for making decisions about vocabulary as part of a dictionary [9], [20]. In TEXT and TAILOR, this separation of conceptual and linguistic decisions has allowed focus on conceptual problems that had previously gone unaddressed, although we have developed a tactical component as well.

It should be noted that while many researchers have used and continue to use this division of processing, there is also interest in examining the interaction that must occur across the boundaries [10], [24]. For example, in Ritchie's model [24], two separate components for conceptual and linguistic decisions are maintained. While the conceptual module is invoked before the linguistic one, on the generation of noun phrases, the linguistic module re-invokes the conceptual one to provide more details on the object the noun phrase must refer to. Similarly, Hovy's model of generation [10] specifies points at which interaction between the modules must occur. And while Mann's [17] efforts have focused mainly on the development of a grammar for generation, the systemic formalism that they are using calls for choice points at which other parts of the system (such as the knowledge base or the text planner) are queried to provide further information. Finally, in recent work of our own [4] we looked at the influence of vocabulary on the order of the text showing how earlier conceptual decisions can be retracted if warranted.

Other researchers have developed integrated models of language generation with no clear separation between phases (e.g., Appelt [1]; Danlos [7]). Appelt characterizes language generation as a planning problem. He shows how a planning paradigm can be used to solve problems at all levels of the generation process, including linguistic decisions. This unified process means that decisions at any level can influence both earlier and later decisions through the use of backtracking. Danlos, on the other hand, claims that such extensive and unpredictable interaction is required between conceptual and linguistic decisions that no general principles ordering these decisions can be developed. Instead, for each new domain, a new ordering of decisions must be developed. In the terrorist domain in which she works, decisions about vocabulary are made before the order of the text is determined.

A. Strategy: Deciding What to Say and When to Say It

If the content of a response is not predetermined by a search of the knowledge base, the text generation module must be able to determine what information to convey given a request for communication. For certain questions, such as requests for definitions in the database domain, there may be a potentially large amount of information that could be used to answer the question. The system must be able to filter out information in its knowledge base which can be ignored and pinpoint information which should be included.

One way in which information can be filtered out for inclusion in the text is by making use of a *discourse strategy* such as the strategy used for the discourse goal *define* in Fig. 1. By identifying the strategies that people commonly use for discourse goals and encoding them formally, a generation system can use them to aid in determining the order and content of the texts it generates. In the next sections, the use of discourse strategies in two systems, TEXT and TAILOR, is discussed.

1) The TEXT System: TEXT is part of a natural language interface to a database system and provides paragraph length responses to questions about database structure. It can respond to three types of high-level questions: requests for definitions, questions about available information, and questions about the differences between objects. The database used for TEXT contains information about military vehicles and weapons.

One of the strategies formalized for TEXT is the *constituency* strategy that was used in Fig. 1 for defining Flagey-Echezeaux. This same strategy was identified in many naturally occurring texts for the purposes of definition and thus could be abstracted out as a standard pattern. It is characterized by four main steps:

1) Identify the item as a member of some generic class.
2) Present the constituents of the item to be defined.
3) Present characteristic information about each constituent in turn.
4) Present additional information about the item to be defined.

This strategy is formalized in TEXT as a *schema* using a graph representation. The *constituency* schema is shown in Fig. 2.[4] Each arc of the graph represents one of the steps above and is labeled by a predicate which characterizes the type of information required. The graph begins with the **identification** predicate, indicating identification of the generic class is required. The **attributive** predicate is an alternative that will only be taken if the discourse goal is not *define*. The second arc is labeled **constituency** and indicates that the constituents, or subclasses, of the item should be included next. Step 3 is represented by the two arcs emanating from the state CONST/CONST and the arc from state CONST/ID. These arcs indicate that **identificational** or **attributive** (i.e., attributes of an object) information and **evidence** (e.g., attributes supporting an object's classification in the database) are to be provided next. The two arcs going to the final node in the graph, CONST/END, represent step 4 and indicate that **attributive** or **analogy** information is to be provided.

To generate the content of a response, TEXT traverses the schema graph. Each predicate in the schema has a function associated with it which retrieves information matching the predicate from the underlying knowledge base. For example, the **identification** predicate has an associated function which takes as input the object to be identified and returns a *proposition* which includes the object to be identified, its superordinate, and any defining attributes. The information extracted for a single predicate will eventually be translated to a single sentence. As TEXT traverses an arc, it extracts information from the underlying knowledge base using the

[4]Four schemata were developed and implemented in TEXT. See [21] for a description of the others.

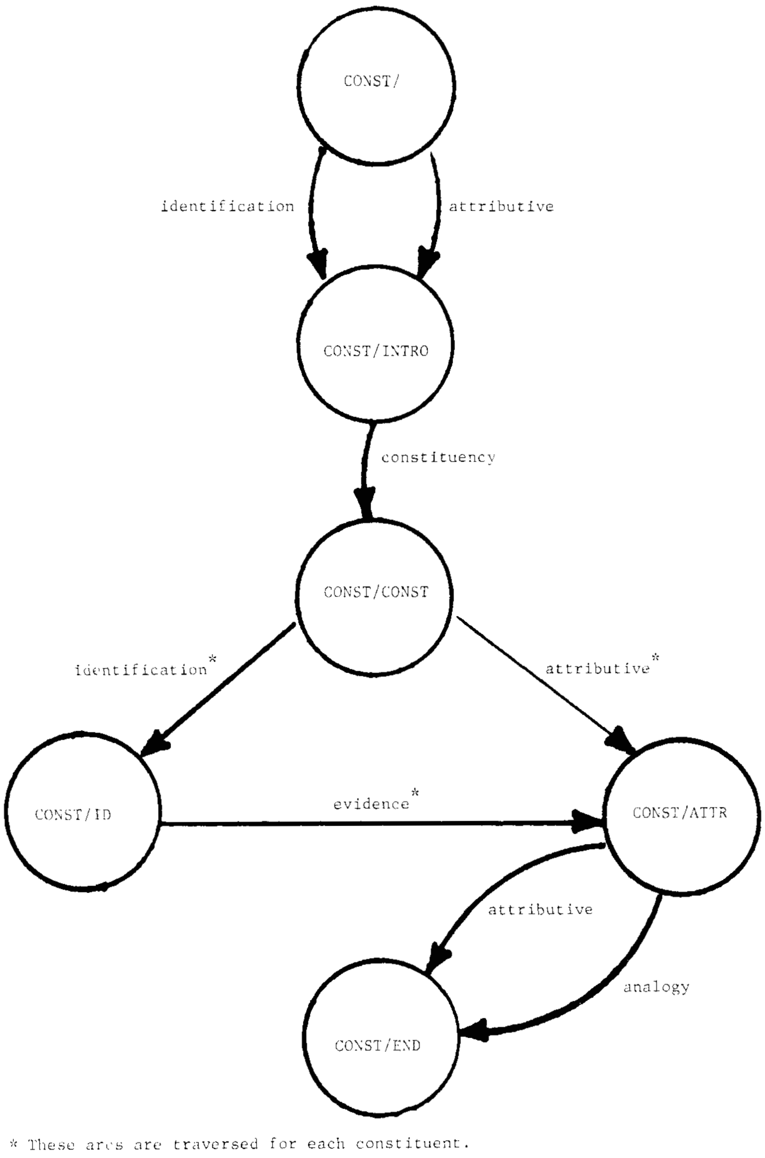

Fig. 2. The constituency schema.

function associated with the predicate labeling the arc. Where there are alternatives in the schema (several arcs emanate from a single state) each predicate is matched and the information that ties in best with the text it has already generated is selected (see [21]).

TEXT used the *constituency* schema to generate the paragraph shown in Fig. 3 in response to the question "What is a guided projectile?" The numbered predicates shown were used to extract information for the corresponding sentences.

2) TAILOR: TAILOR [22] is a question-answering system that was developed for RESEARCHER [13], a system that reads patent abstracts, builds a knowledge base representing what it has read, and generalizes from different patents to learn abstract concepts relating the different objects it has read about. Since RESEARCHER primarily contains information about physical objects, its question-answering component must be able to respond to high-level questions requesting descriptions of the objects.

One main problem for this domain is determining the

```
(definition GUIDED)
;
;What is a guided projectile?
;

Schema selected: constituency

1. identification: guided projectile
2. constituency: guided projectile
3. identification: missile
4. identification: torpedo
5. evidence: missile
6. evidence: torpedo
7. attributive: guided projectile

Message through dictionary. Entering tactical component


    1. A guided projectile is a projectile that is self-propelled. 2. There are 2 types of guided
projectiles in the ONR database: torpedoes and missiles. 3. The missile has a target location in
the air or on the earth's surface. 4. The torpedo has an underwater target location. 5. The
missile's target location is indicated by the DB attribute DESCRIPTION and the missile's flight
capabilities are provided by the DB attribute ALTITUDE. 6. The torpedo's underwater
capabilities are provided by the DB attributes under DEPTH (for example, MAXIMUM
OPERATING DEPTH). 7. The guided projectile has DB attributes TIME TO TARGET &
UNITS, HORZ RANGE & UNITS and NAME.
```

Fig. 3. Response generated by TEXT.

amount of detail that should go into any given description. For example, given a patent about a disk drive with improved air flow, information is available in memory to describe it in either of the two ways shown in Fig. 4 below. Depending on the user's knowledge of the domain, one of the two answers will be more appropriate.

1) *It consists of* an enclosure, 2 filters, and air guide means for directing air flow.

2) It has a normal enclosure, 2 filters of the same type (breather filters) making this a fairly simple filter system in contrast to other disc drives having multiple filter types. The location of the 2 filters (one on enclosure top and one on enclosure bottom) plus an air guide means causes air to flow from the outside, pass directly on the spindle and disc, and pass out the bottom filter to the outside. By providing positive air pressure through this air flow in combination with an obstruction, a positive air pressure differential is created causing improved air cleanliness.

Fig. 4. Two descriptions of a disk drive.

To determine how much detail is appropriate for different users, Paris [22] analyzed encyclopedia entries for various physical objects, using entries for the same object from both adult and junior encyclopedias.[5] She found that rather than provide more detail for the naive reader and less for the expert (or *vice versa*), an approach that had been taken previously in natural language generation, different kinds of detail were given for the different readers. This difference in detail could be captured by two different discourse strategies. For adults, or domain experts, a description of the object's parts was given and its structure could be captured by the *constituency* schema. Given a request for a description of a telephone, her system can currently produce a response for the domain expert as shown in Fig. 5.

identification: telephone

constituency: telephone

constituency: transmitter

identification: housing

identification: line

constituency: wire

The telephone is a device. The telephone has a transmitter, a housing, a line and a receiver. The transmitter has a doubly resonant system and a diaphragm. The housing is a kind of cover. The line is a wire. The receiver has a diaphragm, an air gap and an electromagnet.

Fig. 5. Expert description generated by TAILOR.

For novices, on the other hand, encyclopedia entries generally provided a description of the process that makes the object work. Paris has formalized this strategy by indicating how the system should trace the causal relations in the underlying knowledge base to produce the text. Given the same request for a description of a telephone, her system will produce a response for the novice as shown in Fig. 6. The propositions output from the strategic component are shown with the generated English along side it. Each proposition consists of a causal relation (labeled by a unique identifier such as &MR0 and a mnemonic such as M-CAUSES) between two other relations (again labeled by a unique identifier such as &REL3 and mnemonic P-SPEAKS-INTO). Thus proposition 1 indicates that one relation (speaking) causes another relation (hitting). Only part of the generated text for the novice is shown. The example

[5]The adult is assumed to have more domain expertise in any given domain than the child.

```
1) &REL3 (P-SPEAKS-INTO):                    ; That a person speaks
       subject : (&MEM27) {ONE]
       object  : (&MEM2) [TRANSMITTER]

   ===>   &MR0 {M-CAUSES}                    ; causes

&REL4 (P-HITS):                              ; waves to  hit
       subject : (&MEM28) [SOUNDWAVES]       ; the diaphragm.
       object  : (&MEM3) [DIAPHRAGM-T]

 --------

2) &REL4 (P-HITS):                           ; This fact causes
       subject : (&MEM28) [SOUNDWAVES]       ;
       object  : (&MEM3) [DIAPHRAGM-T]

   ===>   &MR1 {M-CAUSES}

&REL5 (P-VIBRATES):                          ; the diaphragm to
        subject :                            ;  vibrate
        object  : (&MEM3) {DIAPHRAGM-T]

 --------

3) &REL5 (P-VIBRATES):
        subject :
        object  : (&MEM3) {DIAPHRAGM-T]

   ===>   &MR2 {M-EQUIVALENT-TO}             ; in the same manner as

&REL8 (P-VIBRATES)                           ; the molecules of air
        subject :                            ; are vibrating.
        object  : (&MEM26) {AIR-MOLECULES]
```

Fig. 6. Journal for the process trace.

continues by describing the substeps of each relation. See Paris [22] for the full example.

3) Other Influences: Other generation systems have made use of concepts similar to discourse strategies. In earlier work, Weiner [30] made use of an explanation grammar to generate text. His grammars consisted of fewer predicates than TEXT's strategies and were developed for the discourse goal of providing explanations. In more recent work, following the development of TEXT, Mann [18] did an extensive analysis of a large corpus of texts to discover strategies that were used. The schemata resulting from his analysis consist for the most part of a main predicate and a satellite, they do not specify the order of the two predicates, and they can be combined recursively with other schemata to produce a large variety of structures. Thus strategies have been used in a number of systems as one method for determining the content and order of generated text.

There are other influences in addition to discourse strategies on determining the content and organization of a generated text. In each sentence of a text, a writer centers his/her attention on one object (or event) over others. This act of centering one's attention is called *focusing*. In TEXT, a representation of what the system is focusing on in each sentence and how the system's focus shifts as the text is produced is maintained in order to avoid having the text jump around from one topic to another. By singling out one object in each sentence as the system's focus and using a set of rules dictating when and how the system can change focus, TEXT can rule out pieces of information to add to its text that do not conform to its set of rules. TEXT uses these rules to choose between alternatives in a schema.

For example, one of TEXT's focus rules states that if the system must choose between continuing to focus on the same object and returning to focus on an object that was in focus earlier, it is better to continue to focus on the same object. This rule guarantees that the system will finish what it has to say on a current subject before returning to an earlier one. In the *constituency* schema, the use of this rule means that the system presents all information about an object's constituents before returning to talk about the object itself.

Other researchers have shown the influence of knowledge representation on the generated text. A text generation system cannot say more than it knows about, as represented in its knowledge base. In order to generate particular types of text, it may be necessary to specify additional information to add to the knowledge base. Both Swartout [28] and Clancey [6] showed that support knowledge for the rules in an expert system must be added in order to produce acceptable justifications of the system's advice. Swartout made use of domain principles to provide suitable explanations for doctors as part of the Digitalis Therapy Advisor, while Clancey added information about the structure of domain knowledge and system strategy in order to create a tutorial expert system, GUIDON, based on MYCIN [25].

Finally, depending on who the system is talking to when a question is asked, different information will be relevant. Appelt [1] has shown how information about the current user's beliefs and knowledge should influence what the system says. While Appelt assumes the user is a novice, Appelt's system also explicitly keeps track of facts the user knows about. It learns about such facts through the conversation: by the statements or questions the user asks as well as by what the system tells the user. A simple way that the system can make use of such knowledge is to avoid telling the user what s/he already knows. On a more sophisticated level, by using such knowledge, the system may be able to satisfy several goals in a single utterance. For example, by saying "Use the wheelpuller" while pointing to a tool on a nearby table, a speaker is able to indicate what tool to use next in a task while at the same time identifying the wheelpuller for the listener. Similarly, in TAILOR we have shown how information about user type, whether naive or expert, can influence how much detail to include in a text. If TAILOR is conversing with a user who knows very little about the domain (a novice), detail about process information is given, while if the user is a domain expert, details about object parts are given.

B. Tactics: Deciding How to Say It

The text generation system must also be able to determine what the surface text should look like. This involves making decisions about what vocabulary to use (and in particular, how to choose between synonyms), when to use a pronoun instead of a full noun phrase to refer to an object or concept, whether to use a sequence of simple sentences or to combine several simple sentences into a single complex sentence, and how to arrange the words in each sentence. Almost all of these decisions are influenced by syntactic constraints on language and thus one component of a language generation system is a grammar.

In TEXT and TAILOR, we have developed a functional unification grammar [11] to transform the message produced by the strategic component into natural language. A dictionary is also used in this process to determine what words to use in the text.

To see how the actual English is produced, consider the output for the process strategy used in TAILOR, shown in Fig. 6. The tactical component will be invoked separately for each numbered proposition in the output and will produce a single sentence for each.[6] For a single proposition, the dictionary will be accessed to determine the verb and its arguments (which eventually will translate as the subject and object of the sentence). Before invoking the grammar, the first proposition is represented as shown in Fig. 7. All vocabulary has been chosen for the sentence, but how the words will be combined syntactically has yet to be decided. Currently, the verb is selected based on the semantics of the predicate it represents and the semantic features of its arguments.

```
((cat s)
 (verb ((v === cause)))
 (prot ((embed ((relpro === that)
                (verb (v === speak)))
               (prot ((n === person) (article === indef)))))))
 (goal ((embed ((verb ((v === hit) (aspect inf)))
                (prot ((n === wave) (number plur)))
                (goal ((n === diaphragm) (article ===def))))))))
```

Fig. 7. Intermediate representation of Proposition 1.

We plan to have user-type influence the choice of vocabulary as well as the choice of strategy. A domain novice requires less technical vocabulary than does a domain expert. We found that terminology varied significantly between a junior encyclopedia entry and an adult encyclopedia entry for the same object. For example, when discussing the core of a transformer, the adult entry described it as being composed of "laminated steel" while the junior entry indicated it consists of "many layers of thin strips of steel." Our program should also be able to choose vocabulary according to the user's background.

To produce the actual sentence, the intermediate representation shown in Fig. 7 is *unified* with the grammar which is represented in the same formalism as the input. The unification process is based on the unification process used for resolution theorem proving. During the process the syntactic structure of the sentence is constructed, choices such as whether the active voice or passive voice should be used are made, and the tree structure so constructed is linearized to produce the sentence "That a person speaks causes waves to hit the diaphragm."

1) Other Approaches: Slightly different models for the tactical component are used in other research. In a PROLOG definite clause generator developed at Columbia [8], a list of propositions is input to the generator instead of one proposition at a time. Each proposition is represented using basically the same formalism shown in Fig. 7, but one argument of each proposition is singled out as the focus of that proposition. The generator has a set of rules that make use of the information to combine several propositions into a single complex sentence using relative clauses and conjunction when appropriate. When its rules indicate that

[6]While the grammar and strategic component are currently operational, the interface between the two, including the dictionary is partially complete. The grammar for TAILOR was based on TEXT's grammar, but extended by Kwee Tjoe Liong.

a complex sentence is not appropriate, a sequence of simple sentences is generated instead.

In McDonald's [20] generator, MUMBLE, rather than accessing the dictionary before the syntactic tree is constructed, the dictionary is consulted *as* the tree is built. Starting at the root of tree, the verb is selected first and its arguments located as subtrees of the verb. The system does a tree traversal, expanding the tree every time a leaf is reached by consulting the dictionary and grammar, thus constructing the full tree. The tree is then linearized to produce the final sentence.

Influences on the generated English other than syntactic constraints include information about the person the text is intended for, semantic constraints, and information about the discourse structure of the text. Information about user type can be used to select appropriate vocabulary (the naive user will not understand the expert's terminology). Similarly, information about the user's beliefs and knowledge can be used to generate noun phrase descriptions so that the user can successfully identify what is referred to by the description [2]. For example, a system should not use the noun phrase "the wheelpuller" if the user does not know what a wheelpuller is. Danlos [7] shows how the choice of a single word may depend on the semantic features of other words in the sentence and the order in which various facts are presented, as well as syntactic constraints. Finally, knowledge about how a given sentence fits in with the rest of the text can be used to choose the best word order for a sentence and to decide whether or not to use pronouns [21].

V. Summary

Language generation is becoming an increasingly important component of systems that interact with their users. As a discipline, it can be characterized mainly as involving problems of choice, requiring researchers to identify constraints on the various decisions a system must make. This paper has illustrated some of the factors that play a role, notably discourse strategies and grammars, showing how they have been used in both TEXT and TAILOR. For further information on other factors identified and used within generation systems, the interested reader is referred to two bibliographies of language generation research [3], [16].

References

[1] D. E. Appelt, "Planning natural language utterances to satisfy multiple goals," Ph.D. dissertation, Stanford University, Stanford, CA, 1981.

[2] _____, "Some pragmatic issues in the planning of definite and indefinite noun phrases," in *Proc. 23rd Annu. Meet. Assoc. Comput. Linguistics* (Chicago, IL, July 1985), pp. 198–203.

[3] _____, "Collected abstracts from the second international workshop on language generation," Tech. Rep., SRI Int., 1985.

[4] M. Blumenstyk and K. R. McKeown, "Monitoring conceptual and linguistic decisions to generate coherent text," Columbia Univ., Tech. Rep., Jan. 1986.

[5] D. Chester, "The translation of formal proofs into English," *Artificial Intell.*, vol. 7, pp. 261–275, 1976.

[6] W. J. Clancey, "The epistemology of a rule-based expert system—A framework for explanation," *Artificial Intell.*, vol. 20, pp. 215–251, 1983.

[7] L. Danlos, "Conceptual and linguistic decisions in generation," in *Proc. 10th Int. Conf. on Computational Linguistics* (Stanford, CA, July 1984), pp. 319–325.

[8] M. A. Derr and K. R. McKeown, "Using focus to generate complex and simple sentences," in *Proc. 10th Int. Conf. on Computational Linguistics* (Stanford, CA, July 1984), pp. 501–504.

[9] N. M. Goldman, "Conceptual generation," in R. C. Schank, Ed., *Conceptual Information Processing.* Amsterdam, The Netherlands: North-Holland, 1975.

[10] E. H. Hovy, "Integrating text planning and production in generation," in *Proc. 9th Int. Joint Conf. on Artificial Intelligence* (Los Angeles, CA, Aug. 1985), pp. 848–851.

[11] M. Kay, "Functional grammar," in *Proc. 5th Annu. Meet. Berkeley Linguistic Soc.*, 1979.

[12] K. Kukich, "Knowledge-based explanation generation," presented at the 2nd Annu. Language Generation Workshop, Stanford University, Stanford, CA, July 8–10, 1984.

[13] M. Lebowitz, "RESEARCHER: An experimental intelligent information system," in *Proc. 9th Int. Joint Conf. on Artificial Intelligence* (Los Angeles, CA, Aug. 1985), pp. 858–862.

[14] W. G. Lehnert, "Narrative complexity based on summarization algorithms," in *Proc. 8th Int. Joint Conf. on Artificial Intelligence* (Karlsruhe, West Germany, Aug. 1983), pp. 713–716.

[15] A. Malhotra, "Design criteria for a knowledge-based English language system for management: an experimental analysis," Tech. Rep. MAC TR-146, MIT, Cambridge, MA, 1975.

[16] W. C. Mann, "Text generation: The state of the art and the literature," ISI/TR-81-101 and Univ. of Pennsylvania Tech. Rep. MS-CIS-81-9, Dec. 1981.

[17] _____, "An overview of the Penman text generation system," in *Proc. Nat. Conf. on Artificial Intelligence* (Washington DC, Aug. 1983), pp. 261–265.

[18] _____, "Discourse structures for text generation," in *Proc. 10th Int. Conf. on Computational Linguistics* (Stanford, CA, July 1984), pp. 367–375.

[19] J. McDermott, "Building expert systems," in *Proc. 1983 NYU Symp. on Artificial Intelligence Applications for Business.* New York, NY: New York Univ., 1983.

[20] D. D. McDonald, "Natural language production as a process of decision making under constraint," Ph.D dissertation, MIT, Cambridge, MA, 1980.

[21] K. R. McKeown, *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text* (Studies in Natural Language Processing, A. K. Joshi, Ed.). Cambridge, England: Cambridge Univ. Press, 1985.

[22] C. L. Paris, "Description strategies for naive and expert users," in *Proc. 23rd Annu. Meet. Assoc. Comput. Linguistics* (Chicago, IL, July 1985), pp. 238–245.

[23] J. Paterson, *The Hamlyn Pocket Dictionary of Wines.* New York, NY: Hamlyn, 1980.

[24] G. Ritchie, "A rational reconstruction of the proteus sentence planner," in *Proc. 10th Int. Conf. on Computational Linguistics* (Stanford, CA, July 1984), pp. 327–329.

[25] E. H. Shortliffe, *Computer-Based Medical Consultations.* New York, NY: Elsevier, 1976.

[26] R. Simmons and J. Slocum, "Generating English discourse from semantic networks," *Commun. ACM*, vol. 15, no. 10, pp. 891–905, Oct. 1972.

[27] S. Stolfo and G. Vesonder, "ACE: An expert system supporting analysis and management decision making," Tech. Rep., Dep. Comput. Sci., Columbia Univ., New York, NY, 1982, to appear in *Bell Syst. Tech. J.*

[28] W. R. Swartout, "Producing explanations and justifications of expert consulting programs," Tech. Rep. MIT/LCS/TR-251, MIT, Cambridge, MA, Jan. 1981.

[29] H. Tennant, "Experience with the evaluation of natural language question answers," working paper 18, Univ. of Illinois, Urbana-Champaign, IL, 1979.

[30] J. L. Weiner "BLAH, A system which explains its reasoning," *Artificial Intell.*, vol. 15, pp. 19–48, 1980.

[31] R. Wilensky, "A knowledge-based approach to language processing: A progress report," in *Proc. 7th Int. Joint Conf. on Artificial Intelligence* (Vancouver, BC, Canada, Aug. 1981), pp. 25–30.

[32] T. Winograd, *Language as a Cognitive Process: Syntax*, vol. I. Reading, MA: Addison-Wesley, 1983.