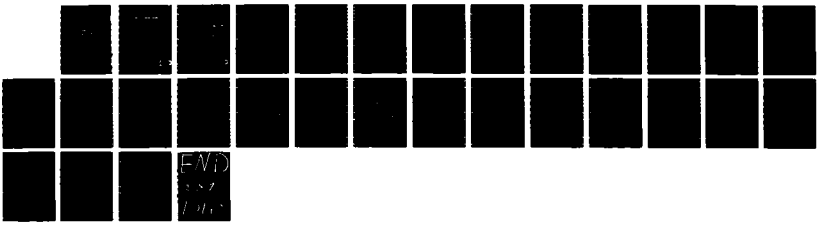
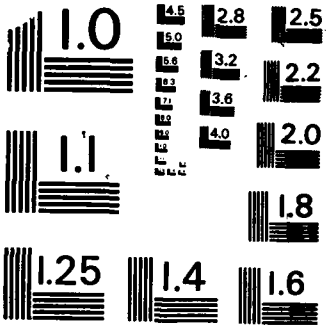
AD-A175 135 NATURAL LANGUAGE GENERATION(U) MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER AND INFORMATION SCIENCE D D MCDONALD FEB 86 CPTM-12 N00014-85-K-0017 1/1

UNCLASSIFIED F/G 5/7 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

The Counselor Project

Department of Computer and Information Science
University of Massachusetts
Amherst, Mass. 01003

Natural Language Generation

David D. McDonald
February, 1986
CPTM #12

This series of internal memos describes research in artificial intelligence conducted under DARPA Contract N00014-85-K-0017

2

Natural Language Generation

David D. McDonald

February, 1986
CPTM #12

Natural Language Generation

David D. McDonald[1]
University of Massachusetts at Amherst
February 1986

To appear in the **Encyclopedia of Artificial Intelligence**, published by John Wiley & Sons 1986.

[1] Preparation of this article was supported in part by contract number SU353-9023-3 from Rome Air Defence Center, and contract number N00014-85-K-0017 from the Defence Advanced Research Development Agency.

1. INTRODUCTION

Generation is the process of deliberately constructing a natural language text[2] in order to meet specified communicative goals. The goals come from another program, perhaps an expert reasoning system or a ICAI tutor, that is motivated to talk to a human user. The texts that are produced may range from a single phrase given in answer to a question, through multi-sentence remarks and questions within a dialog, to full-page explanations.

Generation is a different matter from simply having programs use English: programs have been printing natural language messages at their users for as long as there have been computers, yet one does not want to think of an error message from a FORTRAN compiler as either constructed or goal-directed, however well written it may be. An error message does not mean anything to the program that prints it--any connection between the string of words and the program's situation is strictly within the mind of the programmer who wrote that pre-programmed, "canned" text. Even the use of parameterized "format" statements, where the canned word string can be augmented by including variables in the statement that are substituted for at runtime names or simple descriptions, is not really generation. These "fill-in-the-blank", or "template", techniques depend for their effectiveness on a tacit limitation in the number and complexity of the situations in which the program will need to use them; that they have been adequate up to now for expressing what programs have had to say is more of a comment on the simplicity of today's programs than on the capabilities of template-driven generation.

In contrast with template-based engineering treatments, basic research on natural language generation, like the other areas of its parent field of computational linguistics, has as its goal not just competent performance by a computer, but the development of a computational theory of the human capacity for language and the processes that engage it. For generation, this focuses on the explanation of two key matters: versatility and creativity. What do people know about their language, what processes do they employ that enables them to be

(1) versatile, varying their texts in form and emphasis to fit an enormous range of speaking situations, and

(2) creative, with the potential to express any object or relation in their mind as a natural language text?

The need to accommodate these capabilities is the prime organizing force behind generation theories, and is the basis of the special contributions that the people who work on generation make to the rest of computational linguistics and artificial intelligence.

This article describes AI research on natural language generation with a historical perspective, emphasizing the special character of the problems to be solved. It begins by contrasting generation with language understanding, establishing basic concepts about the breakdown of the process into components and the flow of information and decisions through it. A section of excerpts from the output of illustrative generation systems follows, showing what kinds of performance are possible and where the difficulties are. In the remainder of the article the common approaches to generation are surveyed, including characteristic messages and the nature of a generator's lexicon. A separate section

[2] Throughout this article I will refer to the output of a generation program as a "text". This is intended as a general, recursive term that can apply to utterances or parts of utterances of any size, spoken or written. In people, whether a text is spoken or written has implications for the amount of deliberation and editing that may have gone on. If we identify "spoken" language with a lack of revision, then most programs today "speak", even though nearly all only print words on a terminal's display screen. Since the choice of whether to revise or whether to use print or voice is usually not an option for a generation program today, these particulars will only be mentioned when they are an issue in a program's design .

continues the survey with alternative approaches to the representation and uses of a grammar.

1.1 The character of the generation process

To understand why generation has the organization that it does, it helps to make a brief comparison with its more studied complementary process, natural language understanding. In contrast with the organization of the understanding process---which to a first approximation can follow the traditional stages of a linguistic analysis: morphology, syntax, semantics, pragmatics/discourse---the generation process has a fundamentally different character. This fact follows directly from the intrinsic differences in the information flow in the two processes. Understanding proceeds from texts to intentions; generation does the opposite. In understanding, the "known" is the wording of the text (and possibly its intonation). From the wording, the understanding process constructs and deduces the propositional content conveyed by the text and the probable intentions of the speaker in producing it. Its primary effort is to scan the words of the text in sequence, during which the form of the text gradually unfolds; the scanning requirement forces a process based on the management of multiple hypotheses and predictions that feed a representation that must be expanded dynamically . Major problems are caused by ambiguity--one form can convey a range of alternative meanings, and by underspecification--the audience receives more information from situationally motivated inferences than is conveyed by than by the actual text. In addition, mismatches in the speaker's and audience's model of the situation (and especially of each other) lead to unintended inferences.

Generation has the opposite information flow. It proceeds from content to form, from intentions and perspectives to linearly arrayed words and syntactic markers. Its "known" is its awareness of its intentions, its plans, and the text it has already produced. Coupled with its model of the audience, the situation, and the discourse, they provide the basis for making choices among the alternative wordings and constructions that the language provides--the primary effort in constructing a text deliberately. Most generation systems do produce the surface texts sequentially from left to right, but only after having made decisions top-down for the content and form of the text as a whole. Ambiguity in a generator's knowledge is not possible (indeed one of its problems is to notice that it has inadvertently introduced an ambiguity into the text). Rather than under-specification, a generator's problem is to choose from its over-supplied sources how to adequately signal intended inferences to the audience and what information to omit from explicit mention in the text.

With its opposite flow of information, one might assume that a generation process could be organized like an understanding process but with the stages in opposite order. To a certain extent this is true: identification of intention (goals) largely preceeds any detailing of the conceptual information the audience should be given; the planning of the rhetorical structure that will be imposed on the information largely proceeds any construction of syntactic structures to realize it; and the syntactic context of a word must be fixed before the precise morhological and suprasegmental form it should take can be known. But to emphasize this ordering of linguistic representational levels would be to miss generation's special character, namely that **generation is above all a planning process.** It entails realizing goals in the presence of constraints and limitations on resources. Its efforts consist of making decisions: decisions to use certain words or syntactic constructions, decisions to post constraints on later decisions. It is best organized as a process of progressive refinement.

This perspective on generation as planning permeates the views of the people who work on it. A language's syntax and lexicon become both resources and constraints,

defining the elements available for the construction of the text and also the dependencies between them that determine their valid combinations. These dependencies, and the fact that they tacitly govern when the information on which each decision depends can become available are the fundamental reason why generation programs do largely follow the conventional stages identified by linguists. Goal identification preceeds content selection and rhetorical planning, which preceeds syntactic construction, only because that is a natural order in which to make decisions; it is simpler to go with the flow of the dependencies rather than jump ahead and take the chance that a premature decision will have to be undone because it later turns out to be inconsistent. Today's research concentrates on understanding how best to represent what decisions are possible and the dependencies among them, as well as how to represent the constraints and opportunities earlier decisions place on later ones as the process proceeds.

The focus on planning and intention in generation research puts the underlying program in a pivotal position methodologically. Computational theories of processes must be implemented---embodied in a program that actually performs the behaviors under study---before they can be tested for coherence and procedural adequacy. One cannot test a theory of talking without having the underlying program motivated talk about something--planning and realization must be in the service of some actual goals. One is therefore forced to generate "for" some underlying program or else run the risk of basing one's theory on an unrealistic, incoherent foundation. Unfortunately, underlying programs that one can pick up "off the shelf" have inevitably been designed without the concerns of generation in mind. They turn out to be lacking in conceptual support for subtleties of intention and representation that generation researchers need, and to have structured their internal expressions in ways that make it difficult for a generation system to use alternative perspectives or groupings

Faced with the potential problems of using underlying programs built to suit independent concerns, generation researchers have adopted various approaches. Some develop their generators as standalone facilities and concentrate on studying grammar or planning in isolation [39,3,2]. Others have dedicated a great deal of their own development effort to building a task-based conceptual program for their generator to give it something substantive to talk about [17,13,35]. Still others work from an independently developed program but have interposed some kind of independent "planning" system in between to patch over the differences [46,48]. None of these approaches will lead soon to a general purpose generation facility that can be attached freely to any underlying program, though some work has been directed that way [22,43,40].

1.2 Standard Components and Terminology

The natural language generation component does not stand by itself. It fits within a man-machine interface, which it shares with a component that does natural language understanding--the input to the system.[3] Bridging the two is a representation of the ongoing discourse, which they both add to and use for reference. The interface may end here, or it may extend further back with other shared components such as a discourse controller that directs the actions the generator takes and coordinates the interpretations made by the understander. Behind the interface is the non-linguistic reasoning or database program that human users employ the interface to talk to. This program will be referred to here uniformly as the underlying program. It can be almost any type of AI system one can imagine: cooperative database, expert diagnostician, ICAI tutor, commentator, apprentice,

[3] In a good man-machine interface today one would also expect provisions for coordinated graphical input and output, complementing the natural language I/O.

advisor, mechanical translator. The nature of the underlying program presently has no significant influence on the generator's design.

Today most generation researchers work most often with underlying programs that are expert advisors, e.g. [63,46]. With an advisor program, the control of where the conversation goes is most likely to rest with the program rather than the person using it. In addition, advisor programs and intelligent machine tutors are likely to have a good understanding of what their human interlocutors are thinking. These features make them able to motivate fairly sophisticated texts, which makes them attractive to those generation researchers who are looking for already developed programs to work with.

The generation process starts within the underlying program when some event leads to a need for the program to speak. In the simplest case this may be the need to answer a question from the user; with a sophisticated discourse controller it may be the perception of a need to interrupt the user's activities in order to point out an impending problem. Once the process is initiated, three kinds of activities must be carried out:

(1) identifying the goals the utterance is to achieve,

(2) planning how the goals may be achieved, including evaluating the situation and the available communicative resources, and

(3) realizing the plans as a text.

Goals are typically to impart certain information to the audience or to prompt them to some action or reasoning. People, of course, talk for social and psychological reasons as well as practical ones; but as these needs are beyond the ken of today's computer programs, AI research on generation is forced to largely ignore them. Planning involves the selection (and deliberate omission) of the information units to appear in the text (e.g. concepts, relations, individuals), and the adoption of a coordinating rhetorical framework or schema for the utterance as a whole (e.g. temporal progression, compare-and-contrast). Particular perspectives may be imposed on the units to aid in the signaling of intended inferences.

Realization is the process of manifesting the planner's directives as actual text. It depends upon a sophisticated knowledge of the language's grammar and rules of discourse coherency, and typically constructs a syntactic description of the text as an intermediate representation. The term realization is used technically within the field: for example one speaks about choosing to "realize" a modification relationship as either an adjective or a relative clause. It emphasizes not only attention to linguistic form but also knowledge of the criteria that dictate how those forms are used. In many research projects the process that does grammatical realization is called the linguistic component [43], and in some the planning and goal identification processes are together called the strategic component [62]. Usually it is only the linguistic component that has any direct knowledge of the grammar of the language being produced. What form this grammar takes is one of the points of greatest difference between generation projects, though all projects largely agree on the function a grammar should serve in generation.

For the traditional linguist, a grammar is a body of statements in a notation. The content of the statements--the specific facts of a given natural language--is of less interest to the linguist, by and large, than the theoretical properties of the notation. These properties are measured by how expressive the notation is, what primitives it identifies, and what representations and principles it makes use of. The situation is not much different in theories of generation except that the notation--the procedural and representational framework--is designed to serve a very specific function with which the traditional linguist is not concerned, namely to guide and constrain the process of generating a text with a specific content and goals and in the presence of a specific audience. This has an overriding effect on the form grammars take; more importantly, it also strongly influences

the information they must include. The grammar is now responsible for defining the choices that a language allows in form and vocabulary, and it must further include criteria of usage. Generation researchers must ask what circumstances lead to deciding on one alternative rather than another, as well as what functions the various constructions of the language serve that make them appropriate for fullfilling a certain goal. Only by including such information can a grammar serve as a resource defining the options available to the text planner. The other, more obvious, function of a grammar is to insure that the texts that are produced follow the rules of the language--that they are "grammatical". How exactly this is done is another point where the different schools of generation often part ways, but a common theme is that the grammar functions by defining dependencies and constraining decisions.

The non-linguistic plan or specification that directs realization is typically called the message; some researchers talk about "realizing the message", and speak of the conceptual and rhetorical representations maintained by the planning and goal-identification processes as being at "the message level" (as opposed to realization activities at "the surface-structure level"). This is a convenient and commonsense terminology, but one must be careful not to presume too much from it. The typical mental image evoked by the term "message" is of written notes passed from one person to another, for instance as the result of a telephone conversation; however this image does not fit the situation: Researchers who study both planning and realization continually make the point that there is no clean line between the two activities (see for example [1,16,46]). Planning proceeds in layers of refinement and must appreciate the linguistic consequences of its decisions; the realization of units in early layers creates a grammatical context that imposes constraints on the range of realizations that can be planned for later. Goals may emerge or change in priority opportunistically as planning and even realization proceeds.

2. THE STATE OF THE ART

There is a firm consensus within the field [38] that versatility and creativity in machine generated text is possible only if

(1) the generator incorporates a comprehensive, linguistically principled grammar,

(2) the underlying program has a sophisticated, commonsensical, conceptual view, and

(3) the text planner can make use of models of the audience and the discourse.

Unfortunately such generators are still only the subject of research today. When none of these conditions are met, the state of the art in generated text is still about the same as it was in 1970 in Terry Winograd's SHRDLU program [64]. SHRDLU produced original sentences, which it constructed dynamically, as replies to the questions it was asked. It took program expressions out of its model of the state of the blocks on its table and the actions it had performed, and applied what today would be called a "direct replacement" procedure to make simple grammatical adjustments to the verbs and linearize the expressions to yield comfortably readable texts such as the one below.

24. *When did you pick up [the green pyramid] ?*

While I was stacking up the red cube, a large red block, and a large green cube.

By the late 1970's, generation systems of this simple but effective sort had become quite important in the early rule-based expert systems. They were needed to translate the large numbers of rules in these systems into an easily appreciated format in stylized English. A generator of some kind is required within these systems because the number of rules is large and their internal variation is too high to capture with a set of fixed, fill-in-the-blank templates. It is a straight-forward matter to provide a simple generation capability for any program where the objects in the knowledge base have a consistent structure, and there

is only one situation—one communicative context—in which the text must appear. Such capabilities are developed quickly, typically on an ad-hoc basis as the rule-based system is developed [19,20].

Generation researchers, however, are interested in more complex texts than the context-free presentation of an expert system's rules can motivate. Today this almost invariably means that as well as working on their generator they must develop their own underlying programs to provide an adequate conceptual source to work from, but there are numerous technical problems in generation that can be profitably approached with only a minimal base. As an example, here is a simple description from a program by Bengt Sigurd [57]. Sigurd's point was to study how grouping is signaled though intonational effects; this text is actually spoken by a Votrax speech production system.

> "*The submarine is to the south of the port. It is approaching the port, but is not close to it. The destroyer is approaching the port too.*"

While its content will not win it a place on the New York Times Best Seller list, its structure, especially its use of the inference-directing function words "*but*" and "*too*", represents an important contribution. The source propositions in the data base of a expert system that reasoned about submarines and destroyers would not be "packaged" with the conceptual equivalents of such function words already in place and able to be read out by a simple template. This is because the inferences the words control are specific only to one particular choice of what facts are being mentioned and how they have been grouped—a planning decision that is not part of the reasoning system's job, but cannot be omitted in generation.

A similar technical problem that is not yet well enough understood is "subsequent reference" [42]. What wording should be chosen when a reference to an object appears more than once in the text? Always using a pronoun may introduce ambigities; in general careful reasoning can be needed about how the audience will characterize the actors in a text in order to judge what phrasing to use. Below is an example text from a recent study of this problem by Robert Granville [24]. He classifies the relations between a referent and its last point of mention and develops a set of structural rules for making subsequent references based on it.

> "*Pogo cares for Hepzibah. Churchy likes her, too. Pogo gives a rose to her, which pleases her. She does not want Churchy's rose. He is jealous. He punches Pogo. He gives a rose to Hepzibah. The petals drop off. This upsets her. She cries.*"

The principal problem with that text as a piece of prose is that it is "choppy": no attempt has been make to group its individual propositions into larger units and the resulting sentences feel too short. Ultimately such textural decisions require a linguistically sensitive analysis of text style; but they also require a conceptual basis for the grouping, and an appreciation of what a grouping will signal to the audience. This information is not easy to come by in today's candidates for underlying programs.

It is no wonder then that the very best performances by generators have come from systems where the generation researcher was also the person who developed the underlying program. That way one is sure that there will be a basis in the underlying representation for any rhetorical attitutes or distinctions that the subject matter calls for and will be a conceptual perspective by which to organize groupings. An important case in point is the program PROTEUS, developed by Anthony Davey in 1974 [17]. This program produced descriptions of games of tic-tac-toe (also called "Naughts and Crosses") that are still among the most fluent texts ever produced by a machine.

"The game started with my taking a corner, and you took an adjacent one. I threatened you by taking the middle of the edge opposite that and adjacent to the one which I had just taken, but you blocked it and threatened me. I blocked your diagonal and forked you. If you had blocked mine, you would have forked me, but you took the middle of the edge opposite of the corner which I took first and the one which you had just taken and so I won by completing my diagonal."

The naturalness of PROTEUS's descriptions come largely from its appreciation of tic-tac-toe as a game: it has a rich model of how specific moves may be seen as threats or counters to threats, and it incorporates the rhetorical principle that one should put in a text only the most salient information in a situation, for example missed opportunities or forks, while leaving the other information to be communicated implicitly by inference. PROTEUS has the equivalent of an underlying program in its routines for the analysis of the tic-tac-toe moves. These provide an annotation of the moves in terms of threats, blocks, etc., providing input to a planning facility which selects the best level of description for each move (e.g. "block" versus "fork"). The planner then groups the moves two or three at a time into sentences according to what game-level relationship seems to provide the best description of their motivation (e.g. "threat-but-block" or "although-threat-block-&-counter"). A grammar and realization facility then take the described and grouped moves, work out the details of their form as English sentences, and produce the words of the text.

A rival to Davey's PROTEUS in fluency is John Clippinger's 1974 program ERMA [13]. ERMA is the only program to date that has attempted to deal with the fact that people speak in real time and continue to think and plan as they do so. People reflect on what they are saying and notice, while they are talking, omissions or unintended interpretations which they fix by dynamically replanning and restarting their speech in mid-utterance. To model this behavior, Clippinger, working with an undergraduate assistant, Richard Brown [20], analysed 40 hours of transcripts of a patient in psychoanalysis in order to understand that patient's motivations and reasoning patterns well enough to provide a computational account of one of the paragraphs in that transcript (shown below), which the program ERMA was able to reproduce in every detail.[4] The text segments in parenthesis are what ERMA was planning to say before it cut itself off and restarted.

"You know for some reason I just thought about the bill and payment again. (You shouldn't give me a bill.) <Uh> I was thinking that I (shouldn't be given a bill) of asking you whether it wouldn't be all right for you not to give me a bill. That is, I usually by (the end of the month know the amount of the bill), well, I immediately thought of the objections to this, but my idea was that I would simply count up the number of hours and give you a check at the end of the month."

Clippinger and Brown developed an architecture of five major interlocking components that took a thought from its first appearance as an interpersonal goal, through a fleshing-out and lexicalization, evaluation for social acceptability, interjection of attenuating phrasings, and sometimes a complete reworking to soften harsh impacts, while all the time realizing and uttering whatever text plan was in force at that moment. This required something of a tour-de-force in computer programming for 1974, and the project was not carried further.

It is quite striking to realize that two of the most competent generation programs ever developed, Davey's PROTEUS and Clippinger's ERMA, are also among the oldest in the

[4] Actually the original transcribed paragraph included several additional "uhs" and a "you know"; there was also no attempt to account for the specific time delays that occurred, or for some of the sentence-initial perseverations.

field. There are two reasons for this: (1) until the early 1980's comparatively few people had ever worked on the problem of generation, and (2) the problem is very hard---harder in this writer's opinion than language understanding, the area where most of the AI work in natural language processing has concentrated. These matters are not independent. A good deal of work on generation was in fact going on in the early 1970's, principally in the context of Ph.D. dissertations that built upon the first rush of significant results in language processing that had come a few years before with the work of Winograd on SHRDLU [64] and of Woods with Augmented Transition Networks [66]. In addition to Davey and Clippinger, there was the work of Bob Simmons and Jonathan Slocum on the adaption of ATNs to generation [56], and the thesis of Neil Goldman focusing on how to organize word choice when generating from conceptual dependency networks [22], as well as other work [7,28,41,53]. It is fair to say however that the initial reports of that generation work, principally at the important "TINLAP" meeting in 1975 [9,12,23], fell largely on deaf ears, and research on generation went into something of a hiatus for the last half of the decade.[5]

Until the early 1980's generation was considered by most people in AI (those who did not work on it) to be a relatively simple problem. Indeed, it is a simple matter to take a statement in an internal representation of the sort people used in the middle 1970's, say **(#supports :block6 :block3)**, couple it with attributes stored separately for the individuals, and produce "*The big red block supports a green one* ": Winograd's SHRDLU could do this in 1970. If this were all the competence one needed, then generation would not be an important research problem. However as soon as one begins to consider the various ways that that simple sentence could be rewritten--the versatility that the English language invites speakers to make use of--the difficulties begin to emerge. In that text for example, should one always say "*a green one*", and not "*a green block*"; what kind of circumstances call for one but not the other? Suppose one wanted to use the **Support** assertion as an attribute of the green block, for example as a way to distinguish it from the other green blocks: "*...the green block that's supported by the big red one*". How does one represent the grammatical knowledge that allows a generator to use its representation of the syntactic structure of the statement form of the text to produce the corresponding relative clause? How does the generator represent to itself in a general way the fact that the relative clause is even available, or that such a use for the assertion is possible? Few people worked on generation in the later 1970s (or stayed with the problem for more than a year or two) either because they found the task too simple to be interesting (when working forwards from the sorts of texts that reasoning programs needed at that time), or because they found it too difficult to make any headway (when working backwards from the complexities of actual human texts).

3. COMMON APPROACHES

It is difficult to identify the common elements in the different research projects on generation. By contrast, in language understanding research one can identify any number of primary approaches to the problem: using ATNs, semantic grammars, deamon-based systems grounded in conceptual dependency representation, procedural semantics, and many more. These schools of thought have names, a body of literature, and a coherent historical development over decades or more. Generation research cannot yet be said to

[5] This is not to say that no work was done during those years; rather that generation was not perceived by the larger community as an important problem to be working on. By contrast, today there are entire sessions on generation at any large conference where natural language processing is included as a topic. There have also been three international workshops of generation specialists since 1983 with an ever increasing number of participants.

have any schools in this sense. This is partially because historically only a small number of individual have made this their primary area of research (as just discussed); large research groups focused on generation have formed in only the last few years. A more significant reason is the fact the nature of generation research has made it difficult to see the commonalities among the different generation systems. The principal problem is the lack of a common starting point: Unlike parsing research, where it is obvious that one must start by identifying and grouping the words of the text, independent research efforts in generation inevitably construct their messages using different internal representation languages, use differing amounts of planning, and focus on orthogonal technical problems. This lack of any immediate basis of comparison has made it hard for people to build on each other's work or even to test their own examples on another researcher's system. Nevertheless, the various generation projects have more in common with each other than not. There are common threads running through the projects: similar approaches, similar representations, similar grammars.

Two organizing questions are of common concern. The first is how to confront the diversity of forms in natural languages to develop functional accounts of them--to answer the question of why a person will use one form rather than another, and to do so with a formal, computational account that a machine can use in dealing with people. Put another way, what is a person's model of the differences between syntactically or lexically similar versions of the same text, and of the impact they will have on an audience? The second question is control of the generation process. What defines the choices that have to be made in a given speaking situation? What provides the basis for ordering them? How does one organize and represent the intermediate results? What awareness does the system have of the dependencies between choices? How are these dependencies represented and made to influence the control algorithms?

Alternative answers to these two questions will be described throughout the rest of this article. This section covers the nature of messages and approaches to the lexicon; the following section considers various treatments of grammar.

3.1 Control by progressive refinement of the message

All treatments of the diversity of forms have been bound together with accounts of control, making control the proper place to start in now looking at the schools of thought in how generation is actually done. Among generation systems that were built specifically to work from underlying systems, the predominant approach to control is to treat the message from the text planner as a kind of program, i.e. to see it as an expression that one evaluates with a special kind of interpreter. Again a caution in is order. These "messages" are not simply expressions whose context and form are isomorphic to the target text that happen to be encoded in a non-natural computer language. They cannot just be translated. Of course in the simplest treatments of generation translation might be sufficient (as in most existing expert systems), but in treatments that focus on generation, the relations and arguments in a message are best viewed as instructions to achieve a certain effect by linguistic means. The evaluation proceeds by progresssive refinement from outermost instructions to inner. This control technique is natural to the developers of the systems since it mimics the style of the programming languages that they use and takes advantage of the almost unconscious preference among practicioners of AI to follow a function/parameters, predicate/arguments, style of representation.

The most common messages today are not constructed by any planner but are simply data structures extracted from the underlying program and given a special interpretation by the generator. This is common practice in programs that need to explain their reasoning [60,11], with one of the clearest and earliest examples being to explain the reasoning embodied in simple natural deduction proofs in the predicate calculus. Below is such a proof and the text produced for it by an early version of David McDonald's program

MUMBLE [43]; the proof itself is taken from earlier generation work by Daniel Chester [10].

INPUT:

Line 1: premis

Exists(x) [barber(x) and Forall(y)..shaves(x,y) iff not.shaves(y,y)]

Line 2: existential instantiation (1)

barber(g) and Forall(y)..shaves(g,y) iff not.shaves(y,y)

Line 3: conjunction reduction (2)

Forall(y)..shaves(g,y) iff not.shaves(y,y)

Line 4: universal instantiation (3)

shaves(g,g) iff not.shaves(g,g)

Line 5: tautology (4)

shaves(g,g) and not.shaves(g,g)

Line 6: conditionalization (5,1)

(Exists(x) [barber(x) and Forall(y)..shaves(x,y) iff not.shaves(y,y)])
implies (shaves(g,g) and not.shaves(g,g))

Line 7: reductio-ad-absurdum (6)

not (Exists(x) barber(x) and Forall(y)..shaves(x,y) iff not.shaves(y,y))

OUTPUT:

"*Assume that there is some barber who shaves everyone who doesn't shave himself (and no one else). Call him Giuseppe. Now, anyone who doesn't shave himself would be shaved by Giuseppe. This would include Giuseppe himself. That is, he would shave himself, if and only if he did not shave himself, which is a contradiction. This means that the assumption leads to a contradiction. Therefore it is false, there is no such barber.*"

The fluency of this text derives from an ad-hoc model of the communicative force that accompanies a given instance of an inference rule of natural deduction (e.g. "premis" or "universal instantiation"). The model provides an account of the motivations of the proof writer in selecting what rule to apply, for example that the point of the right hand side of the biconditional in the first line is to place a restriction on the variable "Y" ("...*who doesn't shave himself*"). These motivations license the decisions to realize the lines of the proof in specific ways. These motivations, however, do not appear anywhere in the proof (which was the sole input to the program). They are only presumed, and so are valid only for a few example proofs written with that particular personal style of natural deduction.

The paucity of information or motives and perspectives in the messages of the underlying program is a perennial problem of work on generation: computational linguists are forced to read into the data structures of the underlying programs because they do not already include the kinds of rhetorical instructions the generator needs if it is to employ the syntactic constructions of the language in the way that a person would. Without such "extra" information, the coherency of what is said--especially for texts more than a few sentences in length--will depend on how consistent and how thorough the authors of the underlying programs have been in their representational conventions: A generator has no

choice but to treat a symbol like "premis" or the biconditional operator in the same way each time it sees them in the same context. If consistency is maintained, the imaginative designer can make up for the deficiencies by embellishing the data structures once they are inside the linguistic component.

When a text planner is brought into the process, messages can be built from a combination of data structures from the underlying program and instructions about perspective and rhetorical effect that the planner introduces [46]. Below is an example of a complex message--a "generation program"--that leads to text of the quality a person would produce (taken from a design study reported in [45]). Specification of effects to achieve are marked by colons in front of the symbols. The content information to be conveyed is given by reference to internal frame objects named in angle brackets. This content is to be put in specific perspectives (e.g. **main-event** and **particulars**), and the effect is to direct reasoning about linguistic alternatives in the presence of given rhetorical, and eventually grammatical, constraints. If the researchers's goal is to approximate the fluency and specificity of texts authored by people then messages will normally be as complex as this.

SPECIFICATION

```
(the-day's-events-in-the-Gulf-tanker-war
      :events-require-certification-as-to-source
      (main-event #<same-event-type_varying-patient
                        #<hit-by-missiles Thorshavet>
                        #<hit-by-missiles Liberian> >
            :unusual #<number-of-ships-hit 2>
            :identify-the-ships )
 (particulars  #<damage-report Thorshavet Oslo-officials>
               #<damage-report Liberian Lloyds> ))
```

OUTPUT

"Two oil tankers, the Norwegian-owned Thorshavet and a Liberian-registered vessel, were reported to have been hit by missiles Friday in the Gulf. The Thorshavet was ablaze and under tow to Bahrain, officials in Oslo said. Lloyds reported that two crewmen were injured on the Liberian ship."

The goal of fluency and intentional specification of form motivates many of the more elaborate bits of computational machinery that constitute the common threads running through different research projects, particularly the use of phrasal lexicons and an intermediate linguistic representation. Stepping through a simple example will show why these are needed. Consider the logical formula below, given in the prenext notation that a program would typically use internally. (This example follows the treatments of Chester and McDonald described above.) This is the commonest kind of message one will find today: an expression straight from the model of an underlying program (the natural deduction proof system), now given a special interpretation because it is being used to specify a text.

```
(exists x
   (and  barber(x)
         (forall  y
            (if-and-only-if  shaves(x,y)
                             (not  shaves(y,y) )))))
```

In this formula the generator is immediately confronted with choices of realization. Should the quantification be expressed literally (*"There exists an X such that..."*), or should it be folded within the body as determiner information on the realization of the variables (*"...some barber"*)? Should the biconditional **if-and-only-if** be realized literally

as a subordinating conjunction, or interpreted as a range restriction on the variable (yielding the modifying relative clause "*anyone who doesn't shave himself*"?). A predication like **barber(x)** should presumably always be decoded and converted to a specification of how the variable is to be described since it reflects the logician's convention of expressing type restriction through initial conjunctions; the alternative of using an extra sentence ("*X is a barber*") would be too unnatural. The other choices are substantive however, and need to be deliberated over.

In message-directed progressive refinement treatments, such deliberations are usually managed by grouping the alternatives according to the type of object involved. The objects that populate the "mind" of the underlying program, in this case logical connectives, predicates, and bound variables, are all linked to the words and grammatical constructs that are appropriate for realizing them through "specialist procedures" maintained within the generator. These procedures are the equivalent of the lexicon in an understanding system. The specialists build a realizing phrase by drawing on lexical information associated directly with the individual logical objects. They are able to look at properties of the objects such as when they were last mentioned or what kinds of objects they have as arguments. Each object typically has associated lexical items: a constant may have a name, a predicate may have an adjective or a verb. The specialist does its work by putting these into a phrasal context that will be completed by the recursive application of other specialists, e.g. the two-place predicate **shaves(x,y)** becomes the clause "**x** *shaves* **y**".

In this control regimen, the execution of each of the specialists is compartmentalized and taken up in the order dictated by the hierarchical form of the controlling expression, in this case the formula. The quantifier **exists** would be dealt with first, then the **and**, the **forall**, and so on. Consideration of how an element of the formula is to be interpreted is delayed until it is actually reached in the stepwise, incremental refinement process. Relations provide linguistic templates by which to order the realizations of their arguments, and the process proceeds recursively. This provides the benefits of the principle of least committment, expediting the generation process as a whole by avoiding the possibility of having to "backup" out of prematurely made realization decisions that turn out to be incompatible with the grammatical context defined by a higher template.

3.2 Lexical choice

Some approaches to machine reasoning emphasize the selection of a small set of primitives and the statement of a program's knowledge as a set of expressions over these primitives plus a set of constant terms for individuals. This has the advantage for reasoning of giving the commonalities among situations a structural prominance. This makes inferences easy to draw because they can be bundled into natural groups by the primitives. However, the reduction of the range of human actions to a set of, for example, only 13 conceptual primitives means that a great deal of the specificity that the words of the language carry, in this case the verbs, will have been distributed throughout the expressions and will have to be collected and discriminated during generation if specific verbs are to be used. Neil Goldman pioneered this use of discrimination nets to determine the best words for realizing whole expressions in his thesis on generation from conceptual dependency representation [33]. He demonstrated how one would determine word choice by working outwards from the core primitives, testing the other parts of an expression for certain properties. For example from the action primitive **ingest** one might get the verbs

"drink", "eat", "inhale", "breath", "smoke", or "ingest",[6] by testing whether, e.g., the object ingested was a fluid or smoke.

The fact that one is forced to make deliberate discriminations and word choices when working from expressions over neutral, underspecified primitives means that the problem will receive a good deal of attention. A discrimination net design invites the generation researcher to go beyond the base distinctions by object type and to include contextual factors like the speaker's emotional perspective in the decisions. Consequently, generation work based on underlying programs written using conceptual dependency has involved some of the most creative and interesting work on coordinated word choice of any in the field. Below is a sample from work by Eduard Hovy [29]. Hovy's aim is to bias the text to emphasize a desired point of view, in this case to report on this February primary in such a way that the results look good for Carter even though he lost.

> "*Kennedy only got a small number of delegates in the election on 20 February. Carter just lost by a small number of delegates. He has several delegates more than Kennedy in total.*"

In contrast, representations based on frames, for whatever historical reason, tend to involve the use of a very large number of "primitive" terms, in principle at least one for every word sense in a natural language, with the commonalities among terms indicated by reference to an abstraction-generalization network. When working from such representations, lexical choice is often a non-issue since each term can be uniquely associated with a natural language word. This is not to say that choice of wording on the basis of affective perspective or degree of specificity for words can not take place; rather that they are now seen as conceptual decisions rather than linguistic decisions. As a pragmatic matter, generation research that works off of such fine-grained representations tends to largely ignore the problem of lexical choice and put its energies elsewhere.

3.3 Phrasal Lexicons

What word to associate with simple conceptual terms like **barber** or **shaves** is obvious; however for the objects in complex underlying programs, lexical choice can be more problematic. Representations based on frame systems employ structured objects that denote encapsulations of entire conceptual schema, whose "names" will consist of a single, highly hypenated symbol, e.g. **example-intrinsic-similarities-with-competitive-product**. Such conceptually uninterpreted "primitives" have a reasonable place in underlying programs, at least pragmatically, since an expert system can note qualitative properties of a phenomena without having the common sense to understand it in enough detail to derive the term compositionally the way a person could. Technically these terms can be a considerable problem for generators, since they may encode entire sentences at once, yet will be used in rhetorical contexts where they may need to be modified with adverbs or adjectives, or elaborated by subordinated clauses.

The natural recourse in this situation is to use a phrasal lexicon. This notion was identified in 1975 by Joseph Becker [4], and is an important tool of generation systems. Linguistically, a "phrasal" lexicon is a conceptual extension of a standard, word-based

---

[6] Notice that one of the available words was "ingest", the least marked (most abstract) alternative that the discrimination net allowed. It is inevitable in computer programs developed by people that the internal symbols will correspond to natural language words, and indeed there is invariably an intended correspondence in at least the back of an AI programmer's mind between the symbol and the word when they use it. Careful representation researchers point out that their conceptual terms have no real meaning in and of themselves: they could perfectly well be replaced with artificial print forms like G007 and the programs would continue to work perfectly well.

lexicon to include entire phrases as unanalyzed wholes on the same semantic basis as words. This provides a means of capturing in a natural way the open-ended idioms and manners of speech that people use every day. Since people appear to use these "fixed phrases" as undigested wholes, programs need to be able to do the same. This means that there need not be any internally represented expressions whose parts and relations are the direct source of the words and syntactic relations of the phrase--precisely what is needed to deal with heavily hypenated symbols. Such texts can be quite good even though the underlying program understands little of what it is saying. The example below is from work by Karen Kukich [35]; another notable effort specifically employing a phrasal lexicon is that of Paul Jacobs [30].

> "*Wall Street securities markets meandered upward through most of the morning, before being pushed downhill late in the day yesterday. The stock market closed out the day with a small loss and turned in a mixed showing in moderate trading.*"

This information announcement was computed directly from an analysis of the data for the day's market behavior. Qualitative points in the results were paired directly with the steriotypical phrases of such announcements: "*a small loss*", "*a mixed showing*", "*in moderate trading*". Objects, actions, and time points were mapped directly into the appropriate word strings: "*Wall Street securities markets*", "*meandered upward*", "*<be> pushed downhill*", "*late in the day*". The compositional template driving the assembly of these phrases into a text was based on clauses built out of the S-V-Advp phrase: <market> <action> <time point>. The clauses were then grouped into sentences according to a few heuristics.

4. TREATMENTS OF GRAMMAR

In the study of generation, the choice of formalism for representing the language's grammar has always been bound up with the choice of control protocol. Broadly speaking there are three approaches to this combined design decision that can be identified:

(1) stating the grammar as an independent body of statements and filtering against it (with functional unification grammar as the prime example);

(2) using the grammar to specify all the valid surface structures that texts the language can have, and then stating the planner's choices and the output of realization in terms of surface structure (message-driven approaches, TAG grammars); and

(3) stating the grammar as a traversable graph structure and giving it control of the whole process once a text plan has been constructed (ATNs and most uses of systemic grammars).

There has yet to be any thorough comparative evaluation of these three alternative designs; individuals have adopted one or the other largely because of accidents of their own history: who they studied with, what was available locally, etc.. This article will maintain a studied neutrality. Each approach will be considered in turn, from the perspective of the problems that have particularly motivated its use.

Some of the details that make a text "grammatical" arguably do not and should not have any counterparts in a message from an underlying program. Person and number agreement of subject and verb are an obvious case in English; relative pronouns (e.g. "who" vs. "whom"), the infinitive marker "to", and very large numbers of other linguistic phenomena are the same. This is not to say that these have no conceptual counterparts: agreement can be viewed as an expression of the semantic relation of predication, the lack of tense is often an indication of the action being generic, and so on. The point is rather that this class of

grammatically motivated information is not relevant to the text planner--it is not a natural part of the message and consequently should originate in the linguistics component. The question for the generation researcher is (1) how to state this information, and (2) how to insure that it is brought to bear at the appropriate moment.

Parsimony encourages the computational linguist to attempt to share as much of this information as possible between both generation and understanding systems. Given the radical differences in the intrinsic character of the control and information flow in the two processes, this leads researchers to declarative accounts of language rather than procedural one, with elaborate derivational paradigms like generative grammar being ruled out of consideration quickly.[7] Among the long-standing linguistic traditions, about the most neutral paradigm that survives this criterion of being able to provide a declarative account is a system of rewrite rules.[8] There are, of course, new linguistic paradigms, many of them now put forward by people with computational backgrounds. One of these, functional unification grammar developed by Martin Kay [32], has been employed in generators, and is deliberately put forward as a "reversible" grammar, i.e. able to serve equally well as a controlling description in generation and understanding.

4.1 Functional unification grammar in generation

As presently used, functional unification grammars (FUGs) give a generator a modular, independent way of supplying the purely linguistic information that the process must have, and do so without imposing specific demands on its control structure.[9] The term "functional" in the name of the paradigm speaks to an intention on the part of its practitioners to go beyond description of the structure of linguistic forms to address the reasons why language is used. In contrast with the practice in systemic grammars however, the functional elements in FUGs are thus far only a minimal extension beyond the standard categorical linguistic vocabuary used traditionally to describe syntactic form (e.g. "clause", "noun phrase", "adjective"), and are more in keeping with their paradigmatically close neighbor, "lexical functional grammar" [6]. In the FUGs actually employed in generators, i.e. the Telegram grammar developed by Doug Appelt [2] and the

---

[7] When the purpose of the generation system is not to provide a communications facility for a mechanical actor, systems based literally on versions of transformational generative grammar have been quite appropriate. Two cases in point are the rule testing facility developed by Joyce Friedman for the use of linguists to check the consistency of large sets of rules [21], and the pedagogical ICAI system of Lyn Bates that has been used in the teaching of English as a foreign language [3].

[8] One of the very earliest mechanical generation systems of any sort was developed by Victor Yngve in 1959 using a pushdown automata and a body of context-free phrase structure rules with ad-lib lexical insertion [67]. Though it was not message-driven and generated text that was semantic nonsense and consequently would be uninteresting as a generator today, it did establish the legitimacy of the enterprise of providing explanatory accounts of psycholinguistic phenomena through appeal to the computational properties of a virtual machine operating over representations of linguistic rules, a methodology that is becoming increasingly important to computational and noncomputational linguists alike.

[9] The lack of demands to specify a control structure carries the entailment that one must be willing to live with whatever control structure is supplied. For Kay's FUG this is nondeterministic unification. If efficiency of execution is not relevant then this of course is no problem, however there are indications that the generality of the FUG notation gives them undesirable computational complexity properties, i.e. generating a structure from an arbitrary FUG appears to be NP-complete (51). Certainly a specific individual grammar may not require this complexity to process, however this result means that implementers of FUG generators must be especially careful in the construction of their algorithms since the formalism itself is not efficient.

realization component written by Steve Bossie [5] for the generation system of Kathy McKeown [48], the extensions are just the addition of terms like "subject", "premodifier" or "head"--descriptions of the role a constituent plays within the category that dominates it. Classically functional concerns, such as the distinction between "given" and "new" information in a sentence studied by the Prague School [15], or the similar distinction between "theme" and "rheme" defined by the Firthian tradition [18], have not yet been incorporated into FUGs.

Figure one shows an example taken from Appelt [1a]. It describes the constituent roles that accompany the phrasal category noun phrase. A full definition of the notation may be found in Kay's 1984 paper [32]; briefly the brackets define systems of features and values: square brackets define conjunctive sets, a description must specify all of the features within them; and curly brackets define disjunctive sets, where only one of the conditions defined by the feature-value pairs must be met.

[insert Figure One about here]

FUGs are used to flesh out minimal, conceptually derived functional descriptions,[10] for example that the head of some noun phrase is to be the word "screwdriver". FUGs are used in a process of sucessive mergers, constrained by the rules that govern how two descriptions may be unified. The key idea is that the planner first constructs a minimal description of a phrase, which it can do using specialists in the conventional way (e.g. that it wants to produce a clause with a certain verb and two NPs whose heads are certain nouns). To flesh out the description to the point where it would be valid grammatically, it is then unified with the grammar: The description of the phrase and the specification of the grammar are progressively merged, with specified features in one being melded into unspecified or compatibly constrained features in the other. The instantiation of some of the description's previously unspecified features by grammar-supplied constants then brings about a ripple effect throughout the whole system: decisions that are dependent on a just-instantiated feature force further unifications cyclically until a grammatically complete description of the utterance has been formed. In addition, elements in the planner's description will force selections among the disjunctive specifications in the grammar. For example, specifying a verb will force choice of grammatical subcategorization, which in turn will force a selection among the alternative clause orderings patterns that the grammar defines, since only one of them will have a compatible specification.

The complete description will amount to a rooted tree of subdescriptions (constituents) as defined by the "pat" (pattern) feature which dictates sequential order at each level. The actual production of the text is performed by scanning this tree and reading out the words in the lexical features of each constituent. Constraint has come about tacitly through the unification process--only compatible partial descriptions survive into the final result. This has the benefit that the planner need not be concerned with grammatical constraints and dependencies, but also implies the corresponding potential deficit that the planner cannot make use of knowledge of the grammatical constraints should it want to.

From the point of view of grammar development, FUGs are a satisfying treatment because they allow one to state the facts of the language compactly, i.e. interactions between statements need not be explicitly spelled out in the notation (as they would have to

---

[10] Recent work by Tony Patten [49] uses a systemic grammar in very much the same way. Operations at a semantic level, of the kind performed in other approaches by planning level specialists, specify a set of output features within the systemic grammar, the equivalent of the initial functional description that drives a FUG. A backwards and then forwards chaining sweep through the systemic grammar then determines what additional linguistic features must be added to the specification for a grammatical text to result.

be in unaugmented treatments of phrase structure grammar) since they will come about automatically through the action of unification.

4.2 Surface Structure as an intermediate level of representation

Faced with the difficulties under a message-directed direct-replacement approach of realizing conceptual relations directly as words, a number of generation researchers have independently choosen to interpose a level of explicitly linguistic representation between the levels of the message and the words of the text (McDonald [41,44], Kempen and Hoenkamp [34], Jacobs [31], Swartout [55]). They believe that a syntactic description of the text under construction is the best means of dealing with the problems of grammatically motivated detail and the implementation of linguistically defined constraints and dependencies. The specifics of their individual treatments differ, but a common thread is clearly identifiable. The linguistic structures are produced as the output of realization, which tends to be organized as choices made by specialists. The representations consist of a phrase structure of one or another sort, i.e. hierarchies of nodes and constituents. They incorporate functional concepts like "subject" and "focus". They are most aptly characterized as a kind of "surface structure" in the generative linguist's sense, i.e. they undergo no derivation, and are a proper description of the syntactic properties of the text that is produced.

Loosely speaking, this intermediate level of surface structure is used by the control structure in the same manner in all treatments. It is given as a tree, and its constituency pattern is used directly as the specification of a path--topdown and left to right through the tree--that controls the sequence and environment of realization and the order in which the words appear. The crucial consequence of this "folding together" of the process of realizing the elements of the message and traversing the surface structure is to provide an explicit, examinable representation of the grammatical context in which an element will appear, and thus make it available to constrain the choices open to realization and the text planner.

The most elaborated theory of surface structure as an intermediate representation is McDonald's. His design incorporates several points beyond the common elements of this approach. The diagram below is from [45]. It shows a surface structure as it would be in the middle of producting the text "*Two oil tankers were reported hit by missiles*".

[[insert figure two about here]]

The traversal path through the structure is indicated by the arrows; the system is just about to select a realization for the underlying program predicate #<hit-by-missiles>. The realization is performed in the context of the constraints dictated by its position as a constituent within the sentence, which is represented by the labels in brackets above it. The labeled circle marks an "attachment point" where the surface structure may be extended by splicing in additional phrase structure, in this case the verb phrase and complement structure for the verb "*report*". This provides the capacity for producing texts whose hierarchical structures are different from that of the message that lead to them, the customary form of texts constructed under a message-driven control structure.

4.3 Direct control of realization by the grammar: systemic grammar and ATNs

The augmented transition network, or ATN, was adapted for use in generation almost from the moment of its definition. It was used first by Robert Simmons and Jonathan Slocum in 1970 [58,59], whose system was then used by Neil Goldman [22]. It was also independently adapted by Stuart Shapiro [53,56], whose generator is the most elaborate of the group. All of the systems have a similar design. They scan a data structure provided by an underlying program, in effect "parsing" it. The networks follow the top-down

format found in most ATN parsers, leading naturally to a progressive refinement process as the generator scans its governing data structure from the most important, widest scope, relations on down. For the early ATNs this structure was a semantic net based on the concept of verb-centered case frames (another "functional" linguistic system). A special node in the network, a "modality vector", specified the root-level information such as tense and aspect, or whether a sentence was to be active or passive. The primary function of the ATN in the early systems was (1) to linearize a network structure that was for the most part already encoded in a linguistic vocabulary, and (2) to supplement the conceptual information in the semantic net with the purely linguistic information that all grammars must provide in generation.

As a linguistic formalism, ATNs are essentially a procedural encoding of a generative grammar [66]. The registers that give them their "augmented" power are used as a deep-structure representation of grammatical relations, and the paths through the network encode all of the alternative surface-level constituent sequences. Constraints propagate from higher parts of the surface structure tree to lower (i.e. to recursive subnets of the ATN) through the values in designated registers, bringing the activity of those subnets under contextual control. Shapiro's ATN design is particularly enlightening, as his controlling data structure is the underlying program's entire computational state. (This state is encoded in a particularly sophisticated intensional network formalism known as SNEPS [54].) The "parsing" his ATN performs amounts to the construction of an assessment, in terms appropriate for directing the generation of a text, of the steps that must be taken to satisfy the program's intended communicative goals--in effect an implicit dynamic message.

A further aspect of the ATN design, the fact that the means of actually producing the words of the text is the execution of a side-effect action on the traversal of an arc, brings out the fact that this approach commits the generator to action almost at the very moment that a situation is perceived, e.g. identification of the object that is to serve as the subject is followed directly by its realization and actual production. That this is possible is particularly striking when one appreciates that Shapiro's ATN never backs up [55]. This is quite unusual behavior for an ATN, given that they are usually thought of as expressly nondeterministic devices, and it serves to emphasize the fact that generation is in its essence a process of planning. Since modern planning processes are characteristically determinate, proceeding by incremental refinement and the posting of constraints rather than trial and error, the behavior of Shapiro's ATN is to be expected.

Viewed as a planner, the most significant deficit of the ATN designs is the difficulty of decoupling perception from action. Generators based on systemic grammar deal with this problem directly by introducing an intermediary representation in the form of a set of features--abstract symbols that serve as partial specifications of the text. To make a choice is to select a feature, which in turn creates a need to make certain other choices while rendering still others irrelevant. As was the case with surface structure, the use of an intermediary representation allows the specification of a text to be accumulated gradually, giving constraints an opportunity to propagate and influence later decisions. In this instance the abstract linguistic properties doing the constraining are not already bundled and formed as a phase structure but are distributed as a feature space.

The overall specification of the text is determined in recursive layers topdown, as it is in nearly all of the approaches (the prime exception being systems that use phrasal lexicons). Features are accumulated at a given level, e.g. the main clause of a sentence, until all of the aspects in which clauses can vary have been considered and the options settled. During this phase the issue is what functions are appropriate for the clause to carry out, given the situation and the speaker's intentions; with those determined, the functional features are realized as a group and specify the clause's form. That form now creates an environment for the constituents of the clause. The determination of what functions each of them should serve is then carried out, and when completed, will lead to the realization of

their forms, which in turn will lead to a functional analysis of their own constituents, and so on recursively until the constituents are words; at which point the text is read out as it would be from the description constructed with a FUG.

As a linguistic tradition, systemic grammar owes its form and perspective principally to one person, Michael A. K. Halliday [26], who was himself influenced by the London School of functionalism lead by Firth [18]. The influence of systemic grammar on generation research is considerably wider than just the systems that employ it directly, since it is the sole well known linguistic formalism that has as its very basis the identification of the choices implicit in a language. Choices form the notational basis of systemic grammars, which, like ATNs are written as traversable graph structures which define the space of possible control flow for at least the linguistic portion of the generation process. The very small fragment of a grammar shown below in Figure Three (taken from [27]) illustrates how the graph is formed.

[place Figure Three about here]

Choice systems are given either as "and" paths (leading curled brace), where one choice must be made from each of the systems named on the right, or as "or" paths (leading square brace), where only one of the alternative features listed may be selected. The selection of a feature opens the system that it names (n.b. the feature is the leftward "rootnode" of the tree on its side that constitutes a system within the network), which means that a choice from that system must now be made. Choices continue as the locus of control moves left to right through the network (usually simultaneously active in several choices at once due to the presence of the "and" systems), until a rightmost system is reached that consists of a bare feature without an accompanying system. These rightmost nodes are the concrete elements from which specifications of form are built up. Leftward pointing curled braces indicate path mergers in the control flow, where decisions in disjoint systems have a combined influence.

Two important generation systems have been based on systemic grammar, Davey's PROTEUS [17] (discussed earlier), and William Mann and Christian Mattheissen's NIGEL [36,39]. NIGEL is the largest systemic grammar in the world and very likely one of the largest machine grammars of any sort. Besides the quite important contribution simply of articulating a systemic grammar so thoroughly, Mann and Mattheiessen have developed an original technique for formalizing the usage criteria that govern the choices the grammar defines [37]. A set of criterial predicates, "choosers", are defined for each choice system in the grammar, which act as functions from the internal state of the planner and underlying program to features. The generation process is carried out by starting at the leftmost entry system of the nextwork and applying sucessive chooser procedures to determine the path through the network (i.e. the feature set) that best captures the speaker's intentions.

5. OTHER RESEARCH AREAS

The field of natural language generation, even as seen only by researchers in AI, is considerably larger than this article has been able to accomodate. Two areas must at least be mentioned in passing.

5.1 Planning

Pioneering work by Doug Appelt [1,2] supplied a rigorous logical framework by which to encode basic notions such as intention and reference. His planning technique, the progressive elaboration of goals through the use of Sacerdoti's procedural networks formalism [52], builds on a tradition of viewing the articulation of a generator's goals by chaining backwards from fundamental communications goals [14,49].

From a complementary direction, Kathleen McKeown has presented a theory of the organization of paragraphs into groups of conversational moves [48], drawing on earlier work by Grimes [25]. She employs paragraph schemas as realizations of high level moves such as "compare and contrast". The schemas act as templates to organize the content selection and rhetorical structuring that the planner does.

5.2 Psycholinguistic Theory

Once there are generation systems that have a significant capability, it becomes possible to consider deliberately choosen restrictions on the power of the virtual computational engine underlying the system's capacity. Such restrictions may provide explanitory accounts of aspects of the human generation process by appealing to intrinsic properties of the machine that make it impossible for its behavior to be otherwise. There has been work towards this end by Kempen and Hoenkamp for restarting phenomena [34], and by McDonald for an account of people's fluency and lack of grammatical error, and for certain classes of speech errors [44].

Generation is a young research area. It is populated by a vigorous, mutually identifying group of researchers that is growing at an ever inceasing rate. The intellectual climate within the generation community is not unlike that of the language understanding community of about 1974, with a roughly similar number of players and a similar feeling in the air of significant things happening. There is every reason to believe that the further development and contributions of generation research to AI as a whole in the next twelve years will be every bit as large as the contributions of understanding research in the last twelve.

6. BIBLIOGRAPHY

[1] D. Appelt, "Problem Solving Applied to Language Generation", Proc. ACL, Philadelphia, 1980, 59-63.

[1a] Ref. 1, p108.

[2] _____, **Planning English Sentences**, Cambridge University Press, Cambridge U.K., 1985.

[3] M. Bates & R. Ingria, "Controlled transformational sentence generation", Proc. ACL Stanford CA, 1980.

[4] J. Becker, "The Phrasal Lexicon", Proc. TINLAP-I, ACM, 1975, 60-64; BBN Report 3081.

[5] S. Bossie, **A Tactical Component for Text Generation: Sentence Generation Using a Functional Grammar**, Univ. Pennsylvania, TR MS-CIS-81-5, 1981.

[6] J. Bresnan (ed), **The Mental Representation of Grammatical Relations**, MIT Press, Cambridge, Mass., 1984.

[7] G. Brown, **Some Problems in German to English Machine Translation**, MIT LCS TR 142, 1974.

[8] R. Brown, **Use of multiple-body interrupts in discourse generation**, Bachelor's thesis, MIT Dept. EE&CS, 1974.

[9] B. Bruce, "Generation as social action", TINLAP-I, ACM, 1975, 74-78.

[10] D. Chester, "The translation of formal proofs into English", **Artificial Intelligence** 8(3), 261-278, 1976.

[11] W. Clancey, "Tutoring Rules for Guiding a Case Method Dialog", **IJMMS II**, 25-49, 1979.

[12] J. Clippinger, "Speaking with many tongues: Some problems in modeling speakers of actual discourse", Proc. TINLAP-I, ACM, 1975, 68-73.

[13] _____, **Meaning and Discourse: a computer model of psychoanalytic speech and cognition**, Johns Hopkins, 1977.

[14] P. Cohen, **On Knowing What to Say: Planning Speech Acts**, Univ. Toronto TR 118, 1978.

[15] F. Danes, **Papers on Functional Sentence Perspective**, Academia, Czech. Acad. Sci. 1974.

[16] L. Danlos, "Conceptual and Linguistic Decisions in Generation", Proc. COLING, Stanford CA, 1984, pp 501-504.

[17] A. Davey, **Discourse Production**, Edinburgh University Press, Edinburgh U.K., 1979.

[18] J.R. Firth, **Papers in linguistics 1934-1951**, Oxford Univ. Press, Oxford U.K., 1957.

[19] K. Forbus, A. Stevens, "Using Qualitiative Simulation to Generate Explanations", **Cognitive Science** 3, 1981.

[20] C. Frank, **A Step Towards Automatic Documentation**, MIT AI Lab WP-213, 1980.

[21] J. Friedman, "Directed random generation of sentences", **CACM** 12(6), 40-46, 1969.

[22] N. Goldman "Conceptual Generation", in Schank, R. **Conceptual Information Processing**, North-Holland/Elsevier, 289-372, 1975.

[23] _____, "The boundaries of language generation", Proc TINLAP-I, ACM, 1975, 74-78.

[24] R. Granville, "Controlling Lexical Substitution in Computer Text Generation", COLING, Stanford CA, 381-384, 1984.

[25] J. Grimes, **The Thread of Discourse**, Mouton, The Hague, 1975.

[26] M.A.K. Halliday, "Notes on transitivity and theme in English", **J. Ling.** 3(1), 37-81, 1967.

[27] _____ & J. Martin (eds) **Readings in Systemic Linguistics**, Batsford Academic, London, 1981.

[28] G. Heidorn, "Augmented phrase structure grammar", Proc. TINLAP-I, ACM, 1-5, 1975.

[29] E. Hovy, "Integrating Text Planning and Production in Generation, Proc. IJCAI, Los Angeles, August 1985, 848-851.

[30] P. Jacobs, **PHRED: A generator for natural language interfaces**, Berkeley CS Dept. TR 85/198, 1985.

[31] _____, **A Knowledge-Based Approach to Language Production**, Berkeley CS Dept TR 86/254, 1985.

[32] M. Kay, "Functional Grammar", Proc. Berkley Ling. Soc., 1979.

[33] _____, "Functional Unification Grammar: a formalism for machine translation", Proc. COLING, Stanford CA, July 1984, 75-78.

[34] G. Kempen & E. Hoenkamp, "Incremental sentence generation: implications for the structure of a syntactic processor", Proc. COLING, Prague, August 1982.

[35] K. Kukich, **Knowledge-Based Report Generation: A Knowledge Engineering Approach to Natural Language Report Generation**, Ph.D. Thesis, Inf. Sci. Dept. Univ. Pittsburgh, 1983.

[36] W. Mann, **The Anatomy of a Systemic Choice**, ISI TR/RS-82-104. 1982

[37] _____, **Inquiry semantics: a functional semantics of natural language**, ISI Tr/RS-83-8, 1983.

[38] _____, M. Bates, B. Grosz, D. McDonald, K. McKeown, W. Swartout, "Text Generation: the state of the art and literature", **JACL** 8(2), 1982.

[39] _____& Matthiessen, "Nigel: a Systemic Grammar for Text Generation", in Freedle (ed.) **Systemic Perspectives on Discourse: Selected Theoretical Papers of the 9th Intl. Systemic Workshop**, Ablex, 1985.

[40] _____ & J. Moore, "Computer generation of multi-paragraph English text", **JACL** 7(1), 1981.

[41] D. McDonald, "A Preliminary Report on a Program for Generating Natural Language", Proc. IJCAI-75, Wm. Kaufman, 401-405, 1975.

[42] _____, "Subsequent Reference: syntactic and rhetorical constraints", in **Theoretical Issues in Natural Language Processing II**, ACM, 38-47, 1978.

[43] _____, "Natural Language Generation as a Computational Problem: an introduction", in Brady & Berwick (eds) **Computational Models of Discourse**, MIT Press, 1983, 209-266.

[44] _____, "Description Directed Control: Its implications for natural language generation", in Cercone (ed), **Computational Linguistics**, Plenum Press, 403-424, 1984.

[45] _____, J. Pustejovsky, "TAGs as a Grammatical Formalism for Generation", Proc ACL, Chicago, July 1985, 94-103.

[46] _____, "Description-Directed Natural Language Generation", Proc IJCAI, Los Angles, 799-805, 1985.

[47] R. McGuire, **Political primaries and words of pain**, ms. Yale AI Group, 1980.

[48] K. McKeown, **Text Generation**, Cambridge Univ. Press, Cambridge U.K., 1985.

[49] T. Patten, "A Problem Solving Approach to Generating Text from Systemic Grammars", Proc. Eur. Mtng. Assoc. Comp. Ling., 251-256, 1985.

[50] R. Power, "The organisation of purposeful dialogues", **Linguistics** 17, 1979, 107-151.

[51] G. Ritchie, "The computational complexity of sentence generation using functional unification grammar", Proc. COLING, Bonn, West Germany, August 25-29, 1986.

[52] E. Sacerdoti, **A Structure for Plans and Behavior**, Elsevier North-Holland, 1977.

[53] S. Shapiro, "Generation as parsing from a network into a linear string", **JACL** Fiche 33, 45-62, 1975.

[54] S.C. Shapiro, "The SNePS semantic network processing system", in Findler (ed) **Associative Networks**, Academic Press, 1979.

[55] _____, pers. comm., SUNY at Buffalo, August 1979.

[56] _____, "Generalized Augmented Transition Network Grammars for Generation from Semantic Networks", **JACL** 8(1), 1982, 12-25.

[57] B. Sigurd, "Computer Simulation of Spontaneous Speech Production", Proc. COLING, Stanford CA, July 1984.

[58] R. Simmons & J. Slocum, "Generating English discourse from semantic networks", CACM 15(10) 1972, 891-905.

[59] J. Slocum, **Question Answering via Cannonical Verbs and Semantic Models: Generating English from the Model,** Univ. Texas, Dept. C.S., TR NL-23, 1973.

[60] W. Swartout, **A Digitalis Therapy Advisor with Explanations,** MIT LCS Technical report, 1977.

[61] W. Swartout, pers. comm., Information Sciences Institute, Los Angeles July 1984.

[62] H. Thompson, "Strategy and tactics: A model for language production", Proc. Chicago Ling. Soc., 1977.

[63] R. Wilensky, Y. Arens, D. Chin, "Talking to UNIX in English: An overview of UC", CACM, 577-593, June 1984.

[64] T. Winograd, **Understanding Natural Language,** Academic Press, 1972.

[65] H.K.T. Wong, **Generating English Sentences from Semantic Structures,** Univ. Toronto Dept. C.S. TR 84, 1985.

[66] W. Woods, "Transition network grammars for natural language analysis", CACM 13(10), 1970, 591-606

[67] V.H.A. Yngve, "A model and a hypothesis for language structure", Proc. Am. Phil. Soc., 444-466, 1960.

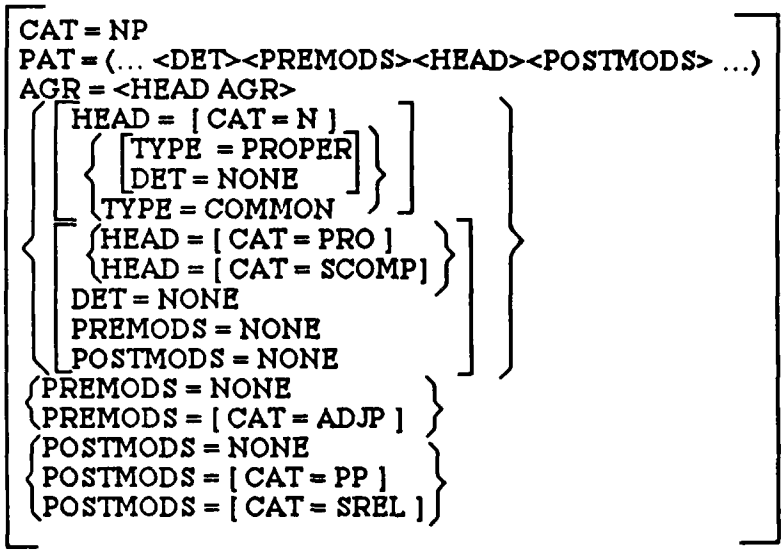

CAT = NP
PAT = (... <DET><PREMODS><HEAD><POSTMODS> ...)
AGR = <HEAD AGR>
HEAD = [CAT = N]
TYPE = PROPER
DET = NONE
TYPE = COMMON
HEAD = [CAT = PRO]
HEAD = [CAT = SCOMP]
DET = NONE
PREMODS = NONE
POSTMODS = NONE
PREMODS = NONE
PREMODS = [CAT = ADJP]
POSTMODS = NONE
POSTMODS = [CAT = PP]
POSTMODS = [CAT = SREL]

Figure One

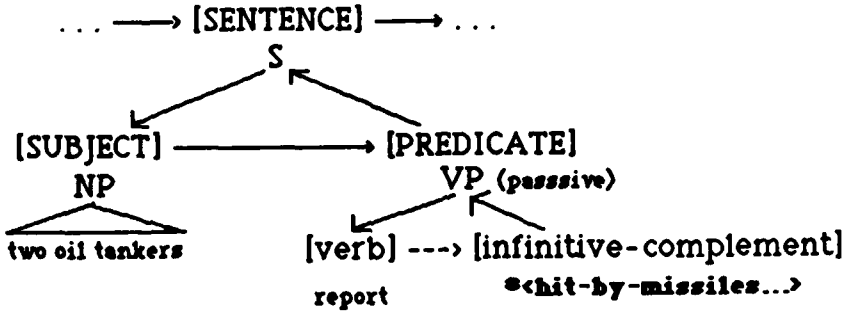

. . . → [SENTENCE] → . . .
S
[SUBJECT]
[PREDICATE]
NP
VP (passive)
two oil tankers
[verb] ---> [infinitive-complement]
report
*<hit-by-missiles...>

Figure Two

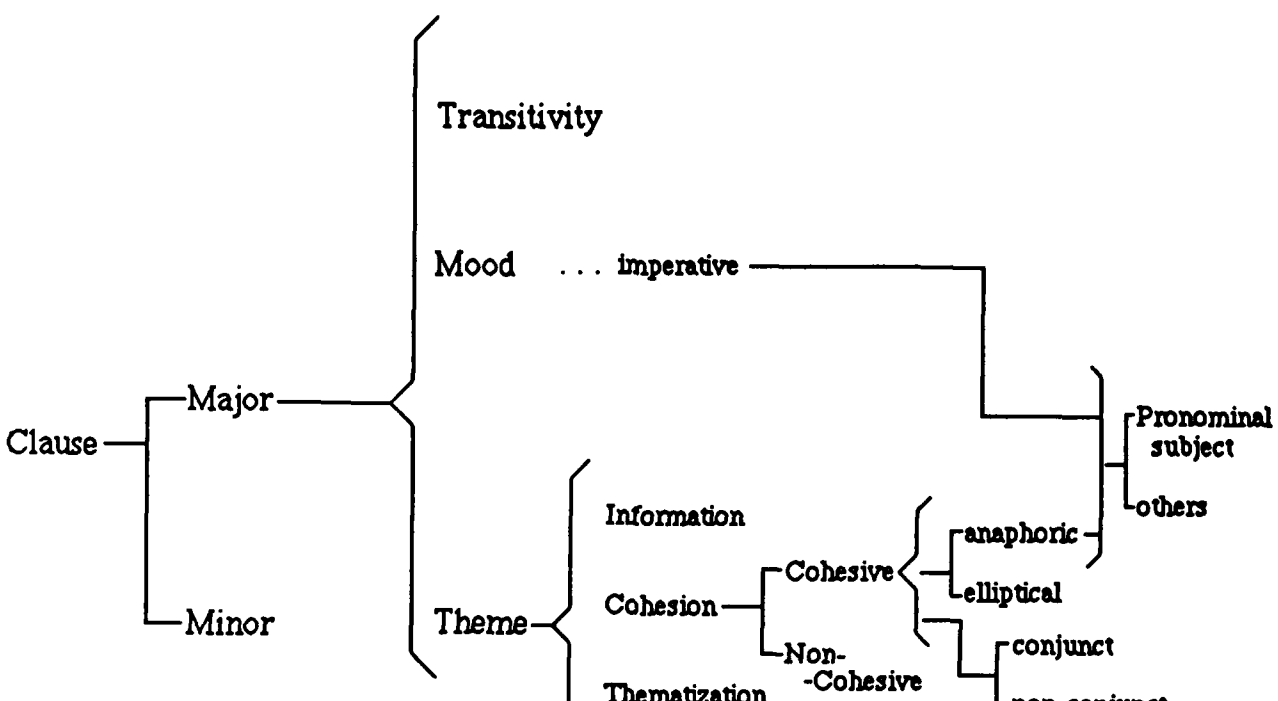


Figure Three

END

2-87

DTIC