Second Edition

ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE

Volume 2
M-Z

Awarded

American Library Association's

Outstanding Reference Source

Association of American Publishers Award

Best New Professional and Scholarly Publication

Stuart C. Shapiro
Editor-in-Chief

Natural Language

David D. McDonald, NATURAL LANGUAGE GENERATION, 983-997

From Shapiro, Stuart C., Editor-in-Chief, Encyclopedia of Artificial Intelligence, 2nd Ed., John Wiley & Sons, Inc., NY, 1992.

NATURAL LANGUAGE GENERATION

Natural language generation is the process of deliberately constructing a natural-language text in order to meet specified communicative goals. The term "text" 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 "spoken" language is identified with a lack of revision, most programs today "speak" even though nearly all only display words on a 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 are only mentioned when they are an issue in a program's design.

The goals come from another program, perhaps an expert reasoning system or an 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 multisentence remarks and questions within a dialogue 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 preprogrammed, "canned" text. Even the use of parameterized "format" statements, where the canned word string can be augmented by names or simple descriptions by substituting for variables, 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 such "engineering treatments," research on natural language generation, like the other areas of its parent field of computational linguistics (qv), 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 enable them to be versatile, varying their texts in form and emphasis to fit an enormous range of speaking situations, and 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 AI.

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 in order to establish 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 entry the common approaches to generation are surveyed, including characteristic messages and the nature of a generator's lexicon. A separate section continues the survey with alternative approaches to the representation and uses of a grammar.

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 (qv). In contrast with the organization of the understanding process—which to a first approximation can follow the traditional stages of a linguistic analysis: morphology (qv), 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 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 the actual text. In addition, mismatches in the speaker's and audience's model of the situation (and especially of each other) led 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 underspecification, a generator's problem is to choose from its oversupplied 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 precedes 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 precedes any construction of syntactic structures to realize it; and the syntactic context of a word must be fixed before the precise morphological 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 (qv) 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 and 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 precedes content selection and rhetorical planning, which precedes 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 on 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 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 stand-alone facilities and concentrate on studying grammar or planning in isolation (Mann and Mathiessen, 1985; Appelt, 1985; Bates and Ingria, 1980). Others have dedicated a great deal of their own development effort to building a task-based conceptual program on their generator and give it something substantive to talk about (Davey, 1979; Clippinger, 1977; Kukich, 1983). Still others work from an independently developed program but have interposed some kind of independent "planning" system in between to patch over the differences (McDonald, 1985; McKeown, 1985). 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 (Goldman, 1975; McDonald, 1983; Mann and Moore, 1981).

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. In a good man–machine interface today one would also expect provisions for coordinated graphical input and output, complementing the natural language I/O. 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 nonlinguistic 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, 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 (eg, McDonald, 1985; Wilensky and co-workers, 1984). 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
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 (eg, concepts, relations, individuals) and the adoption of a coordinating rhetorical framework or schema for the utterance as a whole (eg, 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 on 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 (McDonald, 1983), and in some the planning and goal-identification processes are together called the *strategic component* (Thompson, 1977). 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 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 fulfilling 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 ensure that the texts that are produced follow the rules of the language, ie, 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 nonlinguistic 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, eg, 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, eg, McDonald, 1985; Appelt, 1980; Danlos, 1984). 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.

STATE OF THE ART

There is a firm consensus within the field (Mann and coworkers, 1982) that versatility and creativity in machine-generated text is possible only if all three of the following apply:

1. The generator incorporates a comprehensive linguistically principled grammar
2. The underlying program has a sophisticated, commonsensical, conceptual view

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 Winograd's SHRDLU program (1972). 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.

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 1970s 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-blanks templates. It is a straightforward 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 (Forbus and Stevens, 1981; Frank, 1980).

Generation researchers, however, are interested in more complex texts than the context-free presentation 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 Sigurd (1984). 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.

Although 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 database of an 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" (McDonald, 1978). What wording should be chosen when a reference to an object appears more than once in the text? Always using a pronoun may introduce ambiguities; 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 Granville (1984). 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 made 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 in which 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 attitudes 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 Davey (1979) in 1974. 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, eg, 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 that selects the best level of description for each move (eg, "block" vs "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 (eg, "threat-but-block" or "although-threat-block-&-counter"). A grammar and realization facility than takes the described and grouped moves, works out the details of their form as English sentences, and produces the words of the text.

A rival to Davey's PROTEUS in fluency is Clippinger's 1974 program ERMA (1977). 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, midsentence, omissions or unintended interpretations that they fix by dynamically replanning and restarting their speech in midutterance. To model this behavior, Clippinger, working with an undergraduate assistant, Brown (1974), analyzed 40 hours of transcripts of a patient in psychoanalysis in order to understand that patient's motivations and reasoning patterns sufficiently well 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. (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.) 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 terms of computer programming for 1974, and the project was not carried further.

HISTORICAL PERSPECTIVES ON THE PROBLEM

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 field. There are two reasons for this: first, until the early 1980s, comparatively few people had ever worked on the problem of generation and, second, 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 1970s, 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 (1972) on SHRDLU and of Woods (1970) with augmented transition networks (ATNs) (see GRAMMAR, AUGMENTED TRANSITION NETWORK). In addition to Davey and Clippinger, there was the work of Simmons and Slocum on the adaptation of ATNs to generation (Shapiro, 1982) and the thesis of Goldman (1975) focusing on how to organize word choice when generating from conceptual-dependency (qv) networks as well as other works (Brown, 1974; Halliday and Martin, 1981; McDonald, 1975; Shapiro, 1975; Wong, 1985). It is fair to say, however, that the initial reports of that generation work, principally at the important TINLAP meeting in 1975 (Bruce, 1975; Clippinger, 1975; Goldman, 1975), fell largely on deaf ears, and research on generation went into something of a hiatus for the last half of the decade. This is not to say that no work was done during those years; rather, 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.

Until the early 1980s 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 1970s, say, (#supports :block 6 :block 3), 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, generation would not be an important research problem. However, as soon as one begins to consider the various ways that simple sentence could be rewritten—the versatility 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 forward 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 backward from the complexities of actual human texts).

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 (see GRAMMAR, SEMANTIC), demon-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 have any schools in this sense. This is partially because historically only a small number of individuals 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 that 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 questions will be described throughout the rest of this entry. This section covers the nature of messages and approaches to the lexicon; the following section considers various treatments of grammar.

Control by Progressive Refinement of Message

All treatments of the diversity of forms have been bound together with accounts of control, making control the proper place to start in looking at the schools of thought as to 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, ie, to see it as an expression that one evaluates with a special kind of interpreter. Again a caution is in 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 progressive 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 practitioners of AI to follow a function-parameter, predicate-argument 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 (Swartout, 1977; Clancey, 1979) (see EXPLANATION), 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 is by an early version of McDonald's program MUMBLE (1983); the proof itself is taken from earlier generation work of Chester (1976).

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 (eg, "premis" or "universal instantiation"). The model provides an account of the motivations of the proof writer in selecting what rule to apply, eg, that the point of the right 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 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 (McDonald, 1985). 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 McDonald and Pustejovsky, 1985). 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 (eg, 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 researcher's goal is to approximate the fluency and specificity of texts authored by people, 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 crewman 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, pred-

icates, 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, eg, the two-place predicate "shaves(x,y)" becomes the clause template "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," etc. 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 commitment, expediting the generation process as a whole by avoiding the possibility of having to "back up" out of prematurely made realization decisions that turn out to be incompatible with the grammatical context defined by a higher template.

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 prominence. 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, eg, 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. 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 (Kay, 1984). He demonstrated how one would determine word choice by working outward 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" by testing whether, eg, the object ingested was a fluid or smoke.

Notice that one of the available words was "ingest," the least marked (most abstract) alternative 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.

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 Hovy (1985). 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 votes. 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 nonissue 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 cannot take place; rather, 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.

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 hyphenated symbol, eg, "example-intrinsic-similarities-with-compeditive-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 phenom-

ena without having the commonsense 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 by Becker (1975) and is an important tool of generation systems. Linguistically, a "phrasal" lexicon is a conceptual extension of a standard, word-based 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 hyphenated 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 Kukich (1983); another notable effort specifically employing a phrasal lexicon is that of Jacobs (1985).

> 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 stereotypical 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.

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, three approaches to this combined design decision 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)
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 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 (eg, "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, etc. 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 how to state this information and how to ensure 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 ones, with elaborate derivational paradigms like generative grammar being ruled out of consideration quickly. 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 Friedman (1969) for the use of linguists to check the consistency of large sets of rules and the pedagogical ICAI system of Bates that has been used in the teaching of English as a foreign language (Bates and Ingria, 1980).

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. One of the very earliest mechanical generation systems of any sort was developed by Yngve in 1959 using a pushdown automata and a body of context-free phrase-structure rules with ad-lib lexical insertion (Yngve, 1960). Though it was not message-driven and generated text that was semantic nonsense which consequently would make it uninteresting as a generator today, it did establish the legitimacy of the enterprise of provid-

As a linguistic formalism, ATNs are essentially a procedural encoding of a generative grammar (Woods, 1970). 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 (ie, to recursive subnets of the ATN) through the values is 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 (qv) (Shapiro, 1979).] 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; eg, 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 (Swartout, 1984). 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 phrase structure but are distributed as a feature space.

The overall specification of the text is determined in recursive layers top-down, 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, eg, 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, Halliday (1967), who was himself influenced by the London School of functionalism lead by Firth (1957). 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 that 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 in Figure 3 illustrates how the graph is formed.

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 selected. The selection of a feature opens the system that it names (*note:* the feature will be the leftward "root node" 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 (1979) (discussed earlier) and Mann and Matthiessen's (1985) NIGEL (Mann, 1982). 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 Matthiessen have developed an original technique for formalizing the usage criteria that govern the choices the grammar defines (1983). A set of criterial predicates 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 successive "chooser" procedures to determine the path through the network (ie, the feature set) that best captures the speaker's intentions.

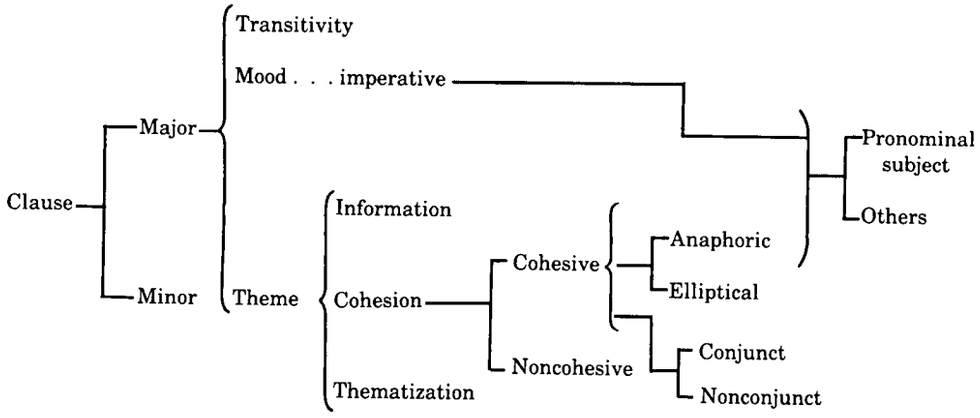

Figure 3. From Halliday and Martin (1981).

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 accommodate. Two areas must at least be mentioned in passing.

Planning. Pioneering work by Appelt (1980, 1985) 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 (1977), builds on a tradition of viewing the articulation of a generator's goals by chaining backward from fundamental communications goals (Power, 1979; Cohen, 1978).

From a complementary direction, McKeown (1985) has presented a theory of the organization of paragraphs into groups of conversational moves, drawing on earlier work by Grimes (1975). 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.

Psycholinguistic Theory. Once there are generation systems that have a significant capability, it becomes possible to consider deliberately chosen restrictions on the power of the virtual computational engine underlying the system's capacity. Such restrictions gain the possibility of providing an explanatory account 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 toward this end by Kempen and Hoenkamp (1982) for restarting phenomena and by McDonald (1984) for an account of people's fluency and lack of grammatical error and certain classes of speech errors.

Generation is a young research area. It is populated by a vigorous, mutually identifying group of researchers that is growing at an ever-increasing 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 12 years will be every bit as large as the contribution of understanding research in the last 12.

BIBLIOGRAPHY

D. Appelt, "Problem Solving Applied to Language Generation," *Proceedings of the ACL,* Philadelphia, Pa., 1980, pp. 59–63.

D. Appelt, *Planning English Sentences,* Cambridge University Press, Cambridge, U.K., 1985.

M. Bates and R. Ingria, "Controlled Transformational Sentence Generation," *Proceedings of the ACL,* Stanford, Calif., 1980.

J. Becker, "The Phrasal Lexicon," *Proceedings of TINLAP-I,* ACM, pp. 60–64; also as Report 3081, Bolt Beranek and Newman, Cambridge, Mass., 1975.

S. Bossie, "A Tactical Component for Text Generation: Sentence Generation Using a Functional Grammar," TR MS-CIS-81-5, University of Pennsylvania, Philadelphia, 1981.

J. Bresnan, ed., *The Mental Representation of Grammatical Relations,* MIT Press, Cambridge, Mass., 1984.

G. Brown, *Some Problems in German to English Machine Translation,* MIT LCS TR 142, MIT, Cambridge, Mass., 1974.

R. Brown, *Use of Multiple-Body Interrupts in Discourse Generation,* Bachelor's thesis, MIT, Cambridge, Mass., 1974.

B. Bruce, "Generation as Social Action," *Proceedings of TINLAP-I,* ACM, 1975, pp. 74–78.

D. Chester, "The Translation of Formal Proofs Into English," *Artif. Intell.* 8(3), 261–278 (1976).

W. Clancey, "Tutoring Rules for Guiding a Case Method Dialog," *Proceedings of IJMMS II,* 1979, pp. 25–49.

J. Clippinger, "Speaking with Many Tongues: Some Problems in Modeling Speakers of Actual Discourse," *Proceedings of TINLAP-I,* ACM, 1975, pp. 68–73.

J. Clippinger, *Meaning and Discourse: A Computer Model of Psychoanalytic Speech and Cognition,* Johns Hopkins Press, Baltimore, MD, 1977.

P. Cohen, *On Knowing What to Say: Planning Speech Acts,* TR 118, University of Toronto, 1978.

F. Danes, *Papers on Functional Sentence Perspective,* Academia, Czechoslovakian Academy of Science, 1974.

L. Danlos, "Conceptual and Linguistic Decisions in Generation," *Proceedings of the COLING,* Stanford, Calif., 1984, pp. 501–504.

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

J. R. Firth, *Papers in Linguistics 1934–1951,* Oxford University Press, Oxford, U.K., 1957.

K. Forbus and A. Stevens, "Using Qualitative Simulation to Generate Explanations," *Proceedings of the Third Annual Conference of the Cognitive Science Society,* Berkeley, Calif., August, 1981, pp. 219–221.

C. Frank, *A Step Towards Automatic Documentation,* WP-213, MIT AI Laboratory, 1980.

J. Friedman, "Directed Random Generation of Sentences," *CACM* **12**(6), 40–46 (1969).

N. Goldman, "The Boundaries of Language Generation," *Proceedings of TINLAP-I,* ACM, 1975, pp. 74–78.

N. Goldman, "Conceptual Generation," in R. Schank, ed., *Conceptual Information Processing,* North-Holland/Elsevier, Amsterdam, 1975, pp. 289–372.

R. Granville, "Controlling Lexical Substitution in Computer Text Generation," *Proceedings of COLING,* Stanford, Calif., 1984, pp. 381–384.

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

M. A. K. Halliday, "Notes on Transitivity and Theme in English," *J. Ling.* **3**(1), 37–81 (1967).

M. A. K. Halliday and J. Martin, eds., *Readings in Systemic Linguistics,* Batsford Academic, London, 1981.

E. Hovy, "Integrating Text Planning and Production in Generation," *Proceedings of the Ninth IJCAI,* Los Angeles, 1985, pp. 848–851.

P. Jacobs, "PHRED: A Generator for Natural Language Interfaces," TR 85/198, Berkeley Computer Science Department, Berkeley, Calif., 1985.

P. Jacobs, "A Knowledge-Based Approach to Language Production," TR 86/254, Berkeley Computer Science Department, Berkeley, Calif., 1985.

M. Kay, "Functional Grammar," *Proceedings of the Berkeley Linguistic Society,* Berkeley, Calif., 1979.

M. Kay, "Functional Unification Grammar: A Formalism for Machine Translation," *Proceedings of COLING,* Stanford, Calif., 1984, pp. 75–78.

G. Kempen and E. Hoenkamp, "Incremental Sentence Generation: Implications for the Structure of a Syntactic Processor," *Proceedings of COLING,* Prague, 1982.

K. Kukich, *Knowledge-Based Report Generation: A Knowledge Engineering Approach to Natural Language Report Generation,* Ph.D. dissertation, University of Pittsburgh, 1983.

D. McDonald, "A Preliminary Report on a Program for Generating Natural Language," *Proceedings of the Fourth IJCAI,* Tbilisi, USSR, 1975, pp. 401–405.

D. McDonald, "Subsequent Reference: Syntactic and Rhetorical Constraints," *Theoretical Issues in Natural Language Processing II,* Association of Computing Machinery, New York, 1978, pp. 38–47.

D. McDonald, "Natural Language Generation as a Computational Problem: An Introduction," in M. Brady and R. Berwick, eds., *Computational Models of Discourse,* MIT Press, Cambridge, Mass., 1983, pp. 209–266.

D. McDonald, "Description Directed Control: Its Implications for Natural Language Generation," in Cercone, ed., *Computational Linguistics,* Plenum, New York, 1984, pp. 403–424.

D. McDonald, "Description-Directed Natural Language Generation," *Proceedings of the Ninth IJCAI,* Los Angeles, 1985, pp. 799–805.

D. McDonald and J. Pustejovsky, "TAGs as a Grammatical Formalism for Generation," *Proceedings of the ACL,* Chicago, 1985, pp. 94–103.

K. McKeown, *Text Generation,* Cambridge University Press, Cambridge, U.K., 1985.

W. Mann, *The Anatomy of a Systemic Choice,* TR/RS-82-104, Information Sciences Institute, 1982.

W. Mann, *Inquiry Semantics: A Functional Semantics of Natural Language,* TR/RS-83-8, Information Sciences Institute, 1983.

W. Mann, M. Bates, B. Grosz, D. McDonald, K. McKeown, and W. Swartout, "Text Generation: The State of the Art and Literature," *JACL* **8**(2) (1982).

W. Mann and C. Matthiessen, "Nigel: A Systemic Grammar for Text Generation," in Freedle, ed., *Systemic Perspectives on Discourse: Selected Theoretical Papers of the Ninth International Systemic Workshop,* Ablex, Norwood, N.J., 1985.

W. Mann and J. Moore, "Computer Generation of Multi-Paragraph English Text," *JACL* **7**(1) (1981).

R. Power, "The Organisation of Purposeful Dialogues," *Linguistics* **17,** 107–151 (1979).

G. Ritchie, "The Computational Complexity of Sentence Generation using Functional Unification Grammar," *Proceedings of COLING,* Bonn, 1986.

E. Sacerdoti, *A Structure for Plans and Behavior,* Elsevier/North-Holland, Amsterdam, 1977.

S. Shapiro, "Generation as Parsing From a Network Into a Linear String," *JACL Fiche* **33,** 45–62 (1975).

S. C. Shapiro, "The SNePS Semantic Network Processing System," in Findler, ed., *Associative Networks,* Academic Press, New York, 1979.

S. C. Shapiro, "Generalized Augmented Transition Network Grammars for Generation From Semantic Networks," *JACL* **8**(1), 12–25 (1982).

B. Sigurd, "Computer Simulation of Spontaneous Speech Production," *Proceedings of COLING,* Stanford, Calif., 1984.

R. Simmons and J. Slocum, "Generating English Discourse From Semantic Networks," *CACM* **15**(10), 891–905 (1972).

J. Slocum, *Question Answering via Cannonical Verbs and Semantic Models: Generating English from the Model,* Department of Computer Science TR NL-23, University of Texas, 1973.

W. Swartout, *A Digitalis Therapy Advisor with Explanations,* MIT LCS Technical Report, MIT, Cambridge, Mass., 1977.

W. Swartout, personal communication, Information Sciences Institute, Los Angeles, July 1984.

H. Thompson, "Strategy and Tactics: A Model for Language Production," *Proceedings of the Chicago Linguistic Society,* 1977.

R. Wilensky, Y. Arens, and D. Chin, "Talking to UNIX in English: An Overview of UC," *CACM,* 577–593 (June 1984).

T. Winograd, *Understanding Natural Language,* Academic Press, New York, 1972.

H. K. T. Wong, *Generating English Sentences from Semantic Structures,* TR84, Department of Computer Science, University of Toronto, 1985.

W. Woods, "Transition Network Grammars for Natural Language Analysis," *CACM* **13**(10), 591–606 (1970).

V. H. A. Yngve, "A Model and a Hypothesis for Language Structure," *Proceedings of the American Philosophical Society,* 1960, pp. 444–466.

D. D. McDonald
University of Massachusetts

NATURAL LANGUAGE UNDERSTANDING

Natural language communication with computers has long been a major goal of AI both for the information it can give about intelligence in general and for its practical utility. Databases, software packages, and AI-based expert systems all require flexible interfaces to a growing community of users who are not able or do not wish to communicate with computers in formal, artificial command languages. Whereas many of the fundamental problems of general natural language processing (NLP) by machine remain to be solved, the area has matured in recent years to the point where practical natural language interfaces to software systems can be constructed in many restricted, but nevertheless useful, circumstances. This article is intended to survey the current state of natural language processing by presenting computationally effective NLP techniques, by exploring the range of capabilities these techniques provide for NLP systems, and by discussing their current limitations. This presentation is organized in two major sections: the first on language recognition strategies at the single-sentence level and the second on language processing issues that arise during interactive dialogues. In both cases the concentration is on those aspects of the problem appropriate for interactive natural language interfaces but relate the techniques and systems discussed to more general work on natural language, independent of application domain.

NATURE OF NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) is the formulation and investigation of computationally effective mechanisms for communication through natural language. To take the bold face phrases in reverse order, first the subject area deals with naturally occurring human languages such as German, French, or English. Second, it is concerned with the use of these languages for communication, both communication between people, the purpose for which these languages evolved, and communication between a person and a computer. Third, NLP does not study natural language communication in an abstract way, but by devising mechanisms for performing such communication that are computationally effective, ie, can be turned into computer programs that perform or simulate the communication. It is this third characteristic that sets the NLP subarea of AI, itself a subarea of computer science, apart from traditional linguistics and other disciplines that study natural language. This article examines the relationship of NLP among two other closely related disciplines: linguistics and cognitive psychology (qv).

Linguistics is traditionally concerned with formal, general, structural models of natural language. Linguists, therefore, have tended to favor formal models that allow them to capture as much as possible the regularities of language and to make the most appropriate linguistic generalizations. Little or no attention was paid in the development of these models to their computational effectiveness. That is, linguistic models characterize the language itself, without regard to the mechanisms that produce it or decipher it. A good example, as shown below, is Chomskian transformational grammar perhaps the best known of all linguistic models, which turns out to be unsuitable as a basis for computationally practical language recognition (Chomsky, 1957, 1965; Petrick, 1965).

The goal of cognitive psychology (qv) on the other hand is not to model the structure of language but rather to model the use of language and to do it in a psychologically plausible way, where plausibility here is defined by correlation with experimental results, especially timing studies of language-understanding tasks (see Anderson (1976) for a good example of the flavor of this approach). This is somewhat closer to the spirit of AI-based NLP in its emphasis on the use of language in communication, but again it is not of primary importance to the cognitive psychologist whether the models are computationally effective. Moreover, the models produced are not often targeted at language understanding per se but at more general aspects of human cognition and memory organization, with natural language serving only as the vehicle through which these related phenomena are studied.

In addition to relating NLP to the study of language in other disciplines, a major division that arises within NLP itself should be pointed out. The distinction is between general and applied NLP. General NLP can be thought of as a way of tackling cognitive psychology from a computer science viewpoint. The goal is to make models of human language use and also to make them computationally effective. The vehicles for this kind of work are general story understanding, as in the work of Charniak (1972), Schank (1975), Cullingford (1978), Carbonell (1979), and others, and dialogue modeling, as in the work of Cohen and Perrault (1979), Allen (1979), Grosz (1977), Sidner (1979), and others. One of the most important lessons learned from this work is that general NLP requires a tremendous amount of real-world knowledge; most of the work just cited is mainly concerned with the representation of such real-world knowledge and its application to the understanding of natural language input. Unfortunately, AI has not yet reached the stage where it can routinely handle the amount of knowledge required for these tasks, with the result that systems constructed in this area tend to be pilot systems that demonstrate the feasibility of a concept or approach but do not contain a large enough knowledge base to make them work on more than a handful of carefully selected example natural language passages or dialogues.

Applied NLP, on the other hand, is not typically concerned with cognitive simulation but rather with allowing people to communicate with machines through natural language. The emphasis is pragmatic. It is less important in applied NLP whether the machine understands its natural language input in a cognitively plausible way than

whether it responds to the input in a way helpful to the user and in accordance with the desires expressed in the input. Typical applications are database interfaces, as in the work of Hendrix (1977), Grosz (1983), Kaplan (1979), and others, and interfaces to expert systems (qv), as in the work of Brown and Burton (1975) and Carbonell (1971) and Carbonell and co-workers (1983). Because such systems must operate robustly with real users, in addition to actually processing well-formed natural language, they must be concerned with the detection and resolution of errors and misunderstandings by the user.

Basic Problem of NLP

If there were one word to describe why NLP is hard, it is *ambiguity*. It arises in natural language in many different forms including the following.

Syntactic (or Structural) Ambiguity.

John saw the Grand Canyon flying to New York.
Time flies like an arrow.

Is it John or the Grand Canyon doing the flying? The answer depends on the ambiguous syntactic role of the word *flying* in this example. Again, is time flying, or are we talking about a species of insect called time flies in the second example. It depends whether *flies* is a noun or a verb. (Actually, the second example here has at least six different parsings.)

Word Sense Ambiguity.

The man went to the bank to get some cash.
and jumped in.

Here the word *bank* refers either to a repository for money or the side of a river, depending on the two different continuations.

Case.

He ran the mile in four minutes
the Olympics

Linguistically, a "case" refers to the relation between a central organizing concept, here an act of running, and a subsidiary concept, here time or location. In both examples the same preposition, *in*, indicates the two quite different relationships.

Referential.

I took the cake from the table and ate it.

What was eaten, the cake or the table? The answer is "obvious," but, independent of real-world knowledge, *it* could refer to either one. For instance, *it* would have a different referent in the example above if *ate* were replaced with *cleaned*.

Literalness.

Can you open the door?
I feel cold.

What are the correct interpretations here? There are some circumstances when the first question might be answered quite reasonably yes or no, eg, before setting off on a long journey to the place where the door is. On the other hand, it is easy to think of circumstances where the speaker might be very unhappy with such a reply. Again, the second example might be a statement of fact or a request to close a window. The ambiguities here lie in whether to interpret the utterance literally or whether to treat it as an indirect speech act (qv) (Searle, 1969), eg, an implicit request as in the examples above.

Because of these and other kinds of ambiguity, the central problem in NLP, and this is true for both the general and applied variety, is the translation of the potentially ambiguous natural language input into an unambiguous internal ie, internal to the program doing the processing, representation, as suggested by Figure 1.

The second layer of Figure 1 shows an example translation of a natural language database query into an expression in a database query language: the one used by the LADDER (Sacerdoti, 1977) system for access to its database of information about U.S. Navy ships. Note how a potentially ambiguous word such as *Kennedy* is resolved into the internal name, John F. Kennedy, of a specific ship, or *captain* is resolved into the name, Commander, of a field of the relational database conceptually underlying the LADDER system. The specific internal representation used here is, of course, highly specialized. In general, there is no standard commonly agreed upon for internal representations, and different types are useful for different purposes. A partial list includes:

- Expressions in a database query language (for DB access).
- Parse trees with word sense terminal nodes (for machine translation).
- LISP expressions (most often for expert system requests).
- Case frame instantiations (for a variety of applications).
- Conceptual dependency (for story understanding).

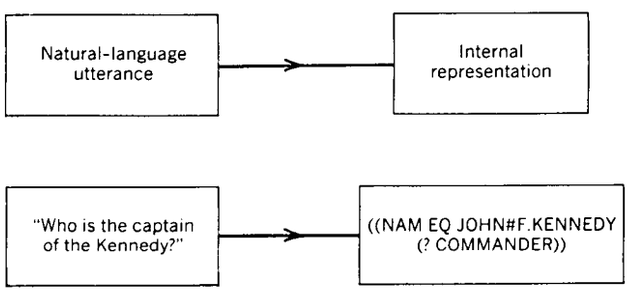

Figure 1. Translation from a natural language utterance to unambiguous internal representation.

In general NLP, translation of an utterance into an unambiguous internal representation can require inference based on a potentially unbounded set of real-world knowledge. Consider, for instance,

> Jack took the bread from the supermarket shelf, paid for it, and left.

Coming up with an unambiguous representation for this requires answers to such questions as

> What did Jack pay for? (the referent of *it*)
> What did Jack leave? (the ellipsed object of *left*)

and possibly even

> Did Jack have the bread with him when he left?

To answer these questions, information on supermarkets, buying and selling, and other real-world topics is required. As mentioned above, AI knowledge representation (qv) techniques have not yet developed to the stage where they can handle at an acceptable level of efficiency the large quantities of such knowledge required to do a complete job of understanding a large variety of topics. Moreover, even if the knowledge could be represented, unresolved problems in inference techniques remain a barrier to applying the correct knowledge to the input in order to produce the desired unambiguous internal representation. The result is that current general NLP systems are demonstration systems that operate with a very small amount of carefully handcrafted knowledge, specifically designed to enable the processing of a small set of example inputs. The main point of such systems is to investigate the feasibility of certain inference or knowledge representation techniques rather than to achieve broad coverage in the NLP they perform.

Applied NLP systems potentially face exactly the same problem, but they finesse it by taking advantage of certain characteristics of the highly limited domains in which they operate. Suppose the input

> How many terminals are there in the order?

was addressed to an expert system that acted as a computer salesman's assistant. Such a system need not consider many of the potential ambiguities lurking in this example. The word *terminals,* for instance, can be assumed to refer to computer terminals, rather than airport terminals, terminally ill patients, or terminal values of a mathematical series. Also, assuming the system processes one sales order at a time, *the order* can be assumed to refer to the current order without considering any others. In general, the technique is to premake as many inferences as possible in a way appropriate to the task at hand. For suitable tasks in many restricted domains, this has been used successfully to reduce the amount of knowledge that must be represented and the number of inferences that must be made to manageable proportions.

By restricting the natural language dealt with by an interface to that required to handle a limited task in a limited domain, it is thus possible to construct performance systems capable of useful natural language communication, and this represents the current state of the art in practical NLP. Clearly, this is far from satisfactory, because in particular, each task and domain that are tackled require careful preanalysis so that the required inferences can be preencoded in the system, thus making it difficult to transfer successful natural language interfaces from one task to another. Some research (Grosz, 1977) is being conducted to improve the portability of current interfaces, but until the problem of preencoding inferences is solved in a more general way, the portability issue will be the one that most hinders the widespread use of natural language interfaces. A practical alternative, however, is the Language Craft (Carnegie-Group Inc.) approach, where a development environment and grammar interpreter are provided to shorten drastically the development of new domain-specific interfaces.

NATURAL LANGUAGE ANALYSIS TECHNIQUES

In this section, several of the more common techniques for natural language analysis are examined in some detail, ie, for translating natural language utterances into a unique internal representation. Virtually all natural language analysis systems can be classified into one of the following categories:

- Pattern matching [eg, ELIZA (qv) (Weizenbaum, 1986); PARRY (qv) (Parkinson and co-workers, 1977)].
- Syntactically driven parsing (qv) [eg, ATNs (Woods, 1970)].
- Semantic grammars (see GRAMMAR, SEMANTIC) [eg, LIFER (qv) (Hendrix, 1977); SOPHIE (qv) (Brown and Burton, 1975)].
- Case frame instantiation [eg, ELI (qv) (see GRAMMAR, SEMANTIC) (Riesbeck and Schank, 1976)].
- Wait and see (Marcus, 1980).
- Word expert (Small and Kleger, 1982).
- Connectionist (see CONNECTIONISM) (Small and co-workers, 1982).
- Skimming [eg, FRUMP (qv) (Dejong, 1979) and IPP (Schank and co-workers, 1980)].

The examples provided with each category are the names of language analysis systems following that approach or the names of builders of such systems. Of these categories, the first four represent the bulk of the language analysis systems already constructed and are the only ones covered in detail. The reader is encouraged to follow up the references provided for further details of the other methods.

Pattern Matching

The essence of the pattern matching approach to natural language analysis is to interpret input utterances as a whole rather than building up their interpretation by combining the structure and meaning of words or other

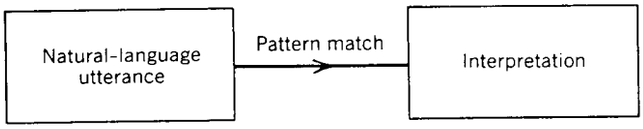

Figure 2. Parsing by pattern matching.

lower level constituents. The approach is thus holistic rather than constructive. With this approach, the interpretations are obtained by matching patterns of words against the input utterance. Associated with each pattern is an interpretation, so that the derived interpretation is the one attached to the pattern that matched. In the simplest case this arrangement is simply a list of correspondences between equivalence classes of utterances (the ones that match a given pattern) and interpretations (the ones associated with each pattern). In more sophisticated variations of the approach, patterns may involve higher level constituents or semantic elements so that some aspects of the interpretation may become constructive, but the basic flavor of the approach still remains to go as directly as possible from the input utterance to the interpretation, as suggested by Figure 2.

One of the simplest and earliest examples of pattern matching parsing was the ELIZA system (Weizenbaum, 1966). This system demonstrated how a very simple mechanism in the form of a computer program was able to mimic apparently intelligent behavior. The carefully selected task of ELIZA was to simulate a Rogerian psychologist interviewing a patient. The style of this school of psychology allowed ELIZA to understand virtually nothing of what was said and still, in many cases, come up with convincing replies. ELIZA did not construct an internal representation of its input as such but instead went directly from the input to its reply, which could therefore be thought of as the interpretation of the input. The input was matched by a small set of single-level patterns, each of which was associated with several replies. The first pattern found to match an input lead to a selected reply from the associated set; if a pattern matched again on a subsequent input, the next reply was given. Variables in the patterns that would match any sequence of words whatsoever allowed text from the input to be incorporated into the reply. Example patterns and corresponding replies with variables represented by *X* and *Y* include

X always *Y*	Can you think of a specific example? When? Really, always?
X and *Y* me	Why do you think I *Y* you? You like to think I *Y* you, don't you? Suppose that I did *Y* you? What would that mean?

These would allow ELIZA (E) to conduct the following dialogue with a patient (P):

P: People always stare at me.	Matches first pattern
E: Can you think of a specific example?	
P: For instance you keep staring at me.	Matches second pattern with *Y* matching *keep staring at*
E: Why do you think I keep staring at you?	First reply for second pattern with appropriate substitution for *Y*
P: Because you hate me.	Second pattern again with *Y* matching *hate*
E: You like to think I hate you, don't you?	Next (second) reply for second pattern

The simplicity of the matching and reply generation meant that most conversations with ELIZA did not go nearly as smoothly as this, but there are several anecdotes about people being fooled into thinking they were talking to a real person.

ELIZA could achieve its results with such a low level of analysis only by ignoring most of what was said. To make more complete analyses of the input using the same techniques would require far too many patterns: in the extreme, one pattern for every possible utterance. Moreover, many of these patterns would contain common subelements because they mentioned the same objects or had the same concepts arranged with slightly different syntax. In order to resolve these problems within the pattern matching approach, hierarchical pattern matching methods have been developed in which some patterns match only part of the input and replace that part by some canonical result. Other higher level patterns can then match on these canonical elements in a similar way, until a top-level pattern is able to match the canonicalized input as a whole according to the standard pattern matching paradigm. In this way similar parts of different utterances can be matched by the same patterns, and the total number of patterns is much reduced and made more manageable.

The best known example of hierarchical pattern matching is the PARRY system (Parkinson and co-workers, 1977; Colby, 1973). Like ELIZA, this program operates in a psychological domain but models a paranoid patient rather than a psychologist. Using the traditional pattern matching paradigm, PARRY interprets its input utterances as a whole by matching them against a set of about 2000 general patterns. The internal representation into which the input is transformed is a set of updates to a simple model of the paranoid patient's mental state plus a representation of any factual content of the input. Replies are generated from the updated paranoid model plus the factual content. However, before the general patterns are applied, PARRY massages its input through a series of eight canonicalizing steps, most of which are based on localized pattern matching. Examples of these steps include

Canonicalizing rigid idioms (eg, "have it in for" → "hate").

Noun phrase bracketing using an ATN (see below).

Canonicalizing flexible idioms (eg, "lend a hand" → "help").

Clause splitting (eg, "I think you need help" → "(I think) (you need help)").

Using rules of this form, an input such as

Do you have it in for me? I want to lend you a hand.

can be canonicalized into a form similar to

(YOU HATE ME) + INTERROGATIVE =?
(I WANT) (I HELP YOU)

An appropriate reply is generated by matching against PARRY's 2000 general patterns.

As well as matching patterns of words, it is also possible to analyze natural language input by matching patterns of semantic elements with potentially very powerful results as shown by the pilot machine translation (qv) system (Wilks, 1975). The goal of this system was to translate English input into French output. To do this, it first analyzed its English input into an internal semantic pattern from which it could generate the French. This analysis was performed by matching the input against a very general set of patterns such as

(MAN FORCE MAN)

which matches all events in which a person compels another person to do something. Other general patterns involved people doing things to objects, objects being in certain states, etc. To allow matches against these patterns, Wilks represented word senses as formulas of the same semantic primitives as appeared in the patterns, so for instance, *interrogate* was

((MAN SUB J) ((MAN OBJE) (TELL FORCE)))

ie, a person forcing another person to tell something, and *crook* was one of the following possibilities:

((((NOTGOOD ACT) OBJE) DO) SUBJ MAN))
((((((THIS BEAST) OBJE) FORCE) SUBJ MAN)) POSS)
LINE THING))

ie, a person who does bad things or a long thin thing that a person uses to force animals (normally sheep) to do something. As well as providing an interpretation of the input, the process of matching these formulas against the general patterns also allowed word senses to be disambiguated. So

The policeman interrogated the crook.

is analyzed by matching it against the (MAN FORCE MAN) pattern, and this also chooses the bad person sense of crook because it matches the second MAN of this pattern. There is also a (MAN FORCE THING) pattern, but this does not match as well because the formula for *interrogate* specifies MAN for its object. Note that the notion of degree of match is present in this system. As shown below, this idea makes parsing by pattern matching considerably more powerful, especially when the input contains grammatical errors.

To summarize this section on parsing by pattern matching, the basic paradigm is to recognize input utterances as a whole by matching them against patterns of words, wildcards, and semantic primitives. The result of the match is the interpretation of the utterance. Unless a very shallow level of analysis is acceptable, the number of patterns required is too large, even for restricted domains. This problem can be ameliorated by hierarchical pattern matching in which the input is gradually canonicalized through pattern matching against subphrases. The number of patterns can also be reduced by matching with semantic primitives instead of words.

Syntactically Driven Parsing

Syntax deals with the ways that words can fit together to form higher level units such as phrases, clauses, and sentences. Syntactically driven parsing (qv) is, therefore, naturally constructive, ie, the interpretations of larger groups of words are built up out of the interpretations of their syntactic constituent words or phrases. In this sense, it is just the opposite of pattern matching, in which the emphasis is on interpretation of the input as a whole. The most natural way for syntactically driven parsing to operate is to construct a complete syntactic analysis of the input utterance first and only then to construct the internal representation or interpretation. This leads to considerable inefficiency, and more recent syntactically driven approaches have tried to intermix parsing and interpretation.

Parse Trees and Context-Free Grammars. The most common form of syntactic analysis is known as a parse tree. Figure 3 shows a parse tree for the sentence

The rabbit nibbled the carrot.

The tree shows that the sentence is composed of a noun phrase (subject) and a verb phrase (predicate). The noun

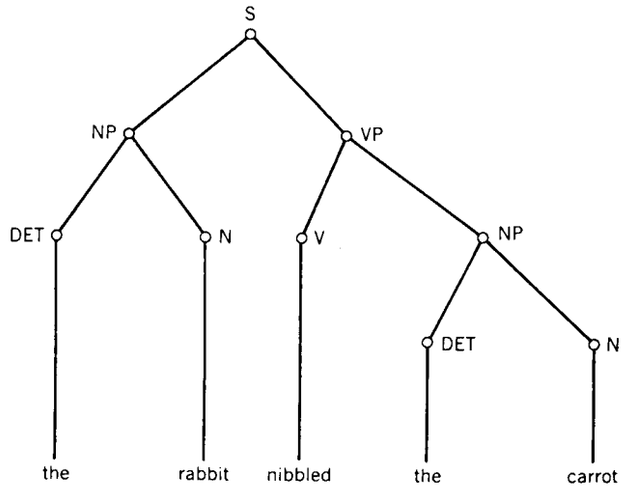

Figure 3. A parse tree for *the rabbit nibbled the carrot.*

phrase consists of a determiner (*the*) followed by a noun (*rabbit*), whereas the verb phrase consists of a verb (*nibbled*) followed by another noun phrase (the direct object), whose determiner is *the* and whose noun is *carrot*.

Syntactic analyses are obtained by application of a grammar that determines what sentences are legal in the language being parsed. The method of applying the grammar to the input is called a parsing (qv) mechanism or parsing algorithm. A very simple style of grammar is called a context-free grammar, which means that the symbol on the left side of a rewrite rule may be replaced by the symbol on the right side regardless of the context in which the left side symbol appears. The context-free grammar consists of rewrite rules of the following form:

$$S \rightarrow NP\ VP$$

$$NP \rightarrow DET\ N \mid DET\ ADJ\ N$$

$$VP \rightarrow V\ NP$$

$$DET \rightarrow the$$

$$ADJ \rightarrow big \mid green$$

$$N \rightarrow rabbit \mid rabbits \mid carrot$$

$$V \rightarrow nibbled \mid nibbled \mid nibble$$

As this example shows, context-free grammars have the advantage of being simple to define. They have been widely used for computer languages, and highly efficient parsing mechanisms (Earley, 1970; Tomita, 1986) have been developed to apply them to their input. However, they also suffer from some severe disadvantages. It should be clear that the above context-free grammar accounts for the parse shown in Figure 3; rewrite rules correspond directly to bifurcations in that tree. Although it accounts for that and several other good sentences, the grammar also allows several bad ones, such as

The rabbits nibbles the carrot.

The problem here is that the context-free nature of the grammar does not allow agreements such as the one required in English between subject and object. To enforce such an agreement, there would have to be two completely parallel grammars, one for singular sentences and the other for plural. Moreover, a grammar that also allowed passive sentences such as

The carrot was nibbled by the rabbit.

would have to have another completely different set of rules, even though the passive and the active forms of the same sentence have a clear syntactic relation, not to mention semantic equivalence. These duplications are multiplicative rather than additive, leading to exponential growth in the number of the grammar rules. Thus, in terms of the number of rules involved and in terms of being unable to capture related phenomena by related rules, context-free grammars turn out to be quite unsuitable for natural-language analysis. Recent work by Gazdar (1983) and others has shown that these problems of exponential rule growth can be masked using notational shorthand devices such as "metarules" plus relatively minor extensions to the context-free formalism and in particular without going to the transformational framework discussed below. However, the computational tractability of generalized phrase structure grammar (qv), as the extended formalism is called, has yet to be determined.

There is one more point to be made with this example, one not specific to context-free grammars, but a serious problem for all syntactically driven parsing. The above grammar also allows

The rabbit was nibbled by the carrot.

This is an example of a sentence that is perfectly good syntactically but makes no sense at all. For utterances that are ambiguous syntactically (and for more comprehensive grammars, syntactic ambiguity is very common), such acceptance of nonsensical interpretations can lead to the highly inefficient generation of multiple parses, only one of which has a reasonable translation into the final internal semantic representation.

Transformational Grammar. The problems mentioned above as specific to context-free grammars were tackled by linguists, in particular Chomsky (1957, 1965), through transformational grammar. As shown in Figure 4, their answer was to add another type of rule to a context-free grammar. The basic idea was to use the context-free grammar to generate a parse tree just as before but add onto it certain tags, such as one for a plural sentence. The set of transformations on the parse tree would then rearrange things so that the pluralness was transmitted to all parts of the tree concerned and the required agreements could be enforced. The transformations that enforced agreements were called obligatory transformations. A second class of optional transformations was used to capture the relations between, for instance, active and passive sentences; the active and passive versions of the same sentence had the same representation in the base component produced by the context-free grammar, but the passive version was the result of applying an extra optional transformation. Transformations are context-sensitive rules that map a parse tree into a related parse tree.

Although transformational grammar did a much better job of accounting for the regularities of natural language than context-free grammar, from the point of view of computational effectiveness, it was much worse. (However, significantly, a complete transformational grammar of English has never been produced.) As the above description implied, it was set up as a generative model, ie, it told you how to produce a sentence starting from the symbol *S*. Running the model in reverse to do sentence analysis turned out to be a computational nightmare, largely be-

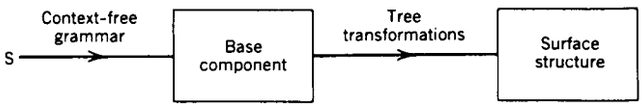

Figure 4. Transformational grammar.

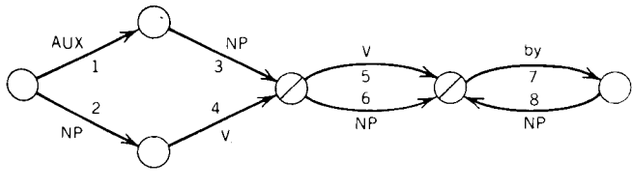

Figure 5. Example ATN.

cause transformations operate on trees, not strings of words, and so are highly nondeterministic when run backward. For instance, the "equi-NP deletion" transformation deletes without trace the second occurrence of a coreferential noun phrase in certain structures, and it is impossible to run a deletion backward if there is no clue as to what was deleted. Consequently, although some attempts have been made (Petrick, 1965), parsers based on transformational grammar have not played a major role in NLP.

Augmented Transition Networks. Largely in response to the problems of transformational grammar, Bobrow and Fraser (1969) proposed, and Woods (1970) subsequently developed, a method of expressing a syntactic grammar that was computationally tractable and yet still could capture linguistic generalizations in a concise way, in many cases more concisely than transformational grammar itself. The formalism Woods developed was known as an augmented transition network (ATN) (see GRAMMAR, AUGMENTED TRANSITION NETWORK). It consisted of a recursive transition network (formally equivalent in expressive power to a context-free grammar) augmented by a set of tests to be satisfied before an arc was traversed and a set of registers that could be used to save intermediate results or global state. An example ATN is shown in Figure 5. The network recognizes simple sentences with just a subject, verb, and direct object in all combinations of active, passive, declarative, and interrogative. The symbols attached to the arcs show what constituent must be recognized to traverse the arc; AUX is an auxiliary verb (like *is* or *have*); NP is a noun phrase, which is defined by another network in the same formalism as this one; V is a verb; and "by" is the word *by*. The numbers on the arcs serve as indices to Table 1, which list the tests that must be true to traverse the arcs and the action that must be performed as the arc is traversed.

In this LISP-like notation, the asterisk refers to the constituent just parsed, and SETR sets a register, whose name is specified by its first argument, to the value of its second argument. A concrete example of the network in operation will make this clearer. Suppose one wanted to parse

The rabbit nibbled the carrot.

One would start at the left-most node in the graph and at the left of the sentence. Two arcs lead from that node, but only arc 2 is applicable because in the input one is not looking at the auxiliary verb required by arc 1 but at a noun phrase, *the rabbit,* as required by arc 2. One can see

Table 1. Tests and Actions for Traversing the Arc

Test	Predicate
1 T	(SETR V*) (SETR TYPE'QUESTION)
2 T	(SETR SUBJ*) (SETR TYPE'DECLARATIVE)
3 (agrees* V)	(SETR SUBJ*)
4 (agrees SUBJ*)	(SETR V*)
5 (AND (GETF PPRT) (= V'BE))	(SETR OBJ SUBJ) (SETR V*) (SETR AGFLAG T) (SETR SUBJ'SOMEONE)
6 (TRANS V)	(SETR OBJ*)
7 AGFLAG	(SETR AGFLAG FALSE)
8 T	(SETR SUBJ*)

from Table 1 that arc 2 has no additional test (indicated by T), so we traverse that link setting the SUBJ register to the thing just parsed, ie, *the rabbit,* and the TYPE register to DECLARATIVE. One is now at a node with only one arc, arc 3, and that arc requires a verb. Fortunately, one is now looking at *nibbled* in the input, so one can try to traverse it. Arc 3 has an additional test requiring that*, ie, the present word in the input (the verb), agree with the contents of the subject register; this is the way agreements are enforced in an ATN. In this case the agreement is correct, and one can traverse the arc, setting the V register to the verb. The node one gets to now has a line through it, indicating that this can be the end of the parse provided that there is no input left to consume, so *The rabbit nibbled* would be accepted here. In this example there is another noun phrase, *the carrot,* and so one follows arc 6, whose test requires that the verb in the V register be transitive, which *nibbled* is. So one ends up at another terminal node with no further input, and so the parse is completed successfully. The result of the parse is the setting of the four registers: SUBJ, TYPE, V, and OBJ, and these can be combined into a tree or whatever representation is desired.

A more interesting use of registers can be seen from the example

The carrot was nibbled by the rabbit.

To parse the first three words, we traverse arcs 2 and 3 much as before, with the difference that now *the carrot* is in SUBJ and *was* is in V. One cannot take arc 6 because one is only up to *nibbled* in the input, but one can take arc 5 because *nibbled* is a verb. The test on arc 5 also requires *nibbled* to be a past participle, which it is, and the contents of V to *be,* and because *was* is a form of the verb *to be,* the test is satisfied. The action on arc 5 is interesting; it puts the contents of the SUBJ register in the OBJ register, overwrites the verb register with the past participle verb, sets a flag to true, and puts a placeholder "someone" in the SUBJ register. This corresponds to recognizing that the sentence is in passive form, and in our case makes *the carrot* the object and *nibbled* the verb. One has reached

"by" in the input and so can follow arc 7, which just requires the passive flag to be set; its only action is to turn this flag off, so that the arc cannot be traversed again. Finally, one gets back to their terminal node via arc 8, which puts *the rabbit* in the SUBJ register. Note that the result of this parse is the same as the result of the first example. Now try to follow the parses of

Did the rabbit nibble the carrot?
Was the carrot nibbled by the rabbit?

These brief examples should give some idea of the power of an ATN and of how its tests and registers can be used to capture the regularities of language in a concise and elegant way. Very large ATN grammars of several hundred nodes (Woods and co-workers, 1972) have been developed that capture large subjects of English. However, ATNs also have several disadvantages:

Complexity and Nonmodularity. As the coverage of an ATN increases, so does its structural complexity. It becomes extremely difficult to modify or augment an existing ATN without causing large numbers of unforeseen side effects. For instance, if another outgoing arc is added to a node with a large number of incoming arcs in order to handle an additional type of phrase that is a valid continuation of the parse represented by one of the incoming arcs, it could to lead to spurious and incorrect parses when the node is reached via a different incoming arc. (Fan-out and fan-in factors of 10 or 20 are not uncommon in large realistic grammars.)

Fragility. The current position in the network is a very important piece of state information for the operation of an ATN. If an input should be slightly ungrammatical, even by a single word, it is very hard to find the appropriate state to jump to that would enable the parse to continue, though see the work by Kwasny and Sondheimer (1981) and Weischedel and Black (1980) on dealing with such extragrammaticality and the work on island-driven ATN parsing for speech input by Woods and co-workers (1976).

Inefficiency through Backtracking Search. Although the above examples are not complex enough to show it, the task of traversing an ATN is in general nondeterministic and requires search. The natural way to search an ATN is through backtracking (qv). Because intermediate failures are not remembered in such a search, major inefficiencies can result through repetition of the same subparses arrived at through different paths through the network. Chart parsing techniques (Kaplan, 1973; Kay, 1973; Frederking, 1981) were designed as alternatives to ATNs precisely to avoid these inefficiencies.

Inefficiency through Meaningless Parses. Normally the grammar of an ATN is purely syntactic, and a complete syntactic parse is produced before any semantic interpretation is performed. In that situation many spurious meaningless parses can be produced, especially if the grammar is large and comprehensive. To combat this, recent parsers (Bobrow, 1978) in the ATN tradition have tried to interpret each constituent as it was produced, thus preventing complete parses based on constituents that could be predicted to be nonsensical.

More information on the relative advantages and disadvantages of ATNs is available (Hayes and Mouradian, 1981; Hayes and Reddy, 1983).

Semantic Grammars

Language analysis based on semantic grammars (qv) is similar to syntactically driven parsing except that in semantic grammars the categories used are defined semantically as well as syntactically. Thus instead of the category "noun phrase" in a syntactic grammar, a semantic grammar might have the category "description of a ship," which is syntactically always a noun phrase but has additional strong semantic constraints. Semantic grammars were introduced by Burton (1976) for use in SOPHIE (Brown and Burton, 1975), a computer-aided instruction system for electronic circuit debugging, to deal with the problems of inefficiency due to the generation of syntactically correct, but meaningless, parses mentioned above for ATN-based syntactic grammars. The goal was to eliminate the production of meaningless parses by setting up the grammar so that only meaningful parses could be produced. To do this, it was necessary to categorize all the objects and actions that the SOPHIE system needed to parse to conduct a conversation in its domain of electronic circuitry and then to construct the grammar so that, for instance, only a description of a switch could be the object of a "close" action. This technique, while retaining the fragility of an ATN, worked well to reduce parsing inefficiency. Because the relevant semantic categories were available at parse time, it also allowed semantic interpretation to proceed as the parse unfolded. However, the technique only works properly in restricted domains, like the one mentioned above, in which all objects and their relations can be categorized in advance, allowing a grammar to be built around the possible semantic relations. Semantic grammars are thus a technique useful only for applied natural language processing, not for general NLP.

For an example of how semantic grammars can be used, consider the following grammar definition in the formalism used by LIFER, a system for building semantic grammars developed by Hendrix (1977).

⟨present⟩ → what is | [can you] tell me

⟨attribute⟩ → length | beam | class

⟨ship⟩ → the ⟨shipname⟩ | ⟨classname⟩ class ship

⟨shipname⟩ → kennedy | enterprise

⟨classname⟩ → kitty hawk | lafayette

An expanded version of this grammar was used for access to a database of information about U.S. Navy ships in the

LADDER (Sacerdoti, 1977) system. Even the above miniversion is capable of recognizing such inputs as

What is the length of the *Kennedy*?
Can you tell me the class of the *Enterprise*?
What is the length of *Kitty Hawk* class ships?

Because the definitions used by LIFER are similar to those used for context-free grammars, the reader should have no difficulty in seeing how these inputs could be recognized by the above grammar. In addition to defining a grammar, LIFER also allowed an interface builder to specify the interpretations to be produced from rules that were used in the recognition of an input. In the above case this resulted in database query language statements corresponding to the inputs being produced as a direct result of the recognition. The database query language statements in effect took the place of a parse tree, and so no separate semantic interpretation stage was required.

Note in the example above that not all the categories are specializations of pure semantic categories; ⟨present⟩, for instance, will parse several phrases, none of which fits into any standard grammatical category. The phrases may differ from each other in their syntactic structure, including the number of verbs they contain. The ability to construct cross-grammatical categories like this allows a semantic grammar to incorporate some features of pattern matching. Also note how strongly directed the recognition is. The word *class* for instance occurs in two quite different ways in the grammar: once as a ship attribute and the other as part of the second type of ship description. Thus in the (rather silly) question

What is the class of *Lafayette* class ships?

the appropriate category for *class* would be used each time it appeared without considering its other role in the grammar. This directedness of recognition is also useful in building spelling correction into the recognition process. In an input like

What is the *legnth* of the *Kennedy*?

the spelling of *legnth* need only be checked against the list of ship attributes rather than the entire system vocabulary because a ship attribute is the only category that can appear at the place where the misspelling occurs. A final advantage of the strong top-down direction available through semantic grammars can be seen in LIFER's ellipsis mechanism, which was intended to deal with input sequences such as

What is the length of the *Kennedy*?
The beam?

Here the fact that *beam* and *length* are in the same semantic grammar category allows the second input to be interpreted as "What is the beam of the *Kennedy*?" rather than say "What is the length of the beam?" See below for discussion on ellipsis mechanisms in general.

In addition to their numerous advantages for limited domain applications, semantic grammars have several disadvantages, chief of which is the requirement that a new grammar be developed for each new domain, since the semantic categories for each domain will be quite different. However, if the applications are similar (eg, both include database access), there will be many parts of the grammar (eg, the basic framework for questions) that are the same. A related disadvantage is that semantic grammars tend to get large very quickly, partly because of the repetition of similar constructions in different semantic categories. This makes nontoy-semantic grammars quite hard to construct and can result in very "spotty" kind of coverage of syntactic variation. For instance, adding a rule that allows the possessive to be apostrophized in the description of a ship attribute (ie, you can say "the *Kennedy*'s length" as well as "the length of the *Kennedy*") does not also allow possessives to be apostrophized in the description of an attribute of a sailor (ie, you might not be able to say "officer's rank" even though you can say "rank of an officer") because the two categories are in different parts of the grammar, and their recognition is unrelated. A second rule would be required.

Three approaches have been tried to resolve these problems. One is to go back to recognition by a syntactic grammar before semantic interpretation, but to try to intermix the semantic and syntactic components much more closely, so that every syntactic constituent is interpreted as soon as it is constructed. The RUS system (Bobrow, 1978) is an example of this approach. It provides some improvement over a pure syntax first approach but is still not as efficient as pure semantic grammars; it is also difficult to incorporate semantic constraints, a process that requires writing different chunks of LISP code, called "I-rules," of each domain.

An alternative approach, as exemplified by the TEAM system (Grosz, 1983), is to focus in on a specific class of applications, access to relational databases, and to abstract out the linguistically common aspects of a semantic grammar for such a class. Building a specific interface, then, requires only instantiating a template, as it were, with the vocabulary and morphological variation required for a specific database. This approach has the potential to produce highly efficient natural language interfaces, but at the cost of some expressive power and inability to go beyond the class of applications without restarting from the ground up.

The third approach is to combine the strengths of several parsing strategies, such as semantic grammars, syntactic transformations, and pattern matching into a single system that maps structures into ore canonical forms before attempting to use the full semantic grammar, thus allowing many redundant and unnecessary constructions to be eliminated (Carbonell and Hayes, 1985; Carbonell, 1981a). This multistrategy approach has been implemented in the DYPAR system (Carbonell, 1985) and applied to database query, expert system command, and operating system command interfaces. Although richer in expressive power, this approach demands more sophistication of the grammar writer, requiring knowledge of how to write transformations, context-free rules, and patterns.

Case Frame Instantiation

A major development in computational linguistics (qv) was the inclusion of case frame instantiation (see GRAMMAR, CASE) in the repertoire of effective parsing techniques. Case frames were popularized by the linguist Fillmore (1968) in his seminal paper, and their computational import was quickly grasped by several researchers in natural language processing, including Simmons (1973), Schank (1975), and Riesbeck (1975). Case frame instantiation is one of the major parsing techniques under active research today. Its recursive nature, and its ability to combine bottom-up recognition of key constituents with top-down instantiation of less structured constituents, gives this method very useful computational properties (see also FRAMES).

What Are Case Frames? A case frame consists of a head concept and a set of roles, or subsidiary concepts, associated in a well-defined manner with the head concept. Initially, only sentential-level case frames were investigated, where the head consists of the main verb, and the cases include the agent that carries out the action, the object acted on, the location in which the action takes place, etc. For instance, consider the sentence

In Elm Street, John broke a window with a hammer for Billy.

In simplified generic notation, the case frame corresponding to this sentence is

[BREAK
  [case frame
    agent: JOHN
    object: WINDOW
    instrument: HAMMER
    recipient:
    directive:
    locative: ELM STREET
    benefactive: BILLY
    co-agent:]
  [modals
    time: past
    voice: active]]

In the notation above, cases, such as *agent,* are written in lowercase, and their fillers are in uppercase.

Case frames, as adopted in computational linguistics, differ markedly from simple, purely syntactic, parse trees. The relations between the head of the case frame and the individual cases are defined semantically, not syntactically. Hence, a noun in the subject position can fill the *agent* case, as in the example above, or it can fill an object case, as in *the window broke* (the window was not the agent that caused the breakage), or it can fill the *instrument* case, as in *the hammer broke the window.* These are different semantic roles played by the same syntactic constituent, "subject." Because the purpose of a nature language interface is to extract the semantics of the input it behooves the case frame representation to encode explicitly semantic differences in otherwise similar syntactic parse trees. Thus parsing into case frames requires semantic knowledge, as well as syntactic information, as shown below.

Consider some other properties of case frames. In the example above, only some of the cases were instantiated. What of the other cases, such as recipient and co-agent? There are examples that illustrate these shortly. First, consider the meaning of each case, as outlined below:

[⟨HEAD VERB⟩
  [case frame
    agent: ⟨the active causal agent instigating the action⟩
    object: ⟨the object on which the action is done⟩
    instrument: ⟨an instrument used to assist in the action⟩
    recipient: ⟨the receiver of an action, often the indirect object⟩
    directive: ⟨the target of an (usually physical) action⟩
    locative: ⟨the location where the action takes place⟩
    benefactive: ⟨the entity on whose behalf the action is taken⟩
    co-agent: ⟨a secondary or assistant active agent⟩
  ]]

If instead of saying *John broke the window with a hammer,* one were to say *John broke the window with Mary,* Mary would fill the co-agent case. Presumably John did not swing Mary over his head and use her as a battering ram to shatter the window, much as he would use an instrument like a hammer or a tree branch. Because Mary is taking part in causing the action to happen, regardless of whether her action is independent of, or in support of, John's action, she fills the co-agent case.

In order to illustrate the *directive* case, consider *John kicked the ball toward the goal* and *John flew the airplane to New York.* In the former example *the goal* fills the directive case, and in the latter *New York* fills the same case, because both express the direction in which each respective action was performed. In some early formulations of case frames no distinction was made between locative and directive, but the need to encode stative vs dynamic information explicitly (plus the need to represent sentences such as *In Yankee Stadium, John threw the ball at the catcher* that instantiate both cases) led to the acceptance of two semantically distinct cases, one encoding global location, the other a local change in location.

The recipient case is filled by *Mary* in both of the following: *John gave Mary a ball* and *John gave a ball to Mary.* Note that in this instance there are syntactically distinct sentences that map onto a unique semantic case frame representation, to wit:

[GIVE
  [case frame
    agent: JOHN

recipient: MARY
object: BALL]
. . .]

Required, Optional, and Forbidden Cases. Each case frame defines some required cases, some optional cases, and some forbidden cases. A required case is one that must be present in order for the verb to make sense. For instance, *break* requires the object case. A sentence is not complete without it, but no other case is required. *The window broke* is a complete, if not very informative, sentence. An optional case is one that, if present, provides more information to the case frame representation but, if absent, does not harm its semantic integrity. For instance agent, co-agent, and locative are optional cases of *break*. Forbidden cases are those that cannot be present with the head verb. The directive and recipient cases are forbidden for the *break* case frame.

Conceptual Dependency. It is often useful in natural language processing to employ a semantic representation that represents information in as canonical a manner as possible. In the ideal canonical representation, different ways of stating the same information would be represented identically, and propositions that encode similar information would map into semantic encodings that highlighted the similarities while retaining the differences in an explicit manner. The best known attempt at a canonical semantic representation is the conceptual dependency (CD) (qv) formalism (Schank, 1975; Shank and Abelson, 1977; Schank and Riesbeck, 1980) as a reductionistic case frame representation for common action verbs. Essentially, it attempts to represent every action as a composition of one or more primitive actions, plus intermediate states and causal relations.

To use Schank's example, suppose one wants to represent, in a case frame notation, *John gave Mary a ball* and *Mary took a ball from John*. These sentences differ syntactically, they differ in terms of verb selection, and they differ in how their cases are instantiated (eg, *John* is the agent of the first sentence and *Mary* of the second sentence). However, both sentences express the proposition that a ball was transferred from John to Mary, and in both cases one can infer that John had the ball before the action took place, that Mary has it after the action, and that John no longer has it after the action. The only significant difference is that in the first sentence, John performed the action, and in the latter Mary did so.

In CD there is a primitive action called ATRANS (for Abstract TRANSfer of possession, control, or ownership) that encodes the basic semantics of both of these verbs and many more. The CD representation of these sentences is

[ATRANS	[ATRANS
rel: POSSESSION	*rel:* POSSESSION
actor: JOHN	*actor:* MARY
object: BALL	*object:* BALL
source: JOHN	*source:* JOHN
recipient: MARY]	*recipient:* MARY]
"John gave Mary a ball"	"Mary took a ball from John"

(Some readers may be acquainted with Schank's complex notation of double and triple arrows. The direct simplified notation (shown above) is virtually isomorphic, somewhat clearer, and closer to the data structures used by most of the computer programs that parse into CD and other case frame representations.)

These two structures are very simple to match against each other to determine precisely in what aspects the two propositions differ and in what aspects they are identical. Moreover, inference rules associated with ATRANS can be invoked automatically when *give* and *take* are parsed into these structures. There are many more verbs that contain the ATRANS primitive (such as *bequeath, donate, steal, sell, buy, appropriate, expropriate,* etc). Sometimes ATRANS is used in conjunction with other CD primitives that capture other aspects of the meaning. The verb *sell,* for instance, involves two ATRANS primitives in mutual causation:

[ATRANS		[ATRANS
rel: OWNERSHIP	CAUSE	*rel:* OWNERSHIP
actor: JOHN	⟶	*actor:* MARY
object: APPLE	⟵	*object:* 24 CENTS
source: JOHN	CAUSE	*source:* MARY
recipient: MARY]		*recipient:* JOHN]

"John sold an apple to Mary for 25 cents"

The cases used in CD are similar but not identical to the set used originally in case grammars, although the basic ideas are the same. One refinement in CD was to separate *agent* into *actor* and *source,* as the two can be instantiated by different entities in the underlying semantic primitives. Other CD primitive actions include

PTRANS	Physical transfer of location.
MTRANS	Mental transfer of information.
MBUILD	Create a new idea or conclusion from other information.
INGEST	Bring any substance into the body.
PROPEL	Apply a force to an object.
ATTEND	Focus a sense organ (eg, eyes, ears).
SPEAK	Produce sounds of any sort.

Later work (Schank and Carbonell, 1979) has extended this list to include social and other interpersonal actions.

Parsing into Case Frames. The discussion of case frames thus far has focused on their structural properties, including parsimony and clarity of representation. Now the uses of case frames in parsing natural language are discussed, in particular certain parsing techniques available to parsers whose target representation is based on case frames. In essence, parsers built around case grammars help to combine bottom-up recognition of structuring constituents with more focused top-down instantiation of less structured, more complex constituents. This essential property is demonstrated in the example case frame recognition algorithm presented below (see also PARSING).

Thus far case frames have been mentioned that consist of a header and a collection of semantically defined cases. There is a bit more to it than that. Each case consists of a

case of *too* in this example. *Too* is a real word that might well be in the system's vocabulary, and without the strong prediction that it should be a preposition marking a case of *transfer,* the system would be unable to correct it (a match against the whole vocabulary would make *too* the best match), or even notice that it is mispelled.

Whereas spelling correction can be dealt with at the lexical level, other forms of grammatical deviation require modification to a NLP system's grammatical expectations. The way in which this can be accomplished differs markedly by approach. In pattern matching, for instance, the obvious approach is partial pattern matching as attempted in the FlexP system (Hayes and Mouradian, 1981). Patterns are deemed to match partially if most, but not all, their elements actually do match the input. Clearly, this can be useful for missing or extra words, but is not useful in the case of unusual word order. Moreover, in practice, it turns out that some elements of a pattern are more important than others, and unless allowance is made for these differences, it is difficult to decide exactly how much of a pattern needs to match before the pattern as a whole can be declared to have matched.

Dealing with grammatical deviation in an ATN-based system turns out to be extremely difficult. The current position in the network is a very important piece of state information for the operation of an ATN. If an input should be slightly ungrammatical, even by a single word, it is very hard to find the appropriate state to jump to that would enable the parse to continue. This assumes, moreover, that it is possible to determine exactly where the input has departed from the grammar's expectations. The backtracking search used with most ATNs can make this difficult. Work by Weischedel and Black (1980) has dealt with extragrammaticality caused by incorrect agreements that can be resolved by relaxing the predicates on ATN arcs, and Kwasny and Sondheimer (1981) have looked into adding extra arcs to ATNs on a dynamic basis to make the grammar fit the input. Earlier work on speech parsing (Woods and co-workers, 1976) also tried to use ATNs in an island-driven mode.

A more recent development in robust parsing (Hayes and Carbonell, 1981a; Carbonell and Hayes, 1981) uses a construction-specific approach that fits in well with semantic grammars and case frame instantiation. The basic idea is to tailor parsing strategies to specific construction types; this not only results in efficient parsing of grammatical input but also permits built-in recovery strategies that exploit the characteristics of the particular construction type. For instance, the following simple recovery strategy works quite well for simple imperative case frames:

> Skip over unexpected input until a case marker is found; parse skipped segments against unfilled cases, using only semantic constraints.

If this strategy is applied to

> transfer Economics 247 to Physics 317 Smith

Economics 247 and *Smith* will initially be skipped over, with *to Physics 317* being correctly parsed because *to* is a valid case marker. Then the skipped segments will be correctly parsed against the unfilled cases *source-course* and *student* respectively, leading to a parse identical to that for

> transfer Smith from Economics 247 to Physics 317

Such methods of robust parsing are under active investigation at the moment with the chief outstanding problem being the coordination of multiple, independent, construction-specific parsing strategies on the same input.

DIALOGUE PHENOMENA

In addition to recognizing individual sentences, the problem of interactive communication through natural language, be it communication between man and machine or communication between two people, entails discourse phenomena that transcend individual sentences (see DISCOURSE UNDERSTANDING).

Anaphora. Pronouns and other anaphoric references (words like *it, that,* or *one*) refer to concepts described previously in a dialogue. Anaphoric resolution entails identifying the referents of these place holder words. Interactive dialogues invite the use of anaphora much more than simpler database query situations. Therefore, as natural language interfaces increase in complexity and expand their domain of application, anaphoric resolution becomes an increasingly important problem.

Definite Noun Phrases. Noun phrases often serve as another type of anaphoric reference by referring to previously mentioned concepts, much like the less specific anaphors do. Usually such phrases are flagged by a definite pronoun (eg, *the*). As Grosz (1977) noted, resolving the referent of definite noun phrases or any other anaphors often requires an understanding of the planning structure underlying cooperative discourse.

Ellipsis. People often use sentence fragments to express a complete proposition. These terse utterances must be filled out in the context of the dialogue. Sentential level ellipsis (qv) has long been recognized as ubiquitous in discourse. However, semantic ellipsis, where ellipsis occurs through semantically incomplete propositions rather than through syntactically incomplete structures, is also an important phenomenon. The ellipsis resolution method presented below addresses both kinds of ellipsis.

Extragrammatical Utterances. Interjections, dropped articles, false starts, misspellings, and other forms of grammatical deviance abound. Developing robust parsing techniques that tolerate errors has been the focus of much recent work (Weischedel and Black, 1980; Hayes and Carbonell, 1981a, 1981b; Carbonell and Hayes, 1981; Kwasny and Sondheimer, 1979), as discussed in the preceding section.

Metalinguistic Utterances. Intrasentential metalanguage has been investigated to some degree (Ross, 1970) but its more common intersentential counterpart has received little attention (Carbonell, 1982). However, utterances about other utterances (eg, corrections of previous commands, such as "I meant to type X instead" or "I should have said . . .") are not infrequent, and an initial stab is being made at this problem (Hayes and Carbonell, 1983). Note that it is a cognitively less demanding task for a user to correct a previous utterance than to repeat an explicit sequence of commands (or worse yet, to detect and undo explicitly each and every unwanted consequence of a mistaken command).

Indirect Speech Acts. Occasionally users of natural language interfaces will resort to indirect speech acts (qv) (Allen and Perrault, 1980; Perrault and coworkers, 1978; Searle, 1975), especially in connection with intersentential metalanguage or by stating a desired state of affairs and expecting the system to supply the sequence of actions necessary to achieve that state.

Empirical studies suggest that users of natural language interfaces avail themselves of discourse phenomena whenever such devices help in formulating short, succinct linguistic expressions over lengthier, more explicit ones. This observation is summarized as follows:

> Terseness Principle: Users of natural-language interfaces insist on being as terse as possible, independent of task, communication media, typing ability, or instructions to the contrary, without sacrificing the flexibility of expression inherent in natural-language communication [This principle may be viewed as a surprisingly strong form of Grice's (1975) maxim of brevity].

Case Frame Ellipsis Resolution

In order to illustrate the ubiquity of ellipsis in interactive dialogues through a natural language interface, look at the XCALIBUR project, whose objective is to provide flexible natural language access (comprehension and generation) to the XSEL expert system (McDermott, 1982). XSEL, the Digital Equipment Corporation's automated salesman's assistant, advises on selection of appropriate VAX components and produces a sales order for automatic configuration by the R1 system (McDermott, 1980). Part of the XSEL task is to provide the user with information about DEC components, hence subsuming the database query task. However, unlike a pure database query system, an expert system interface must also interpret commands, understand assertions of new information, and carry out task-oriented dialogues (Grosz, 1977). XCALIBUR, in particular, deals with commands to modify an order, as well as information requests pertaining to its present task or its database of VAX component parts. In the following example dialogue, user inputs are preceded with a ">" prompt.

>What is the largest 11780 fixed disk under $40,000?
The rp07-aa is a 516 MB fixed pack disk that costs $38,000.
>The largest under $50,000?
The rp07-aa.
>Add two rp07-aa disks to my order.
Line item 1 added: (2 rp07-aa)
>Add a printer with graphics capability
fixed or changeable font?
>fixed font
lines per minute?
>make it at least 200, upper/lowercase.
OK. Line item 2 added: (1 lxy11-sy)
>Tell me about the lxy11
The lxy11 is a 240-l/m line printer with plotting capabilities.

Details of the XCALIBUR interface are available (Carbonell and co-workers, 1983, 1985; Carbonell, 1983). In this article, only illustrating the case frame ellipsis resolution method is discussed.

The XCALIBUR system handles ellipsis at the case frame level. Its coverage is a superset of the LIFER/LADDER system (Hendrix, 1977; Sacerdoti, 1977) and the PLANES (qv) ellipsis module (Waltz and Goodman, 1977). Although it handles most of the ellipsed utterances encountered, it is not meant to be a general linguistic solution to the ellipsis phenomenon. The following examples are illustrative of the kind of sentence fragments the current case frame method handles. For brevity, assume that each sentence fragment occurs immediately following the initial query below.

> INITIAL QUERY: "What is the price of the three largest single port fixed media disks?"

"Speed?"
"Two smallest?"
"How about the price of the two smallest?"
"also the smallest with dual ports"
"Speed with two ports?"
"Disk with two ports?"

In the representative examples above, punctuation is of no help, and pure syntax is of very limited utility. For instance, the last three phrases are syntactically similar (indeed, the last two are indistinguishable), but each requires that a different substitution be made on the parse of the preceding query.

Ellipsis is resolved differently in the presence or absence of strong discourse expectations. In the former case the discourse expectation rules are tested first, and if they fail to resolve the sentence fragment, the contextual substitution rules are tried. If there are not strong discourse expectations, the contextual substitution rules are invoked directly.

An exemplary discourse expectation rule follows:

> IF: The system generated a query for confirmation or disconfirmation of a proposed value of a filler of a case in a case frame in focus,

THEN: EXPECT one or more of the following:
1. A confirmation or disconfirmation pattern.
2. A different but semantically permissible filler of the case frame in question (optionally repeating the attribute or providing the case marker).
3. A comparative or evaluative pattern.
4. A query for possible fillers or constraints on possible fillers of the case in question. [If this expectation is confirmed, a subdialogue is entered, where previously focused entities remain in focus.]

The following dialogue fragment, presented without further commentary, illustrates how these expectations come into play in a focused dialogue:

>Add a line printer with graphics capabilities.	
Is 150 lines per minute acceptable?	
>No, 320 is better	Expectations 1, 2, & 3
(or) other options for the speed?	Expectation 4
(or) Too slow, try 300 or faster	Expectations 2 & 3

The utterance "try 300 or faster" is syntactically a complete sentence, but semantically it is just as fragmentary as the previous utterances. The strong discourse expectations, however, suggest that it be processed in the same manner as syntactically incomplete utterances since it satisfies the expectations of the interactive task. The terseness principle operates at all levels: syntactic, semantic, and pragmatic.

The contextual substitution rules exploit the case frame representation of queries and commands discussed in the previous section. The scope of these rules, however, is limited to the last user interaction of appropriate type in the dialogue focus, as illustrated below. The rules search the ellipsed fragment for case fillers (or case marker and filler pairs) to substitute for corresponding cases in the parse of the previous input. Substitution can occur at a top-level (sentential) case frame or in embedded (relative-clause or noun phrase) case frames.

>What is the size of the 3 largest single-port fixed-media disks?
>And the price and speed?

and

>What is the size of the 3 largest single-port fixed-media disks?
>disks with two ports?

Note that it is impossible to resolve this kind of ellipsis in a general manner if the previous query is stored verbatim or as a semantic-grammar parse tree. *Disks with two ports* would be best corresponded to some ⟨disk-descriptor⟩ nonterminal and hence, according to the LIFER algorithm (Hendrix, 1977; Sacerdoti, 1977), would replace the entire phrase *single-port fixed-media disks* that corresponded to ⟨disk-descriptor⟩ in the parse of the original query. However, an informal poll of potential users suggests that the preferred interpretation of the ellipsis retains the previous information in the original query. The ellipsis resolution process, therefore, requires a finer grain substitution method than simply inserting the highest level nonterminals in the ellipsed input in place of the matching nonterminals in the parse tree of the previous utterance.

Taking advantage of the fact that a case frame analysis of a sentence or object description captures the meaningful semantic relations among its constituents in a canonical manner, a partially instantiated nominal case frame can be merged with the previous case frame as follows:

Substitute any cases instantiated in the original query that the ellipsis specifically overrides. For instance *with two ports* overrides *single port* in our example, as both entail different values of the same case filler regardless of their different syntactic roles. (*Single port* in the original query is an adjectival construction, whereas *with two ports* is a postnominal modifier in the ellipsed fragment.)

Retain any cases in the original parse that are not explicitly contradicted by new information in the ellipsed fragment. For instance, *fixed media* is retained as part of the disk description, as are all the sentential-level cases in the original query, such as the quantity specifier and the projection attribute of the query (*size*).

Add cases of a case frame in the query that are not instantiated therein but are specified in the ellipsed fragment. For instance, the *fixed-head* descriptor is added as the media case of the disk nominal case frame in resolving the ellipsed fragment in the following example:

>Which disks are configurable on a VAX 11-780?
>Any configurable fixed-head disks?

In the event that a new case frame is mentioned in the ellipsed fragment, wholesale substitution occurs, much like in the semantic grammar approach. For instance, if after the last example one were to ask "How about tape drives?" the substitution would replace *fixed-head disks* with *tape drives* rather than replacing only *disks* and producing the phrase *fixed-head tape drives*, which is meaningless in the current domain. In these instances of wholesale context switch the semantic relations captured in a case frame representation and not in a semantic grammar parse tree prove immaterial.

The key to case frame ellipsis resolution is matching corresponding cases rather than surface strings, syntactic structures, or noncanonical representations. Although correctly instantiating a sentential or nominal case frame in the parsing process requires semantic knowledge, some of which can be rather domain specific, once the parse is attained, the resulting canonical representation, encoding appropriate semantic relations, can and should be exploited to provide the system with additional functionality such as the present ellipsis resolution method. More details and examples of the rules that perform case frame substitution are available (Carbonell and co-workers, 1985).

More Complex Phenomena

In addition to ellipsis and anaphora, there are more complex phenomena that must be addressed to understand and simulate human discourse. This type of deeper under-

standing has not yet been incorporated into practical natural language interfaces. However, as natural language interfaces increase in sophistication (as they surely will), these more complex phenomena require attention, so, as the final topic of this article, some examples of these more esoteric discourse phenomena are discussed.

Goal Determination Inference. The interpretation of an utterance may depend on the inferred conversational goals of the speaker. Consider the following set of examples, in which the same utterance spoken in somewhat different contexts elicits radically different responses. These responses depend on the interpretation of the initial utterance, in which the attribution of goals to the speaker plays a dominant role.

PASSER-BY: Do you know how to get to Elm Street?

PERSON ON THE STREET CORNER: Walk toward that tall building and Elm Street is the fifth or sixth on your left.

The passer-by's question was quite naturally interpreted as an indirect speech act, because the information sought (and given) was not whether the knowledge of getting to Elm Street was present but rather how actually to get there. Lest the mistaken impression be given that it is a simple matter to identify indirect speech acts computationally, consider the following variant to the example:

PASSER-BY: Do you know how to get to Elm Street?

PERSON READING A STREET MAP AND HOLDING AN ENVELOPE WITH AN ELM STREET ADDRESS ON IT: No, I haven't found it; could you help me?

In the second example, the listener infers that the goal of the passer-by is to render assistance, and therefore the initial utterance is interpreted as a direct query of the knowledge state of the listener in order to know whether assistance is required. Hence, the passer-by's question is not an indirect speech act in this example.

Nor is the task of the interpreter of such utterances only to extract a binary decision on the presence or absence of a speech act from goal expectations. The selection of which indirect speech act is meant often rests on contextual attribution of different goals to the speaker. Consider, for instance, the following contextual variant of our previous example:

PASSER-BY: Do you know how to get to Elm Street?

WAITING CABBIE: Sure, hop in. How far up Elm Street are you going?

In this example, the cabbie interpreted the goal of the passer-by as wanting a ride to an Elm Street location. Making sure the cabbie knows the destination is merely instrumental to the inferred goal. The social relation between a cabbie and a (potential) customer is largely responsible for triggering the goal attribution. Thus the passer-by's utterance in this example is also interpreted as an indirect speech act, but a different one from the first example (ie, wanting to be driven to the destination vs. wanting to know how to navigate to the destination). In summary, three totally different speech acts (qv) are attributed to identical utterances as a function of different goals inferred from contextual information [additional discussion of goal determination inferences in discourse comprehension is available (Frederking, 1981; Perrault and co-workers, 1978; Carbonell, 1981b)].

Example	Speech Act
Original example	Indirect information request
Map reader	Direct information request
Cabbie example	Indirect action request

Social Role Constraints. The relative social roles of the discourse participants affect their interpretation of utterances as illustrated below:

ARMY GENERAL: I want a juicy Hamburger.

AIDE: Yes sir!

CHILD: I want a juicy Hamburger.

MOTHER: Not today, perhaps tomorrow for lunch.

PRISONER 1: I want a juicy hamburger.

PRISONER 2: Yeah, me too. All the food here tastes like cardboard.

Clearly, the interpretation of the sentence "I want a juicy hamburger" differs in each example with no context present beyond the differing social roles of the participants and their consequent potential for action. In the first example a direct order is inferred, in the second a request, and in the third only a general assertion of a (presumably unattainable) goal. Therefore, comprehending a dialogue rests critically on knowledge of social roles (Carbonell, 1981b; Grosz, 1979). Moreover, social role constraints provide part of the setting essential in making goal attributions and therefore impinge (albeit indirectly) on goal determination inferences discussed in the previous section. In unconstrained discourse there is strong interaction between goal expectations, social role constraints, indirect speech acts, and metalanguage utterance interpretation.

CONCLUSION

This article has presented a brief overview of the current state of the art of NLP: the process of developing computer systems that communicate with their users through natural language. The computational approach to NLP differs from the more general open-ended approach to natural language in linguistics and cognitive psychology. As shown above, practical natural language interfaces can currently be constructed to perform limited tasks within restricted domains, and the various techniques that have been employed to construct such interfaces have been examined and compared. Further details on any of the systems or techniques described can, of course, be obtained by following the large set of references provided. Further

general information has been published (Charniak and Wilks, 1976; Schank and Colby, 1973) and some implementation details of systems illustrative of the cognitive simulation approach are also available (Schank and Riesbeck, 1980; "Teaching Computers," 1986).

BIBLIOGRAPHY

J. F. Allen, *A Plan Based Approach to Speech Act Recognition,* Ph.D. dissertation, University of Toronto, 1979.

J. F. Allen and C. R. Perrault, "Analyzing Intention in Utterances," *Artif. Intell.* **15**(3), 143–178 (1980).

J. R. Anderson, *Language, Memory, and Thought,* Lawrence Erlbaum, Hillsdale, N.J., 1976.

R. J. Bobrow, *The RUS System,* BBN Report 3878, Bolt, Beranek, and Newman, Cambridge, Mass., 1978.

D. G. Bobrow and J. B. Fraser, "An Augmented State Transition Network Analysis Procedure," in *Proceedings of the First IJCAI,* Washington, D.C., Morgan-Kaufmann, San Mateo, Calif., 1969, pp. 557–567.

J. S. Brown and R. R. Burton, "Multiple Representations of Knowledge for Tutorial Reasoning," in D. G. Bobrow and A. Collins, eds., *Representation and Understanding,* Academic Press, Inc., New York, 1975, pp. 311–349.

R. R. Burton, *Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems,* BBN Report 3453, Bolt, Beranek, and Newman, Cambridge, Mass., Dec. 1976.

J. G. Carbonell, *Subjective Understanding: Computer Models of Belief Systems,* Ph.D. dissertation, Yale University, New Haven, Conn., 1979.

J. G. Carbonell, "Towards a Robust, Task-Oriented Natural Language Interface," in *Workshop/Symposium on Human Computer Interaction,* Georgia Technical Information Sciences, Mar. 1981a.

J. Carbonell, *Subjective Understanding: Computer Models of Belief Systems,* UMI Research Ann Arbor, Mich., 1981b.

J. G. Carbonell, *Interpreting Meta-Language Utterances,* Preprints of the Workshop: L'Analyze du Language Naturel par L'Ordinateur, Cadarache, France, 1982.

J. G. Carbonell, "Discourse Pragmatics in Task-Oriented Natural Language Interfaces," *Proceedings of the Twenty-First Annual Meeting of the Association for Computational Linguistics,* Cambridge, Mass., 1983.

J. G. Carbonell, "Robust Man–Machine Communication, User Modelling and Natural Language Interface Design," in S. Andriole, ed., *Applications in Artificial Intelligence,* Petrocelli, Boston, 1985.

J. G. Carbonell, W. M. Boggs, M. L. Mauldin, and P. G. Anick, "The XCALIBUR Project, A Natural Language Interface to Expert Systems," in *Proceedings of the Eighth IJCAI,* Karlsrube, FRG, Morgan-Kaufmann, San Mateo, Calif., 1983, pp. 653–656.

J. G. Carbonell, W. M. Boggs, M. L. Mauldin, and P. G. Anick, "The XCALIBUR Project, A Natural Language Interface to Expert Systems and Data Bases," in S. Andriole, ed., *Applications in Artificial Intelligence,* Boston, Mass. 1985.

J. G. Carbonell and P. J. Hayes, "Dynamic Strategy Selection in Flexible Parsing," in *Proceedings of the Nineteenth Annual Meeting of the Association for Computational Linguistics,* Stanford University, Stanford, Calif., 1981, pp. 143–147.

J. G. Carbonell and P. H. Hayes, "Robust Parsing Using Multiple Construction-Specific Strategies," in L. Bolc, ed., *Natural Language Parsing Systems,* Springer-Verlag, New York, 1985.

J. G. Carbonell, J. H. Larkin, and F. Reif, *Towards a General Scientific Reasoning Engine,* CIP #445, Carnegie-Mellon University Computer Science Department, Pittsburgh, Pa., 1983.

J. R. Carbonell, *Mixed-Initiative Man-Computer Dialogues,* Bolt, Beranek, and Newman, Cambridge, Mass., 1971.

E. C. Charniak, *Toward a Model of Children's Story Comprehension,* TR-266, MIT AI, Cambridge, Mass., 1972.

E. Charniak and Y. Wilks, eds., *Computational Semantics,* North-Holland, Amsterdam, The Netherlands, 1976.

N. Chomsky, *Syntactic Structures,* Mouton, The Hague, 1957.

N. Chomsky, *Aspects of the Theory of Syntax,* MIT Press, Cambridge, Mass., 1965.

P. R. Cohen and C. R. Perrault, "Elements of a Plan-Based Theory of Speech Acts," *Cog. Sci.* **3**, 177–212 (1979).

K. M. Colby, "Simulations of Belief Systems," in R. C. Schank and K. M. Colby, eds., *Computer Models of Thought and Language,* W. H. Freeman, San Francisco, 1973, pp. 251–286.

R. Cullingford, *Script Application: Computer Understanding of Newspaper Stories,* Ph.D. dissertation, Yale University, New Haven, Conn., 1978.

G. Dejong, *Skimming Stories in Real-Time,* Ph.D. dissertation, Yale University, New Haven, Conn., 1979.

J. Earley, "An Efficient Context-Free Parsing Algorithm," *CACM* **13**(2), 94–102 (1970).

C. Fillmore, "The Case for Case," in E. Bach and R. Harms, eds., *Universals in Linguistic Theory,* Holt, Rinehart, and Winston, New York, 1968, pp. 1–90.

R. Frederking, "A Rule-Based Conversation Participant," in *Proceedings of the Nineteenth Annual Meeting of the ACL,* 1981.

G. Gazdar, "Phrase Structure Grammars and Natural Language," in *Proceedings of the Eighth IJCAI,* 1983, pp. 556–565.

H. P. Grice, "Conversational Postulates," in D. A. Norman and D. E. Rumelhart, eds., *Explorations in Cognition,* W. H. Freeman, San Francisco, 1975.

B. J. Grosz, "The Representation and Use of Focus in a System for Understanding Dialogues," in *Proceedings of the Fifth IJCAI,* Cambridge, Mass., Morgan-Kaufmann, San Mateo, Calif., 1977, pp. 67–76.

B. J. Grosz, "Utterance and Objective: Issues in Natural Language Communication," in *Proceedings of the Sixth IJCAI,* Tokyo, Morgan-Kaufmann, San Mateo, Calif., 1979, pp. 1067–1076.

B. J. Grosz, "TEAM: A Transportable Natural Language Interface System," in *Proceedings of the Conference on Applied Natural Language Processing,* Santa Monica, Calif., Feb. 1983.

P. J. Hayes, "A Construction Specific Approach to Focused Interaction in Flexible Parsing," in *Proceedings of the Nineteenth Annual Meeting of the ACL,* 1981, pp. 149–152.

P. J. Hayes and J. G. Carbonell, "Multi-Strategy Construction-Specific Parsing for Flexible Data Base Query and Update," in *Proceedings of the Seventh IJCAI,* Vancouver, B.C., Morgan-Kaufmann, San Mateo, Calif., 1981*a*, pp. 432–439.

P. J. Hayes and J. G. Carbonell, "Multi-Strategy Parsing and Its Role in Robust Man–Machine Communication," CMU-CS-81-118, Carnegie-Mellon University Computer Science Department, Pittsburgh, Pa., May 1981*b*.

P. J. Hayes and J. G. Carbonell, "A Framework for Processing Corrections in Task-Oriented Dialogs," in *Proceedings of the Eighth IJCAI,* 1983.

P. J. Hayes and G. V. Mouradian, "Flexible Parsing," *Am. J. Comput. Ling.* **7**(4), 232–241 (1981).

P. J. Hayes and D. R. Reddy, "Steps Toward Graceful Interaction in Spoken and Written man–machine Communication," *Int. J. Man-Machine Stud.* **19**(3), 211–284 (Sept. 1983).

G. G. Hendrix, "Human Engineering for Applied Natural Language Processing," in *Proceedings of the Fifth IJCAI,* 1977, pp. 183–191.

R. M. Kaplan, "A General Syntactic Processor, in R. Rustin, ed., *Natural Language Processing,* Algorithmics, New York, pp. 193–241, 1973.

S. J. Kaplan, *Cooperative Responses from a Portable Natural Language Data Base Query System,* Ph.D. dissertation, University of Pennsylvania, Philadelphia, 1979.

M. Kay, "The MIND System," in R. Rustin, ed., *Natural Language Processing,* Algorithmics, New York, 1973, pp. 155–188.

S. C. Kwasny and N. K. Sondheimer, "Ungrammaticality and Extragrammaticality in Natural Language Understanding Systems," in *Proceedings of the Seventeenth Meeting of the Association for Computational Linguistics,* San Diego, Calif., 1979, pp. 19–23.

S. C. Kwasny and N. K. Sondheimer, "Relaxation Techniques for parsing Grammatically Ill-Formed Input in Natural Language Understanding Systems," *Am. J. Comput. Ling.* **7**(2), 99–108 (1981).

J. McDermott, *R1:A Rule-Based Configurer of Computer Systems,* Carnegie-Mellon University, Pittsburgh, Pa., 1980.

J. McDermott, "XSEL: A Computer Salesperson's Assistant," in J. Hayes, D. Michie, and Y-H. Pao, eds., *Machine Intelligence,* Vol. 10, John Wiley & Sons, Inc., New York, 1982, pp. 325–337.

M. A. Marcus, *A Theory of Syntactic Recognition for Natural Language,* MIT Press, Cambridge, Mass., 1980.

R. C. Parkinson, K. M. Colby, and W. S. Faught, "Conversational Language Comprehension Using Integrated Pattern-Matching and Parsing," *Artif. Intell.* **9,** 111–134 (1977).

C. R. Perrault, J. F. Allen, and P. R. Cohen, "Speech Acts as a Basis for Understanding Dialog Coherence," in *Proceedings of the Second Conference on Theoretical Issues in Natural Language Processing,* Cambridge, Mass., 1978.

S. R. Petrick, *A Recognition Procedure for Transformational Grammars.* Ph.D. dissertation, MIT, Cambridge, Mass., 1965.

C. Riesbeck, "Conceptual Analysis," in R. C. Schank, ed., *Conceptual Information Processing,* North-Holland, Amsterdam, The Netherlands, 1975, pp. 83–156.

C. R. Riesbeck and R. C. Schank, *Comprehension by Computer: Expectation-Based Analysis of Sentences in Context,* Yale University, New Haven, Conn., 1976.

J. R. Ross, "Metalinguistic Anaphora," *Ling. Inq.* 1(2), 273 (1970).

E. D. Sacerdoti, "Language Access to Distributed Data with Error Recovery," in *Proceedings of the Fifth IJCAI,* 1977, pp. 196–202.

R. C. Schank, *Conceptual Information Processing,* North-Holland, Amsterdam, The Netherlands, 1975.

R. G. Schank, and R. P. Abelson, *Scripts, Goals, Plans and Understanding,* Lawrence Erlbaum, Hillsdale, N.J., 1977.

R. C. Schank and J. G. Carbonell, "Re The Gettysburgh Address: Representing Social and Political Acts," in N. V. Findler, ed., *Associative Networks,* Academic Press, Inc., New York, 1979, pp. 327–362.

R. G. Schank and K. M. Colby, eds., *Computer Models of Thought and Language,* W. H. Freeman, San Francisco, 1973.

R. C. Schank, M. Lebowitz, and L. Birnbaum, "An Integrated Understander," *Am. J Comput. Ling.* **6**(1), 13–30 (1980).

R. Schank and C. Riesbeck, *Inside Computer Understanding,* Lawrence Erlbaum, Hillsdale, N.J., 1980.

J. R. Searle, *Speech Acts,* Cambridge University Press, Cambridge, UK, 1969.

J. R. Searle, "Indirect Speech Acts," in P. Cole and J. L. Morgan, eds., *Syntax and Semantics, Vol. 3, Speech Acts,* Academic Press, Inc., New York, 1975.

C. L. Sidner, *Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse,* TR-537, MIT AI Laboratories, Cambridge, Mass., 1979.

R. F. Simmons, "Semantic Networks: Their Computation and Use for Understanding English Sentences," in R. C. Schank and K. M. Colby, eds., 1973, pp. 63–113.

S. Small, G. Cotrell, and L. Shastri, "Toward Connectionist Parsing" in *Proceedings of the Second National Conference on Artificial Intelligence,* Pittsburgh, Pa., AAAI, Menlo Park, Calif., 1982, pp. 247–250.

S. L. Small and C. Rieger, "Parsing and Comprehending with Word Experts (A Theory and its Realization)," in M. Ringle and W. Lehnert (eds., *Strategies for Natural Language Processing,* Lawrence Erlbaum, Hillsdale, N.J., 1982, pp. 89–147.

"Teaching Computers Plain English," *High Technol.,* 16 (May 1986).

M. Tomita, *Efficient Parsing for Natural Language,* Kluwer Academic Publishers, Boston, 1986.

D. L. Waltz and A. B. Goodman, "Writing a Natural Language Data Base System," in *Proceedings of the Fifth IJCAI,* 1977, pp. 144–150.

R. M. Weischedel and J. Black, "Responding to Potentially Unparseable Sentences," *Am. J. Comput. Ling.* **6,** 97–109 (1980).

J. Weizenbaum, "ELIZA—A Computer Program for the Study of Natural Language Communication Between Man and Machine," *CACM* **9**(1), 36–45 (Jan. 1966).

Y. A Wilks, "Preference Semantics," in Keenan, ed., *Formal Semantics of Natural Language,* Cambridge University Press, Cambridge, UK, 1975.

T. Winograd, *Language as a Cognitive Process, Vol. 1, Syntax.* Addison Wesley Publishing Co., Inc., Reading, Mass., 1982.

W. A. Woods, "Transition Network Grammars for Natural Language Analysis," *CACM* **13**(10), 591–606 (Oct. 1970).

W. A. Woods, W. M. Bates, G. Brown, B. Bruce, C. Cook, J. Klovstad, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf, and V. Zue, *Speech Understanding Systems,* Final Technical Report 3438, Bolt, Beranek, and Newman, Cambridge, Mass., 1976.

W. A. Woods, R. M. Kaplan, and B. Nash-Webber, *The Lunar Sciences Language System,* Final Report, 2378, Bolt, Beranek, and Newman, Cambridge, Mass., 1972.

JAIME G. CARBONELL
PHILIP J. HAYES
Carnegie Mellon University
and Carnegie Group Inc.

This research was sponsored in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539, and in part by the Air Force Office of Scientific Research under Contract F49620-79-C-0143. The views and conclusions contained in this document are those of the authors

and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, the Air Force Office of Scientific Research, or the U.S. Government.

NEAR-MISS ANALYSIS. See CONCEPT LEARNING; LEARNING, MACHINE.

NETtalk

A neural network model that was developed in 1985, NETtalk learns to read English aloud [see T. Sejnowski and C. Rosenberg, "Parallel Networks that Learn to Pronounce English Text," *Complex Systems* **1**, 145–168, (1987)]. The input to the neural network is a string of letters and the output is a corresponding string of phonemes. When coupled to a speech synthesizer, a simulation of the network on a workstation can pronounce words at the rate of 10 letters per second. The network was trained using the backpropagation learning algorithm on a 20,000 word dictionary of aligned phonemes. The NETtalk database has become a benchmark for other learning systems and is available via FTP from nn-bench-Request@b.gp.cs.cmu.edu

TERRENCE J. SEJNOWSKI
University of California, San Diego

NEURAL NETWORKS

Historically, artificial intelligence grew out of work in neural networks, cybernetics (qv), information theory, and automata theory (as exemplified in, eg, Shannon and McCarthy, 1956). The serial symbol-manipulation approach dominated AI in the 60s and 70s, but the 80s saw a resurgence of interest in distributed computation and in neural networks in the AI community. [For a historical account, see Arbib (1987), and the collection of classical papers edited by Anderson and Rosenfeld (1988).] The term neural networks (NNs) may refer to the circuitry of real brains or to technological devices for a mode of parallel computation which complements approaches to AI based on serial processing and symbol manipulation. This mode of computation is known as neural computing, the study of artificial neural networks (ANNs); it is also referred to as connectionism (qv) or parallel distributed processing. While many people identify NNs with the training of layered NNs using techniques like back propagation (qv) to set the weights in such a net, a broader view is shown in this article, stressing the biological roots of the subject as well as theoretical approaches to networks based on optimization theory and statistical physics. Issues in learning and self-organization are emphasized, with applications to vision and motor control, and the importance of oscillations in NNs is stressed. (For a complementary view, see CONNECTIONISM.)

The language of force fields and energy landscapes provides a useful tool for the analysis of network properties, viewing a computer not in terms of programmed passage through a sequence of discrete states but rather in terms of the motion of a dynamic system towards some attractor set or in terms of oscillatory behavior described by limit cycles. Whereas time in sequential programs is a bookkeeping variable, so that the result of executing a program is not affected by the execution speed, the interaction between different subunits in massively parallel systems like NNs depends crucially on time. Neurons can change internal states of other neurons if particular messages are exchanged at the right time. The same messages might have no influence on the receiving neurons at other times. The additional degrees of freedom provided by the relative timing of neurons have to be taken into account when a program is designed and can be exploited for computation.

Thus, rather than thinking of computation as serial, the concern is with the new paradigm of *cooperative computation* (a shorthand for computation based on the competition and cooperation of concurrently active subsystems) in which local interactions yield an overall result through global self-organization without explicit executive control. Cooperation self-organizes a pattern of "strengthened alliances" between mutually consistent hypotheses about aspects of a problem; it is as a result of competition that hypotheses which do not meet the evolving (data-guided) consensus lose activity. A related theory is that of cooperative behavior in automata developed by Tsetlin (1973). Intriguingly, this work has roots in a classic but neglected study of the conduction of impulses in a network of connected excitable elements, specifically in cardiac muscle (Wiener and Rosenblueth, 1946; Selfridge, 1948) which has been suggested by Minsky (1969) as a possible theoretical basis for studies of epilepsy.

In one of the foundation papers for the study of cooperative computation, Kilmer and co-workers (1969) modeled the reticular formation of the brainstem as a series of interconnected modules, each receiving partial sensory input. They suggested a scheme whereby the modules (by continued interaction; not by executive fiat) could reach a consensus as to the correct global mode of activity for the organism. The modules cooperate while the modes compete. This approach to competition and cooperation can be placed in perspective by considering two early models, one a maximum selector and the other for stereo vision (qv), that also introduce themes to be developed in later sections. Didday (1976) modeled the frog tectum as a distributed network to convert a sensory input signaling a number of prey objects into an output array commanding a snap at one of them. This model embodies pure competition (the neuron that "wins" the competition determines which prey the frog will snap at), and is the earliest example of a "winner-take-all" network. Dev's (1975) model of stereopsis embodies both competition (between neurons encoding different depths in a given direction) and cooperation (so that neurons encoding similar depths in nearby directions will excite each other, thus favoring stable states that encode surfaces rather than rapid fluctuations in depth with changing visual direction). The Dev model

ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE

This extensively revised and expanded *Second Edition* of the *Encyclopedia of Artificial Intelligence* defines the discipline by bringing together the core of knowledge from all fields encompassed by AI. It covers the latest developments in current AI topics such as neural networks, fuzzy logic, machine vision, natural language generation, and many more. Includes:

- Over 450 articles—all entries written expressly for the *Encyclopedia*
- Over 5,000 literature references; 454 illustrations and color photographs
- Over 50% new and revised material
- Exemplary indexing and cross-referencing for easy, complete information access to all topics

Praise for the *First Edition*...

"The *Encyclopedia* is a wonder of clarity and scope: surprisingly easy to read...the clarity is an especially pleasant surprise, considering the articles were all written by AI experts...It's a treasure house of easily accessible knowledge."
—*Language Technology*

"Excellent bibliographies are attached to most of the articles, and diagrams and sketches are clear and helpful. The indexing and cross-indexing are exemplary. As the editor points out, the reader will be led by the extensive cross-references to almost every other article..." —*Artificial Intelligence Reporter*

"The *Encyclopedia* is a first-class piece of work that will be an indispensable part of any AI library." —*Computing Reviews*

"...A tour de force...a truly fantastic encyclopedia which no one in the field of artificial intelligence can afford to be without."
—*Systems Research & Information*

WILEY-INTERSCIENCE
John Wiley & Sons, Inc.
Professional, Reference and Trade Group
605 Third Avenue, New York, N.Y. 10158-0012
New York • Chichester • Brisbane • Toronto • Singapore
ISBN 0 471-50307-X (Two-Volume Set)
ISBN 0 471-50305-3 (Vol. 1)
ISBN 0 471-50306-1 (Vol. 2)

ISBN 0-471-50306-1

9 780471 503064 90000