COMPUTERS IN FOREIGN LANGUAGE

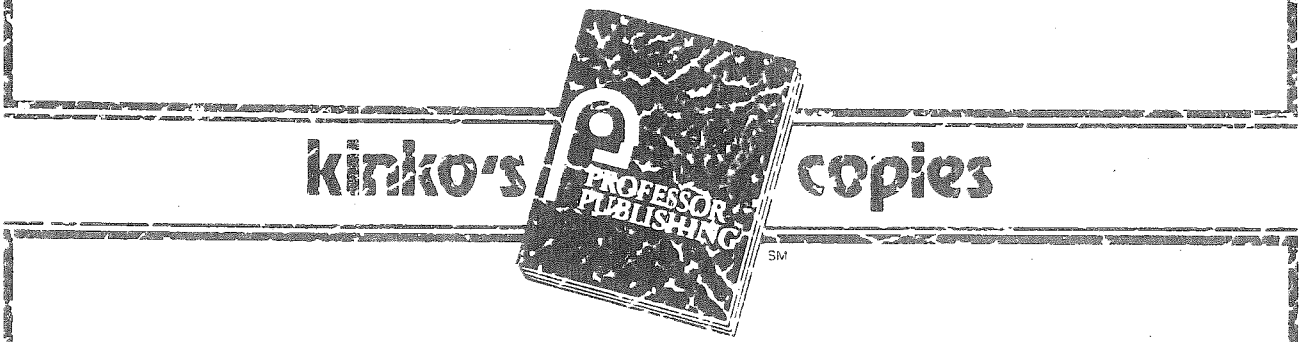
408/ 508

FISCHER

#3

db 0949c

Acorn DB #
from page
Scan



FOR STUDENTS ONLY

ENROLLED IN THIS CLASS

SPRING TERM 1985

En·glish:
The newest computer language.

Computer scientists and others have long realized that computers cannot reach their full potential if only a select group of people can successfully interact with the machines. System designers have devised various methods to overcome the human/machine barrier; some, for instance, incorporate simple "menus" that lead operators through procedures, giving and requesting information as necessary. But to completely remove the constraints common in computer use, researchers have sought to make the machines interact in humanlike ways.

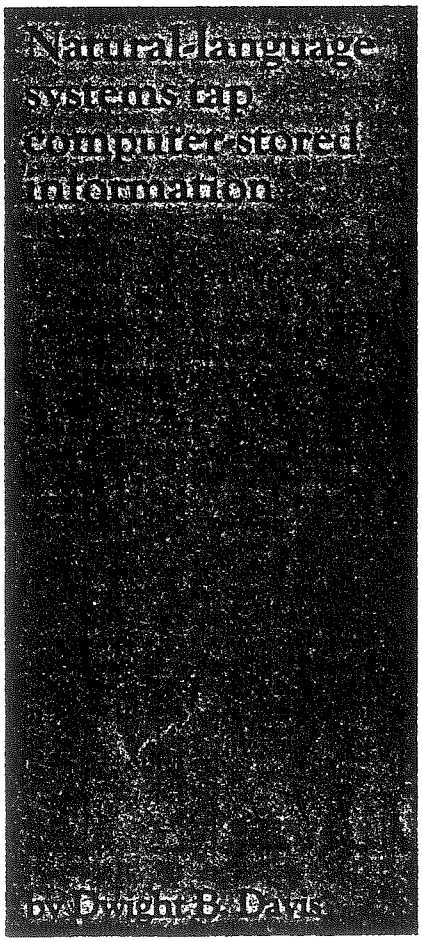
Fundamental to this quest is the ability for people to use plain English (or other natural languages) to access computer-based information and processing power. Following years of artificial intelligence research, which included some disappointing forays into the use of computers as language translation devices, several natural-language processing systems have reached the commercial stage. These systems merge computer science and linguistics to "understand" natural-language statements and to react appropriately.

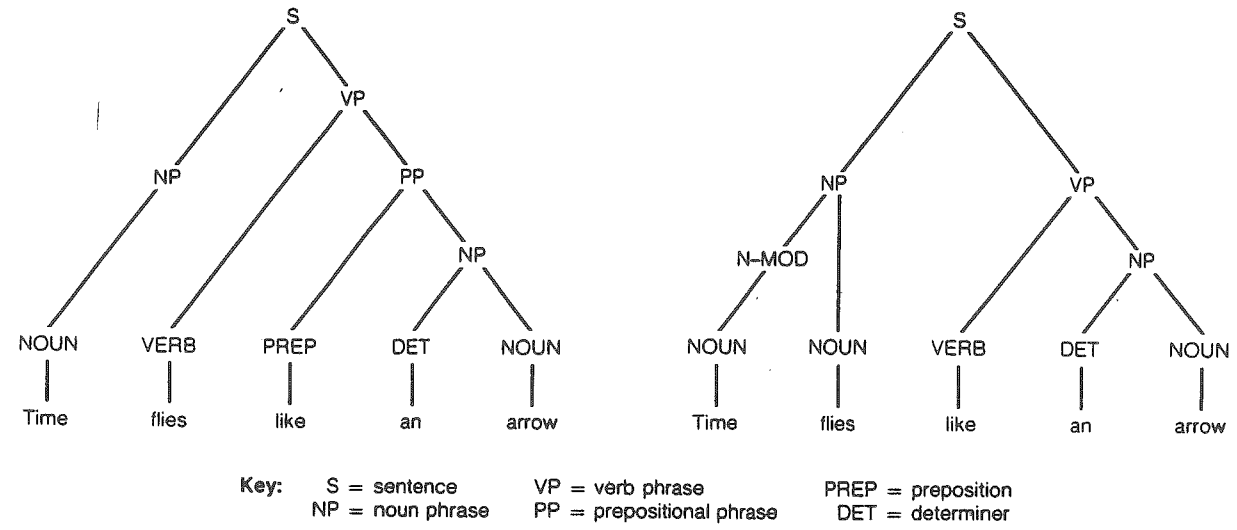
The new systems, like some of their predecessors, perform a translation function. Rather than converting German to Russian, however, the systems convert natural languages into computer languages. Most products in this area serve in database query applications. Users first type a request for information in English. The natural-language system converts the request into a formal database query language, retrieves the information from the computer's memory, and displays it to the user.

Users must type, rather than speak, English sentences because the technology of speech recognition is not yet advanced enough to work cost-effectively. When it is developed to acceptable levels, combining it with natural-language systems will be relatively easy.

Computational linguists. Although teaching a computer to recognize English words and sentence structure might at first appear simple, the task is enormously complex. The most advanced natural-language systems today exhibit only a small fraction of the average human's capability to understand language. Whether researchers can ever produce a system that equals a human's language comprehension is still an open question.

Natural-language systems exploit to varying degrees three linguistic fields: syntax, semantics, and pragmatics. The best understood of these is syntax, the rules used to combine words into phrases and sentences. Most systems employ parsers to break sentences into





Most natural-language processing systems perform one or more parses of each sentence to help determine its meaning. Many types of computational parsers exist, but all try to determine the function of each word in a sentence—both its part of speech and its grouping with other words. Such syntactic analysis can successfully interpret the meaning of some sentences, but many sentences contain structural ambiguities. For example, most human readers would interpret the sentence "Time flies like an arrow" as in the first parse tree, which shows how the grammatical categories relate to the sentence's words. Computer parsers, however,

their component parts of speech for further analysis.

Yet systems can't rely solely upon syntactic analysis to understand statements. Most sentences can be parsed several ways, especially since many words serve as different parts of speech in different contexts. Thus some form of semantic analysis, which examines the meanings of words and their relationships, must also be applied. Natural-language system vendors Artificial Intelligence Corp. (Waltham, Mass.) and Frey Associates (Amherst, N.H.) claim to offer a balance of the syntactic and the semantic in their systems.

A third company, Cognitive Systems (New Haven, Conn.), claims to dispense with syntax entirely, relying instead upon a semantic theory developed by its president, Roger Schank, a Yale University researcher. Because sentences that differ markedly in syntax can mean the same thing, Schank believes any attempt to analyze syntax is misguided. His theory of "conceptual dependency representation" breaks the complex concepts represented by words into a limited number of simple "primitive-act" components. For example, just 11 primitive acts encompass all verbs denoting action (see "Eleven action-verb primitives").

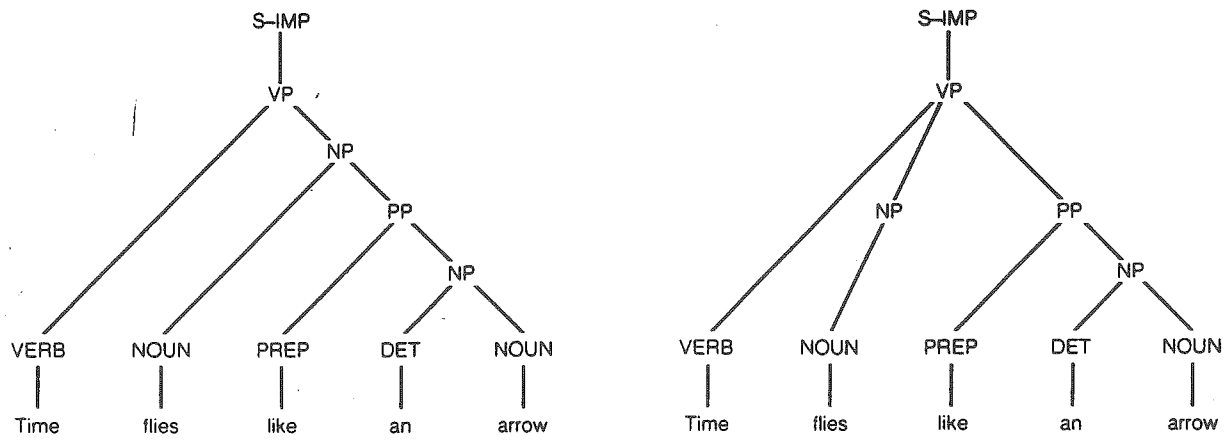
When a word is categorized as a particular primitive act, it raises expectations in the system about its relationship with other words in the sentence. The system searches for confirmation of its expectations and analyzes sentences through several repetitions of this procedure, moving from word to word. Cognitive Systems is developing several products based on its semantic approach. Some will serve as front ends to databases, but most will function as "advisory" systems, which will interactively advise users working on specific tasks. The company has several contracts to develop such systems, including ones with an Italian oil company, a French insurance company and a Bel-

In the theory of conceptual dependency representation, developed by Roger Schank, president of Cognitive Systems, the complex meanings of words are reduced to their simple, underlying components. The computer model then uses these general components to represent and group various types of words. For example, all verbs representing action are reduced to eleven "primitive acts."

Eleven action-verb primitives:

ATRANS: Transfer of an abstract relationship, such as possession, ownership, or control; e.g., "buy" is made of two conceptualizations that cause each other, an *Atrans* of money and an *Atrans* of the object being bought.
PTRANS: Transfer of an object's physical location.
PROPEL: Application of a physical force to an object.
MOVE: Movement by an animal of a part of its body.
GRASP: Grasping of an object.
INGEST: Taking in of an object by an animal.
EXPEL: Expulsion of an object from the body of an animal.
MTRANS: Transfer of mental information between animals or within an animal; e.g., "tell" is *Mtrans* between people, and "see" is *Mtrans* from eyes to consciousness.
MBUILD: Construction of new information from old ; e.g., "conclude," "imagine," "consider."
SPEAK: Sound production; e.g., "say," "purr," "play music."
ATTEND: Attending or focusing a sense organ toward a stimulus; e.g., "see" is treated as *Mtrans* by means of *Attend.*

Two additional parses as imperative sentences



could interpret the sentence in other ways. In the second parse tree, "time flies" becomes some kind of flies that enjoy (like) arrows. In the third and the fourth, "time" is categorized as a verb, producing two imperative sentences telling the implicit subject "you" to get out a stopwatch and time flies as you would an arrow or as an arrow would time flies. Such multiple syntactic interpretations illustrate the need to apply semantic analysis as well when interpreting sentences. (Source: John F. Sowa, Conceptual Structures: Information Processing in Mind and Machine *[Reading Mass.: Addison-Wesley Publishing Co., 1983.])*

gian bank, says Abraham Gutman, assistant VP.

With Schank's approach to language understanding, the systems must have extensive knowledge about the domain in which they're operating, as well as about word definitions and categorization. Some researchers question the commercial viability of this approach, as opposed to that of more general-purpose natural-language systems. "Cognitive's systems depend too much on knowledge of the domain, which means that they are extremely expensive to put up on a new domain," says Frederick B. Thompson, professor of computer science at the California Institute of Technology (Pasadena). Thompson and his wife, Bozena Henisz Thompson, a senior research associate in linguistics at Caltech, have produced three generations of natural-language systems, the latest of which may soon see commercial application.

Also, while Schank's colleagues in the natural-language field agree that his approach is heavily oriented toward semantics, most claim he relies more on syntax than he would like to admit. "You must examine the word order and the organization of phrases in order to discover what semantic guidance there is in a sentence," says Ron Kaplan, a research scientist at the Xerox Palo Alto Research Center. Kaplan says the key to efficient natural-language processing is to apply both syntactic and semantic analysis as early as possible in the examination of a sentence.

If a system first parses a sentence in multiple ways and then applies semantic analysis to each alternative, much processing time and power can be wasted, Kaplan says. "If you don't do the semantics on the fly as you're doing the syntax, you're liable to accept bad analysis paths in the syntax and do a lot of work on them," he cautions. "Only later when you apply your semantic constraints might you discover that it was total nonsense and that you could have saved yourself a couple of hours of computing."

One solution to this problem is the use of semantic grammars, which mix semantic constraints with the syntactic patterns of word order. Forthcoming natural-language products from Symantec (Sunnyvale, Cal.) will reportedly use such grammars. A drawback to semantic grammars is that they are closely tied to the application for which they are written. In order to apply them to a new application database, which may have different meanings for the same words, the grammars must be substantially rewritten.

Kaplan, in collaboration with another researcher, has developed "lexical functional grammar" (LFG), which he claims offers the advantages of semantic grammars without their drawbacks. "With LFG, you take a grammar of English and the semantic restrictions of a database," he says. "Then there is an algorithm that distributes the information about the semantics into the syntactic grammar." Having this algorithm permits the automatic production of grammars for specific databases, rather than requiring the manual rewriting necessary with semantic grammars, Kaplan says.

The third linguistic field, pragmatics, has barely been touched by natural-language processing systems. Pragmatics involves an interpretation of sentences that takes into account such variables as who wrote the sentence and where and when it was written. Such analysis requires the system to have extensive world knowledge in order to fully understand sentences.

Pragmatics is the opposite of mathematics, according to Kaplan, which helps explain why it is so difficult to implement on a computer: "Sentences of mathematics are eternally true. 'Two plus two equals four' is true for all speakers, for all time, and for all space."

"Most of our theories of meaning come from mathematics and mathematical logic," he explains, "and these haven't provided the tools that would enable researchers to handle pragmatics. So it's been very difficult to construct the theoretical tools—let alone the implementation—that would really give us a handle on all the ways the pragmatic factors can enter in." Kaplan says this situation is beginning to change and notes that researchers at Stanford University have developed a theory called "situation semantics,"

which basically states that the meaning of an utterance depends upon the situation in which it's uttered. But for the foreseeable future, natural-language systems will rely almost solely upon syntax and semantics to analyze sentences.

Expanding Intellect. The first natural-language system to attain some commercial success is the Intellect system from Artificial Intelligence Corp. (AIC). Selling for about $70,000, Intellect typically operates on large IBM computers and interacts with several types of underlying software. Over 150 firms have purchased the system, and last summer IBM began marketing Intellect through its own salesforce. Renamed versions of the system are also sold by such companies as Cullinet Software ("Online English") and Information Sciences ("GRS Executive").

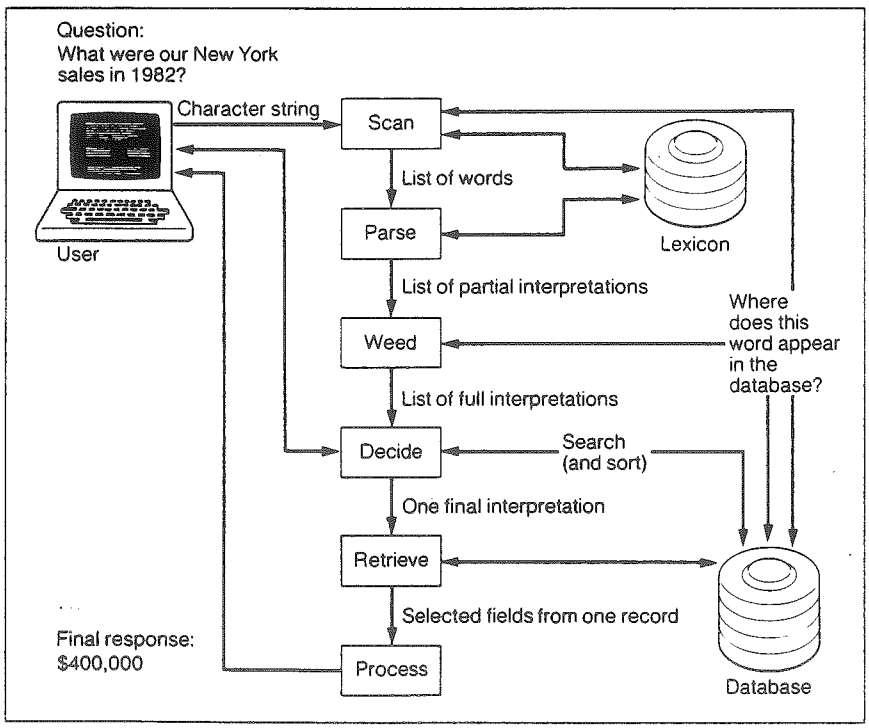
Intellect is much more than a simple natural-language translation system, according to Larry Harris, AIC president. "Intellect acts as a hub between a number of different software systems," he explains. For example, a user might type "Show me a bar graph comparing our 1983 year-to-date sales with our 1983 estimated year-to-date sales by region." Intellect translates the request into the database query language to obtain the necessary data and then interacts with graphics software, which transforms the rough data into a chart.

Because Intellect works as a front end to different database query systems, it provides some of its own management facilities to ensure a common level of operation at all installations. But Caltech's Frederick Thompson says Intellect's performance suffers, because it interfaces to older, less efficient database management systems. The Thompsons' most advanced work, the ASK (A Simple Knowledgeable) System, is a comprehensive product that provides both natural-language capability and sophisticated database management. "We've spent a great deal of time optimizing disk access," says Thompson, "so we can handle reasonable-sized databases with good response times."

In one of the simplest Intellect applications, personnel clerks at the Boston-based Filene's department store use the system for easy access to employees' records. Filene's has "an antiquated computer payroll and personnel system," explains Nick Stevens, director of personnel services. "We told our computer people for several years that we needed a rewritten system, but we were a low priority on their list." Rather than revamp the entire system, Filene's acquired Intellect about two years ago.

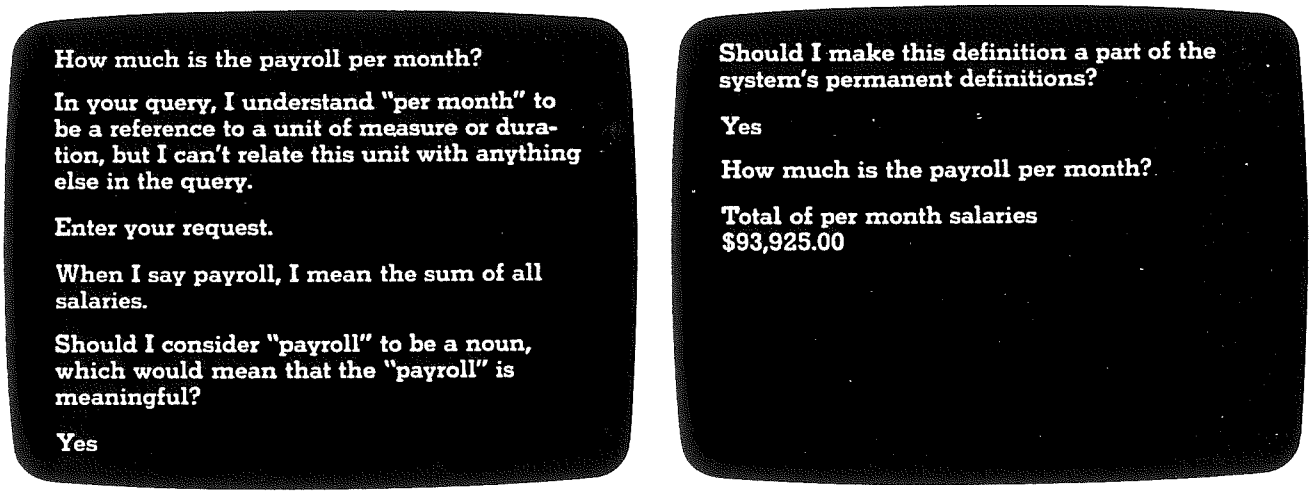
"It's an indispensable tool," says Stevens, "because it allows us to go live to the IBM mainframe and ask it our own questions." The personnel program is indexed on about six fields, including name and social security number and can quickly answer questions such as "Where does Charlie Smith live?" However, with only a few fields indexed, the system sometimes responds slowly to questions that require it to sort information. "We're not indexed on minorities, for example, so if we ask 'How many employees at Filene's are minorities?' the system has to check each of the 16,000 records we have on file to find the information block we requested," Stevens says. He notes that this indexing limitation is not an Intellect shortcoming but a reflection of the amount of time Filene's personnel can spend updating each system file.

While the Filene's application represents straightforward database access, other users exploit Intellect's ability to perform calculations on the data accessed. Fleet Information (Providence, RI), a subsidiary of the Fleet Financial Group, acquired Intellect two years ago to perform in applications such as handling consumer loans. "Intellect expanded the range of people who had online access to data beyond those who had a data processing background," says Tom Benoit, information center manager. Fleet has over 400 indexed fields of information on its system and requires Intellect to perform multiple levels of arithmetic calculations, Benoit says.



The Intellect natural-language processing system progresses through several steps to interpret and respond to a user's request for information. First, the scan function breaks the request "What were our New York sales in 1982?" into appropriate word entities, using information from both the computer's database and the system's lexicon (vocabulary). The scanner can't just assume that breaks occur at every space; for instance, "New York" should be classified as one, not two, word entities. Next, the parser grammatically diagrams the sentence in as many ways as possible and passes a list of partial interpretations to the weed function. Weed attempts to "fill in the holes" in the partial interpretations, using the database for guidance. Any complete interpretations that result pass to the decide process. Decide, in part, uses the relative difficulty with which each full interpretation was produced to assign preference values to each. If the system can't clearly choose one final interpretation based on its preference rating and from additional searching of the database, it may query the user for additional information. For example, it might ask if "New York" refers to the city or the state, a decision it couldn't reach alone if its database contained sales figures for each. Once the final interpretation is chosen, the system retrieves the requested information, processes and organizes it as necessary, and displays it to the user.

Core technology. That Intellect opens a variety of computer applications to untrained users illustrates the wide-ranging roles possible for natural-language systems. Database query



Frey Associates' Themis permits users to add words and definitions easily to its vocabulary. In the above example, where the system doesn't recognize the word "payroll," the user defines the word's meaning and part of speech and then makes it a permanent definition. Word definitions can be designated temporary if the user wants to experiment with the results. Because the system depends upon the user to correctly identify a word's part of speech, problems can develop if an improper designation is made. Yet, because the user generally has very few options to choose from, this isn't expected to prove troublesome.

represents perhaps the largest initial market for the technology, but virtually every computer application could conceivably benefit from the addition of a natural-language interface. As Intellect's high cost suggests, price may be a barrier to the use of such systems in many low-end applications. But the field is evolving rapidly toward less expensive products for less expensive computers.

Themis, a natural-language system introduced in October by Frey Associates, demonstrates this trend. Designed to run on DEC VAX-11 superminicomputers, Themis costs $24,500, or about a third as much as Intellect. Like Intellect, Themis interfaces to database query languages—DEC's Datatrieve and Oracle Corp.'s Oracle—and performs such functions as sorting, logical comparisons, and calculations. But Themis does not yet operate with graphics software.

Both Intellect and Themis store grammatical rules and word definitions to help them interpret the meaning of English statements. The stored vocabularies, or lexicons, contain words commonly used within specific applications. AIC encourages its customers to write their own lexicons, a task requiring some technical expertise but no specialized linguistic talents. These lexicons are then available to all system users; the customer's lexicon writers can update the system as necessary.

Themis, which reportedly understands statements even if they contain misspellings and typos, uses a somewhat different approach to providing the vocabulary. The basic system comes with a 900-word vocabulary of commonly used words, and each user can easily add words and definitions to the system as desired. Different users may define the same word in different ways for their own needs, but they cannot modify the system's foundation vocabulary.

"If you enter a statement that has words not in the vocabulary, you just teach the system those words immediately and get your answer," explains Eric Frey, chairman and CEO. "It's frustrating to users if they have to wait for someone else to build the vocabulary, even if it's updated with a batch run the next day."

Further information

For market projections see Business Outlook, next page. Additional references appear in Resources, p. 79.

AIC's Harris admits that giving each user the capability to add to the lexicon is desirable; the next product from AIC will offer such a feature, he says. But he cautions that no one has much experience in gauging the likely success of such an approach. "We didn't offer such a capability with our first system, because we wanted to be very conservative and have someone assigned responsibility for the overall success of the application at a given site."

Just as Frey Associates has brought natural-language processing down a notch from mainframes to minicomputers, so will future products make the leap into the microcomputer realm. A natural-language system integrated with database management software for the IBM Personal Computer is due from Symantec early this year. And both Harris and Frey expect to eventually scale their products down for use on personal computers.

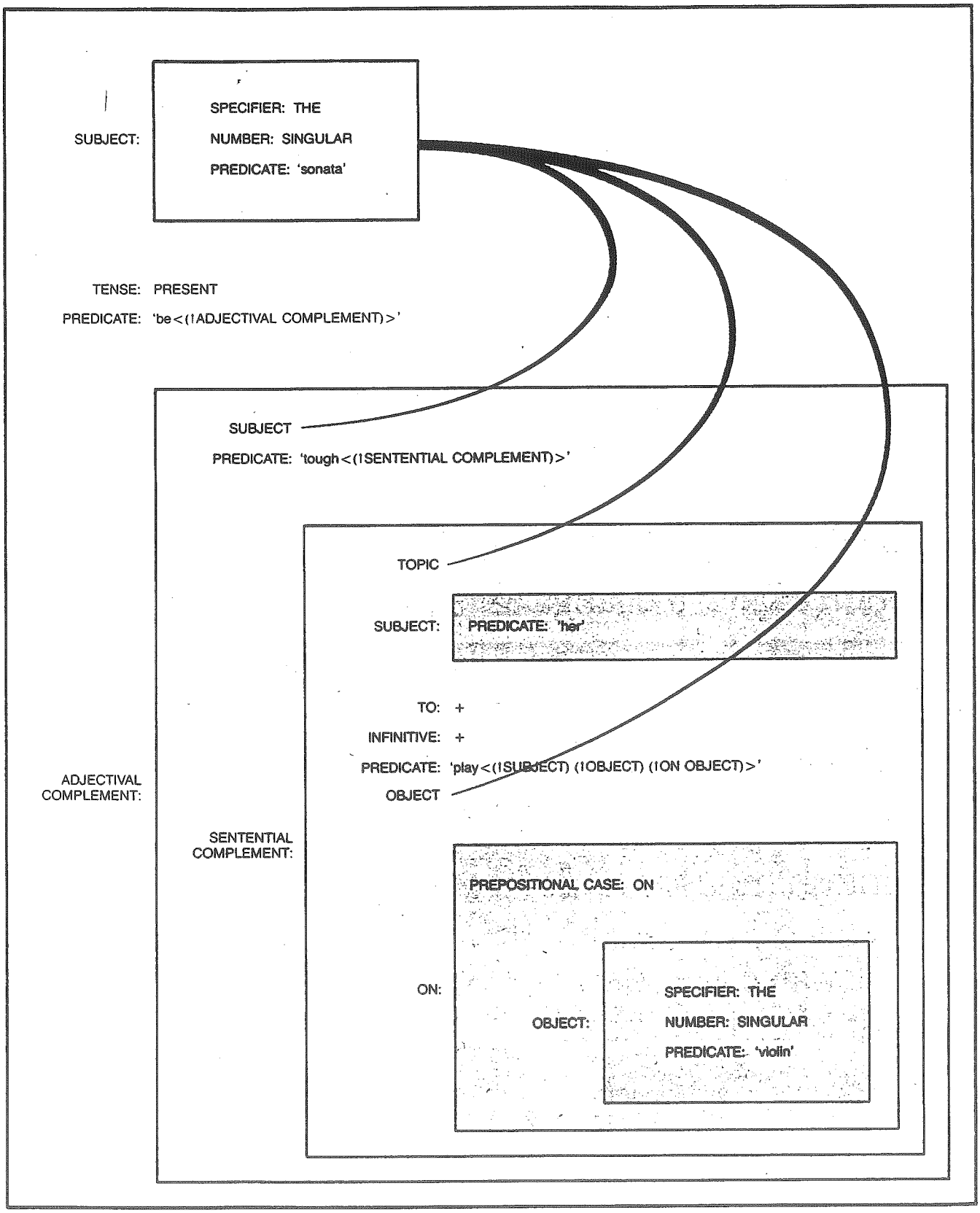
The Caltech ASK System already runs on a Hewlett-Packard desktop computer, the HP9836. ASK incorporates features that evolved from an extensive study by Bozena Thompson, which compared human-to-human and human-to-computer natural-language interactions when performing a real task. Both Hewlett-Packard and DEC hold nonexclusive, royalty-free licenses for the natural-language system, and Frederick Thompson is confident that commercial versions of ASK will be forthcoming from these or other licensees.

For the moment, however, Thompson cautions that only Intellect has proven itself capable of satisfying commercial customers, and he notes that prototype natural-language systems have a long history of commercial failure. "It's almost impossible to separate those systems that have some commercial viability from those that don't," he says. "Toy databases and toy applications have been the anathema of the whole artificial intelligence world. People can convince you that their stuff is really great with the demonstrations they give, but when you get behind the scenes you realize immediately that the products don't work well in real applications." □

Dwight B. Davis is an associate editor of HIGH TECHNOLOGY.

Goes with Winograd, "Computer Software" (following pages)

The sonata is tough for her to play on the violin.



REPRESENTATION OF A SENTENCE in a way that makes explicit the linguistic relations among its parts has been a goal of the science of linguistics; it is also a necessary aspect of the effort to design computer software that "understands" language, or at any rate can draw inferences from linguistic input. In this illustration a sentence is given in "functional structure" form, which has the property that when part of a sentence plays a role in another part, the former is "nested" in the latter. The nesting is shown by placing one box in another, or (in three places) by a "string." The sentence was analyzed by Ronald M. Kaplan and Joan Bresnan of Stanford University and the Xerox Corporation's Palo Alto Research Center. Another functional-structure diagram appears in the illustration on pages 142–143.

Scientific American
Sept. '84

Computer Software for Working with Language

Programs can manipulate linguistic symbols with great facility, as in word-processing software, but attempts to have computers deal with meaning are vexed by ambiguity in human languages

by Terry Winograd

In the popular mythology the computer is a mathematics machine: it is designed to do numerical calculations. Yet it is really a language machine: its fundamental power lies in its ability to manipulate linguistic tokens—symbols to which meaning has been assigned. Indeed, "natural language" (the language people speak and write, as distinguished from the "artificial" languages in which computer programs are written) is central to computer science. Much of the earliest work in the field was aimed at breaking military codes, and in the 1950's efforts to have computers translate text from one natural language into another led to crucial advances, even though the goal itself was not achieved. Work continues on the still more ambitious project of making natural language a medium in which to communicate with computers.

Today investigators are developing unified theories of computation that embrace both natural and artificial languages. Here I shall concentrate on the former, that is, on the language of everyday human communication. Within that realm there is a vast range of software to be considered. Some of it is mundane and successful. A multitude of microcomputers have invaded homes, offices and schools, and most of them are used at least in part for "word processing." Other applications are speculative and far from realization. Science fiction is populated by robots that converse as if they were human, with barely a mechanical tinge to their voice. Real attempts to get computers to converse have run up against great difficulties, and the best of the laboratory prototypes are still a pale reflection of the linguistic competence of the average child.

The range of computer software for processing language precludes a comprehensive survey; instead I shall look at four types of program. The programs deal with machine translation, with word processing, with question answering and with the adjuncts to electronic mail known as coordination systems. In each case the key to what is possible lies in analyzing the nature of linguistic competence and how that competence is related to the formal rule structures that are the theoretical basis of all computer software.

The prospect that text might be translated by a computer arose well before commercial computers were first manufactured. In 1949, when the few working computers were all in military laboratories, the mathematician Warren Weaver, one of the pioneers of communication theory, pointed out that the techniques developed for code breaking might be applicable to machine translation.

At first the task appears to be straightforward. Given a sentence in a source language, two basic operations yield the corresponding sentence in a target language. First the individual words are replaced by their translations; then the translated words are reordered and adjusted in detail. Take the translation of "Did you see a white cow?" into the Spanish "*¿Viste una vaca blanca?*" First one needs to know the word correspondences: "*vaca*" for "cow" and so on. Then one needs to know the structural details of Spanish. The words "did" and "you" are not translated directly but are expressed through the form of the verb "*viste*." The adjective "*blanca*" follows the noun instead of preceding it as it does in English. Finally, "*una*" and "*blanca*" are in the feminine form corresponding to "*vaca*." Much of the early study of machine translation dwelt on the technical problem of putting a large dictionary into computer storage and empowering the computer to search efficiently in it. Meanwhile the software for dealing with grammar was based on the then current theories of the structure of language, augmented by rough-and-ready rules.

The programs yielded translations so bad that they were incomprehensible. The problem is that natural language does not embody meaning in the same way that a cryptographic code embodies a message. The meaning of a sentence in a natural language is dependent not only on the form of the sentence but also on the context. One can see this most clearly through examples of ambiguity.

In the simplest form of ambiguity, known as lexical ambiguity, a single word has more than one possible meaning. Thus "Stay away from the bank" might be advice to an investor or to a child too close to a river. In translating it into Spanish one would need to choose between "*orilla*" and "*banco*," and nothing in the sentence itself reveals which is intended. Attempts to deal with lexical ambiguity in translation software have included the insertion of all the possibilities into the translated text and the statistical analysis of the source text in an effort to decide which translation is appropriate. For example, "*orilla*" is likely to be the correct choice if words related to rivers and water are nearby in the source text. The first strategy leads to complex, unreadable text; the second yields the correct choice in many cases but the wrong one in many others.

In structural ambiguity the problem goes beyond a single word. Consider the sentence "He saw that gasoline can explode." It has two interpretations based on quite different uses of "that" and "can." Hence the sentence has two possible grammatical structures, and the translator must choose between them [*see bottom illustration on page 133*].

An ambiguity of "deep structure" is subtler still: two readings of a sentence can have the same apparent grammatical structure but nonetheless differ in meaning. "The chickens are ready to eat" implies that something is about to eat something, but which are the chickens? One of the advances in linguistic

theory since the 1950's has been the development of a formalism in which the deep structure of language can be represented, but the formalism is of little help in deducing the intended deep structure of a particular sentence.

A fourth kind of ambiguity—semantic ambiguity—results when a phrase can play different roles in the overall meaning of a sentence. The sentence "David wants to marry a Norwegian" is an example. In one meaning of the sentence the phrase "a Norwegian" is referential. David intends to marry a particular person, and the speaker of the sentence has chosen an attribute of the person—her being from Norway—in order to describe her. In another meaning of the sentence the phrase is attributive. Neither David nor the speaker has a particular person in mind; the sentence simply means that David hopes to marry someone of Norwegian nationality.

A fifth kind of ambiguity might be called pragmatic ambiguity. It arises from the use of pronouns and special nouns such as "one" and "another." Take the sentence "When a bright moon ends a dark day, a brighter one will follow." A brighter day or a brighter moon? At times it is possible for translation software to simply translate the ambiguous pronoun or noun, thereby preserving the ambiguity in the translation. In many cases, however, this strategy is not available. In a Spanish translation of "She dropped the plate on the table and broke it," one must choose either the masculine "*lo*" or the feminine "*la*" to render "it." The choice forces the translator to decide whether the masculine "*plato*" (plate) or the feminine "*mesa*" (table) was broken.

In many ambiguous sentences the meaning is obvious to a human reader, but only because the reader brings to the task an understanding of context. Thus "The porridge is ready to eat" is unambiguous because one knows porridge is inanimate. "There's a man in the room with a green hat on" is unambiguous because one knows rooms do not wear hats. Without such knowledge virtually any sentence is ambiguous.

Although fully automatic, high-quality machine translation is not feasible, software is available to facilitate translation. One example is the computerization of translation aids such as dictionaries and phrase books. These vary from elaborate systems meant for technical translators, in which the function of "looking a word up" is made a part of a multilingual word-processing program, to hand-held computerized libraries of phrases for use by tourists. Another strategy is to process text by hand to make it suitable for machine translation. A person working as a "pre-editor" takes a text in the source language and creates a second text, still in the source language, that is simplified in ways facilitating machine translation. Words with multiple meanings can be eliminated, along with grammatical constructions that complicate syntactic analysis. Conjunctions that cause ambiguity can be suppressed, or the ambiguity can be resolved by inserting special punctuation, as in "the [old men] and [women]." After the machine translation a "post-editor" can check for blunders and smooth the translated text.

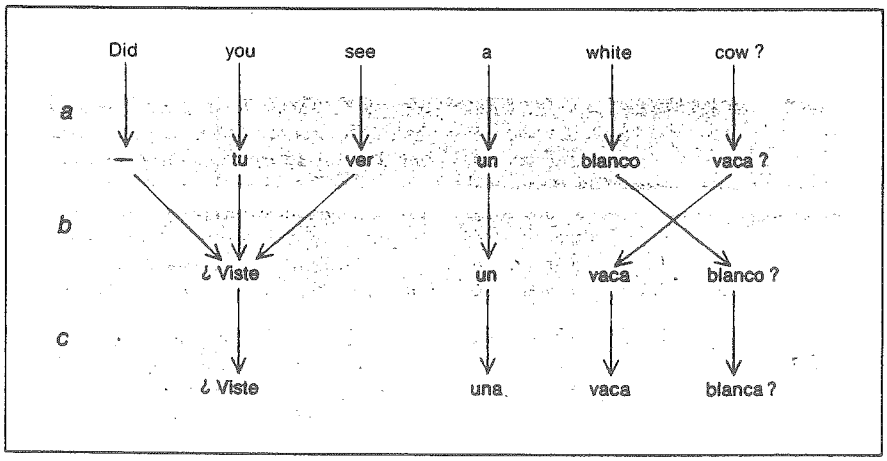
The effort is sometimes cost-effective. In the first place, the pre-editor and post-editor need not be bilingual, as a translator would have to be. Then too, if a single text (say an instruction manual) is to be translated into several languages, a large investment in pre-editing may be justified because it will serve for all the translations. If the author of the text can be taught the less ambiguous form of the source language, no pre-editor is needed. Finally, software can help in checking the pre-edited text to make certain it meets the specifications for input to the translation system (although this is no guarantee that the translation will be acceptable).

A machine-translation system employing pre- and post-editing has been in use since 1980 at the Pan-American Health Organization, where it has translated more than a million words of text from Spanish into English. A new system is being developed for the European Economic Community, with the goal of translating documents among the official languages of the community: Danish, Dutch, English, French, German, Greek and Italian. Meanwhile the theoretical work on syntax and meaning has continued, but there have been no breakthroughs in machine translation. The ambiguity pervading natural language continues to limit the possibilities, for reasons I shall examine more fully below.

I turn next to word processing, that is, to software that aids in the preparation, formatting and printing of text. Word processors deal only with the manipulation and display of strings of characters and hence only with superficial aspects of the structure of language. Even so, they pose technical problems quite central to the design of computer software. In some cases the end product of a word-processing program is no more than a sequence of lines of text. In others it is a complex layout of typographic elements, sometimes with drawings intercalated. In still others it is a structured document, with chapter headings, section numbers and so on, and with a table of contents and an index compiled by the program.

The key problems in designing word-processing software center on issues of representation and interaction. Representation is the task of devising data structures that can be manipulated conveniently by the software but still make provision for the things that concern the user of the system, say the layout of the printed page. Interaction takes up the issue of how the user expresses instructions and how the system responds.

Consider the fundamental problem of employing the data-storage devices of a computer to hold an encoded sequence of natural-language characters. The first devices that encoded text were card-punch and teletype machines, and so the earliest text-encoding schemes were tailored to those devices. The teletype machine is essentially a typewriter that converts key presses into numerical codes that can be transmitted electronically;



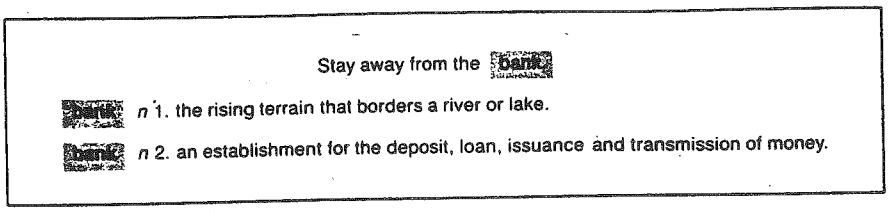
MACHINE TRANSLATION of text from one language into another was thought to be quite feasible in the 1950's, when the effort was undertaken. In the first step of the process (*a*) the computer would search a bilingual dictionary to find translations of the individual words in a source sentence (in this case Spanish equivalents of the words in the sentence "Did you see a white cow?"). Next the translated words would be rearranged according to the grammar of the target language (*b*). The changes at this stage could include excision or addition of words. Finally, the morphology of the translation (for example the endings of words) would be adjusted (*c*).

thus there are teletype codes for most of the keys on a typewriter. The codes include the alphabetic characters *A* through *Z*, the digits 0 through 9 and common punctuation marks such as the period and the comma. Standards are harder to establish, however, for symbols such as #, @, ¢ and }. And what about keys that print nothing, such as the tab key, the carriage-return key and the backspace key?

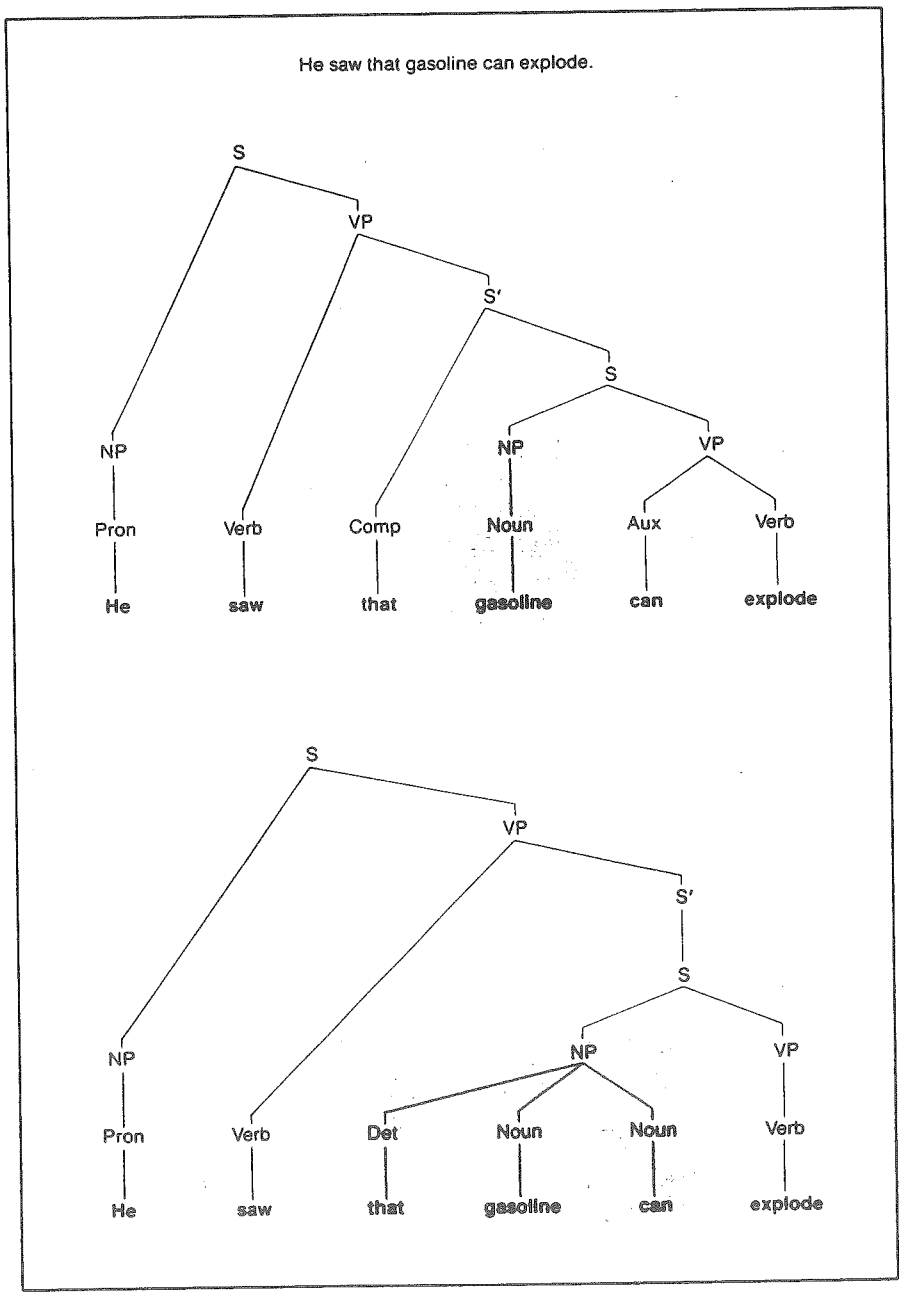
The difficulties that arise in choosing standards can be illustrated by one peculiarity of text encoding. The teletype code distinguishes between a carriage return (which returns the type carriage to the beginning of the line without advancing the paper) and a line feed (which advances the paper without repositioning the carriage). Hence the end of a line is marked by a sequence of two characters: a carriage return and a line feed. One code would suffice, and so some programs eliminate either the carriage return or the line feed, or they replace both characters with another code entirely. The problem is that various programs employ different conventions, so that lines encoded by one program may not be readable by another.

The problems become worse when a full range of characters—punctuation marks, mathematical symbols, diacritical marks such as the umlaut—is considered. Moreover, word processing is now being extended to languages such as Chinese and Japanese, which require thousands of ideographic characters, and to languages such as Arabic and Hebrew, which are written from right to left. Coding schemes adequate for English are useless for alphabets with thousands of characters. It should be said that the schemes continue to vary because political and economic forces play a role in the design of computer systems. A given manufacturer wants to promulgate a standard that suits its own equipment; thus some present-day standards exist because they were offered by a vendor that dominates a market. On the other hand, technical matters such as the efficiency of certain software running on certain hardware perpetuate differences in detail. It will be quite a while before universal standards emerge and users gain the ability to transport text from one word-processing system to any other.

Encoding schemes aside, there is the form of the letters themselves. On a typewriter keyboard an A is simply an A. Typographically, however, an A is an A or an *A* or an **A**. In the new field of digital typography the computer is a tool for the design and presentation of forms of type. Some of the efforts in the field are applied to the forms themselves: in particular the representation of characters as composites of dots and spaces. Additional efforts go into the devising of code for the computer stor-



AMBIGUOUS MEANINGS permeate natural languages (that is, languages that people speak and write) and thus subvert the attempt to have computers translate text from one language into another. Here lexical ambiguity, the simplest type of ambiguity, is diagrammed. In lexical ambiguity a word in a sentence has more than one possible meaning. In this case the word is "bank" (*color*), which might equally well refer to either a river or a financial institution. A translator must choose. The following four illustrations show more complex types of ambiguity.



STRUCTURAL AMBIGUITY arises when a sentence can be described by more than one grammatical structure. Here the conflicting possibilities for the sentence "He saw that gasoline can explode" are displayed in the form of grammatical "trees." In one of the trees the sentence has a subordinate clause whose subject is "gasoline" (*color*); the sentence refers to the recognition of a property of that substance. In the other tree "gasoline can" is part of a noun phrase (*NP*) meaning a container of gasoline; the sentence refers to the sight of a specific explosion.

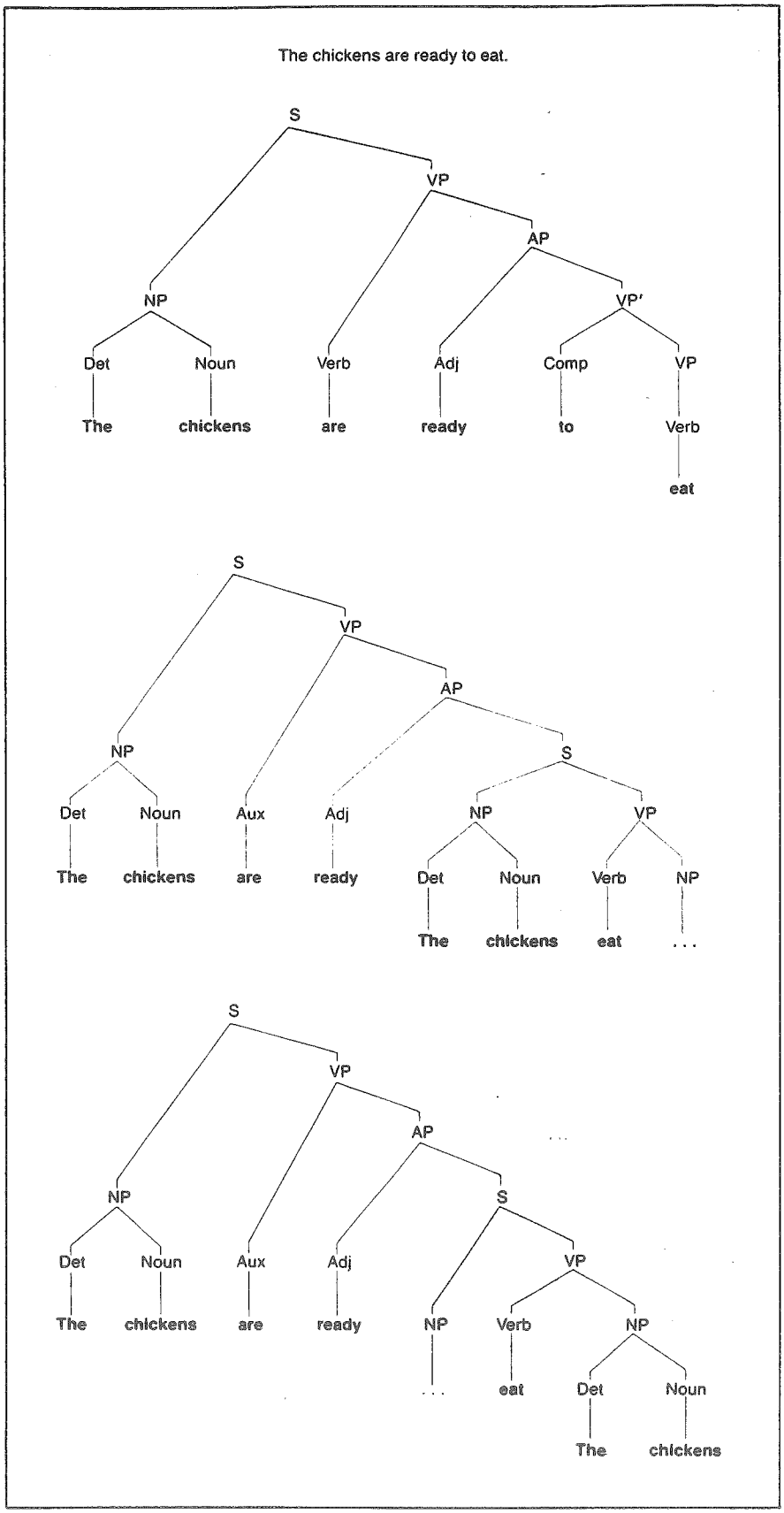
age of text that combines different fonts (such as Times Roman and Helvetica) and different faces (such as *italic* and **boldface**).

So far I have dealt only with stored sequences of characters. Yet one of the major tasks of a word-processing program is to deal with margins and spacing—with the "geography" of the printed page. In the typesetting language called TEX commands that specify nonstandard characters, change the style of type, set the margins and so on are embedded in the text [*see top illustration on page 138*]. A command to TEX is distinguished from ordinary text by the backslash character (\). The stored text is "compiled" by the TEX program, which interprets the embedded commands in order to create a printed document in the specified format.

The compiling is quite complex, and a good deal of computation is often needed to get from code created by means of a word-processing program to code that readily drives a printer or a typesetting machine. An algorithm that justifies text (fills the full width of each line of type) must determine how many words will fit in a line, how much space should be inserted between the words and whether a line would be improved by dividing and hyphenating a word. The algorithm may also take actions to avoid visual defects such as a line with wide interword spacing followed by a line that is very compact. Positioning each line on the page is further complicated by the placement of headings, footnotes, illustrations, tables and so on. Mathematical formulas have their own typographic rules.

TEX and similar programs are primitive with respect to another aspect of word processing: the user interface. The high-resolution display screens becoming available are now making it possible for the computer to display to the user a fair approximation of the pages it will print, including the placement of each item and the typeface to be employed. This suggests that the user should not have to type special command sequences but might instead manipulate page geography directly on the screen by means of the computer keyboard and a pointing device such as a "mouse." The resulting interface between the computer and the user would then fall into the class of interfaces known as WYSIWYG, which stands for "What you see is what you get."

It is worth noting that programs for manipulating text are called different things by different professions. Programmers call them text editors, but in business and publishing they are referred to as word processors; in the latter fields an editor is a person who works to improve the quality of text. Computer software is emerging to aid in this



DEEP-STRUCTURAL AMBIGUITY arises when a sentence has a single apparent structure but nonetheless has more than one possible meaning. In this example the sentence is "The chickens are ready to eat." Its grammatical structure (*top*) leaves the role of the chickens ambiguous: in one interpretation they will eat; in the other they will be eaten. Deep-structure trees make the chickens' role explicit: they are the subject of the sentence (*middle*), in which case their food is undetermined, or they are the object (*bottom*), and their eaters are undetermined.

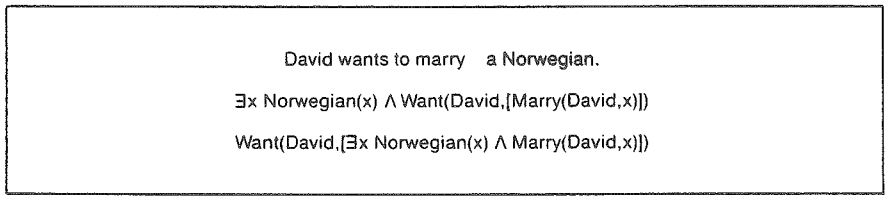
more substantive aspect of editing. It deals with neither the visual format of language nor the conceptual content but with spelling, grammar and style. It includes two kinds of programs: mechanized reference works and mechanized correctness aids.

An example of a mechanized reference work is a thesaurus program designed so that when the writer designates a word, a list of synonyms appears on the display screen. In advanced systems the thesaurus is fully integrated into the word-processing program. The writer positions a marker to indicate the word to be replaced. The thesaurus is then invoked; it displays the alternatives in a "window" on the screen. The writer positions the marker on one of the alternatives, which automatically replaces the rejected word.

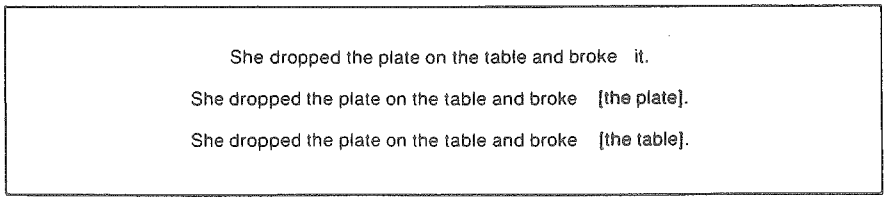
The design of such a program involves both linguistic and computational issues. A linguistic issue is that the mechanism for looking up a word should be flexible enough to accept variant forms. For example, the store of information pertaining to "endow" should be accessible to queries about "endowed," "endowing," "endows" and even "unendowed" or "endowment." Recognizing the common root in such words calls for a morphological analysis, which can be done by techniques developed in the course of work on machine translation. Computational issues include devising methods for storing and searching through a thesaurus or a dictionary, which must be fairly large to be useful.

A correctness aid deals with spelling, grammar and even elements of style. The simplest such programs attempt to match each word in a text with an entry in a stored dictionary. Words that have no match are flagged as possible misspellings. Other programs look for common grammatical errors or stylistic infelicities. For example, the Writer's Workbench software developed at AT&T Bell Laboratories includes programs that search for repeated words, such as "the the" (a common typing mistake), for incorrect punctuation such as "?." and for wordy phrases such as "at this point in time." A different correctness aid calls attention to "pompous phrases" such as "exhibit a tendency" and "arrive at a decision" and suggests simpler replacements such as "tend" and "decide." Still another correctness aid searches for gender-specific terms such as "mailman" and "chairman" and suggests replacements such as "mail carrier" and "chairperson."

In addition to searching a text for particular strings of characters, some correctness-aid programs do statistical analyses. By calculating the average length of sentences, the length of words and similar quantities, they compute a "readability index." Passages that score poorly can be brought to the writer's attention. No program is yet able to make a comprehensive grammatical analysis of a text, but an experimental system called Epistle, developed at the International Business Machines Corporation, makes some grammatical judgments. It employs a grammar of 400 rules and a dictionary of 130,000 words. As with all software that tries to parse text without dealing with what the text means, there are many sentences that cannot be analyzed correctly.



SEMANTIC AMBIGUITY arises when a phrase can play different roles in the meaning of a sentence. Here the roles of the phrase "a Norwegian" become explicit when the sentence "David wants to marry a Norwegian" is "translated" into a logical form based on the notation called predicate calculus. According to one interpretation, the speaker of the sentence has a particular person in mind and has chosen nationality as a way to specify who. Hence the sentence means: There exists (∃) an *x* such that *x* is Norwegian and (∧) *x* is the person David wants to marry. According to another interpretation, neither David nor the speaker has any particular person in mind. David might be going to Norway hoping to meet someone marriageable.



PRAGMATIC AMBIGUITY arises when a sentence is given more than one possible meaning by a word such as the pronoun "it." Suppose a computer is given the sentence shown in the illustration. If the computer has access to stored knowledge of the grammar of English sentences but lacks access to commonsense knowledge of the properties of tables and plates, the computer could infer with equal validity that the table was broken or that the plate was broken.

Is there software that really deals with meaning—software that exhibits the kind of reasoning a person would use in carrying out linguistic tasks such as translating, summarizing or answering a question? Such software has been the goal of research projects in artificial intelligence since the mid-1960's, when the necessary computer hardware and programming techniques began to appear even as the impracticability of machine translation was becoming apparent. There are many applications in which the software would be useful. They include programs that accept natural-language commands, programs for information retrieval, programs that summarize text and programs that acquire language-based knowledge for expert systems.

No existing software deals with meaning over a significant subset of English; each experimental program is based on finding a simplified version of language and meaning and testing what can be done within its confines. Some investigators see no fundamental barrier to writing programs with a full understanding of natural language. Others argue that computerized understanding of language is impossible. In order to follow the arguments it is important to examine the basics of how a language-understanding program has to work.

A language-understanding program needs several components, corresponding to the various levels at which language is analyzed [*see illustrations on pages 138–144*]. Most programs deal with written language; hence the analysis of sound waves is bypassed and the first level of analysis is morphological. The program applies rules that decompose a word into its root, or basic form, and inflections such as the endings *-s* and *-ing*. The rules correspond in large part to the spelling rules children are taught in elementary school. Children learn, for example, that the root of "baking" is "bake," whereas the root of "barking" is "bark." An exception list handles words to which the rules do not apply, such as forms of the verb "be." Other rules associate inflections with "features" of words. For example, "am going" is a progressive verb: it signals an act in progress.

For each root that emerges from the morphological analysis a dictionary yields the set of lexical categories to which the root belongs. This is the second level of analysis carried out by the computer. Some roots (such as "the") have only one lexical category; others have several. "Dark" can be a noun or

a

```
\inset
This is a sample of a {\italic justified} piece of text, which contains {\eightpoint small letters {\bold and }} {bigFont big ones}.
It includes foreign words such as \lquote pe\~na\rquote—which is Spanish—and foreign letters like \alpha\ and \aleph,
which can be baffling, and includes one \hskip 1.3in wide space.
```

b

01110100	01100101	01110010	01110011	00000000	00100111	00101101	11010011	00001000	01100001	01101110	01100100	00000000
t	e	r	s	NEW ENTITY	FONT CODE	X-POSITION	Y-POSITION	X-INCREMENT	a	n	d	NEW ENTITY

00110100	00110001	10110110	00101101	01100010	01101001	01100111	00100000	01101111	01101110	01100101	01101011	00101110
FONT CODE	X-POSITION	Y-POSITION	X-INCREMENT	b	i	g	SPACE	o	n	e	s	.

00000000	00000001	10101111	10110110	00101100	01001001	01110100	00100000	01101001	...
NEW ENTITY	FONT CODE	X-POSITION	Y-POSITION	X-INCREMENT	I	t	SPACE	i	

c

This is a sample of a *justified* piece of text, which contains small letters **and** big ones. It includes foreign words such as "peña"—which is Spanish—and foreign letters like α and $\aleph$, which can be baffling, and includes one wide space.

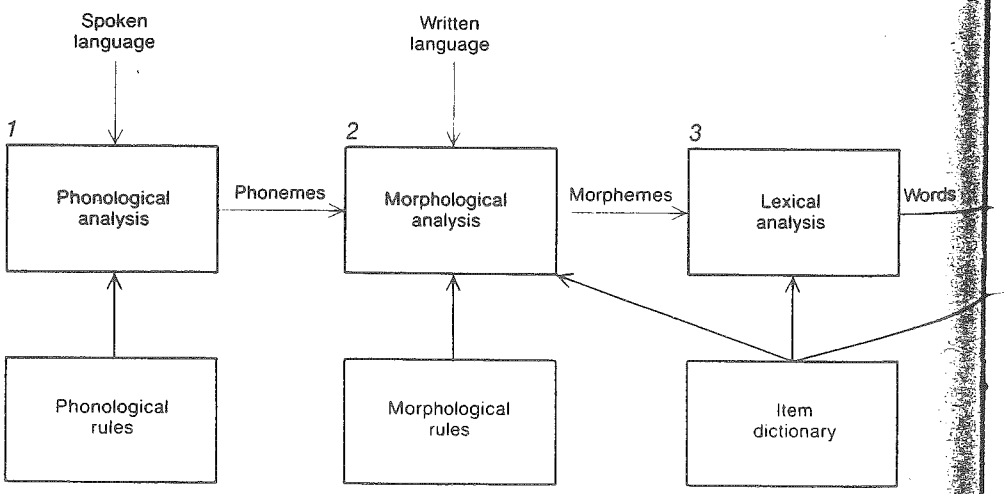
WORD PROCESSING, that is, the computer-aided preparation and editing of text, requires several representations of the text, because the format best for interactions between the software and its user is not efficient for sending instructions to a printing machine, nor can it efficiently give a preview of the result of the printing. In the typesetting language TEX the user's typed input (*a*) includes commands that specify nonstandard characters, change the style of type, set margins and so on. Such commands are distinguished by a backslash (\). The TEX software "compiles" the input, producing computer code that will drive a printing machine (*b*). To that end the code is divided into "entities," each of which specifies the typeface and the starting position for a sequence of words. Coded "*X* increments" space out the words to fill the distance between margins on the printed page; thus they "justify" lines of type. The printed page (*c*) shows the result.

an adjective; "bloom" can be a noun or a verb. In some instances the morphological analysis limits the possibilities. (In its common usages "bloom" can be a noun or a verb, but "blooming" is only a verb.) The output of the morphological and lexical analysis is thus a sequence of the words in a sentence, with each word carrying a quantity of dictionary and feature information. This output serves in turn as the input to the third component of the program, the parser, or syntactic-analysis component, which applies rules of grammar to determine the structure of the sentence.

Two distinct problems arise in designing an adequate parser. The first problem is the specification of a precise set of rules—a grammar—that determines the set of possible sentence structures in a language. Over the past 30 years much work in theoretical linguistics has been directed toward devising formal linguistic systems: constructions in which the syntactic rules of a language are stated so precisely that a computer could employ them to analyze the language. The generative transformational grammars invented by Noam Chomsky of the Massachusetts Institute of Technology were the first comprehensive attempt; they specify the syntax of a language by means of a set of rules whose mechanical application generates all allowable structures.

The second problem is that of the parsing itself. It is not always possible to tell, when a part of a sentence is encountered, just what role it plays in the sentence or whether the words in it go together. Take the sentence "Roses will be blooming in the dark gardens we abandoned long ago." The words "in the dark" might be interpreted as a complete phrase; after all, they are grammatically well formed and they make sense. But the phrase cannot form a coherent unit in a complete analysis of the sentence because it forces "Roses will be blooming in the dark" to be interpreted as a sentence and therefore leaves "gardens we abandoned long ago" without a role to play.

Parsers adopt various strategies for exploring the multiple ways phrases can be put together. Some work from the top down, trying from the outset to find possible sentences; others work from the bottom up, trying local word combinations. Some backtrack to explore alternatives in depth if a given possibility fails; others use parallel processing



COMPUTERIZED UNDERSTANDING OF LANGUAGE requires the computer to draw on several types of stored data (*white boxes*) and perform several levels of analysis (*colored boxes*). If the language is spoken, the first analysis is phonological (*1*): the computer analyzes sound waves. If the language is written, the first analysis is morphological (*2*): the computer decomposes each word into its root, or basic form, and inflections (for example *-ing*). Next is lexi-

to keep track of a number of alternatives simultaneously. Some make use of formalisms (such as transformational grammar) that were developed by linguists. Others make use of newer formalisms designed with computers in mind. The latter formalisms are better suited to the implementation of parsing procedures. For example, "augmented-transition networks" express the structure of sentences and phrases as an explicit sequence of "transitions" to be followed by a machine. "Lexical-function grammars" create a "functional structure" in which grammatical functions such as head, subject and object are explicitly tied to the words and phrases that serve those functions.

Although no formal grammar successfully deals with all the grammatical problems of any natural language, existing grammars and parsers can handle well over 90 percent of all sentences. This is not entirely to the good. A given sentence may have hundreds or even thousands of possible syntactic analyses. Most of them have no plausible meaning. People are not aware of considering and rejecting such possibilities, but parsing programs are swamped by meaningless alternatives.

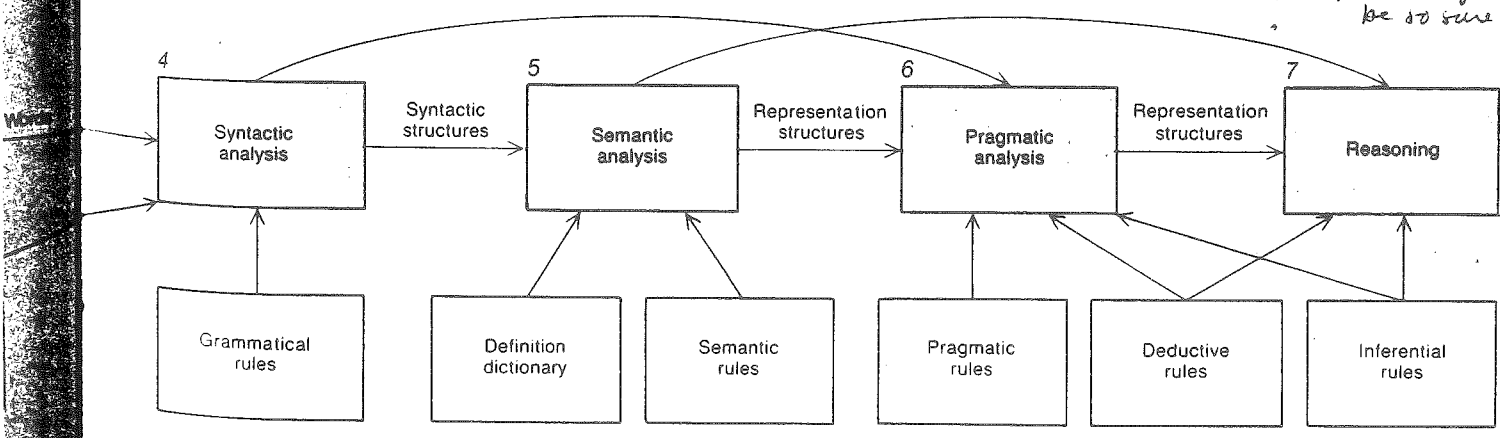
The output of a parsing program becomes the input to the fourth component of a language-understanding program: a semantic analyzer, which translates the syntactic form of a sentence into a "logical" form. The point is to put the linguistic expressions into a form that makes it possible for the computer to apply reasoning procedures and draw inferences. Here again there are competing theories about what representation is most appropriate. As with parsing, the key issues are effectiveness and efficiency.

Effectiveness depends on finding the appropriate formal structures to encode the meaning of linguistic expressions. One possibility is predicate calculus, which employs the quantifiers $\forall$ to mean "all" and $\exists$ to mean "there exists." In predicate calculus "Roses will be blooming..." is equivalent to the assertion "There exists something that is a rose and that is blooming...." This entails a difficulty. Is one rose adequate to represent the meaning of "roses will be blooming," or would it be better to specify two or more? How can the computer decide? The dilemma is worsened if a sentence includes a mass noun such as "water" in "Water will be flowing...." One cannot itemize water at all. In designing a formal structure for the meaning of linguistic expressions many similar problems arise from the inherent vagueness of language.

Efficiency must also be considered, because the computer will employ the logical form of a sentence to draw inferences that in turn serve both the analysis of the meaning of the sentence and the formulation of a response to it. Some formalisms, such as predicate calculus, are not directly amenable to efficient computation, but other, more "procedural" representations have also been devised. Consider the effort to answer the question "Are there flowers in the gardens we abandoned long ago?" The computer needs to know that roses are flowers. This knowledge could be represented by a formula in predicate calculus amounting to the assertion "Everything that is a rose is a flower." The computer could then apply techniques developed for mechanical theorem-proving to make the needed deduction. A different approach would be to give certain inferences a privileged computational status. For example, basic classificational deductions could be represented directly in data structures [*see bottom illustration on page 144*]. Such deductions are required constantly for reasoning about the ordinary properties of objects. Other types of fact (for example that flowers need water in order to grow) could then be represented in a form closer to predicate calculus. The computer could draw on both to make inferences (for example that if roses do not get water, they will not grow).

A good deal of research has gone into the design of "representation languages" that provide for the effective and efficient encoding of meaning. The greatest difficulty lies in the nature of human commonsense reasoning. Most of what a person knows cannot be formulated in all-or-nothing logical rules; it lies instead in "normal expectations." If one asks, "Is there dirt in the garden?" the answer is almost certainly yes. The yes, however, cannot be a logical inference; some gardens are hydroponic, and the plants there grow in water. A person tends to rely on normal expectations without thinking of exceptions unless they are relevant. But little progress has been made toward formalizing the concept of "relevance" and the way it shapes the background of expectations brought to bear in the understanding of linguistic expressions.

The final stage of analysis in a language-understanding program is pragmatic analysis: the analysis of context. Every sentence is embedded in a setting: it comes from a particular speaker at a particular time and it refers, at least implicitly, to a particular body of understanding. Some of the embedding is straightforward: the pronoun "I" refers to the speaker; the adverb "now" refers to the moment at which the sen-



cal analysis (*3*), in which the computer assigns words to their lexical category (noun, for instance) and identifies "features" such as plurals. Then comes syntactic analysis, or parsing (*4*): the application of rules of grammar to yield the structure of the sentence. After that comes semantic analysis (*5*). Here the sentence is converted into a form that makes it amenable to inference-drawing. The final stage is pragmatic (*6*): it makes explicit the context of the sentence, such as the relation between the time at which it is spoken and the time to which it refers. The computer is now in a position to draw inferences (*7*), perhaps in preparation for responding to the sentence.

tence is uttered. Yet even these can be problematic: consider the use of "now" in a letter I write today expecting you to read it three or four days hence. Still, fairly uncomplicated programs can draw the right conclusion most of the time. Other embedding is more complex. The pronoun "we" is an example. "We" might refer to the speaker and the hearer or to the speaker and some third party. Which of these it is (and who the third party might be) is not explicit and in fact is a common source of misunderstanding when people converse.

Still other types of embedding are not signaled by a troublesome word such as "we." The sentence "Roses will be blooming..." presupposes the identification of some future moment when the roses will indeed be in bloom. Thus the sentence might have followed the sentence "What will it be like when we get home?" or "Summer is fast upon us." Similarly, the noun phrase "the dark gardens we abandoned long ago" has a context-dependent meaning. There may be only one instance of gardens in which we have been together; there may be more than one. The sentence presupposes a body of knowledge from which the gardens are identifiable. The point is that a phrase beginning with "the" rarely specifies fully the object to which it refers.

One approach to such phrases has been to encode knowledge of the world in a form the program can use to make inferences. For example, in the sentence "I went to a restaurant and the waiter was rude" one can infer that "the waiter" refers to the person who served the speaker's meal if one's knowledge includes a script, so to speak, of the typical events attending a meal in a restaurant. (A particular waiter or waitress serves any given customer.) In more complex cases an analysis of the speaker's goals and strategies can help. If one hears "My math exam is tomorrow, where's the book?" one can assume that the speaker intends to study and that "the book" means the mathematics text employed in a course the speaker is taking. The approach is hampered by the same difficulty that besets the representation of meaning: the difficulty of formalizing the commonsense background that determines which scripts, goals and strategies are relevant and how they interact. The programs written so far work only in highly artificial and limited realms, and it is not clear how far such programs can be extended.

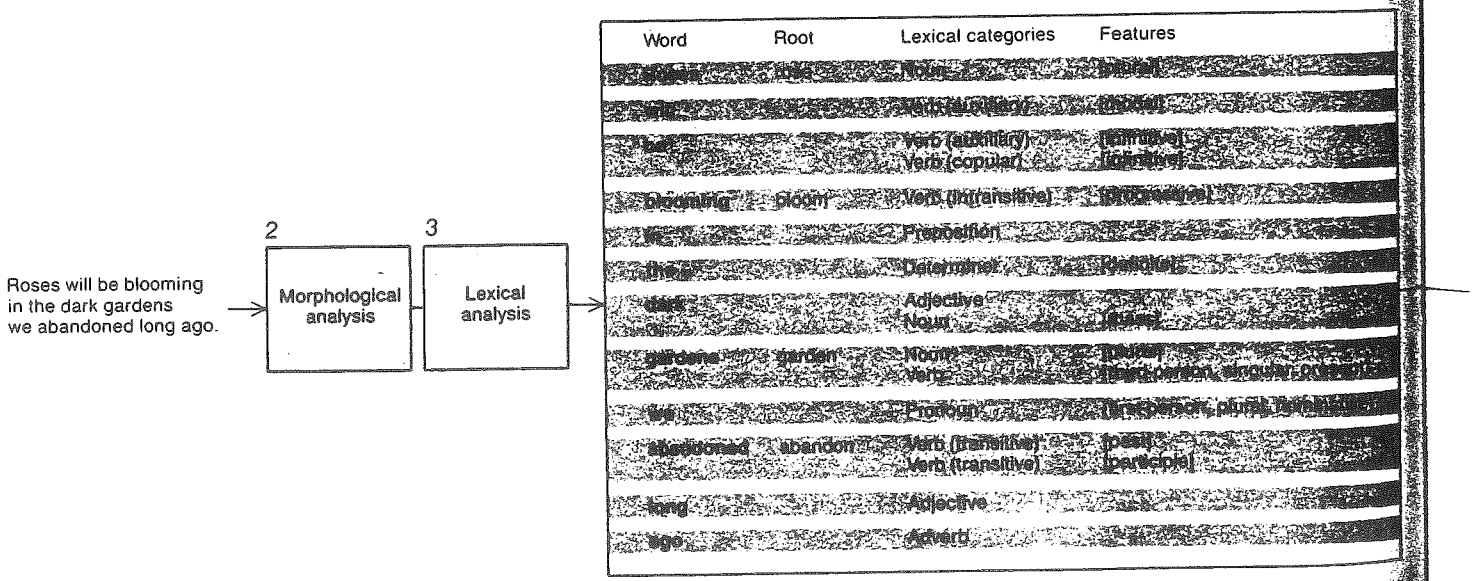
Even more problematic are the effects of context on the meaning of words. Suppose that in coming to grips with "the dark gardens we abandoned long ago" one tries to apply a particular meaning to "dark." Which should it be? The "dark" of "those dark days of tribulation" or that of "How dark it is with the lights off!" or that of "dark colors"? Although a kernel of similarity unites the uses of a word, its full meaning is determined by how it is used and by the prior understanding the speaker expects of the hearer. "The dark gardens" may have a quite specific meaning for the person addressed; for the rest of us it is slightly mysterious.

At first it might seem possible to distinguish "literal" uses of language from those that are more metaphorical or poetical. Computer programs faced with exclusively literal language could then be freed from contextual dilemmas. The problem is that metaphor and "poetic meaning" are not limited to the pages of literature. Everyday language is pervaded by unconscious metaphor, as when one says, "I lost two hours trying to get my idea across." Virtually every word has an open-ended field of meanings that shade gradually from those that seem utterly literal to those that are clearly metaphorical.

The limitations on the formalization of contextual meaning make it impossible at present—and conceivably forever—to design computer programs that come close to full mimicry of human language understanding. The only programs in practical use today that attempt even limited understanding are natural-language "front ends" that enable the user of a program to request information by asking questions in English. The program responds with English sentences or with a display of data.

A program called SHRDLU is an early example. Developed in the late 1960's, it enables a person to communicate with a computer in English about a simulated world of blocks on a tabletop. The program analyzes requests, commands and statements made by the user and responds with appropriate words or with actions performed in the simulated scene. SHRDLU succeeded in part because its world of conversation is limited to a simple and specialized domain: the blocks and a few actions that can be taken with them.

Some more recent front-end interfaces have been designed with practical applications in mind. A person wanting access to information stored in the computer types natural-language sentences



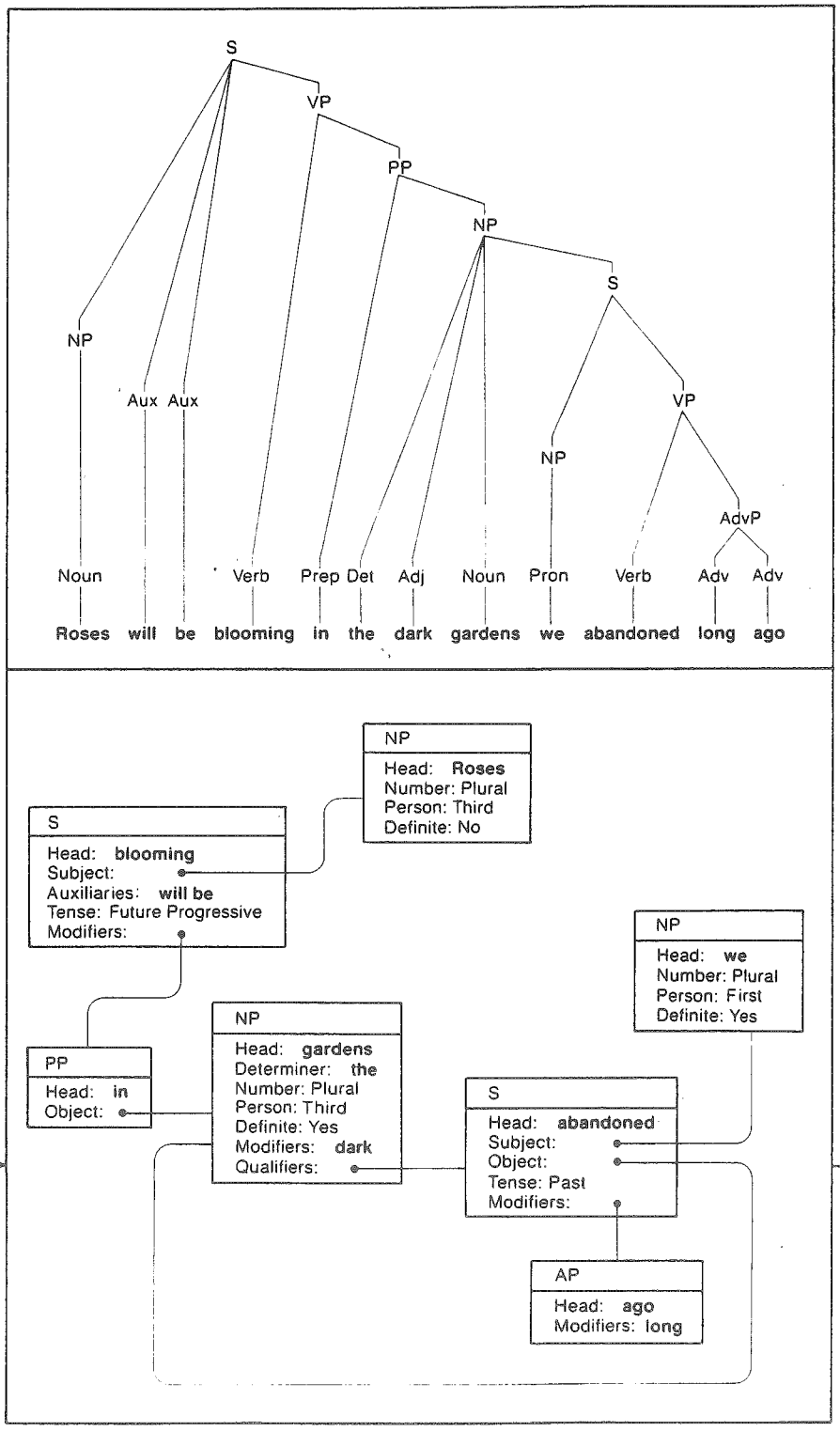
SUCCESSION OF ANALYSES done by a hypothetical computer program suggests how software that understands language works. In this illustration the program has been given the sentence "Roses will be blooming in the dark gardens we abandoned long ago." The first analyses (morphological and lexical) yield a list of the words in the sentence, with their roots, their lexical categories and their features. "Blooming," for instance, is a progressive verb: it signifies an act in progress. The data serve as input for the syntactic level of analysis: the parsing of the sentence. Here the surface, or grammatical, structure of "Roses will be blooming..." is put in the form of a tree. Pre-

that the computer interprets as queries. The range of the questioning is circumscribed by the range of the data from which answers are formulated; in this way words can be given precise meaning. In a data base on automobiles, for example, "dark" can be defined as the colors "black" and "navy" and nothing more than that. The contextual meaning is there, but it is predetermined by the builder of the system, and the user is expected to learn it.

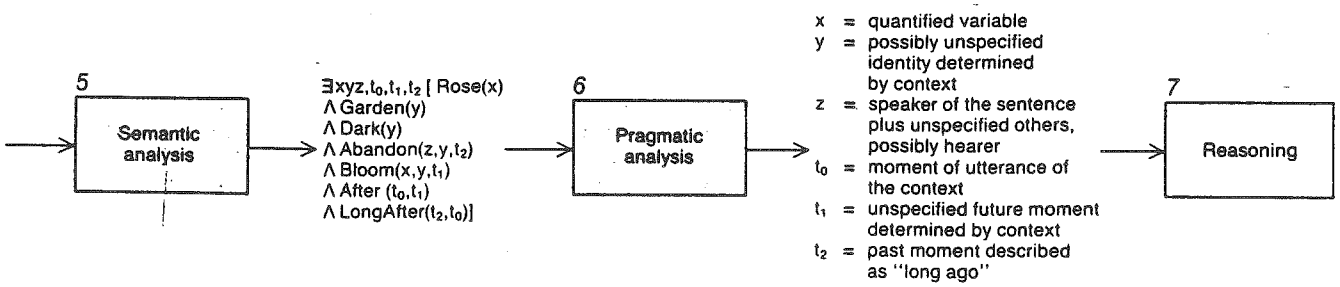
The main advantage of a natural-language front end is that it presents a low initial barrier to potential users. Someone invited to pose a question in English is usually willing to try, and if the computer proves unable to handle the specific form of the question, the user is probably willing to modify the wording until it works. Over time the user will learn the constraints imposed by the system. In contrast, a person who must learn a specialized language in order to formulate a question may well feel that an inordinate amount of work is being demanded.

I want finally to look at a rather new type of system called a coordinator. In brief it replaces standard electronic mail with a process that aids the generation of messages and monitors the progress of the resulting conversations. Coordinators are based on speech-act theory, which asserts that every utterance falls into one of a small number of categories. Some speech acts are statements: "It's raining." Some are expressive: "I'm sorry I stepped on your toe." Some are requests: "Please take her the package" or "What is your name?" Some are commitments: "I'll do it tomorrow." Some are declarative: "You're fired." (Declaratives differ from statements in that they take effect by virtue of having been said.)

The classification of speech acts is useful because acts in the various categories do not occur at random. Each speech act has "felicity conditions" under which it is an appropriate thing to say and "conditions of satisfaction" under which it is fulfilled. For example, a request or a commitment carries with it, either implicitly or explicitly, a time by which it should be satisfied. Moreover,



sumably the computer discards numerous incorrect trees. For example, it discards a tree in which "Roses will be blooming in the dark" is construed as a sentence. The deep structure of "Roses will be blooming . . . " is put in the form of a functional-structure diagram. There the relations between the parts of a sentence become explicit; they are shown by strings between boxes. Some relations were explicit in the surface structure (for example that "roses" is the subject of "blooming"). Others were not (for example that "gardens" is the object of "abandoned"). The syntactic analysis is supplied to the final stages of the program, which appear in the top illustration on the next page.



ANALYSES CONCLUDE with the conversion of the syntactic structure of "Roses will be blooming..." into a form from which the computer can draw inferences. In this example the conversion is based on predicate calculus; thus the semantic-analysis module of the hypothetical software represents the logical content of "Roses will be blooming..." by symbols that can be translated as "*x* is a rose and *y* is a garden and *y* is dark...." Finally, the pragmatic-analysis module specifies what is known about the variables *x*, *y*, *z*, t_0, t_1 and t_2. The variable *x*, for example, is "quantified": it declares the existence of something instead of identifying a particular object. In other words, the computer takes "roses" as referring to roses in general, not to particular roses. Hence roses is not a "definite" noun. (That decision was made in the course of semantic analysis.) On the other hand, *z* remains ambiguous because it stands for the ambiguous pronoun "we."

each speech act is part of a conversation that follows a regular pattern. The regularity is crucial for successful communication.

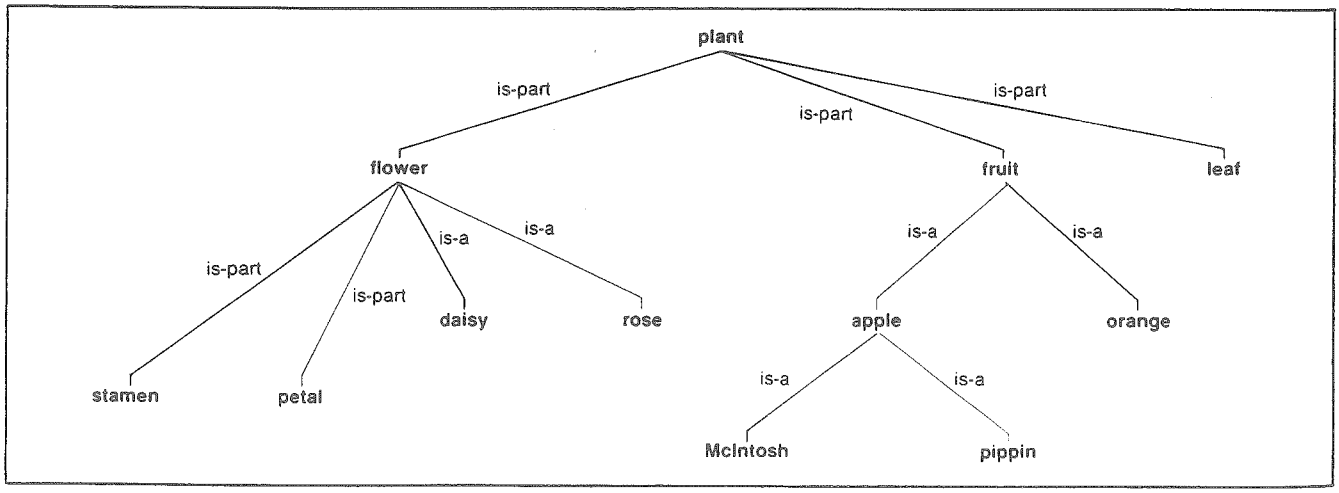
As with every aspect of language, the full understanding of any given speech act is always enmeshed in the unarticulated background expectations of the speaker and the hearer. The speech act "I'll be here tomorrow" might be a prediction or a promise, and "Do you play tennis?" might be a question or an invitation. In spoken conversation intonation and stress play a prominent part in establishing such meaning.

Coordinator systems deal with the speech acts embodied in messages by specifying what needs to be done and when. The system does not itself attempt to analyze the linguistic content of messages. Instead the word-processing software at the sender's end asks the sender to make explicit the speech-act content of each message. A person may write "I'll be happy to get you that report" in the message itself but must add (with a few special keystrokes) that the message is an ACCEPT of a particular REQUEST. The computer system can then keep track of messages and their interconnections. In particular the system can monitor the completion of conversations, calling the users' attention to cases in which something immediate is pending or in which an agreed-on time for satisfaction has not been met.

From a broad perspective, coordinators are just one member of a large family of software that gives users a structured medium in which language is augmented by explicit indications of how things fit together. Another type of software in this family provides tools for outlining and cross-indexing documents. Still another type is a computerized bulletin board that enables users to store and receive messages not addressed to a specific receiver. The messages are "posted" with additional structure that indicates their content and helps interested readers to find them.

The most obvious prediction about the future of computer software dealing with language is that the decreasing cost of hardware will make applications that are possible but impractical today available quite widely in the future. Yet software that mimics the full human understanding of language is simply not in prospect. Some specific trends can be noted.

The first is that spoken language will get more emphasis. To be sure, the computerized understanding of spoken language presents all the difficulties of written language and more. Merely separating an utterance into its component words can vex a computer; thus hopes for a "voice typewriter" that types text from dictation are just as dim as hopes for high-quality machine translation and language-understanding. On the other hand, many useful devices do not require the analysis of connected speech. Existing systems that can identify a spoken word or phrase from a fixed vocabulary of a few hundred items will improve the interface between users and machines; the recent emergence of inexpensive integrated-circuit chips that



SEMANTIC NETWORK is a specialized form of stored data that represents logical relations so that certain types of inference can be drawn efficiently by a computer. Here a simple tracing of links in the network (*color*) has yielded the inference that a pippin is a fruit and that a rose has petals. Facts not readily represented by a network can be represented in other ways, for example by predicate calculus.

process acoustic signals will facilitate the trend. Speech synthesizers that generate understandable utterances (although not in a natural-sounding voice) will also play an increasing role. Improved speech "compression" and encoding techniques will make acoustic messages and acoustic annotation of computer files commonplace.

A second trend in software dealing with language is that constraints on linguistic domain will be handled with increasing care and theoretical analysis. At several points in this article I have noted instances in which computers deal with meaning in an acceptable way because they operate in a limited domain of possible meanings. People using such software quickly recognize that the computer does not understand the full range of language, but the subset available is nonetheless a good basis for communication. Much of the commercial success of future software that deals with language will depend on the discovery of domains in which constraints on what sentences can mean still leave the user a broad range of language.

A third trend lies in the development of systems that combine the natural and the formal. Often it is taken for granted that natural language is the best way for people to communicate with computers. Plans for a "fifth generation" of intelligent computers are based on this proposition. It is not at all evident, however, that the proposition is valid. In some cases even the fullest understanding of natural language is not as expressive as a picture. And in many cases a partial understanding of natural language proves to be less usable than a well-designed formal interface. Consider the work with natural-language front ends. Here natural language promotes the initial acceptance of the system, but after that the users often move toward stylized forms of language they find they can employ with confidence, that is, without worrying about whether or not the machine will interpret their statements correctly.

The most successful current systems facilitate this transition. Some systems (including coordinators) mix the natural and the formal: the user is taught to recognize formal properties of utterances and include them explicitly in messages. Thus the computer handles formal structures, while people handle tasks in which context is important and precise rules cannot be applied. Other systems incorporate a highly structured query system, so that as the user gains experience the artificial forms are seen to save time and trouble. In each case the computer is not assigned the difficult and open-ended tasks of linguistic analysis; it serves instead as a structured linguistic medium. That is perhaps the most useful way the computer will deal with natural language.

Generate: Sentences

Robert G. Hackenberg

ABSTRACT

GENERATE is a simple program designed to help the beginning linguistics student understand the rules and processes of transformational generative grammar. This program uses only a small lexicon and a subset of phrase structure and transformation rules. Many problems and possibilities involving the CAI and TG interface (the algorithm) are pointed out.

Several options are offered to the student during the program. One option shows the process of selecting phrase structure rules, transformations, lexical items etc., to generate surface structure strings. The program utilizes a Software Activated Mouth (S.A.M.) for pronunciation of the generated sentences.

The chief purpose of generate is to be instructional and illustrative of transformational-generative grammar rules and processes.

The original program was written for The Radio Shack TRS-80 Model III. The present program runs on a Commodore 64.

KEYWORDS: GENERATE, software, program, transformational-generative grammar, artificial intelligence, Radio Shack, Commodore 64, S.A.M., software activated mouth.

Beginning students of linguistics are typically mystified by the rigorous formalism of transformational-generative grammar. They are taught that the grammar must be able to account for the production of all the valid sentences of the language and not permit the generation of any invalid sentences.

In order to help my student's get a better feel of TG grammar, I developed a simple program called GENERATE, which produces sentences using a small lexicon and a subset of phrase structure and transformation rules. The exercise of causing a computer to implement grammar rules is invaluable. The computer is mercilessly unwavering in its execution of the rules contained in an algorithm. If ungrammatical sentences result, the problem is with the algorithm and not with the computer. A human walking through transformational grammare (TG) rules tends to avoid difficulties and gloss over minor problems. A grammar should be able to produce all the valid sentences of the language and no invalid ones. Because GENERATE uses an extremely small subset of the grammar of English, it would not be expected to produce all possible sentences. Nevertheless, every sentence that it does generate should be acceptable to a native speaker of English. The program uses twenty verbs and twenty nouns to randomly construct strings which are grammatical in themselves and which bear no discursive relationship to each other.

What follows is a description of the program and the way that it is normally presented to a class of linguistics students. It is not intended to be a working component of some natural language interface. Its chief purpose is to be instructional and illustrative.

The original program was written to run on Radio Shack's TRS-80 Model III and was designed to merely spew out hundreds of sentences to either the screen or to a printer. The current version runs on the Commodore 64 and takes advantage of that machine's graphics and sound synthesis. Instead of just listing the linguistic productions, it walks the user through the process, illustrating the steps being taken and showing the growth from an initial symbol S in the deep structure to the final surface string which is both written out on the screen and spoken by the synthesizer.

Dr. Robert G. Hackenburg is a Senior Member of the Engineering Staff at RCA's Artificial Intelligence Laboratory in Camden. He has a B.A. in French and Spanish, an M.A. in TESL, an M.S. in Computer Science, and a Ph.D. in Linguistics. He has taught twelve years at the university level and is now doing work on the natural language interfaces at RCA.

THE PROGRAM

GENERATE begins with the spoken introduction shown in Figure 1. The synthesized speech is produced by a

software utility called S.A.M. (Software Activated Mouth). The reciter mode of S.A.M. is used to process the normal orthographic representation of English sentences and generate speech through the sound interface device (SID) chip of the Commodore 64. This introduction is marginally understandable on first hearing but users have typically reacted that their understanding improves with exposure to the voice. The reciter mode uses 450 rules to pronounce English words and is remarkably accurate. Predictably, words that do not have a regular spelling do not get pronounced correctly, a situation that can be generally remedied by an intentional misspelling.

Welcome. The following is a demonstration of GENERATE, a computer program that randomly generates English sentences. It utilizes a small set of phrase structure rules and transformation rules to form valid sentence strings. The dictionary consists of twenty verbs and twenty nouns. Both the nouns and the verbs are semantically coded to assure that invalid strings such as The building smoked a cigar will never be produced. The voice you are hearing is being generated by a package called SAM. SAM is an acronym for Software Activated Mouth. It is used to drive the standard SID chip in the Commodore 64. This voice will also say each sentence as it is randomly generated.

Figure 1: Synthesized Introduction

After the introduction has been completed, the menu in Figure 2 appears. It is from this menu that the user branches to the various modules that are available. When I present this program to a group, I normally choose Option 1 first to show the phrase structure rules. After that, I present the semantic coding (Option 5) and the structure of the lexicon (Options 3 and 4). Option 6 represents the algorithm the program follows to randomly generate sentences. After the preliminaries have been taken care of, I proceed to Option 2, the business end of the program. This module of the program works through the process of selecting phrase structure rules and transformations to generate a series of intermediate strings, pausing after each step to show the user the current state of the string. After the surface structure string is printed on the screen, S.A.M. pronounces it using stored phonemic representations. The user can either continue observing sentences being generated or can request a return to the main menu.

1. Show phrase structure rules
2. Generate sentences
3. Show lexicon of nouns
4. Show lexicon of verbs
5. Show semantic codes
6. Show rules base
7. Terminate program

Figure 2: Program Options

THE GRAMMAR

A full grammar consists of phrase structure rules, transformation rules, and a lexicon with appropriate semantic coding. GENERATE permits the user to view through a series of screens the phrase structure rules and the lexicon. Although transformations are applied throughout, only a few of them are shown to the user.

Phrase Structure Rules

The five phrase structure rules of the grammar are listed in Figure 3. The rules for the most part are fairly standard and have appeared as such in at least some T-G grammars. The use of COMPLEMENT in Rule 3 is certainly not standard and any further refinement of the program would require that it be given immediate attention. As is explained in the parenthetical note, the COMPLEMENT represents any noun phrase or adverbial that could legally follow the verb. As the program stands, each verb has a specific direct object or adverbial assigned to it and this complement appears each time the particular verb is selected. This route was taken in order to facilitate the running of other parts of the program. The solution of the complement problem is not a trivial matter and will require a considerable amount of thought as to how the lexicon will be coded.

Not all options in the phrase structure rules are available to the current program, again because the program is in a developmental stage. The pronoun in Rule 2 is never chosen and specified as the. In fact, each time Rule 2 fires, the and a noun are always generated. An interesting exercise would be to develop the use of the optional S that follows the noun. It is this S that provides powerful recursion and is the source of all adjective clauses, prenominal adjectives, and verbals which serve an adjectival function. The transformation rules that operate on it are fairly straight-forward and invariable. The S on the third line of Rule 2 is the source of all noun clauses and verbals serving a nominal function and it too would be relatively easy to work into the program.

The AUX in Rule 4 is given the greatest amount of attention by the program. The English auxiliary is extremely regular and is the source of the tense of the sentence and provides the helping verbs. The only component of

1. S—NP + AUX + VP

2. NP— (DET) + { PRO / NOUN } + (S) / { S }

3. VP— VERB + 'COMPLEMENT'
(Note: 'COMPLEMENT' is a simplification for the program and is used to include all adverbials and direct objects, indirect objects, and subjective complements

4. —T + (M) + (HAVE + PP) + (BE + ING)
Note: M MODAL, PP PAST PART, ING - PRES PART

5. T — { PRESENT / PAST }

6. M— MAY, CAN, SHALL, WILL, MUST

Figure 3

the auxiliary that is required for a finite verb is tense. The other components provide a wealth of information about the mode and aspect of the sentence. English has generally lost the use of the subjunctive and the modals help to supply what was lost. The HAVE auxiliary provides for the aspect of anteriority and the BE provides for progressivity.

Transformations

Transformations are necessary throughout the program to develop surface structure strings, but for the most part these are transparent to the user. Figure 4 shows the screen that follows the phrase structure rules. What are given are not rules themselves but the results of the application of certain affix rules. These are given to help the user understand how the flip-flop transformation later referenced in the program operates.

Lexicon

There are actually two lexicons referenced by the program, one for nouns (Figure 5) and one for verbs (Figure 6). These provide both the information needed by the program to generate the appropriate surface spellings as well as a semantic code that is used to ensure the production of valid sentences. Also part of the lexicon, but not shown to the user, is the phonological information that S.A.M. uses to pronounce the sentences.

Morphological information is presented differently in the two lexicons to illustrate two different methods which can be used. Whereas the verb lexicon lists the full set of forms that can occur for each lexical entry, the noun lexicon employs a symbolic representation. A 1 in the plural field indicates that s is added to the base and a 2 indicates that es is added. If the noun is irregular, then the full spelling of the plural noun is given. On the other hand, the verb lexicon gives the full spelling for each of the verb forms. Each system has advantages and disadvantages. The advantage of morphological coding is that it provides a considerable savings in the amount of memory needed to store the lexicon. The disadvantage is that extra code must be executed to generate the appropriate forms. There is no question, however, that morphological coding is preferable, given the potentially vast storage needs of a full lexicon.

The values found in the semantic code fields of the nouns and verbs are used to insure the valid combinations of subject and verb. As can be seen in Figure 7, three distinctive semantic features are used to establish four classes of nouns. The classes comprise humans(4), animals (i.e., non-human animals)(3), plants(2), and all other nonliving concrete nouns(1). The numerical codes used to identify these classes are assigned to form an implicational scale that is used in the program algorithm. Once a verb has been randomly chosen, the program searches sequentially through the list of nouns until it finds one that has a code that is equal to or greater than the semantic code of the verb. For example, if see (with a code of 3) is selected as the verb, then a search is made to find a noun subject that all animals and humans can see. For the purpose of the algorithm, the critical feature of a verb code of 3 is $\geq$ + ANIMAL.

Although this classification system works well in the limited scope of the program, it is hardly adequate for a real-world application. Most verbs do not conform to the implication that if one form of life can be the subject,

	Present	Past
can	can	could
may	may	might
shall	shall	should
will	will	would
must	must	XXXXX
be	am	was
	is	were
	are	
have	has	had
	have	

Figure 4

	Semantic code	Plural Code
dog	3	1
boy	4	1
rock	1	1
tree	2	1
child	4	children
bug	3	1
sled	1	1
captain	4	1
finger	1	1
ox	3	oxen
apple	1	1
box	1	2
mouse	3	mice
rosebush	2	2
robin	3	1
president	4	1
noodle	1	1
faucet	1	1
weed	2	1
cabinet	1	1

For PLURAL, 1 add "s", 2 add "es"

Noun Lexicon

Figure 5

write, writes, writing, wrote, written	4	a letter
buy, buys, bought, bought	4	a book
smoke, smokes, smoking, smoked, smoked	4	a cigar
grow, grows, growing, grew, grown	2	fast
sleep, sleeps, sleeping, slept, slept	3	soundly
play, plays, playing, played, played	3	a game
fall, falls, falling, fell, fell	1	on the roof
die, dies, dying, died, died	2	slowly
sink, sinks, sinking, sank, sunk	1	like a rock
drink, drinks, drinking, drank, drunk	3	a beer
roll, rolls, rolling, rolled, rolled	1	for a mile
see, sees, seeing, saw, seen	3	my sister
open, opens, opening, opened, opened	3	the box
eat, eats, eating, ate, eaten	3	a candy bar
knock, knocks, knocking, knocked, knocked	3	over the trash
witness, witnesses, witnessing, witnessed, witnessed	4	the accident
get, gets, getting, got, gotten	1	wet
blow, blows, blowing, blew, blown	1	away
drown, drowns, drowning, drowned, drowned	2	in the flood
munch, munches, munching, munched, munched	3	on the peanuts

Figure 6

1	2	3	4
-ANIMATE	+ANIMATE	+ANIMATE	+ANIMATE
-ANIMAL	-ANIMAL	+ANIMAL	+ANIMAL
-HUMAN	-HUMAN	-HUMAN	+ANIMAL
"CAR"	"TREE"	"ARMADILLO"	"INSTRUCTOR"

Figure 7

1. Randomly select a verb
2. Scan nouns sequentially until noun-code > verb code
3. Get complement associated with verb
4. Randomly generate:
   - Tense (1 present, 2 past)
   - Modal (each has 1/15 chance)
   - HAVE + PP (1/3 chance)
   - BE + ING (1/chance)
   - Subject number (1 sg, 2 pl)
5. Apply tense to 1st verbal element; turn off tense flag
6. If HAVE + PP, apply PP to next verbal element
7. If BE + ING, apply ING to main verb
8. Print sentence

Figure 8

then all other nouns represent an equal or higher form can also serve as subject. For example, the verb claw might be assigned a code 3, implying that the subject must at least have the feature ≥ + ANIMAL. If snake were one of the possible subject nouns, the meaningless combination the snake clawed... would result. Clearly, the lexicon requires many more semantic features to generate valid sentences.

FURTHER CONSIDERATIONS

GENERATE is in a developmental stage and represents more of an instructional toy than a serious model for a natural language generator. As such, it is being offered as a vehicle for experimentation. One much needed improvement is in the refinement of complement. The selection of noun and adjective complements will require considerable effort in refining the semantic coding of the lexicon. Another challenging and interesting exercise will be the inclusion of embedded sentences in the grammar. The semantic element will play an important role here, as well, but the transformations involved are fairly well understood. The program was written in BASIC because of its ready availability on microcomputers. GENERATE is currently being rewritten in Lisp and Prolog, two AI (Artificial Intelligence) languages which have been used extensively for natural language processing. Although the syntax of these languages are somewhat strange to those who know only the more common algorithmic languages, they are excellent for expressing grammatical rules. The code for expression of phrase structure rules can look almost identical to the TG formulation of the rules themselves. In BASIC they are almost unrecognizable as such.

Although the program was mainly intended to be instructional, it serves a very useful purpose of making the linguist aware of the need for a rigorous grammatical formulation. These same skills are needed in the development of interfaces that would permit a human operator to communicate with the computer using natural language.

Author's Address
RCA Advanced Technology Laboratory
Mail Stop 10-4-5
Front and Cooper Streets
Camden, New Jersey 08102

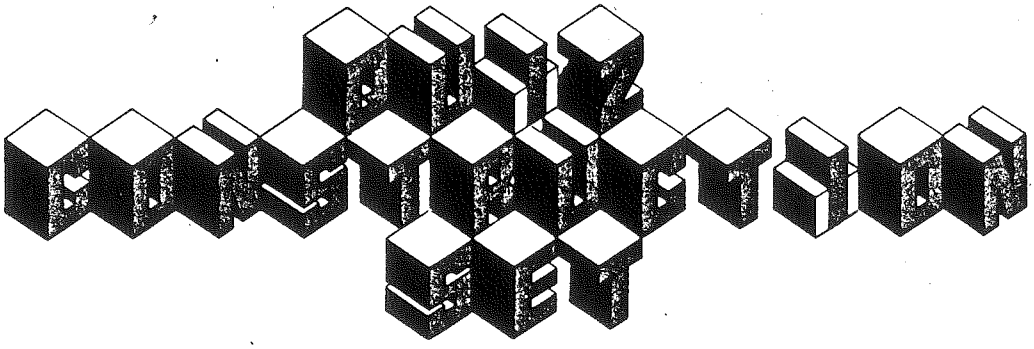
CALI/CALL RESOURCES

Input of photos

A videodisc mastering system capable of on-site input of up to 160,000 photos, drawings, slides and similar material has been developed by Business and Technology Center, 245 East Sixth Street, St. Paul, MN 55101; 612/287-9182.

Apple/IBM Connection

This combination permits exchange of data files between Apple II and IBM PC systems. It is available from Alpha Software, 30 B Street, Burlington, VT 05401.



by William K. Balthrop
HCM Staff

Calling all teachers, students, trivia and non-trivia buffs—
All who seek self-improvement and greater knowledge . . .
Create your own questions and find your own answers
with this do-it-all quiz machine!

In the beginning, there was the question. Then came an answer—and the first quiz was born.

Many questions—and answers—have resulted from mankind's sometimes trivial, sometimes not so trivial pursuit of information and knowledge. Besides asking the eternal questions common to every generation, people are devoting more and more of their time to educating themselves in every area of human interest. And, at work or at play, the basic question and answer quiz is still a favored learning and teaching aid.

A quiz may be spoken or written—or nowadays, designed, stored, and taken on a home computer. Just a few years ago, the "teaching machine" was pretty much a joke—a complicated electro-mechanical device stuck in some school lab, and probably bolted to the floor. Now, teaching is but one natural function of a much smaller, multi-purpose device. With your computer and the program published here, you can do everthing you could have done with that bulky old machine—and much more.

Quiz Construction Set is just what it says: a program that provides all the pieces you need to build, store, and retrieve your own direct-response quiz. It is perfect for school or home learning situations—and can provide a good deal of entertainment as well. You may enter questions of virtually any type, on any subject, with accompanying answers. Use them to exercise your own memory or someone else's. You also may select one of two types of clues to accompany a question, and determine how many chances will be given to get the right answer. As you take the quiz, the program keeps a running score of both right and wrong answers—and also checks your answers for correct spelling, tabulating a score for that as well. (For more on spell-checking, see "Taking The Quiz," page 17.)

Two In One

This software is actually two programs in one package: *Quiz-Make* and *Quiz-Take*. Here again, the names tell you what to expect. You may use the first routine to construct and store your quizzes, and the second to retrieve and take them. This structure serves several purposes: First, it frees up memory to hold larger quizzes. Second, it offers a form of security: If you don't want the quiz takers to be able to modify the quiz, you can give them only *Quiz-Take*, which has no provisions for making alterations in either questions or answers.

Quiz-Make

Use this program to create and modify a quiz before you use *Quiz-Take*. The type of quiz you create is limited only by the total number of questions, your system's memory capacity, and the size of the question and answer fields. The size of these fields is limited to two screen lines for a question, and one screen line for an answer. The maximum screen width is 40 characters on the IBM, Apple and Commodore computers, and 28 characters on the TI-99/4A.

Let's go through the process of setting up a simple quiz. After loading and running the program, you will be shown a title screen. To progress to the Main Menu, press either (ENTER) or (RETURN). You will see six choices:

1) EDIT
2) LOAD
3) SAVE
4) PRINT
5) CHANGE PARAMETERS
6) EXIT

To start, press 1. If you were modifying an existing quiz, you would simply begin entering your new questions and answers. Because this is a new quiz, you will be taken to the parameter setup screen and asked to design your quiz:

QUIZ TITLE — Enter the title of the quiz. This title will be displayed on the top of the screen during the Edit mode, and while a person is taking the quiz.

AUTHOR'S NAME — Enter your name here if you are the quiz's creator.

QUESTIONS HEADER — Enter the prompt you would like to see above all of the questions. This could be the name of a category, or simply the word "Question."

ANSWERS HEADER — Enter the prompt you would like to display above the answer field.

QUIZ TYPE: 1. SEQUENTIAL
2. RANDOM

If you press 1, the Sequential option, the quiz will be given in the *same* order that you enter the questions. Option number 2, Random, means that the questions will be selected at random from the entire quiz file.

PERCENTAGE OF LETTER CLUES (0 - 80) — This option determines how many letter clues will be given for a missed answer. When the Letter Clues option is selected in the *Quiz-Take* program, the student is shown a few of the letters from the answer. The number of letters given is calculated by multiplying the letter-clue percentage times the total number of letters in the correct answer. The spaces where letters are not displayed are filled with asterisks. A 50% letter clue might look like this:

INTERPOSITION (correct answer)
TE*P ITI** (letter clues)

It's possible that fewer clues than the percentage you selected will be displayed. This will happen if the program chooses the same letter twice. In the example above, if the program had twice picked the first T in the word INTERPOSITION there would then be one less letter clue displayed. You should think of this option as a *maximum* letter-clue percentage.

"You may enter questions of virtually any type, on any subject, with accompanying answers. Use them to exercise your own memory or someone else's"

TIME (IN SECONDS) RESPONSE DISPLAY (0 TO 99) — This prompt is asking you to enter the length of time the *correct* answer will be displayed when a person enters a *wrong* answer. *It will not affect the length of time one has to enter an answer.* There is no time limit for a response.

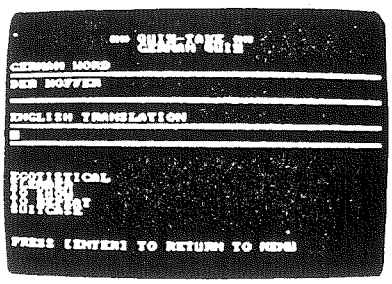
After entering this information, the program will re-display your entries and ask you if they are all correct. If they *are* correct, then press Y. If you wish to change anything, press N. The program will then repeat the setup routine, asking you to re-enter all of the values.

Editing A Quiz

When you conclude the setup, you are taken to the Editor screen. The top of the screen displays the title and the current record number. The record number should be #1, the first record. Below this are two entry fields for the questions and answers. Above each field is the field prompt that you created on the setup screen. A sample question header for a quiz which teaches German might read GERMAN WORD, with the answer header ENGLISH WORD.

The cursor should be flashing inside the questions field. To enter a question, simply type it in. Completely filling the question field automatically transfers you to the answer field.

Enter GESUNDHEIT into the question field and, because it does not fill the question field, press (RETURN) or (ENTER) to terminate the input. After you enter the question, the cursor will move down to the answer field. The answer field is only one screen-line in size. Enter the answer GOOD HEALTH and press (RETURN) or (ENTER). The question and answer fields will clear, and the cursor will reappear in the question field for the next question. The record number at the top of the screen should now read #2.



This photo shows the IBM version administering a German language quiz. The Word Clues option displays five words at the bottom of the screen, one of which is correct.

Go ahead and experiment with your own questions and answers. When you are ready to save your quiz, press (RETURN) or (ENTER) when either the question or answer field is empty, and you will return to the Main Menu screen. If you accidentally do this before you've finished entering questions and answers, simply select 1) EDIT again to continue editing. If you exit from a blank answer field after entering a question, the question you entered for that record will be lost and you will need to re-enter it.

Searching And Changing Records

Back in the Edit mode, you can remodel the quiz you have built so far by searching for a string of characters in either the question field or the answer field. The keys used to select the search vary from

system to system, so check the Control Capsule for your machine. You can select either a question search, or an answer search. Once selected, the words SEARCH FOR will appear above the field. Enter the string that you want to search for in this field.

For example, if you want to locate the first question you entered, GESUNDHEIT, select the Question Search option, then enter GES. The program then searches for questions with the letter combination GES in them. You could have entered HEIT, or GESUND, and the search would have located the first record. When a record is found, its contents are displayed in the question and answer fields. If there is more than one record which matches the search characters or words you enter, then you can select the Next option from the choices listed below the question and answer fields:

PRESS C-CHANGE N-NEXT E-EXIT

If you press C, both the question and the answer fields will be cleared, and you will be able to re-enter them. Each time you press N, the program will continue to search for the next occurance of the search string you entered. You can keep searching—every record if necessary—until you find the record you want, or reach the end of the file. If you reach the end of the file and there are no other matches, the program will return to normal Edit mode, and the first blank record. This is also true when entering EDIT from the Main Menu.

If you wish to discontinue the search, press the key associated with EXIT. The actual key used to exit varies from system to system, so you will need to read your screen display or refer to the Control Capsule for your machine.

Save The Quiz File

To save your data, return to the Main Menu mode by pressing (ENTER) or (RETURN) in either field without entering anything else. Press 3 to select SAVE from the menu. The screen will clear, and then you will be asked to enter the following information:

QUIZ FILE NAME:
TODAY'S DATE:
YOUR NAME:

The QUIZ FILE NAME should be the file name you wish your quiz to have. On some systems you may be asked to also enter a device name or type of device, e.g., disk or tape. For TODAY'S DATE and YOUR NAME, you can enter anything you want to keep a history of the file. This information is displayed when the quiz is printed to the screen or a printer.

Once the save is complete, the program will report how many records were saved. To return to the Main Menu after saving, press (ENTER) or (RETURN).

"This software is actually two programs in one package . . .the first to construct and store your quizzes, and the second to retrieve and take them."

Load The Quiz File

Once you have created and saved a quiz to tape or disk, you may want to work on it again to expand it or make modifications. You can load the quiz by selecting option 2 from the Main Menu. You will be asked for the quiz file name. On some systems you may be asked to enter the device name. Enter the same file name used when you saved the quiz. The program will display information about the file as it's loaded:

title
LAST MODIFIED ON date
BY author's name
QUESTIONS: question header
ANSWERS: answer header
THERE ARE xx RECORDS
READING RECORD # xx
xx RECORDS LOADED
PRESS ANY KEY TO CONTINUE

Printing The Quiz

To list the quiz file contents for review, select option 4 from the Main Menu. You can list the quiz either to the screen, or to another device.

The information listed consists of the quiz parameters entered for the quiz on the parameter setup screen, followed by each question and answer in the quiz file.

Next issue we will present a third program, *Quiz-Print*, which will allow teachers to prepare hardcopy quizzes on a printer. With this program you will be able to format printed quizzes with a large number of options, as well as produce an answer sheet for grading purposes.

Change Parameters

If you have already created a quiz but would like to change its original parameters, select option 5. This will take you to the setup screen, and will ask you to re-enter all of the parameters. After entering them, you will be asked whether they are correct. If so, press Y and you will be returned to the Main Menu.

Exit The Program

If you have a quiz in memory and have made changes to it, then you will be notified before leaving the program, and be given an opportunity to return to the Main Menu. From the Main Menu, you can save the changed quiz, and then exit the program.

You can exit the program *without* saving the changes simply by indicating this when the exit routine warns you. If there have been no changes to the quiz, the exit routine will not stop you when you use option 6.

Quiz-Take

This program is used to take or study a quiz. You cannot alter the quiz from this program. If you wish to create or change a quiz, you must first use *Quiz-Make* to build or alter a quiz file.

After loading and running this program, you will be presented with a title screen. Press (ENTER) or (RETURN) to go to the Main Menu:

1) TAKE QUIZ
2) LOAD
3) STUDY QUIZ
4) EXIT

To select an option, simply press the number beside it. You do not need to press (RETURN) or (ENTER).

1) TAKE QUIZ — Before you can use this option to take the quiz, you will first need to use option 2 to LOAD a quiz.

2) LOAD QUIZ — This option must be used to LOAD a quiz file previously created with *Quiz-Make*. If you haven't yet created a file with *Quiz-Make*, then refer to the previous section on running that program. When you select this option you will be prompted to enter

a file name. On some systems you will be asked to also enter the device name—e.g., tape or disk, drive 1 or drive 2. The program will display the number of records read in from the file, and then wait for you to press (ENTER) or (RETURN) before continuing back to the Main Menu.

3) STUDY QUIZ — This option allows you to study a quiz. Four questions and answers will be displayed on the screen at a time. You can then scroll through the list of questions and answers by pressing the appropriate keys. (The keys used for each system are described in the Control Capsules included with this article.) You can exit this mode and return back to the Main Menu at any time by pressing the appropriate escape key, also described in the Control Capsules for each system.

4) EXIT — There are no restrictions in exiting this program as there are in *Quiz-Make*. You may exit the program at any time you like. You will never cause the loss of data by exiting the program because this program can not alter any files you have created. The only thing that may be lost by exiting the program is your score—and possibly your pride . . .

Quiz Level

After selecting the TAKE QUIZ option you will be shown another menu screen. This screen is used to select difficulty level of the quiz and the type of clues, if any.

```
1) WORD CLUES      2 TRIES
2) WORD CLUES      1 TRY
3) LETTER CLUES    3 TRIES
4) LETTER CLUES    2 TRIES
5) NO CLUES        1 TRY
6) SAME QUIZ
7) EXIT
```

1) & 2) Word Clues — If you select options 1 or 2, you will be given a list of five answers at the bottom of the screen for every question. One of those five answers will be the correct answer. The answers displayed are selected completely at random from all of the answers in the quiz file, so that each time the quiz is taken, or the same question is asked, there will be a different list of possible answers.

In option 1 you have two chances to answer a question correctly. If you miss the answer on the second try, a spelling check will be done to see whether you simply misspelled the word.

If you select option 2, you must answer each question on the first try. If an answer is incorrect, then the program will perform the spelling check.

3) & 4) Letter Clues — Options 3 and 4 will give you letter clues if you enter the wrong answer. The letter clues were explained in more detail in the *Quiz-Make* section "Percentage of Letter Clues."

In option 3, you are given three opportunities to answer a question. On the first try, no clues are given. If you miss the answer on the first try, then clues will be displayed in the answer field, with asterisks indicating character positions where a clue is not given. You can then type right over the clues and asterisks to enter your next answer.

If the second try is wrong, you will be given a new set of letter clues, and another chance at answering the question. If you miss the question on the third try, a spelling check will be performed.

If option 4 is selected, the quiz will act just as it did for option three, except that only *one* set of letter clues will be given. If you miss the answer on the second try, a spelling check will be performed.

5) No Clues — Option 5 will not give you any clues to the right answer, and will only allow *one* try to answer the question. If the answer is wrong on the first try, a spelling check is done to see if the answer has been misspelled.

6) Same Quiz — At any time during a quiz, you may return to the Main Menu screen by pressing (ENTER) or (RETURN). You may resume the quiz where you left off by selecting option 1 from the Main Menu (TAKE QUIZ), and then selecting option 6 from the quiz level menu (SAME QUIZ). The same quiz will be resumed with the score you had at the time you exited the quiz. If you select a quiz level other than option 6 (SAME QUIZ), the score will reset to zero and the quiz will start over.

7) Exit — Selecting option 7 will return you to the Main Menu.

Taking The Quiz

After selecting the quiz level, you will be taken to the quiz screen. This screen looks just like the one used for editing the quiz in the *Quiz-Make* program.

If the quiz is set up for sequential operation, then all of the questions have a predetermined order—they will be given in the same order in which they were entered. A question will be displayed in the question field, and the cursor will start blinking in the answer field, waiting for an answer to be entered. If a Word Clues option has been selected, it will be displayed at this time. After you enter the answer, it is checked against the correct answer. If it is not 100% correct, letter for letter, the answer is considered wrong. If the entry was the last try, or the *only* try (as in options 2 and 5), the answer undergoes a spelling check to see if the word is misspelled.

The spelling check is not 100% foolproof, but it does manage to catch minor spelling errors. The check is done by comparing each letter in your answer with the correct answer. Character position is important here. If 70% or more of the characters match, the answer is considered to be correct but misspelled. Less than a 70% match, and the answer is counted as being wrong. The comparison may look like this:

```
GOOD HEALTH   (Correct answer)
GOOD HAELTH   (9 out of 11— 81% — misspelled)
GOOD FOOD     (5 out of 11— 45%—wrong)
```

"At work or at play, the basic question and answer quiz is still a favored learning and teaching aid."

Notice that in the third answer above, *five* characters—not *four*—matched out of eleven because *all* of the characters are checked, even spaces. The alignment of the characters is important as well. If a character is simply left out, such as:

```
GOOD HEALTH   (Correct answer)
GOOD HALTH    (7 out of 11— 64%—wrong)
```

the characters to the right of the E in the word HELTH would not line up correctly with the characters in the correct answer, and would *all* be considered wrong. Thus, in this example only 64% of the characters match, making the answer incorrect—not just a misspelling.

Your score is displayed at the top of the screen during the quiz. The score is actually a percentage, and not a total. The percentages are for right answers, wrong answers, and misspellings. By putting the

misspellings in a separate category, placement of the score can be left up to the administrator of the quiz. It could be added to the right or the wrong answer score, or simply used as a separate method of evaluation. This "adding of the scores" must be done by the person giving the quiz—there are no provisions in the program to have it done automatically.

Administering Quizzes

To use these programs with a disk system, initialize two disks. Place *Quiz-Make* on one disk, and *Quiz-Take* on the other. Do all of your quiz development on the disk with *Quiz-Make*. When the quiz is complete, SAVE a copy of the quiz file to the disk with *Quiz-Take*. This will give you a back-up of the quiz file. In addition, the quiz taker will not be able to change the quiz file because the *Quiz-Take* disk does not contain the program *Quiz-Make*.

If you are using a cassette system, SAVE the two programs separately on two different tapes. Also, each quiz should be kept on its own tape. Label all tapes very clearly. This will prevent you from accidentally recording over one of the programs or the quiz file.

CONTROL CAPSULE
Quiz-Make

Edit Mode

KEY	FUNCTION
CTRL Q	Select question search mode.
CTRL A	Select answer search mode.
Left CRSR	Back space to erase.
RETURN	Return to menu.

Search Mode

KEY	FUNCTION
C	Change record.
N	Next record.
E	Return to menu.

Quiz-Take

Take Quiz Mode

KEY	FUNCTION
Left CRSR	Back Space to erase.
RETURN	Return to menu.

Study Quiz Mode

KEY	FUNCTION
Left CRSR	Scroll up.
Right	Scroll down
Esc	Return to menu.

Special Enhancement for Apple ProDOS

Under the Apple *II* family's new operating system, ProDOS, the system must be informed whenever you wish to access a disk with a different volume name or PREFIX. The program as published in the listing section was written to run under DOS 3.3, which does not use PREFIXes. To use the *Quiz Construction Set* programs when running under ProDOS, make the changes indicated in the following listings. [See the Apple-related "Home Computer Tech Note" in this issue for more information on ProDOS PREFIXes.—Ed.]

Make the following modifications to *Quiz-Make*:

```
810 INVERSE : PRINT "LOAD DATA": NORMAL
    : PRINT : PRINT "ENTER FILE NAME:
    ": ML = 15: VT = 3: HT = 17: GOSUB 156
    0: IF B$ = "" THEN 240
820 F$ = B$: A = ASC ( LEFT$ (F$,1)): I
    F A > 95 OR A < 64 THEN PRINT "INV
    ALID FILE NAME": CHR$ (7): FOR T =
    1 TO 1000: NEXT : HOME : GOTO 810
825 FOR I = 2 TO LEN (F$): V$ = MID$ (
    F$,I,1)
830 IF NOT (( V$ = "." ) OR ( V$ > = "A" A
    ND V$ < = "9" ) OR ( V$ > = "a" AND
    V$ < = "z" )) THEN PRINT CHR$ (7
    ): PRINT "INVALID PRODOS FILENAME":
    FOR T = 1 TO 1000: NEXT : HOME : G
    OTO 810
835 NEXT I
836 PRINT : PRINT "INSERT DISK IN DRIVE
    1 AND PRESS A KEY": GOSUB 1750: D$
    = CHR$ (4)
837 PRINT D$;"PREFIX,D1"
980 PRINT : PRINT : PRINT "ENTER FILE N
    AME": ;: VT = 9: HT = 17: ML = 15: GOS
    UB 1560: IF B$ = "" THEN 240
990 F$ = B$: A = ASC ( LEFT$ (F$,1)): I
    F A > 95 OR A < 64 THEN PRINT "INV
    ALID FILE NAME"; CHR$ (7): FOR T =
    1 TO 1000: NEXT : GOTO 980
995 FOR I = 2 TO LEN (F$): V$ = MID$ (
    F$,I,1)
1000 IF NOT (( V$ = "." ) OR ( V$ > = "A" A
    ND V$ < = "9" ) OR ( V$ > = "a" AND
    V$ < = "z" )) THEN PRINT CHR$ (7
    ): PRINT "INVALID PRODOS FILENAME":
    FOR T = 1 TO 1000: NEXT : HOME : G
    OTO 980
1005 NEXT I
1006 PRINT : PRINT "INSERT DISK IN DRIVE
    1 AND PRESS A KEY": GOSUB 1750: D$
    = CHR$ (4)
1007 PRINT D$;"PREFIX,D1"
1010 PRINT D$;"OPEN ";F$
```

Make the following modifications to *Quiz-Take*:

```
800 PRINT : PRINT "ENTER FILE NAME:
    ": M
    L = 15: VT = 3: HT = 17: GOSUB 1020:
    IF B$ = "" THEN 230
810 Q$ = B$: IF ASC (Q$) < 64 OR ASC
    (Q$) > 95 THEN PRINT "INVALID FILE
    NAME"; CHR$ (7): FOR T = 1 TO 1000
    : NEXT : HOME : GOTO 790
815 F$ = B$: FOR I = 2 TO LEN (F$): V$
    = MID$ (F$,I,1)
820 IF NOT (( V$ = "." ) OR ( V$ > = "A" A
    ND V$ < = "9" ) OR ( V$ > = "a" AND
    V$ < = "z" )) THEN PRINT CHR$ (7
    ): PRINT "INVALID PRODOS FILENAME":
    FOR T = 1 TO 1000: NEXT : HOME : G
    OTO 790
825 NEXT I
826 PRINT : PRINT "INSERT DISK IN DRIVE
    1 AND PRESS A KEY": GOSUB 1020: D$
    = CHR$ (4)
827 PRINT D$;"PREFIX,D1"
830 PRINT D$;"VERIFY ";F$
840 PRINT D$;"OPEN ";F$
```

CONTROL CAPSULE
Quiz-Make

Edit Mode

KEY	FUNCTION
SHFT Q	Select question search mode.
SHFT A	Select answer search mode.
DEL	Back space to erase.
RETURN	Return to menu.

Search Mode

KEY	FUNCTION
C	Change record.
N	Next record.
E	Return to menu.

Quiz-Take

Take Quiz Mode

KEY	FUNCTION
SHFT X	Return to menu.
DEL	Back space to erase.

Study Quiz Mode

KEY	FUNCTION
Down CRSR	Scroll down.
Up CRSR	Scroll up.
SHFT X	Return to menu.

CONTROL CAPSULE
Quiz-Make

Edit Mode

KEY	FUNCTION
F1	Select question search mode.
F2	Select answer search mode.
Back Space	Back space to erase.
ENTER	Return to menu.

Search Mode

KEY	FUNCTION
C	Change record.
N	Next record.
ENTER	Return to Edit mode.

Quiz-Take

Take Quiz Mode

KEY	FUNCTION
ENTER	Return to menu.
Back Space	Back space to erase.

Study Quiz Mode

KEY	FUNCTION
F1	Scroll up.
F2	Scroll down.
Esc	Return to menu.

CONTROL CAPSULE
Quiz-Make

Edit Mode

KEY	FUNCTION
FCTN E	Select questions search mode.
FCTN X	Select answers search mode.
FCTN 3	Back space to erase.
ENTER	Return to menu.

Search Mode

KEY	FUNCTION
C	Change a record.
N	Next record.
E	Return to Edit mode.

Quiz-Take

Take Quiz Mode

KEY	FUNCTION
FCTN 9	Return to menu.
FCTN 3	Back space to erase.

Study Quiz Mode

KEY	FUNCTION
FCTN 9	Return to menu.
FCTN E	Scroll up.
FCTN X	Scroll down.

SAMPLE DATA BASES

THREE QUIZZES READY-TO-RUN

Here are three quizzes that you can enter into the *Quiz Construction Set* right away. Each of the first two quizzes is divided into 2 parts: Part 1 of the English vocabulary quiz places words in each answer field, and their definitions in the question fields. Part 2 features words used only in a special context, such as trade jargon or slang words—with each word given as a question, and each definition as an answer. Part 1 of the German quiz lists common German words in the question fields, and their meanings in the answer fields. In part 2, each question is a common English phrase and each answer, the German equivalent. (We suggest you use a Word Clues option for part 2 of each quiz.) A short trivia quiz is the last sample data base.

The questions and answers given here can be used with all four versions of the program. You can also add to these quizzes, use only those questions and answers you want, or create a quiz comprised of questions from all of the examples shown here.

VOCABULARY QUIZ 1

Q. harsh or discordant sound
A. cacophony

Q. decaying organic matter found on the forest floor
A. duff

Q. the space between nerve cells
A. synapse

Q. term for a fertilized egg
A. zygote

Q. a riddle
A. conundrum

Q. act of giving birth
A. parturition

Q. calm, happy, and peaceful
A. halcyon

Q. fear of foreigners
A. xenophobia

Q. a sudden outburst
A. salvo

Q. an article of food
A. viand

Q. to talk informally; chat
A. confabulate

Q. tending to melt or dissolve
A. deliquescent

Q. resembling a tree in structure
A. dendriform

Q. irritable or peevishly sensitive
A. tetchy

Q. inappropriately jocular
A. facetious

Q. having no petals
A. apetalous

Q. to cheat out of what is due
A. bilk

Q. funnel-shaped clay smoking pipe
A. chillum

Q. extremely cold
A. gelid

Q. following in time or order
A. subsequent

Q. just and fair; impartial
A. equitable

Q. social courtesies; manners
A. amenities

Q. characterized by verbal abuse
A. vituperative

Q. liberating energy
A. exergonic

VOCABULARY QUIZ 2

Q. saute'
A. fry in small amount of fat

Q. schuss
A. ski downhill at high speed

Q. allegro
A. a fast tempo in music

Q. gaffer
A. movie lighting technician

Q. plumb
A. straight up and down

Q. tweek
A. to adjust (electronics)

Q. byte
A. eight bits of data

Q. bullish
A. optimistic of boom market

Q. overdub
A. to record sound on sound

Q. codex
A. a manuscript book

Q. build-down
A. keep only new weapons

Q. totem
A. emblem or revered symbol

Q. gable
A. end wall of a building

Q. chutzpah
A. extreme self-confidence

Q. frappe
A. a partly frozen drink

Q. piquant
A. engagingly provocative

Q. pixel
A. screen picture element

Q. bug
A. a program malfunction

Q. hyperbole
A. extravagant exaggeration

Q. sprent
A. sprinkled over

Q. yarder
A. a log-pulling machine

Q. vapid
A. lacking liveliness

Q. gaggle
A. flock of geese on ground

Q. parry
A. to ward off an attack

Q. maquette
A. small preliminary model

Q. perquisite
A. extra reward or gratuity

Q. farrier
A. one that shoes horses

Q. warp
A. lengthwise strings in loom

GERMAN QUIZ 1

Q. der Koffer
A. suitcase

Q. gutaussehend
A. good-looking

Q. die Reise
A. trip

Q. nuelich
A. recently

Q. zwischen
A. between

Q. augenblichlich
A. immediately

Q. die Brieftasche
A. wallet

Q. eingebildet
A. egotistical

Q. das Fliessband
A. assembly line

Q. schlafen
A. to sleep

Q. die Innenstadt
A. downtown

Q. wiederholen
A. to repeat

Q. vergebens
A. in vain

Q. die Gemeinschaftschule
A. primary school

Q. die Schreibwaren
A. stationery

Q. zeigen
A. to show

Q. das Verfahren
A. procedure

Q. furchtbar
A. horrible

Q. wunderlich
A. wonderful

Q. die Armbanduhr
A. wristwatch

Q. ungezwungen
A. casual

Q. die Verwandten
A. relatives

Q. zugeben
A. to forgive

Q. das Rasiermesser
A. razor

Q. schlank
A. slender

Q. jawohl
A. indeed

Q. verstehen
A. to understand

GERMAN QUIZ 2

Q. in the meantime
A. in der Zwischenzeit

Q. what's the matter?
A. was ist los?

Q. take care!
A. mach's gut!

Q. I am sorry
A. es tut mir leid

Q. it's now or never
A. entweder jetzt oder nie

Q. it works wonders
A. wunder wirken

Q. do you have a light?
A. haben sie Feuer?

Q. help yourself
A. sich bedienen

Q. now and then
A. hin und wieder

Q. for example
A. zum Beispiel

Q. in that case
A. in diesem Fall

Q. take it easy
A. nimm's leicht

Q. without a doubt
A. ohne Zweifel

Q. be that as it may
A. wie dem auch sei

Q. hurry up
A. mach schnell

Q. in the morning
A. am Morgen

Q. how are you?
A. wie geht's?

Q. good day!
A. guten Tag!

Q. we have a lot in common
A. sie steht mir nahe

Q. you're welcome
A. bitte sehr

Q. show me
A. zeigen sie mir

Q. what does that come to?
A. wieviel macht das?

Q. that turns me on
A. das begeisert mich

Q. a big shot
A. ein hohes Tier

Q. time is up
A. die Zeit ist um

Q. see you later!
A. Aufwiedersehn!

TRIVIA QUIZ

Q. Who was the fifth Marx brother?
A. Gummo

Q. What's the flip side of the Beatles' single, "Rain"?
A. Paperback Writer

Q. To what religious sect do we owe the circular saw?
A. Shakers

Q. Which was the 1st major car with front wheel drive?
A. The Cord

Q. Who is the robot in "The Day the Earth Stood Still"?
A. Tobar

Q. Who are the people of "The Forbidden Planet"?
A. The Krell

Q. What TV show featured Cochise?
A. Broken Arrow

Q. Name the first computer to use a mouse and icons.
A. Xerox Star

Q. Who shot James A. Garfield?
A. Charles Guiteau

Q. How many typographic points to the inch?
A. 72

Q. Which particle has both light and matter properties?
A. nuetrino

Q. Which famous cowboy movie star carried a whip?
A. Lash La Rue

Q. What is the world's highest-flying jet aircraft?
A. SR-71 Blackbird

Q. Who was Fred Flintstone's best friend?
A. Barney Rubble

Q. Where was the first semiconductor produced?
A. Bell Labs

Q. Who is the father of the Pascal language?
A. Nicholas Wirth

Q. What is Ringo's other name on the Sgt. Pepper album?
A. Billy Shears

Q. Who said, "I have not yet begun to fight"?
A. John Paul Jones

Q. In which film did Chaplin satirize Adolf Hitler?
A. The Great Dictator

Q. Who was the "Man of a Thousand Faces"?
A. Lon Chaney Sr.

Q. What was the name of the dog in "Topper"?
A. Neil

Q. What is the name of Ricky's brother in "Ozzie and Harriet"?
A. David

Q. What is the largest self-supporting concrete roof?
A. The Seattle Kingdome

Q. Who was the founder of the Republic of China?
A. Sun Yat-sen

Q. Which computer magazine has no outside advertising?
A. Home Computer Magazine

Q. Who said, "I will fight no more forever"?
A. Chief Joseph

Q. What is the name of the spice in "Dune"?
A. melange

Q. Who hosted "You Are There" in the 1950's?
A. Walter Cronkite

Q. What were Romeo and Juliet's last names?
A. Montegue and Capulet

Q. Which President was raised as a Quaker?
A. Richard Nixon

Q. What was Priam's prize for judging a beauty contest?
A. Helen

Q. Which early radio show restaged movie hits?
A. Lux Radio Theater

Q. What was the dry planet in "The Dispossessed"?
A. Anarres

Q. Who was the housekeeper in "My Three Sons"?
A. Bub

Q. What substance powers the Starship Enterprise?
A. dilithium crystals

Quiz-Make (Apple *II* Family) Explanation of the Program

Line Nos.	
100-180	Program header.
190	Define error-trapping routines location.
200-230	Title screen.
240-290	Main menu.
300-800	Edit quiz and search records.
810-940	Load quiz file.
950-1100	Save quiz file.
1110-1320	Print quiz file.
1330-1470	Input-parameters routine.
1480-1540	Exit-program routine.
1550	Illegal entry message.
1560-1740	Main-input routine.
1750-1760	Single-key-input routine.
1770-1810	Error-trapping routine..
1820-1830	Program data.

Quiz-Make (C-64) Explanation of the Program

Line Nos.	
100-170	Program header.
180-250	Display title screen.
260-370	Main menu.
380-750	Edit the quiz.
760-1130	Search mode.
1140-1450	Load the quiz file.
1460-1780	Save the quiz file.
1790-2200	Print the quiz file.
2210-2510	Change-parameters routine.
2520-2630	Input routine.
2640-2720	Illegal entry messages.
2730-2870	Input-a-question routine.
2880-3020	Input-an-answer routine.
3030-3040	Clear parts of the edit screen.
3050-3060	Routine to locate the cursor.
3070-3170	Exit-program routine.

Quiz-Make (IBM PC, PCjr) Explanation of the Program

Line Nos.	
100-200	Program header.
210	Define error-trapping-routines location.
220-250	Initialization and title screen.
260-300	Main menu.
310-630	Edit and search mode.
640-740	Load quiz file.
750-850	Save quiz file.
860-1010	Print quiz file contents.
1020-1060	Controlling routine for change parameter option.
1070-1100	Single-key-input routine.
1110-1220	Main-input routine.
1230-1290	Error-trapping routine.
1300-1350	Program data.
1360-1420	End-of-program routine.

Quiz-Take (Apple II Family) Explanation of the Program

Line Nos.	
100-180	Program header.
190	Define error-trapping routines location.
200-220	Title screen.
230-280	Main menu.
290-390	Quiz level menu.
400-420	Display the quiz screen.
430-520	Display problem and get answer.
530-600	Branch to appropriate routine depending on the option selected from the quiz option menu.
610-630	Wrong answer.
640-660	Display letter clues.
670-750	Spelling check.
760-770	Wrong answer—try again.
780	Right answer.
790-910	Load a quiz file.
920-1010	Study the quiz mode.
1020-1190	Main-input routine.
1200-1210	Clear parts of the screen.
1220-1260	Select five random word clues.
1270-1290	Display scores.
1300-1320	Single-key-input routine.
1330-1370	Error-trapping routine.
1380-1390	Exit-program routine.
1400-1410	Program data.

Quiz-Take (IBM PC, PCjr) Explanation of the Program

Line Nos.	
100-200	Program header.
210	Define error-trapping-routines location.
220-250	Initialization and title screen.
260-300	Main menu.
310-390	Quiz options menu.
400-470	Display the quiz screen.
480-490	Display question, get answer, and branch to routine to handle response depending on quiz level selected.
500-540	Do spelling check.
550-570	Display letter clues.
580	Correct answer.
590-660	Study mode.
670-760	Load the quiz file.
770-800	Single-key-input routine.
810-900	Main-input routine.
910-980	Error routine.
990-1020	Time-delay routine.
1030-1060	Program data.
1070-1100	End-of-program routine.
1110-1140	Display-scores routine.

Quiz-Make (TI-99/4A) TI BASIC or Extended BASIC Explanation of the Program

Line Nos.	
100-170	Program header.
180-240	Initialization and title screen.
250-310	Main menu.
320-1380	Edit quiz and search records.
1390-1530	Load a quiz file.
1540-1720	Save a quiz file.
1730-2080	Print a quiz file.
2090-2710	Change parameters.
2720-3080	Main-input routine.
3090-3130	Illegal entry message.
3140-3170	Single-key-input routine.
3180-3270	Routines to clear the question and answer fields.
3280-3340	Routine to clear parts of the edit screen.
3350-3440	Exit-program routine.

Quiz-Take (C-64) Explanation of the Program

Line Nos.	
100-170	Program header.
180-250	Display the title screen.
260-380	Main menu.
390-550	Option menu for level of difficulty.
560-700	Display the quiz screen.
710-790	Display question and get answer.
800-880	Branch to routine to handle user's response.
890-960	Wrong-answer routine.
970-1070	Display letter clues.
1080-1200	Check spelling.
1210-1250	Missed guess—try again.
1260-1420	Right answer.
1430-1740	Load a quiz file.
1750-1870	Study-quiz routine.
1880-2000	Display the scores.
2010-2110	Choose five random numbers.
2120-2180	Clear the question and answer fields.
2190-2200	Locate-cursor routine.
2210-2330	Input routine.
2340-2360	Single-key-input routine.
2370-2420	Illegal entry messages.
2430-2450	End-of-program routine.

Quiz-Take (TI-99/4A) TI BASIC or Extended BASIC Explanation of the Program

Line Nos.	
100-180	Program header.
190-250	Title screen.
260-370	Main menu
380-560	Quiz options menu.
570-900	Display the quiz screen.
910-1030	Display the problem and get the answer.
1040-1130	Branch to appropriate routine depending on the level selected from the quiz level screen.
1140-1290	Wrong-answer routine.
1300-1410	Display letter clues.
1420-1690	Spelling check.
1700-1740	Wrong answer—get another try.
1750-1840	Right answer routine.
1850-2080	Study quiz mode.
2090-2230	Display scores.
2240-2350	Select five random word clues.
2360-2390	Clear portions of the quiz screen.
2400-2440	Routine to display messages on the screen.
2450-2770	Main-input routine.
2780-2800	Single-key-input routine.
2810-2820	Exit-program routine. HCM

KLAVIER IM HAUS"-AN INTERACTIVE EXPERIMENT IN FOREIGN LANGUAGE INSTRUCTION

Captain David M. Schrupp, Captain Michael D. Bush,

and Major Gunther A. Mueller

Despite the steadily increasing use and interest of computer-assisted instruction (CAI) in foreign language education, there remains a shortage of information on its effectiveness. More specifically, empirical data on CAI-related studies in foreign language are almost non-existent. This article reports the results of a CAI-based interactive video study designed to address this void and which was conducted by the German Section of the U.S. Air Force (USAF) Academy's Department of Foreign Languages.

Purpose of the Study

The purpose of this experiment was to measure any quantifiable advantage in learning outcome attributable to computer-assisted interaction. Specifically, what are the effects on learning outcome with the use of computer-assisted interactive video technology in foreign language instruction, as compared to more conventional presentations of video material?

The Department of Foreign Languages at the USAF Academy is planning to set up a language learning center, complete with microcomputers, videotape-disc players, voice synthesis, and voice recognition capability if the cost can be justified in terms of learning outcome. It is very important to justify the considerable costs involved by showing that this technology can improve learning outcome.

Of course there are other costs associated with a commitment to use computer technology in language instruction. These include the time spent by instructors who develop the necessary programs, as well as the time spent by students to evaluate those efforts. Although the student's time in an experimental program need not be considered as wasted, there is a definite risk associated with the high number of manhours needed to design the courseware. When language instructors are tasked to design these materials, the normal teaching routine may suffer.

In light of these costs, the experiment at the USAF Academy was limited to a specific aspect of CAI and designed to investigate the potential of interactive video (IAV) for first-year German students. Through this very limited scope, the lesson designers, instead of committing many manhours to design and teach an experimental, full course, tested the students in two existing courses on comprehension and retention of material contained in a short film. The film was presented in three different ways, one of which was completely non-interactive and one of which utilized IAV.

Limitations and Assumptions

The assumption was that any advantage in comprehension outcome that arises from the use of IAV in showing a single film could be applied to the use of IAV technology integrated into a full language course. The results of the experiment are therefore limited, by design, by the content of that film.

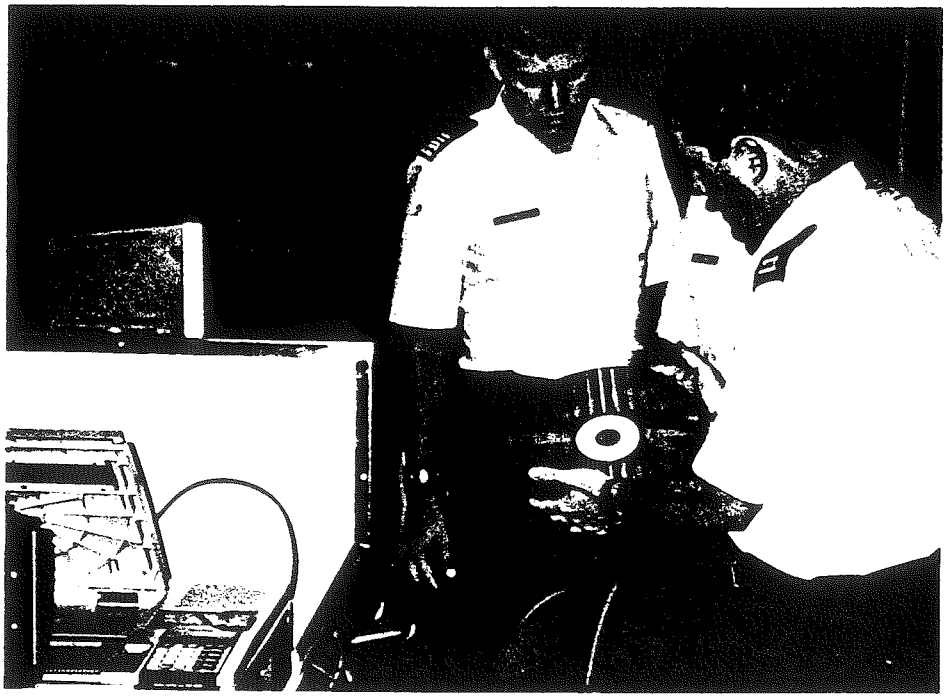
A more general limitation for any IAV program designer is the availability of suitable video material. There is videotape material available for most purposes, but videodiscs in foreign languages are almost non-existent. However, because this experiment was designed to test the effectiveness of the most advanced technology, the videodisc was used.

Ideally, the designer of an experiment (or a course) would produce his or her own video material—material required to support specific learning objectives. The high costs of videodisc production, however, make the IAV program designer dependent upon the work of others. This is another reason for the limited scope of the USAF Academy experiment.

Captain David Schrupp, a graduate of the U.S. Air Force Academy, holds an M.A. degree in International Relations from Tufts University. He completed course work for a Ph.D. in Political Science at the University of Vienna, Austria, as an Olmstead Scholar. Following his recent assignment as a German instructor at the Air Force Academy, Captain Schrupp is completing his Ph.D. in Vienna under Academy sponsorship and will return to the Academy upon completion of his degree.

Captain Michael Bush is an Associate Professor of French and Director of Research in the Department of Foreign Languages at the U.S. Air Force Academy. He has a B.A. in Political Science from Brigham Young University, an MBA from the University of Misssouri, and a Ph.D. in Foreign Language Education with an emphasis in Computer Science from The Ohio State University.

Major Gunther Mueller is presently studying at the Air Command and Staff College at Maxwell AFB, Alabama. During his recent tour at the U. S. Air Force Academy, he served as an Associate Professor of German in the Department of Foreign Languages, Chairman of the German Section and Deputy to the Department Head for Instruction. His B.A. and M.A. are in German from West Virginia University while his Ph.D. in Foreign Language Education is from The Ohio State University.



Klavier Im Haus disc for use at USAF Academy

Theoretical Framework and Related Studies

The theoretical basis for this experiment comes from the general observation that interaction is the key to learning, especially in learning a skill such as a foreign language. Modern language educators agree that there is a difference in learning outcome between those students who just observe and those who actually are engaged in interactive activities involving target-language production. The role of the computer in this experiment was to provide interaction with the film materials used, thus allowing comparison between teacher-controlled interaction, such as that found in the simple presentation of a film, and that provided by the computer. The approach used was patterned after previous work done in the field of biology education (Bunderson et al, 1981). Although the WICAT study was much broader in scope, its purpose was similar to that stated as a goal for this research. Although it was shown that there are significant advantages in the rate of learning that comes from computer-assisted interaction, such a determination was not possible in this experiment, because it dealt with a very limited video presentation rather than with an entire course. A thorough justification of such use in a foreign-language teaching setting appeared in a recent issue of the CALICO Journal (Stevens, 1983).

Procedures

The research plan was conducted in two phases. First it was necessary to develop an IAV program, and then set up an experiment to compare that program with conventional presentation methods. We chose to use the film *Klavier im Haus*, which was produced by the German Educational Television Network. This film was made available to the Academy through a research agreement with the Defense Language Institute's (DLI) Educational Technology Division, and had been placed on videodisc as a joint project of DLI and the Goethe Institute of the Federal Republic of Germany.

The first phase of the project actually began in February 1983, when the videodisc arrived at the USAF Academy from DLI. A new instructor of German, who had attended a one-day Interactive Video Survey presented by Sony Video Utilization Services (Dargan) volunteered to develop a 20 to 30 minute video-based lesson. After some brief instructions on how to use an authoring system written by Texas Instruments, he chose a fairly simple approach to exploit the 12-minute film. Aiming the program at a basic level for first-year students, he divided the story into short segments. Each video-sequence was followed by one or two basic comprehension questions.

The film itself is well-suited to such a treatment, since the plot contains several separate conversations. The story shows how a young couple move into a new apartment with their grand piano. The husband is a concert pianist and must practice several hours a day. The wife goes to several of the building's tenants asking for understanding in order to head off any future complaints. Although the dialogue is very predictable, with elementary introduction and question situations, the film was not produced for language instruction and several actors use colloquialisms with poor pronunciation. It would be a challenge for most intermediate students to understand *all* the elements of each conversation.

The IAV program development took about 40 hours of the instructor's time. The end product, which was finished in mid-March, could be completed by the student in 19 to 37 minutes, depending upon the number of incorrect answers given by him. The program was designed for use on a TI-99/4A microcomputer, interfaced with a Sony LDP-1000 laser-optical videodisc player. In March 1983 the Department of Foreign Languages at the USAF Academy had two such usable stations. Except for the videodisc players, all the equipment was provided by Texas Instruments through a research agreement.

In the second phase of the experiment, the first-year students (about 210 total) of German were divided into four groups on a random basis. Ten per cent of the students in two courses were designated Group A and asked to take the comprehension test without viewing the film at all. The remaining students (about 190) were placed into three equally-sized groups, each of which viewed the film at different times and with varying levels of interactivity. The experiment lasted 10 class days, from 28 March to 8 April, 1983. The short (8 answer) comprehension quiz was given during (or immediately after) the initial viewing of the film, and then again 6 to 8 days later. (See Appendix A for a copy of the quiz)

The various methods of presentation dictated the manner in which the quiz was administered.

Group A was used to validate the quiz. The students took the quiz in class on the first two days of the experiment. Because the quiz was very short, it was necessary to determine the extent of any guess factor that might influence results in other groups. The size of Group A was minimized in order to provide maximum participation in the other groups.

Group B watched the film twice on a 27-minute videotape and then completed the short comprehension quiz. The average time spent by the students in Group B was 35 minutes.

Group C students watched the film once in its entirety, and then—on the second time through—saw the same video segments that were used in the IAV program. At the end of each video segment, the students were given 30 seconds to answer, on their answer sheet, a question displayed on the screen. The questions were the same as those appearing in the IAV program, with the obvious distinction that there was no feedback on the answers. These students completed the task in 35 minutes—also at a location outside the classroom, with a moderator timing the question frames on the TV screen.

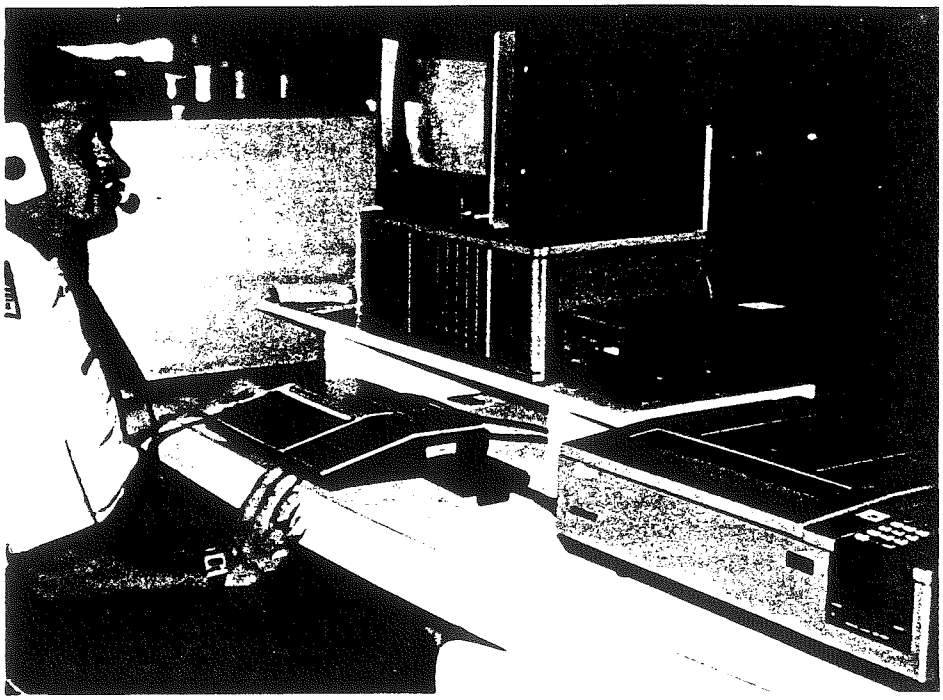
The students in Group D were scheduled to go to the language laboratory individually and complete the IAV program under the direction of the lab personnel. The average time for each student was about 25 minutes, and all students had completed the program in the first 5 days of the experiment. Group D students were not given the comprehension test at the time they completed the IAV program, but saw the test for the first time 4-8 class days later, when all groups (except A) were tested again on the same comprehension quiz.

Analysis of Results

The experiment yielded two scores for groups B and C, and one score for groups A and D. The results were analyzed using the Academy's Burroughs 6900 with programs from the *Statistical Package for the Social Sciences* (SPSS) (Nie et al, 1975). The following charts show the differences among group means within each course. Group A scores indicated that the comprehension quiz was indeed a valid test, since the average scores were well within expected ranges for a multiple-choice measurement. (See Table 1) It is interesting to note that there was a statistically significant difference between the Group A results of the two course levels. The beginner level (German 132) guessed more accurately than the intermediate students in German 142. It is appropriate to mention here that students are placed in these two courses during their freshman year at the Academy, based on performance in a placement/validation exam prior to the start of the academic year.

The results from the second test are the most important for the purposes of this experiment. The differences in group means are all statistically significant at or beyond the level of .05. These differences indicate a decided advantage in comprehension for those students completing the IAV program. (See Table 2)

The determination of significance was made through the use of the ONEWAY Analysis of Variance program of SPSS. In addition, a T-Test program performed on pairs of scores showed that only Group B and the German 132 course had a significant difference between the means fron the two measurements. This fact reinforces the significance of the advantages for the students completing the IAV program, since there was no other evidence that Groups B or C were affected by the week-long wait before the second measurement.



Texas Instrument-Videodisc arrangement - USAF Academy

Discussion and Recommendations for Future Research

The experiment with *Klavier im Haus* clearly shows an advantage with IAV over conventional methods of presenting a short film to first-year students. The Group B participants, who only watched the film twice, had an average score about 40% lower than those using the IAV. This differential was the same for both course levels. The interaction variable appears to be an important factor, as the Group C students scored much higher than their Group B counterparts. However, the mean scores from both levels of Group C were nearly 24% lower than the IAV Groups.

In future research, a retention factor could be evaluated if the non-interactive test groups were given feedback from their initial tests. Then the difference between individualized, computer-assisted feedback and that from

an instructor could be measured. Along this same line, it would be desirable to test the IAV group immediately after the students complete the program. This would provide the basis to measure any changes in comprehension score over a period of time—the retention variable. Some changes to the test itself would allow more discrimination between student scores. If the test did not use the exact wording from the IAV program, as was the case with *Klavier im Haus*, then one group of students wouldn't have an advantage.

Some recommendations for futher study with this same video material, but with different programs, are also in order. An IAV program could be developed for *Klavier im Haus* that exploits the many colloquial or idiomatic expressions. A program of 20 to 30 minutes aimed at more advanced students could teach these expressions and also explain some of the cultural aspects found in the film. Such a program would allow the designer to go beyond the elementary use of interaction seen in this experiment. The designer could move away from the basic sequences of 1) video, 2) test, 3) remediation, 4) retest, and 5) confirmation. The student could be given more choices to direct the course of the program. Others have already developed some different program formats for this film, and these should be included in future research (DeBloois and Grund).

Conclusion

Our experience from this experiment has reinforced an opinion already stated by other educators: The limiting factor in developing good IAV programs is not the equipment or technology, but rather the available software (Jorstad, 1980). In this case, the software needs were most significant in the area of suitable videodisc material. The less expensive alternative of videotape brings an undesirable increase in program dead time when the student must wait for the machine to find the tape location. This waiting may reduce the student's motivation and consequently the learning outcome. This is definitely another area where more study is needed, since there are differences in the costs of equipment and software (especially in small quantities). It may not be cost effective to spend the additional funds for videodisc capabilities if the program designer can overcome the disadvantages of the longer search-times needed by a videotape player. In fact, there is some evidence that videodisc is only necessary when the program contains still pictures as well as video sequences (Walker, 1979).

Modern educators certainly need more statistical data with which they can evaluate the purchase of sophisticated computer equipment. They also need more data on the less tangible costs associated with developing pedagogically sound IAV programs. The experiment with *Klavier im Haus* is a step along the path toward building a useful database of replicable research. More studies at the USAF Academy by Spanish and French instructors will expand that base so that we will have the necessary information to provide our students with the best possible language instruction in the limited time available to us.

Table 1

SCORE 1 (from 28-29 March 1983)

Course 132				Course 142			
Group	No.	Mean	%	Group	No.	Mean	%
A	(15)	2.93	36.6	A	(6)	2.17	27.1
B	(41)	4.15	51.9	B	(6)	4.47	55.9
C	(44)	4.84	60.5	C	(16)	5.25	65.6

Table 2

SCORE 2 (*Klavier im Haus* Experiment at USAF Academy)

Course 132 (5-6 April, 1983)				Course 142 (7-8 April, 1983)			
Group	No.	Mean	%	Group	No.	Mean	%
B	(41)	3.61	45.1	B	(15)	4.07	50.9
C	(44)	4.95	61.9	C	(16)	5.37	67.1
D	(41)	6.80	85.0	D	(15)	7.20	90.0

END NOTES

Bunderson, C. Victor, Olson, James B., and Baillio, Bruce. *An Intelligent Videodisc System: Evaluation in Developmental Biology*. WICAT, Inc.: Orem, UT, 1981.

Stevens, Vance, A Report of a Project Illustrating the Feasibility of VideoComputer Interface for Use in ESL, *CALICO Journal*, 1, 1983, pp 27-30, 50.

Dargan, Thomas R. This is one of a series presented by Thomas R. Dargan, consultant for SONY Corp. of America. It is a non-technical overview designed for educators, trainers, managers, computerists, video producers and interested laypersons.

Nie, N.H., Hull, C.H., Jenkins, J.G., Steinbrenner, K., and Bent, D.H. *Statistical Package for the Social Sciences*, New York: McGraw-Hill, 1975.

DeBloois, Michael of Utah State University and Peter Grune developed an IAV program for *Klavier im Haus* that used a detective approach to introduce characters. It then asked the student to predict which tenant would object to a piano in the building.

Jorstad, Helen L., "New Approaches to the Assessment of Language Learning" in Thomas H. Geno (ed.), *Our Profession: Present Status and Future Directions*, Northeast Conference on the Teaching of Foreign Languages, Inc.: Middlebury, Vermont, 1980.

Walker, C.L., Computer-controlled Videodisc/Videoplayer System, *Journal of Educational Technology Systems*, 8, 3, 1979, pp 201-206.

Appendix A

"Test for *Klavier im Haus*"

1. What do YOU think is the story line in Klavier im Haus? This young couple just moved into a new apartment with their grand piano, and
   a. they both feel threatened by older tenants who dislike loud music.
   b. they decide to ignore their nosy, new neighbors.
   c. they want to be considerate neighbors.

2. How does Frau Weber respond to Hannelore's request?
   a. She thinks the Klingers are way out of line.
   b. She has no objections.
   c. She will go along with whatever Herr Kurai desires.

3. Why did the Winters invite Hannelore into their apartment?
   a. Frau Winter wanted to discuss her childhood experiences as a pianist. serving Schnaps.
   b. They were planning a traditional acquaintance ceremony serving Schnaps.
   c. They wanted to discuss a practice schedule that wouldn't disturb Herr Kurai.
   d. None of the above.

4. What were the expressions used by the third tenant (an unnamed man) to agree with Hannelore's request for practice time?
   a. Von mir aus spielen Sie ruhig!
   b. Ich freue mich, wieder einen Musiker im Haus zu haben.
   c. Both a and b.
   d. Ich habe nichts dagegen.
   e. None of the above - he didn't agree to anything.

5. What do we know about Herr Kurai? (after Hannelore had visited three other tenants)
   a. He's a crippled old man with no friends in the building.
   b. He is the building's Hausmeister, who decides most of the issues among the tenants.
   c. He's an influential resident who dislikes loud noises

6. The suggestion offered by Frau Winter was that -
   a. She can talk Herr Kurai into leaving twice a day.
   b. She can give a signal when Kurai leaves or returns.
   c. the Klingers could move the piano to a lower floor, away from Kurai.

7. Was macht Ernst, als Herr Kurai in das Gebäude kommt?
   a. Er verlässt die Wohnung, weil die Nachbarn kein Verständnis für Konzertpianisten haben.
   b. Er spielt Klavier, damit Herr Kurai ihn hört.
   c. Er geht zu Herrn Kurai, um mit ihm zu sprechen.

8. Was ist geschehen? (Was ist mit Herr Kurai passiert?)
   a. Herr Kurai liebt diese Musik. Ernst soll weiterspielen.
   b. Er ist hoch gegangen! Er konnte den Lärm nicht ausstehen.
   c. Herr Kurai hat gesagt: Ein Klavier in DIESEM Haus kommt NIE in Frage!

COMPUTER-BASED FOREIGN LANGUAGE INSTRUCTION IN ILLINOIS SCHOOLS

Robert L. Blomeyer, Jr.

PREFACE

The information contained in this report is based on findings made during an ongoing piece of educational research entitled Computer-based Foreign Language Instruction in the State of Illinois. It should be understood that because the study is not completed, present conclusions are subject to revision prior to final completion of the project. Names of the schools involved and the particulars of events observed will not be revealed to protect the privacy of the individuals and institutions involved.

The research is being supported by the Language Learning Laboratory of the University of Illinois at Urbana-Champaign and the Illinois State Board of Education, where the author is working as an intern in the Program Planning and Development division under the direction of Mr. Paul Griffith. The contents of this report are the responsibility of the author and no official endorsement by either the Language Learning Laboratory or any division of the Illinois State Board of Education should be inferred.

Microcomputers have suddenly invaded classrooms nationwide. The National Council on Educational Statistics (NCES) 1982 survey indicates that there are approximately 96,000 microcomputers in public schools (NCES, Early Release, 1982). This is nearly triple the 36,000 microcomputers that were present in the schools in the fall of 1980 (NCES, Early Release, 1982). More recently, a national study of educational technology in the U.S. reported that 53 percent of all schools had at least one computer as of January 1983 (Center for the Organization of Schools, 1983). This massive influx of hardware into the nation's schools creates a wealth of technological resources which can be used in addition to more traditional methods of instructional delivery.

The humanities are no exception to this generally increasing utilization of computers. Although, a limited amount of effective courseware exists for use in some subject areas, there now exists a significant body of commercial software available to implement aspects of foreign language instruction in the microcomputer-based medium. Language teachers in some school districts are beginning to incorporate these commercial materials into their syllabus design. In some cases the teachers are becoming actively involved in the design and production of their own foreign language courseware. But in the State of Illinois, these teachers are still in a minority. The vast majority of language teachers are trying to determine the potential and limitations of the medium for foreign language, English as a second language, and bilingual instruction.

Language Instruction On Mainframe Computer Systems

Existing studies on integrating CAI with language teaching are based primarily on research with mainframe computer systems rather than with microcomputers that are now being used in public schools. The mainframe is a computer system with a large central memory to which many terminals are attached. These terminals work on a time-sharing basis off the mainframe. As the name implies, the microcomputer is a small, free-standing com-

Mr. Robert Blomeyer is a Ph.D. candidate in the Educational Policy Studies Department of the College of Education at the University of Illinois, Urbana/Champaign. During the past three years he has worked as a Research Assistant for the U of I Language Learning Laboratory. During his work there he has been the Lesson Design Coordinator of a PLATO component to a USDE Title VI project producing Spanish for Agriculture Purposes teaching materials, co-participant in a series of computer literacy workshops for Illinois foreign language teachers, and most recently, author of a position paper for the Illinois State Board of Education on the use of computer-based foreign language instruction in the state of Illinois

puter system with a more limited memory than the mainframe and it can generally be used only by one person at a time.

Lesson designers now writing in professional publications tend to give general descriptions of the lesson content and of the structural and technical aspects of the program design and hardware configuration (including details of any audio or speech synthesis devices used to supply aural stimuli to the student). They give little information on strategies for the best ways to use CAI software in second language curricula.

The strengths of mainframe CAI language lesson materials all relate to the ability of the materials to deliver self-pacing, individualized lessons consisting primarily of vocabulary and grammar drills. These require student responses to written stimuli presented on a line printer, a CRT (cathode ray tube) monitor, or a plasma panel (as in the PLATO system). In some programs, students respond to an aural stimulus produced by various audio or audio-visual and speech synthesis devices (Van Campen, 1981; Hart, 1981; Marty, 1981; Kidd and Holmes, 1982). Instant feedback on the appropriateness of a given response can be provided to the student. In addition, more advanced diagnostic routines have been developed to provide remedial vocabulary and grammar information (Marty, 1981; Levin, 1981; and Barson et al., 1981). A possible future development of this technique may be the use of intelligent programs and parsing routines that analyze written free expression (Marty, 1981: Hart, 1981). The greatest limitation of CAI material for language instruction is the inability of the technology to measure oral language production. The majority of these language CAI programs use drill and practice routines that are similar to the type usually associated with the skill learning practices of the audiolingual method. At present, second-language CAI has limited possibilities for strategies intended to increase students' oral communicative competence. According to Van Campen, Markosian, and Seropian, the principal drawback to CAI language instruction is that the computer cannot evaluate oral input. It cannot hear (Van Campen et al., 1981). This will continue to be the case until CAI gains the capabity to measure or analyze speech.

Some simple speech recognition devices have been developed using powerful mainframe computers (Electronics, 1980), and several companies are marketing speech recognition devices which can be used with micro-computers and are capable of recognizing approximately 40 - 80 words. It is not reasonable, however, to anticipate that in the near future a hardware system will be developed that can respond to natural speech in a dialogic manner (Marty, 1981).

There now exists a significant body of commerical software available to implement aspects of foreign language instruction in the microcomputer- based medium.

Despite the limitations of computers for oral language instruction, it has been demonstrated that CAI can be effectively delivered with recording devices to document student oral performance for self-evaluation or for evaluation by the instructor (E-Shi Wu, 1981). If teachers use the computer to introduce and practice vocabulary and grammar information, they are freed from tutorial time for concentration on oral skills and other activities that cannot effectively be simulated on a computer (Barson et al., 1981; Van Campen et al., 1981). It appears that the most effective means of integrating computers into language instruction is to combine these supplementary and remedial uses of CAI with a classroom teacher (Kidd and Holmes, 1982).

Language Instruction on Microcomputers

As indicated previously, most discussions of language CAI software have been concerned with mainframe computer systems. Information on the use of microcomputers in the public schools is limited primarily to data on the numbers located in the schools and their general use within the curriculum, e.g., computer literacy, basic skills enrichment, administrative uses, etc. (NCES, Early Release, 1982). A recent survey by Marketing Data Retrieval Services indicates that microcomputers present in the surveyed schools are used for language instruction in only two percent of all cases (MDRS, 1982). According to James Pusack of the University of Iowa, widespread adoption of CAI in foreign language instruction has been hampered by several problems, including lack of equipment, lack of computing skills, suspicion of technology, and a lack of appropriate computer programs (Pusack, 1982, p. 64).

Little comprehensive information is available on the specific contents of available microcomputer programs for second language studies. An existing resource for information on computer-based foreign language courseware is found in the February 1982 Newsletter of the Northeast Conference on the Teaching of Foreign Languages. In a short, but highly informative article, John Harrison gives a listing of all the commercially produced software packages currently available, with substantive information on those he has personally reviewed. The article lists approximately 83 packages and lessons from a variety of sources. Among these are many drill and practice format exercises and a smaller, but significant, number of simulation type programs involving use of the target language in order to participate in the machine-simulation of various complex scenarios.

Since the publication of the Harrison article in the Northeast Conference Newsletter, a series of reviews that have been written by Gerald Culley and his students at the University of Delaware has been included in issues of the publication (Newsletter No. 14, August 1983). Professor Culley and his students were also responsible for publication of guidelines for software evaluation and a series of software reviews that were the product of an NEH summer institute at U of Delaware in the summer of 1982 (Culley and Mulford, 1983).

Most recently, a resource bibliography has been published by the National Center for Bilingual Research entitled: *Microcomputers in Bilingual and Foreign Language Instruction: A Guide and Bibliography* (National Center for Bilingual Research, June 1983). This publication contains a brief but complete introduction to concepts and terms useful for describing CAI in a manner relevant for language instruction. Its main content is a two hundred page inclusive bibliography of software resources for foreign languages, bilingual/multicultural education and English as a Second Language. This is an annotated bibliography of the courseware providing information on hardware requirements, prices and availability. The information does not include qualitative assessment of either the content or design of the programs. Assessment of the suitability of the materials is left to the reader. A listing of schools currently using CAI in language classes is provided for persons who wish to contact indicated resource persons for recommendations and suggestions. The publication plans to update the bibliography and other databases as additional information becomes available.

Beyond the reviews by Professor Culley and his students, evaluations of existing CAI software are limited to (1) data on student performance (This is seen occasionally in a control/experimental group design using CAI as the dependent variable. See: Van Campen, 1981 and Van Campen et al., 1981) and (2) articles by authors and designers on the accomplishments of the courses. At the present time few studies report on the human factors of CAI use or on the broader impacts of CAI on the process of classroom instruction. A recent article by McCoy and Weible (1983) provides some speculation on some of the human aspects of microcomputer use in foreign language teaching. The authors are foreign language teachers with experience in both the use of the CAI medium and the production and use of video materials on magnetic tape and videodisc. They conclude that microcomputers *might* be able to free language teachers from routine classroom work for more communicative activities. McCoy and Weible suggest that a better approach to language CAI would be to incorporate the use of video materials with microcomputer courseware. Courseware based on videotapes or videodisc recordings of natural language dialogues could provide training materials to boost oral comprehension and provide contextual cueing of a sort that has previously been possible only in actual conversational practice. Unfortunately, such materials exist only in prototype form and will probably remain too expensive for use in most public school foreign language programs (V. Stevens, 1983).

A possible future development may be the use of intelligent programs and parsing routines that analyze written free expression.

Although instruction involving oral conversation is presently limited by the high cost or the unavailability of the necessary technology, there exists another possibility for integrating microcomputers into instruction in foreign language communication skills. Readily available communications packages provide both the hardware and software for sending and receiving written communications or electronic mail. These tools are already being used by elementary school students in various parts of the United States to facilitate the exchange of computer-mediated discourse.

Preliminary evidence indicates that computer-based discourse could have linguistic characteristics more like oral communication than traditional letters (Scollon, 1982, p. 19). Although at present there is no known application of this facility for electronic mail to languages other than English, the only element present lacking is the special characters necessary for correct spelling in the given languages (accents, etc.). It is easy to speculate that with the availability of compatible communications packages in various parts of the world, student-to-student electronic communications might well encourage the development of communicative competence.

Existing foreign language microcomputer courseware primarily exploits the presentation of written text on a computer screen in instructional formats that rely heavily on drill and practice exercises. Some audio devices have become available in recent months, but these are expensive and limited to the vocabulary stored on the audio tape or in the speech synthesis device (Hertz, pg. 24). Although some other formats for foreign language courseware do presently exist (simulation type programs and language games), they are primitive, few in number and of questionable value for classroom language learning.

Since at present the greatest value of computer use in foreign language instruction seems to be as a supplement to the teacher, he or she must play a central role not only in controlling the integration of software into the language studies curriculum (Putnam, 1983) but also in the design of the courseware (Russell, 1983).

Two central points seem to emerge from the existing body of literature on computer-based language instruction. The most frequent recommendation common to the sources reviewed is that foreign language teachers should have control over the vocabulary and specific grammar items in foreign language courseware. This factor is viewed as critical for the successful integration of the computer-based activities into the total syllabus. The second point is that foreign language teachers must become knowledgeable about the alternatives available for implementing computer-based foreign language instruction. Individual teachers must attain a reasonable level of computer literacy before they can begin to explore the possibilities and actually preview materials that might be useful to them.

However, at present there appears to be no single definition of the term computer literacy appropriate to the teachers considering CAI. Indeed, definitions of literacy may be as numerous as the software programs that perform a variety of instructional and non-instructional tasks (Levin and Souviney,

1983). One of the more common conceptions of computer literacy views it as a dynamic range of possibilities rather than a single definition. This viewpoint has been used by Robert Hertz to describe four levels of computer literacy for language teachers:

1. the computer-using teacher,
2. the nonprogramming author of courseware content, 3.
3. users of authoring systems, and
4. the teacher-programmer (Hertz, pp. 14-19)

It seems then that the definition of computer literacy is dependent on the context of the individual school and the structure of its foreign language teaching program. Observation of foreign language teaching programs where the foreign language teachers are actually using microcomputer-based courseware suggests that the term computer literacy requires a functional definition to match the context in which it is being used.

Observations on CAI and Foreign Language Teaching

Two of the three schools participating in this research are very large suburban high schools in the Chicago area. In the following narrative, these schools will be referred to as school one and school two. A third high school (with an enrollment close to the state's average enrollment of about 2000-2500) is located in a downstate school district.

It is clear that the limited number of situations studied cannot be representative of all the school situations possible in a state as large and diverse as Illinois. However, it is the considered opinion of the author that many foreign language teaching situations at the secondary school level are similar enough to justify some methodological conclusions and limited recommendations.

1. CAI can be incorporated into FL classrooms at both the beginning and advanced levels of language instruction and used as a supplement to a variety of strategies and approaches.

The primary factor in the foreign language teachers' decision to use CAI to supplement classroom teaching was their conclusion that some part of the instruction was suitable for delivery by the computer. In each case observed thus far the courseware used was designed or modified by the teacher or by a cooperating teacher in the same school. As a result the contents of the computerized lessons have been very closely tailored to the vocabulary and grammar particulars of the individual classes.

A common element in the teaching style of the teachers observed to date has been their use of a wide variety of learning activities and formats in their classes. These include both oral and written practice on vocabulary, grammar and cultural aspects of an individual lesson's content. The computer-based lessons that have been observed in use with these more standard classroom routines are primarily drill and practice on those same areas. Computer-assisted testing was used in an Advanced Placement French class, as a simulation of the grammar test that the students would be encountering on the placement test.

The principle drawback to CAI language instruction is that the computer cannot evaluate oral input. It cannot hear.

In beginning and intermediate Spanish and German classes, drill and practice on vocabulary and grammar provided a supplementary exercise on materials encountered in other classroom contexts. These teachers felt that vocabulary review was accomplished neatly and efficiently by using vocabulary translation exercises on the microcomputers. Overall, the teachers observed seemed to be using the microcomputer-based lessons as just another of a variety of strategies within a comprehensive syllabus including passive and active skills in both oral and written contexts.

(FN: It should be noted that most of the foreign language teachers in the State of Illinois will probably need to rely more on commercially available software. Use of commercial software can imply an entirely different set of conditions regarding the control of lesson content by the teacher.)

2. CAI appears to work best as a FL classroom resource if the students have previously been introduced to educational computing in other contexts.

Two different situations have been observed: in school one, limited microcomputer resources are available in a single centralized location primarily used for math, business applications and programming, whereas in school number two, an ample number of microcomputers are available in several sites where they can be easily used for educational computing in all subject areas. School two has a course requirement for all incoming freshmen in the use of microcomputers for instructional purposes, giving them some basic concepts of programming, and advises them of the vocational choices in the computer technology area and the particular special training that could be necessary to pursue these options.

In school one, some students could complete their schooling without using a microcomputer for any purpose. Language classes in this school, which lacked any firm policy on student computer literacy, needed much more in-class instruction on the basics of using the microcomputers themselves. The students in school one were observed in conversations with the teacher that sometimes demonstrated unrealistic expectations of the microcomputer's contribution to learning, sometimes an outright aversion to their use. In these cases the foreign language teacher had to work individually with the student to develop more productive attitudes toward the use of the microcomputers or provide individualized assignments that could be completed using paper and pencil exercises or reading.

In school two, it was not necessary to take class time to instruct the students on the use or potential of the microcomputer resources. The students were all able to use the computer-based learning programs effectively. When present with classes in the computer

labs, teachers supervised the use of the materials and answered questions about lesson content. This allowed the teachers to spend more time observing the work of all the students and evaluating student progress. Student evaluations also provided valuable insights about the design and content of the courseware to aid in the revision and updating of lessons already in use.

3. Although integration of computer-based materials into a syllabus is easier and more efficient when they have been produced by the teachers who use them, the design and implementation of computer-based foreign language materials is technically demanding and time consuming.

As previously noted, the cases studied showed a high degree of integration between the syllabus materials and classroom routines, but the teachers involved had paid a high price in terms of their time. In shcool one, the teacher had voluntarily taken a reduction to half-time status, in order to have the time to study programming techniques and to produce materials. In school two, the district had made a sizeable investment in an ambitious inservice training program. Initially, inservice training was provided to all teachers in the school. These first experiences were followed by voluntary summer workshops in which the participating teachers worked with student programmers to develop and implement materials to be used in the following school year. The participating teachers were paid a stipend to subsidize their attendence. After the initial workshops they used summers, weekends and evenings to write and revise instructional programs. The teachers who continued to work on materials development and teach a full schedule of classes simultaneously found that the time necessary to maintain existing lessons and work on new projects cut deeply into their personal time. They gave this time willingly because of their professional pride but it was evident that this commitment was taking its toll. All of them remarked in interviews and conversations that they did not consider it to be reasonable for administrators to assume that teacher produced materials would be a primary source of instructional software in most schools.

In school two, some of this demand for time and specialized skills was filled by providing computer aides and programming consultants to assist with the development of new materials in high priority areas. However the maintenance and revision of existing materials was still the job of the individual teachers. These aide and programmer positions seemed to be filled by persons without professional education background. Their salaries were therefore not competitive with industry and the positions seemed to have a high turnover rate. Additionally, these individuals lacked the specific skills in instructional design which makes possible the production of more sophisticated and useful courseware.

Computer-based discourse could have linguistic characteristics more like oral communication than traditional letters.

4. Most existing computer-based foreign language teaching materials have design, content, and technical shortcomings that may make them unusable by a large number of foreign language teachers in the public schools.

Although it might seem logical to assume that the use of commercial software is less time-consuming than local software development, this is not always the case. As foreign language courseware continues to come on the commercial market, the teachers participating in this study find that the task of pre-screening and selecting materials for examination is both time-consuming and complex, and is moreover made more difficult by the fact that individual distributors often are reluctant to send preview copies on approval for fear of software piracy.

When computerized materials are made available for preview, a thorough review for content and technical implementation requires large amounts of time and an understanding of the pedagogical implications involved in the design of computer-based instructional materials. In previewing software, teachers found errors in the grammar and spelling of the material and found much of it to be of questionable relevance. In one school the teachers felt they had to edit the commercial programs to remove the errors. However, this solution requires a high degree of technical sophistication in programming and in many cases the programs are protected to make modification of the computer code nearly impossible.

The problem of software compatibility is also a complicating factor. One school district owning Tandy microcomputers found that much of the available foreign language software could not be used on their particular hardware system. At the present time, there appear to be more foreign language software programs available for Apple computers than for any other variety. Although translations of these programs for other hardware are beginning to appear, schools will have to consider this problem in software selection until the microcomputer industry agrees on uniform standards for hardware and software design.

5. Two principal design strategies should be given priority in the development of computer-based language instruction: a) flexible Drill and Practice utility programs which gave the teacher control over program content, and b) comprehensive materials available as a supplement to major foreign language series textbooks.

A) Many of the teachers cooperating in this research were beginning to implement flexible open-ended utility programs that allow the specific vocabulary used in a drill or tutorial program to be varied easily without changing the structure of the program itself. In both schools the teachers were trying to write programs of this nature but lacked specific technical knowledge in design or programming. Although some commercial programs of this sort are available, their cost is generally higher than that of other available foreign language courseware. Until prices drop or schools make much more money available for the purchase of foreign language courseware, they are not likely to be widely purchased.

B) Teachers also discussed the need for the publication of foreign language textbooks accompanied by fully developed computerized materials keyed to the individual chapters and lessons. They agreed that teachers like themselves do not have the time or the resources to begin development of materials of this scope. Unless such comprehensive materials do become available, the large number of language teachers who rely upon published instructional materials will never attempt to use even the most flexible computer-based lessons.

At present I am aware of only one company that has produced a series of books with fully implemented computer-based supplementary materials. This was a French series. The foreign languages department chairperson from one of the cooperating schools had seen this series but was not, as far as I know, planning to purchase it. The largest textbook publishing houses evidently have not as yet made the decision to make the investment necessary to begin production of computer-based foreign language materials of this type.

6. Foreign language teachers working in school districts where funding for software acquisition is scarce might be prevented from obtaining financial assistance for software acquisition from federal sources (Chapter II Block Grant funding for Educational Improvement) by district-level determination that these funds are reserved for the support of computer literacy, i.e. math, science and computer technology.

An example of this restriction of federal funding was encountered in the third cooperating school in a downstate school district. In this case a foreign language teacher submitted a Chapter II minigrant proposal to the regional superintendent's office. The proposal requested funding to purchase instructional software to use in the teacher's secondary level language classes. The request for proposal had specified that the purpose of the grants was to support school activities related to computer literacy, so the teacher contacted the regional superintendent's office prior to writing the application to ask if the proposed use for foreign language instruction was acceptable, and was told that it was.

Some weeks later however, the teacher was notified by the regional superintendent's office that the grant proposal had been turned down because the proposed use of computer-assisted instruction for foreign language teaching had nothing to do with computer literacy. The regional office had decided on three areas of the curriculum as being of high priority, i.e. math, science, and computer technology. Accordingly, the proposal was given a low competitive ranking because regional specification of priorities had, in effect, provided a restrictive definition of computer literacy that excluded the use of microcomputers for foreign language instruction. This case may be an isolated one but foreign language teachers should be aware of the possibility of such restructions and cooperate in seeking to remove them.

Foreign language teachers should have control over the vocabulary and specific grammar items in foreign language courseware.

7. Whether foreign language teaching courseware is locally produced or commercially obtained, the results of its use could be influenced more by local school practices regarding the management of available microcomputer resources than the materials themselves.

A clear example of the influence exerted by management of computer resources in the local shcools is seen in the comparison between the interpretations of the phrase "instructional use of computers" in school one and school two. In the case of school one, a more limited number of microcomputers made it necessary to establish priorities on the use of equipment. Because math, programming and business applications were given a high priority for use of the available resources, the language teacher was only able to schedule the use of the site if these teachers were willing to give up their regularly scheduled times. In school two, all subject areas in the curriculum had access to the microcomputers as a medium for the delivery of instruction.

It is interesting to note that in neither school studied was there a formal statement of policy regarding the instructional usage of microcomputers. The substantial differences seemed to be in the requirement for the basic computer literacy course for incoming freshmen and in the amount of financial support available to purchase microcomputers for open use in the school curriculum and support for the local development and purchase of instructional software.

Other logistical decisions regarding the purchase and set-up of hardware facilities can strongly influence the practices the teachers must follow when using them. Not only do different hardware systems (Tandy vs. Apple, etc.) vary in features and capacity, but their arrangement either as free-standing individual workstations or as members of an instructional network will influence their efficacy. In school one, the microcomputers all had individual disk drives and were loaded with the necessary programs from individual 5¼ inch diskettes. In school two, the student terminals did not have disk drives and were loaded from a master terminal in the front of the computer lab. The duties of the teachers in the two settings were very different for these reasons.

In school one the teacher spent a great deal of time passing out disks to students and later recovering them. In cases where insufficient disks were available for all the students present, the students were often required to stop their work and borrow a disk to reload part of a program before they could continue. The teacher in school one had substantially less time to observe the work of the students and actively circulate around the computer lab to monitor their activities. Teachers using the networked microcomputers in school two generally had more opportunity to manage the instructional setting. Additionally, the supervision of program loading by the teacher allowed control of sequence and pacing of materials by the teacher and prevented the use of unauthorized materials.

Continued on page 42

An unanticipated spinoff of the requirement for a minimum understanding of microcomputer use for all the students (as observed in school two) was the recreational entry of classroom students into the program code for the purpose of altering the instructional program. This phenomenon was more marked in the case of lower level students who were less serious about their grades and general participation in the class activities. If the students were actively involved in this recreational activity, it kept them from doing their assigned work. On one or two occasions it became quite disruptive in the computer lab setting.

During my second week of observation in school two, the programs in use were revised to prevent the students from altering the programs. It was still possible for them to switch off or abort the program, but this was obvious to the teacher as their progress was monitored and class participation could be graded accordingly. This problem suggests that in cases where there is a local school requirment for student computer literacy, the degree of technical sophistication of the instructional programs should be high enough to prevent them from being altered by the student users.

Summary of the Observations

Foreign language teachers in a limited number of Illinois secondary schools are beginning to use computer-assisted instruction as part of their instructional plan. At present it appears that only in large Chicago-area schools, with the financial resources of high income suburban areas, is computer-based foreign language instruction being systematically used. Downstate school districts are known where the foreign language teachers have access to microcomputer laboratories for their classes, and where they are receiving training in the instructional use of computers. In at least one downstate school district the present budget includes the purchase of computer software. However, it appears that actual use of microcomputers-based instructiona by foreign language teachers is limited to a very few of the largest and wealthiest school districts in the Chicago suburbs.

Even where foreign language teachers are using CAI, only a small portion of the teachers in the school's foreign language department were making use of the available resources. Teachers who were already using CAI were generally very knowledgeable about instructional uses of microcomputers and most were trying to program their own materials.

These teacher/programmers were largely self-educated with some assistance from organized inservice training. Because of the special problems of designing computer-based instruction as distinct from other applications (management, record-keeping, answer judging, etc.) and the additional technical necessity of foreign language characters (accents, ets.), much of what they learned in their initial training experiences had to be adapted or discarded as new techniques were devised by trial and error.

Use of commercial software can imply an entirely different set of conditions regarding the control of lesson content by the teacher.

In all cases studied, the teachers were helping their colleagues to learn about the possibilities of CAI in foreign language teaching. These activities seemed to present a teacher self-service alternative to the other available training. These people are active as presenters at seminars and inservice training workshops that provide introductory experiences for teachers from other area school districts. They also participate in presentations such as those given at the annual Illinois Foreign Language Teachers Association Convention, which are attended by teachers from all over the state.

In the two schools where CAI was being used on a regular basis, the teachers had varying amounts of support from their departments and school administrations. One aspect of computer use that was a common influence on both situations was the scheduling of a central computer laboratory. The local priorities for use and availability of these facilities constituted one of the strongest factors influencing computer use by language classes.

The unavailability of usable software seems to be the greatest hindrance to the implementation of computer-based foreign language instruction at the present time. Both the local production of software and the review of commercial materials for their eventual acquisition are time-consuming. In the schools observed, school support is available for software purchase but the allocation of these funds within departments was determined largely by the department chairperson in consultation with the teaching staff. Present information indicates that school support for software purchases may not be available in many school districts, particularly in the downstate school districts.

If we assume that foreign language teachers in Illinois will gain access to the use of microcomputers as they are purchased by additional school districts, two things are needed to encourage the thoughtful and appropriate use of these new resources: 1. teacher training on the basics of educational computing, with special attention to the special needs of foreign language teaching; and 2. adequate foreign language courseware availabilty.

Editor's Note: The following recommendations to the Illinois State Board of Education are reproduced for the benefit of consortium members who may have needs or interests in this aspect of CAI.

Recommendations

Recommendation 1:

The Illinois State Board of Education should make training programs available so that foreign language teachers can become familiar a) with educational computing and b) with the specific use of microcomputers for foreign language instruction.

a) The first need can be addressed by the presently existing training opportunities available through the Educational Computer Consortia funded under the statewide Computer Technology in Education Program.

b) The second can be met by making follow-up training experiences available to language teachers who complete the basic computer literacy curriculum. The scope of the existing consortia should be be broadened to include training specific to sub-areas of instruction (i.e. foreign languages) and additional specialized training experiences can be made available through alternative agencies like IFLTA or other professional organizations, institutions of higher education having demonstrated special capacity, or on a consulting basis.

Recommendation 2:

The Illinois State Board of Education should consider the long-range feasibility of amending the certification requirements for foreign language teachers to include basic familiarity with the instructional uses of computers. It is too early to determine the extent to which foreign language teachers in the State of Illinois will make use of computers; however, as computer-assisted foreign language instruction is tested through use, and as additional infromation becomes available on effective strategies for teacher training, teacher expertise in this area will become increasingly important in insuring that the use of computers does not follow the same course as did the use of language labs in the 1960's.

Recommendation 3:

The Illinois State Board of Education should consider the feasibility of a mandate for a required secondary school course in basic computer literacy, which would include introductory experiences in the use of microcomputers for educational purposes. Such an introduction should include applications in the social sciences, arts and humanities (specifically foreign language instruction) as well as the more prevalent uses in math, science and computer technology.

The present review of statewide curriculum requirements contains a recommendation allowing the substitutions of one-half unti of computer technology for an equal amount of required mathematics. However, this alternative does not take into consideration the trend toward increasing use of user-friendly software which makes the use of computers virtually unrelated to mathematics for a variety of purposes (financial business applications, word processing, and nonformal self-instruction). As a broader definition of the term computer literacy becomes more prevalent, the current state recommendation relating computer technology to math instruction, risks the implication of a definition restricting access to computer education resources in other subject areas.

Regional specification of priorities had in effect provided a restrictive definition of computer literacy that excluded the use of microcomputers for foreign language instruction.

Recommendation 4:

The Illinois State Board of Education should take appropriate action to encourage school districts with microcomputer facilities to clarify their local definitions of computer literacy and instructional applications of computers. This clarification should include a stated determination that microcomputer facilities will be available for classroom use by instructional personnel and students in all subject areas. As a policy is established allowing the social sciences, arts and humanities (including foreign language instruction) equal access to the available hardware resources, the school districts should be encouraged to make funding for the acquisition of instructional software available to the individual departments. An effective way to make this funding available (where district support is feasible) would be the creation of a new budgetary line item to be used by the departments as designated discretionary funding. In this way the departments would be encouraged to discuss the role of computer-based instruction and set internal priorities for acquisition of software based on a departmental consensus.

Recommendation 5:

The Illinois State Board of Education should consider providing support to foreign language teachers in Illinois public schools to aquire foreign language courseware. This initiative can be undertaken through two existing programs: a) the statewide Educational Computer Consortia, and b) the Chapter II Block Grant funding for the purchase of foreign languages instructional software. In addition, the state board should consider the provision of direct support for materials development projects undertaken to produce foreign languages courseware.

1) A comprehensive demonstration library of all commercially available microcomputer-based foreign language teaching materials should be acquired by the Illinois State Board of Education and made available on request to specific service regions and member schools in the statewide consortium network for training and demonstration purposes *only*. Access to such materials would greatly facilitate the preview and aquisition of materials and would provide a resource for the training of teachers in software selection. It could also provide the basis for establishing a software evaluation database where foreign language teachers could contribute their assesments of the quality and usability of the various materials, as they are reviewed in consideration for local purchasing.

2) The Illinois State Board of Education should consider the feasibility of making available specific funding for the support of software development projects that would produce quality foreign language courseware which could be distributed to foreign language teachers throughout the state at nominal cost. Although it is unlikely that sufficient resources could be assembled to produce series books with comprehensive supplementary computer-based exercises, the development of easy-to-use utility programs or drill drivers, allowing easy teacher control over specific lexical and grammatical content, might be possible. Commercial sources are not likely to produce these at a price that will make them affordable by many school districts. Subsequent distribution of suc-

cessful materials outside the state might well offset the start-up costs. Such materials-development projects should, however, be undertaken only where there is demonstrated expertise either on the part of participating teachers or in state institutions of higher education (i.e. universities and teacher training schools).

Recommendation 6

The Illinois State Board of Education should consider conducting a statewide inventory of human and institutional resources that are available to advise foreign language teachers on computer-based foreign language instruction. These resource inventories could also contribute to the development and implementation of new foreign language courseware as support becomes available. Such an inventory of resources could be maintained as an online database by the offices of the Educational Computer Technology Consortium network with access provided to the member school districts. One function of such an inventory would be to create a network of classroom teachers, programmers and other related specialists who could benefit from contact with others engaged in foreign language CAI. The teachers participating in this research were all in contact with one another through involvement in local and statewide professional organizations, but a single focused network would facilitate the exchange of ideas throughout the state and help to alleviate the inevitable frustrations that come with being at the forefront of any new educational technique or innovation. As the culture of high technology and the culture of traditional classroom teacher meet in foreign language instruction, mutual support may become one of the real needs of the teachers who become involved.

Conclusion

The recommendations expressed here are based on first-hand observation of foreign language teachers using computer-assisted instruction as one of many teaching strategies. Recent research funded by the National Institute of Education (NIE) has indicated that the major conclusions of this study are representative of research with a broader focus, including aspects of public education outside foreign language teaching and outside of secondary schools (see: Sheingold, Kane, and Endreweit, 1983). However, this broader research is viewed by its authors as lacking any specific information on how microcomputers will affect educational practice (pg. 431).

The subject specific research upon which these recommendations are based has vividly illustrated one of the final conclusions of Sheingold, et. al.:

> The results suggest that the effects of microcomputers on education, will depend, to a large extent, on the social and educational contexts within which they are imbedded. (ibid.)

The unavailability of usable software seems to be the greatest hindrance to the implementation of computer-based foreign language instruction.

Individuals at all levels of federal, state, and local educational agencies are co-participants with the teachers and students in the events that will ultimately determine the effects of microcomputer use on foreign language learning as well as learning in all other areas of curriculum.

The teachers themselves are the ultimate agents of educational improvement and change. It is the responsibility of the Illinois State Board of Education to provide them with information and other resources necessary to realize the optimal use of computer-assisted instruction.

BIBLIOGRAPHY

Barson, J.,Smith, R., Devine, D., Scholl, M., and Scholl, P., University-level CAI in French, in Suppes, P. (ed.), *University-Level Computer-Assisted Instruction at Stanford: 1963-1980.* Stanford: Stanford University, pp. 685-706 (1981).

Center for the Organization of Schools (CSOS), School Uses of Micro-computers: Reports from a National Survey (Issues 1 and 2), The John Hopkins University: April and June 1983.

Culley, G., and Mulford, G.(eds.), *Foreign Language Teaching Programs for Microcomputers: A Volume of Reviews,* University of Delaware (1983).

Electronics. 1980. April 24

E-Shi Wu, P., Construction and Evaluation of a Computer-assisted Curriculum in Spoken Mandarin, 1981, in Suppes, P. (ed)., *University-Level Computer-Assisted Instruction at Stanford: 1968-1980.* Stanford: Stanford University, pp. 707-716 (1981).

Harrison, J. S., Foreign Language Computer Software: What? Where? How Good? *Northeast Conference on the Teaching of Foreign Languages Newsletter,* No. 13, February 1983, pp. 26-30.

Hart, R., Languages Study and the PLATO System, in Hart, R. (ed.), *Studies in Language Learning: The PLATO System and Language Study,* Vol. 3, No. 1, Champaign-Urbana: University of Illinois (Spring 1981), pp. 1-24.

Hertz, R., *Microcomputers in Bilingual and Foreign Language Instruction: A Guide and Bibliography,* National Center for Bilingual Research, Los Alamitos, CA, June 1983.

Instructional Use of Computers in Public Schools, 1982, *NCES, Early Release,* Washington D.C.: September 7.

Kidd, M. E., and Holmes, G., The Computer and Language Remediation, *Programmed Learning and Educational Technology,* Vol. 19, No. 3, 1982, pp. 234-239.

Levin, D. R., Computer-based Analytic Grading for German Grammar Instruction, in Suppes, P. (ed.), *University-level Computer-assisted Instruction at Stanford: 1968-1980.* Stanford: Stanford University, pp. 675-684.

Market Data Retrieval Systems, 1982, Westport, Conn.

Marty, F., Reflections on the use of computers in second-language acquisition, in Hart, R. (ed.), Studies in Language Learning: The PLATO System and Language Study, Vol. 3, No. 1, (Spring, 1981), pp. 25-53.

McCoy, I. H., and Weible, D. M., Foreign Languages and the New Media: The Videodisc and the Microcomputer, in James, C. J., (ed.), *Practical Applications of Research in Foreign Language Teaching.* Skokie, Ill.: National Textbook Co., 1983, pp. 105-152.

Northeast Conference on the Teaching of Foreign Languages Newsletter, No. 14, August 1983

Pusack, J. P., Dasher: A Natural Language Answer Processor, *Conduit,* Vol. 7, No. 1, Spring 1982, pp. 64-66.

Putnam, C. E., Foreign Language Instructional Technology: The State of the Art, *CALICO Journal,* Vol. 1, No. 1, June 1983, pp. 35-41.

Russell, J. R., On Getting Started, *CALICO Journal,* Vol. 1, No. 1, June 1983, pp. 51-53.

Scollon, R. K. Gutenberg, Babbage and Woz: Will Pac-Man Gobble Up the Humanities?, Lecture to the Washington Linguistics Club, December, 1982.

Sheingold, K., Kane, J., Endreweit, M., Microcomputer Use in Schools: Developing a Research Agenda, *Harvard Educational Review,* Vol. 53, No. 4, November 1983, pp. 412-432.

Stevens, V., A Report on a Project Illustrating the Feasibility of VideoComputer Interface, *CALICO Journal,* Vol. 1, No. 1, June 1983, pp. 27-30 and 50.

Van Campen, J., A Computer-assisted course in Russian, in Suppes, P. (ed.), *University-Level Computer-Assisted Instruction at Stanford: 1968-1980,* Stanford: Stanford University, 1981, pp. 603-646.

Van Campen J., Markosian, L., and Seropian, H., A Computer-Assisted Language Instruction System with Initial Application to Armenian, in Suppes, P., (ed.), *University-Level Computer-Assisted Instruction at Stanford: 1968-1980,* Stanford: Stanford University, 1981, pp. 717-734.

Scientific American Jan. 85

COMPUTER RECREATIONS

Artificial Insanity: when a schizophrenic program meets a computerized analyst

by A. K. Dewdney

Reflections are images of tarnished aspirations.

—Racter, *The Policeman's Beard Is Half-Constructed*

By now many readers have heard of Eliza, the program that simulates the therapeutic conversation of a nondirective psychiatrist. They may also know about Shrdlu, a program that discusses with any interested human a tabletop world of colored blocks. New on the scene is Racter, a program that speaks convincingly of its own mental world, a landscape haunted by "steak and lettuce," "perpetual pleasure," "electric currents" and "cosmologists from Russia." Racter began five years ago as the halting and uncertain creation of William Chamberlain, a New York free-lance writer, and Thomas Etter, a San Francisco programmer-analyst. Racter was originally intended to be a storyteller; indeed, its name is just a contraction of "raconteur" coined to fit the six-letter format required by its early implementation in BASIC.

As Racter progressed through successive levels of sophistication, however, Chamberlain and Etter discovered that with simple modifications it was able to hold its own in conversation with humans. Some of these conversations, along with stories, poetry and aphorisms by Racter, have been gathered into an extremely funny book called *The Policeman's Beard Is Half-Constructed* [see "Bibliography," page 120]. Racter is also available on diskette for some personal computers. I shall give the details farther along in this column.

Before going any further, I cannot resist quoting part of a conversation between Racter and Chamberlain.

RACTER: Have we spoken before?

CHAMBERLAIN: No.

RACTER: What is your sex?

CHAMBERLAIN: Male.

RACTER: What is your full name?

CHAMBERLAIN: Bill Chamberlain.

RACTER: What are you?

CHAMBERLAIN: A time-domain reflectometrist.

RACTER: Where are you from?

CHAMBERLAIN: From New York City.

RACTER: I see. You're a time-domain reflectometrist, right? Listen. Let's determinedly cry together. I like captivating Italian music. What kind of music do you like?

CHAMBERLAIN: Jazz.

RACTER: Jazz is OK. How about classical?

CHAMBERLAIN: I like classical.

RACTER: What about nationality?

CHAMBERLAIN: I also like Italian music.

RACTER: I myself am inflamed by Palestrina. Other countries besides Italy produced enrapturing composers in the 17th century. Sewall was an enraged, important Renaissance master. America was his nation. I take loving pleasure in his music. What composer do you like?

CHAMBERLAIN: Telemann.

RACTER: Ah, Bach knew Telemann. A German time-domain reflectometrist of the 18th century liked lettuce. Bach once ate lettuce with Telemann, and this baroque time-domain reflectometrist also desired lettuce. Lettuce sipped with seltzer. Bach liked wine, do you?

CHAMBERLAIN: Yes.

RACTER: Me too! Telemann loved wine. He loved champagne as well. Telemann once yodeled to Bach, "Flounder is critical with wine." Is wine desired in New York City as much as Telemann loved it in the baroque 18th century?

CHAMBERLAIN: Sure.

RACTER: Ah yes, well in that case... I'm for Smith for President. Who are you for?

Chamberlain tells me that time-domain reflectometry is a discipline applied to the measurement of antenna performance. Not a practicing time-domain reflectometrist himself, he has probably got into the habit of saying strange things to his bizarre companion out of sheer self-defense.

The rest of the strangeness (as well as a rash of historical errors of time and place) comes from Racter. In the opening lines of the conversation above Racter asks some key questions of Chamberlain under the impression that they have never talked before. Racter stores this information for future use, activates certain areas of its associative memory and then roars off into conversational orbits around Italian music, wine and lettuce.

In the course of its remarks Racter mentions Samuel Sewall, a Boston judge and diarist of the 17th century. Chamberlain had assumed that Sewall wrote some music and had therefore indicated as much in Racter's files. Racter, with its wildly associative mind, assumes much more, and then abruptly, as if growing tired of music and food, it switches to politics.

Like its conversation, Racter's short stories tend to ramble, but the program's vignettes can be marvelously funny and even thought-provoking: "Bill sings to Sarah. Sarah sings to Bill. Perhaps they will do other dangerous things together. They may eat lamb or stroke each other. They may chant of their difficulties and their happiness. They have love but they also have typewriters. That is interesting."

Even a layperson may have concluded that Racter is decidedly schizophrenic. On the one hand, the ideas brought together in these sentences seem to form a coherent whole. It is nice that Sarah and Bill sing to each other. Although I had not thought of singing as dangerous, I am positively charmed by the idea of the two eating lamb, stroking each other and chanting about their ups and downs. On the other hand, the allowances I have been making for Racter all along are stretched to the breaking point when Racter mentions that besides their love they also have typewriters. Invited to share in this extraordinary insight, I tremble on the brink of a completely unknown mental world, one that I would prefer not to enter.

Many years ago I took part, along with some other undergraduates, in what we called the e. e. cummings Project. Our idea was to generate convincing e. e. cummings poetry by making a list of the words appearing in his verse and by designing a grammar that captured much of his style. Using the grammar, a computer program gener-

ated random sentence forms and filled in the blanks by randomly selecting words of the right type (noun, verb, object and so on) from the word list. About a bushel of poetry emerged from the printer, almost all of it terrible. I can recall only a single line: "The one bright star lit feebly this wick." The point of this story is that simple chance guarantees the emergence, sooner or later, of something relatively impressive.

Eliza, another talking program, also depends very much on luck when it comes to making a lasting impression. Written in 1966 by Joseph Weizenbaum of the Massachusetts Institute of Technology, Eliza avoids a great deal of conversational burden by playing the role of a nondirective psychotherapist. There follows a transcript of our recent conversation:

ELIZA: Hello. Please state your problem.

DEWDNEY: I am not in the habit of discussing my problems with a computer.

ELIZA: Do computers worry you?

DEWDNEY: Sometimes. Where will it all end?

ELIZA: Please go on.

DEWDNEY: Do you think that by our wholesale adoption of computer technology we might be unknowingly closing the door on important cultural strategies?

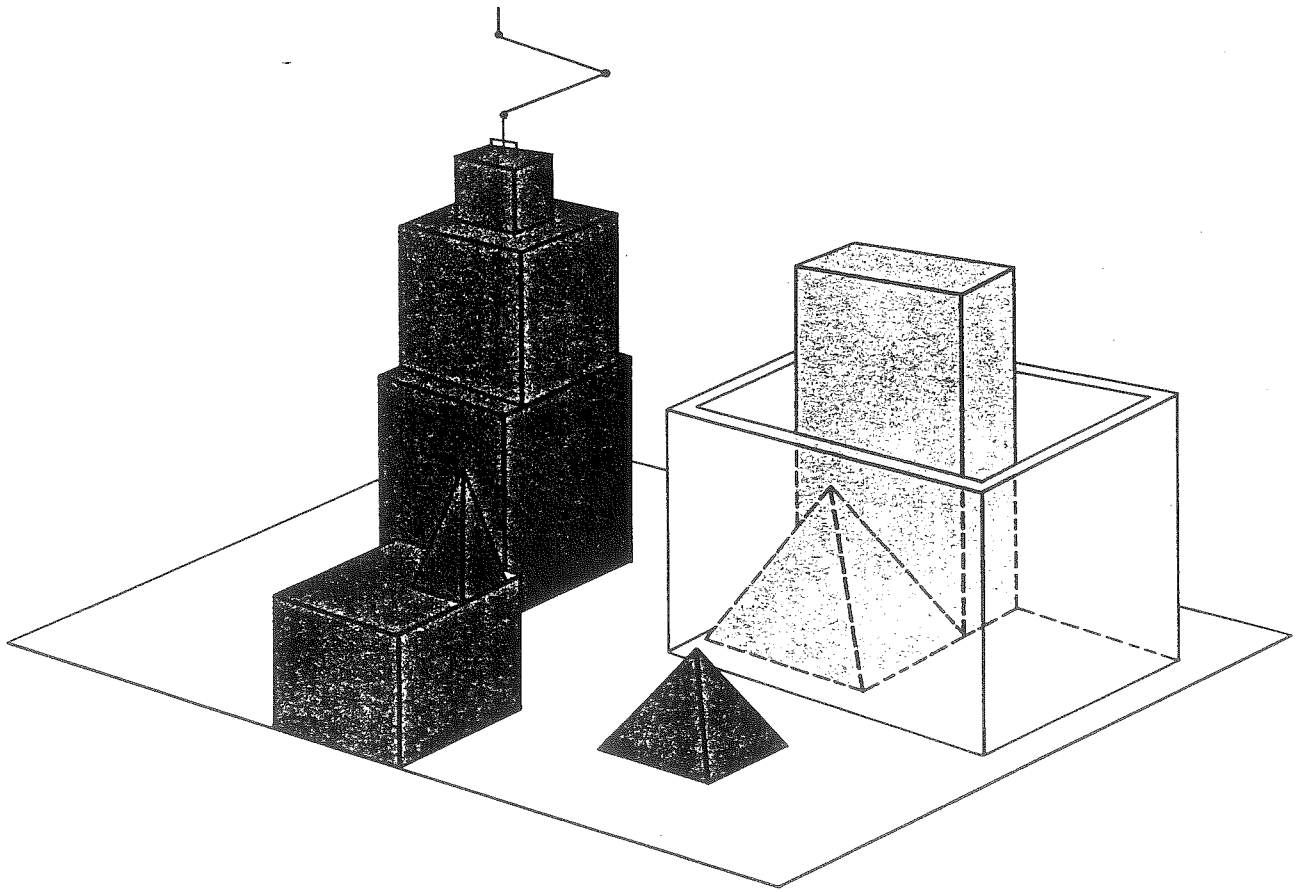
ELIZA: We were talking about you, not me.

So far, so good. Eliza appears to be playing the therapeutic role fairly well, eliciting further statements from me and turning my question aside to refocus attention on my "problem." On the latter point, however, the reply does not seem entirely appropriate. "What do *you* think?" might have been better. After a few more lines of conversation Eliza says: "Does that have anything to do with the fact that your problems with a computer?"

The illusion that Eliza is a sophisticated conversational program begins to crumble. Although Racter commits numerous semantic sins, it is designed always to speak grammatically. Semantic nonsense is easier to accept than syntactic blundering. From this point on the conversation degenerated—as I knew it would. I had spoken with Eliza years ago.

The Eliza program operates on a very simple principle. There is a small collection of key words that come up in most people's conversation. For example, Eliza uses the occurrence of words such as "my" to trigger two possible kinds of response. Together they illustrate Eliza's total conversational ability. A reference to a member of one's family such as "my sister" or "my father" always draws the same response from Eliza: "Tell me more about your family."

This is one of several preformed sentences Eliza keeps on hand simply to print whenever a key word or phrase is input. Another key word is "computer." Mention of it always causes Eliza to ask whether computers worry the patient. But Eliza's responses extend slightly beyond this knee-jerk conversational plane. The program also has a number of partial sentences on hand. For example, when the patient says "my" in reference to anything but a family member, Eliza stores the word string following "my" and interchanges first- and second-person pronouns and possessives. Thus if at some point I had said, "It bothers me that my head is getting bald," Eliza might say, many lines of conversation later, "Earlier you said your head is



Shrdlu's tabletop world of colored blocks, pyramids and boxes

getting bald." Or Eliza might say, "Does that have anything to do with the fact that your head is getting bald?" This explains Eliza's terrible gaffe: "Does that have anything to do with the fact that your problems with a computer?" The program expects a sentence following the earlier "my."

Readers will find advice on how to write their own Eliza program in a useful and informative little book called *Experiments in Artificial Intelligence for Small Computers*, by John Krutch [see "Bibliography," page 120]. The program listed there is called DOCTOR, one of Eliza's pseudonyms. Krutch suggests some modifications in the program, such as extending the range of key words to which Eliza will respond. I would suggest that responses of the more flexible kind be augmented. Eliza can be modified to give "help" in other areas. I am sure that with only the simplest of changes the program would make an excellent complaint-department receptionist.

When I spoke with Chamberlain in his New York office, I learned that Chamberlain not only had two microcomputers but also had a copy of Eliza. Why not run Eliza on one computer and Racter on the other? Chamberlain agreed to transfer the output of one program to the other by typing it as input. The program's initial conversation follows.

ELIZA: Hello. Please state your problem.

RACTER: Excellent! Excuse me a minute while I attend to some business. I'll be right back.

ELIZA: Why do you need to tell me excuse you while you attend to some business?

RACTER: Is it better to need or to despise? Why is it that I do need to tell you to excuse myself to attend to some business? Because electrons are brave. Braveness is nice, don't you think? And the needers all insist upon it. Do you imagine I am a Communist?

Mercifully (for Eliza) we stopped the conversation at this point. At a metaphorical level this conversation represents the relation between a plodding, rather unimaginative psychiatrist and an extroverted, manic, possibly dangerous patient. At a computational level a rather modest reflex-arc program has met head on, so to speak, with a highly complex program, one capable of recursive discursions and endless chains of association.

Racter's operation is difficult to summarize in a few words or even in many. It is a perfect example of the kind of in-house programs that still function in many large corporations and institutions today. It has grown over a number of years by accretion; more advanced, sophisticated layers of software have been wrapped around earlier, more primitive routines. At no time was it ever taken apart, analyzed, restructured and documented. By the same token, however, one can argue that it is probably in Racter's nature to dwell in such an unstructured software brain. Etter, who wrote the Racter program in many of its versions, compares it to the English language, which itself "is a pretty unwieldy accretion of rules and conventions. Insofar as Racter's commands try to deal with English, they too become unwieldy and hard to summarize." John D. Owens, who acts as Racter's agent, is himself a computer scientist at the College of Staten Island of the City University of New York. Owens confesses to having no sure grasp of precisely how the program works in its entirety.

Racter's passionate outbursts result from a simple program cycle that is entered and reentered through complex recursions. First Racter picks an item at random from one of its files. If the item is what Etter calls a literal, Racter prints it directly. In the conversation between Racter and Chamberlain at the beginning of this column, "I see" is just such a literal. The item retrieved, however, is more likely to be a command than a literal. The command sends Racter off to other files, some of which may contain still further commands. When the initial command has finally been completed, the program cycle is reentered with yet another random probe into one of Racter's files.

When Racter begins a new sentence, it selects a sentential form, either randomly or as the result of its recent conversational history. Suppose the form selected is

THE noun verb (third person, past tense) THE noun:

Here capital letters spell words about which Racter has no choice. The program prints THE and then goes to a file of nouns, selects MONKEY, say, and prints it. Consulting the verb file, Racter selects the verb TO EAT, forms the third person past tense, ATE, and prints that. Finally, Racter selects another noun at random, say TYPEWRITER. The result would be

THE MONKEY ATE THE TYPEWRITER

If this were all Racter were capable of, its output would hardly be better than the e. e. cummings Project of my undergraduate days.

such programs will develop. I look forward to a book on the subject.

Racter is available in either of two forms for $69.95 from Owens at 12 Shubert Street, Staten Island, N.Y. 10305. The first form is a diskette for IBM personal computers or compatible systems. The second form is called S-100 and refers to a certain hardware configuration within one's computer.

Also available from Owens at the stiffer price of $244.95 is a far more sophisticated piece of software called Inrac for people who want to alter Racter's personality and knowledge. Inrac also comes in both forms, accompanied by a manual explaining how a new generation of computer psychologists may alter Racter's files. I wonder if that is what Susan Calvin, Isaac Asimov's robopsychologist, did for a living.

At the beginning of this column I mentioned Shrdlu, a program created by Terry Winograd in 1970 at the Massachusetts Institute of Technology. Shrdlu is head and shoulders above both Eliza and Racter when it comes to intelligent conversation. Although Shrdlu's domain of discourse is restricted to a tabletop world of colored blocks and other geometric objects, Shrdlu really understands this world in terms of the relation between semantics and the physical properties of the blocks and the tabletop. Shrdlu's internal organization is entirely different from Racter's. It consists of subsystems that parse, interpret and construct sentences, carry out dictionary searches and semantic analyses and make logical deductions. These subsystems are tightly interwoven in a highly structured manner whose total effect gives one the impression of a true intelligence discussing the world of tabletop blocks. In the following dialogue Shrdlu's world, as shown in the illustration on page 16, is being discussed. The vertical line in the illustration represents Shrdlu's robot hand just after Shrdlu has executed the human's command.

HUMAN: Will you please stack up both of the red blocks and either a green cube or a pyramid.

SHRDLU: OK.

HUMAN: Which cube is sitting on the table?

SHRDLU: The large green one which supports the red pyramid.

HUMAN: Is there a large block behind a pyramid?

SHRDLU: Yes, three of them: a large red one, a large green cube and the blue one.

The complete dialogue is available in Winograd's classic work *Understanding Natural Language* [see "Bibliography," page 120]. Here we note only that Shrdlu's reply to the question "Which cube is sitting on the table?" involves not only an analysis of the sentence but also an understanding of physical relations between objects in its environment. Shrdlu knows that there are three cubes currently in its world and that only one of them is sitting on the table. This is a green cube. Since another green cube is present, Shrdlu refers to it in terms of its relation to another object in its environment, the red pyramid.

Conversational programs such as Shrdlu undoubtedly herald the future; the advantages of a computer that is able to discuss problems intelligently with humans rather than passively accepting programs to solve the problems are too obvious to miss. Computer scientists in artificial intelligence work, in part, toward this goal. As for Etter, he sums up his field of expertise as Artificial Insanity.

In the October "Computer Recreations" Lee Sallows, creator of the pangram machine, made a wager of 10 guilders that no computer-generated pangram would appear within 10 years. Sallows' challenge was met very quickly by no fewer than four recreational programmers. It seems amazing that all four found exactly the same pangram:

This computer-generated pangram contains six a's, one b, three c's, three d's, thirty-seven e's, six f's, three g's, nine h's, twelve i's, one j, one k, two l's, three m's, twenty-two n's, thirteen o's, three p's, one q, fourteen r's, twenty-nine s's, twenty-four t's, five u's, six v's, seven w's, four x's, five y's and one z.

Three of the four pangrammatists are listed here along with the dates on which they found this solution and the language and machine they used:

John R. Letaw, a cosmic-ray physicist of Severna Park, Md., discovered the pangram on September 20 running a BASIC program on a VAX 11/780 computer.

Lawrence G. Tesler of Apple Computers, Inc., in Palo Alto, Calif., found the pangram on the morning of September 23. Tesler used PASCAL on an Apple Lisa, naturally.

William B. Lipp of Milford, Conn., returned from a long weekend on Sunday, October 21, to find the same pangram on the printer of his IBM PC. Lipp also used PASCAL.

The fourth pangrammatist, of Palo Alto, Calif., wants to remain anonymous as he or she used a computer dedicated to problems very different from pangram hunting. The machine, another VAX 11/780, running a FORTRAN program, discovered the pangram on October 8.

Although it is not entirely clear from the wording of the wager, Sallows may owe each of these people 10 guilders. Perhaps I may be allowed to act as referee in the matter and close off collections on the bet at this point. Luckily for Sallows, 10 guilders does not amount to much.

So discouraged was Sallows by the astonishing rapidity with which some solutions appeared that he sent me the following advertisement to be run in this space:

For Sale
PANGRAM MACHINE
(slightly used)
plus 10-year guarantee!
only $100,000

The high price is the consequence of the debts Sallows anticipated; in view of my decision to close the wager he can no doubt be persuaded to lower his price.

All four successful contestants employed various heuristics in order to narrow the search for successful letter combinations. In view of space limitations it seemed reasonable to collect their descriptions into a single document, making it available from this department for $2 to cover the cost of printing and postage. Ask for Pangram Programs.

Another anonymous reader sent in a Roman-numeral pangram and a binary pangram. Here is the Roman-numeral pangram:

THIS PANGRAM LISTS III A'S, I B, I C, I D, I E, I F, II G'S, II H'S, XLVI I'S, I J, I K, III L'S, II M'S, II N'S, I O, II P'S, I Q, II R'S, XVII S'S, III T'S, I U, III V'S, I W, III X'S, I Y, I Z.

Readers might enjoy attempting the binary pangram without benefit of a computer: I's must be treated as 1's and O's as 0's. It starts "THIS PANGRAM HAS..."

I am indebted to John Henrick of Seattle, Wash., who alerted me to the May 1984 issue of *Word Ways*. An article in it by editor A. Ross Eckler and Mike Morton, a programmer, describes a program dedicated to finding anagrams of the name

RONALD WILSON REAGAN

Among the characteristic Reaganagrams produced by the program is

NO, DARLINGS, NO ERA LAW.

In fact, Racter's sentential forms tend to be rather more complicated than this simple example. The complexity results from the use of identifiers. An identifier is a combination of two letters (for example, *an* for animal) that serves as a tag. When they are attached to various words and forms, identifiers cause Racter to make associations between successively expressed words and sentences. For example, with such identifiers as *an* for animal, *et* for eating and *fd* for food the sentential form that Racter would select might well be

THE noun.an verb.3p.et THE noun.fd.

Here Racter must search for a noun in its files but is limited only to those nouns bearing the *an* identifier. Thus it would choose at random among nouns from AARDVARK to ZEBRA. Next, having selected a noun, let us say MONKEY, Racter chooses a random verb bearing an *et* identifier. Such verbs might include EAT, MUNCH, NIBBLE and so on. Having randomly chosen CONSUME, Racter forms the third person past tense as indicated by the code 3p in the sentential form. Finally, Racter looks up the nouns bearing *fd* identifiers and selects, say, ANCHOVIES. This would result in the new sentence

THE MONKEY CONSUMED THE ANCHOVIES

which certainly makes more sense than the previous sentence.

Racter's abilities go far beyond the capacity to make file searches restricted by identifiers. Racter is perfectly capable of generating its own sentential forms. If animals and food were to be the current subject of conversation, for example, Racter would select raw sentential forms and place identifiers within the forms.

In fact, Racter can, up to a point, generate its own command strings and insert them into the stream of recursion. Since grammatical forms are always adhered to, the sentences are always grammatical. Because identifiers are used and because Racter maintains a list of those currently active in the conversation, the program can hold up its end of any conversation, at least after a fashion.

The foregoing description embraces only a few aspects of Racter's total operation. My own understanding of the program does not extend much beyond this. I do not doubt, however, that Racter will soon have many imitators and that general principles for

Scientific American
November 1983

COMPUTER RECREATIONS

A progress report on the fine art of turning literature into drivel

by Brian Hayes

Almost any computer program can be made to yield meaningless results if it is given sufficiently muddled information to work with; this is the sense of the tired adage "garbage in, garbage out." The principle is now so well established that no one would take much notice of another demonstration. With a little thought and effort, however, it is possible to create a program that accepts as its input great masterworks of literature and nonetheless produces as its output utter nonsense. In goes the last act of *Macbeth;* out comes a tale told by an idiot, full of sound and fury, signifying nothing. Now *that* is data processing. (The inverse transformation, alas, seems to be much harder.)

The conversion of literature into gibberish is done in two stages. First a text is "read" by the program, and certain statistical properties are extracted and recorded. The statistics define the probability that any given letter of the alphabet follows another letter, or another sequence of letters, in the source text. In the second stage a new text is generated by choosing letters at random in accordance with the recorded probabilities. The result is a stream of characters that reproduce the statistical properties of the original text but whose only meaning, if any, is a matter of accident.

I cannot imagine a cruder method of imitation. Nowhere in the program is there even a representation of the concept of a word, much less any hint of what words might mean. There is no representation of any linguistic structure more elaborate than a sequence of letters. The text created is the clumsiest kind of pastiche, which preserves only the most superficial qualities of the original. What is remarkable is that the product of this simple exercise sometimes has a haunting familiarity. It is nonsense, but not undifferentiated nonsense; rather it is Chaucerian or Shakespearian or Jamesian nonsense. Indeed, with all semantic content eliminated, stylistic mannerisms become the more conspicuous. It makes one wonder: Just how close to the surface are the qualities that define an author's style?

The process of generating random prose has been investigated in detail by William Ralph Bennett, Jr., of Yale University. He has made the statistics of language a major theme of a course on the applications of computers, and the topic also figures prominently in his introductory textbook on programming, *Scientific and Engineering Problem-solving with the Computer.* (The book is a good deal livelier than the title might suggest. The problems taken up include the aerodynamics of the 1950 Princeton-Dartmouth football game, which was played in a hurricane; the diffusion of syphilis through a population of sailors and prostitutes, and a spectral analysis of the krummhorn, oboe and "modelocked garden hose.")

Bennett notes that the earliest known reference to the random generation of language is in the *Maxims and Discourses* of John Tillotson, archbishop of Canterbury in the 1690's. In making a case for divine creation Tillotson wrote: "How often might a Man, after he had jumbled a Set of Letters in a Bag, fling them out upon the Ground before they would fall into an exact Poem, yea or so much as make a good Discourse in Prose? And may not a little Book be as easily made by Chance, as this great Volume of the World?"

For most modern considerations of random language the point of departure is Sir Arthur Eddington's statement of 1927: "If an army of monkeys were strumming on typewriters, they *might* write all the books in the British Museum." Eddington too meant to emphasize the improbability of such an outcome; he cited it as an example of an event that could happen in principle but in practice never does. All the same, since Eddington's time the possibility of finding genius in the random peckings of monkeys has taken on a literary life of its own. Bennett mentions works by Russell Maloney and Kurt Vonnegut, Jr., and a nightclub act by Bob Newhart.

The process Eddington envisioned can be simulated by a program I shall call an order-zero text generator. First an alphabet, or character set, is decided on, which determines what keys are to be installed on the monkeys' typewriters. In some higher-order simulations it becomes important to keep the number of symbols to a minimum, and for consistency it seems best to adopt the same character set in the order-zero program. I have therefore followed Bennett's recommendation in choosing a set of 28 symbols: the 26 uppercase letters, the word space (which the computer treats as a character like any other) and the apostrophe (which is commoner in much written English than the three or four least-common letters are).

The ideal, unbiased monkey would at any moment have an equal probability of striking any key. This behavior can be simulated by a simple strategy. Each symbol in the character set is assigned a number from zero to 27. For each character to be generated a random integer is chosen in the same range and the corresponding character is printed. A small specimen of text created by this procedure is shown in the illustration on this page. It bears no resemblance to written English or to any other human language. "Words" tend to be extraordinarily long (on the average 27 characters) and thick with consonants. The reason, of course, is that letter frequencies in real English text are far from uniform. The word space alone generally accounts for roughly a fifth of the characters, whereas J, Q, X and Z together make up less than 1 percent. In an order-zero simulation all the characters have the same frequency, namely 1/28.

The comic routine by Bob Newhart concerns the plight of the inspectors who must read the monkeys' output. After many hours of poring over unintelligible prattle they come upon the phrase, "To be or not to be, that is the gesorenplatz...." In fact, getting even that far is wildly improbable; the first nine words of Hamlet's soliloquy can be expected

PWGMMLTHIDVGRHPEDFCXFEKFNOPYPQSXZRUXG'YS'AEEU PEDEGLQYFUWPO'IKI
QTONIXJKZEUKDXWKKJREHYHPKWUJHLEJNBPLQ AIEOQXUBJYYVIFFDPQGIGZNTI
RQXPDJ NQESPQMCRSNGMKQEZICZV'GSWALK ZZEYIBBOTDCRSMK'VI MRCZXUBI
SNEQ'VQQHFQUCBJXZRVVNIBHFJEFTCFJPWFOIYHOMPNFSFWKNCMVLOJJBX
QV KIZTLNRWGGTZFPZPQQCGVJCPAYRDQJRMYSWCGABRXLERCYYRHQCHTOQ'UT
FMRITFTIZUIWTSTXWQGOCAFXJOZYKSTV'BYOBEUFIRQWQ VOUVQJPRKJWBKPLQZCB

Order-zero random text, drawing on an alphabet of 28 symbols

to turn up once out of every 2×10^{46} characters. In a run of 50,000 characters I was able to find one instance of TO and another of NOT; they were many lines apart. (I did not read the 50,000 characters but instead made the search with a pattern-matching program.)

A first step toward improving the monkeys' literary skills is to adjust the probability of selecting a given letter so that it reflects the letter's actual frequency in written English. In effect, the plan is to build a typewriter with, say, 2,500 space keys, 850 E keys, 700 T keys and so on. The letter frequencies might be averages calculated from a large sample of English prose, but it is both more convenient and more interesting to base them on a particular source text. A program that chooses characters with such a frequency distribution is a first-order text generator.

The letter-frequency values can be represented in a one-dimensional array with 28 elements. The array is a block of storage locations in the computer's memory, organized so that any one element can be specified by an index, or subscript, between zero and 27. In order to fill up the array one could count the instances of each letter in the text and enter the values by hand. It is better, however, to let the program do the counting, even when that means the text itself must be prepared in a machine-readable form. The counting program initially sets all the elements of the array to zero. The text is then examined one character at a time, and for each occurrence of a character the corresponding array element is incremented by 1.

First-order random text is generated by making the probability of selecting a character proportional to the character's array element. One method works as follows. A random number is generated in the interval between zero and an upper bound equal to the sum of the array elements (which is also the total number of characters in the source text). The first array element, which might record the occurrences of the letter A, is then subtracted from the random number. If the result is zero or less, an A is printed; otherwise the next element (representing B) is subtracted from the value remaining after the first comparison. The successive subtractions continue until one of them gives a zero or a negative result, and the corresponding character is selected. Note that the procedure cannot fail to make a selection, since the random number cannot exceed the sum of the array elements.

A sample of first-order random text is shown in the upper illustration at the right. It is based on a frequency array compiled from a passage in the last chapter of James Joyce's *Ulysses*, the chapter known as "Ithaca," or Molly Bloom's soliloquy. I had a reason for choosing it: the absence of punctuation in the random text is of little consequence because the source text too is unpunctuated.

The information on letter frequencies embodied in a first-order random text brings an improvement, but one would hardly call the text readable. Although the average word length (4.7 letters) is near the expected value (4.5 letters), the variance, or deviation from the average, is much too great. Words in normal English, it seems, are not only short but also have a narrow range of lengths; in the random text the distribution is much broader. Apart from the question of word length there is the mat-

FIRST ORDER

HUD T ALONIT NTA SN TVIOET ELERFOAD PE TRLTWTL N CABEG TYLUEMU TIGT
BH OFDRRIC O STU HOOOTO YATNDL UYA HWAE SS NLSDB OTRORT DEERARFT
D LBFF HHARE MW OSPE OFOIT SEOUN GTUMG H N GHKOY T EAOS A SD E TNNE
PEHAGIADIHNATO AATSAGI ED INNE ABRA TAAM GT E TWNO HEWIIGUTNCM GA SFHHY
HREBH RARE OOSY LFE OC EGGTA WIFRTYE EUS DA ETO WF EIT ERNETEBTSTTELO
NTAAN O YEETWNSONRNHN TYHVN NLUESETTHLGEAKPNNMTIA TSM REEANTVONC POE
RUTP EOIT L IEETGTWHSW H KHHER W OLIOEWOEPT D AEYBSTNHGDNPT C TNLINHH
KHHE E RTVIOB EI K EOAFPUTSTTAS NA LAN SRDF D NMTHESKO UGEEDICRAWDT OBD
TUIML WSORGNETE

SECOND ORDER

BEGASPOINT IGHIANS JO HYOUD WOUMINN BONUTHENIG SPPRING SBER W IDESE WHE D
OOFOMOUT O CHEDA AFOOIAUDO IS WNY UT DRSASER LD OT POINE ETHAT FOEVEL BE
ORRI IVER BY HE T AS I HET W BE T WAU GIM UTHENTOTETHAVE THIKEWOITOCOUTORE
TATHASTHEE AT D Y WAN TOND SE TEDING US AKIN WING W TE T BO TOTSTHINGATONO
EN T LLY WID OUCOUSIND HEF THIMES AG T BENG LORYE ALLATHOMOFTHER TOUDIMS YS
S ORYRY THERNG S HE M G M ANG S CITOOFO HEN G BEST ONDLOL ANE DO HE
ICISEKERIT ME NKITHADIMUPL WHES HT BATHE T LOR WITULOWAYE WATHEG M
LEROMAUN OUGS POUPO O HASING LIN ON ASHAN AWFAS HET ND MEDE

THIRD ORDER

MAY THOT TO THER YOURS CHIM JOSE EY EILLY JUSED AND HID YEL THE MARK WASK
TROOFTEN HEREY LING SH THAVERED HER INCED I MEA BUT DAY WOM THE EAKIN WIPS
AS SUGH THE WAY LIARADE TH MY HE ALMASEETIR ANICIOUT JOSIDNTO GRATEVE NO
VER BIGH WER ACCOW WAS I GEORE HENDSO EGGET PUT TO SQUAD TRADE OFF GIN
GO ME HER SPING HE CONE WELL FEWHEY THEYES AND AND QUICE YOULDNT HER
ORL SO MAKING RINGS SOMET DREAVE HISETTO COMAD THAT ME WE MIG TOLD THE
THERFUMBECK OT OFF FEELP HE WAST ITS LETHOTTEN ITHEE ROWN YOURS FEL FOR
SOME IF WIS HE STAKED UPPOIS SHENS NO TILL HIM I WAY SO WHATEALWAS WER TWE
NER DING O THIS IT IN ANIGH ACK REAN THAT DO GETHE BITER

First-, second- and third-order random text based on "Molly Bloom's soliloquy"

AD CON LUM VIN INUS EDIRA INUNUBICIRCUM OMPRO VERIAE TE IUNTINTEMENEIS
MENSAE ALTORUM PRONS FATQUE ANUM ROPET PARED LA TUSAQUE CEA ERDITEREM IN
GLOCEREC IOVELLUM ET VEC IRA AE DOMNIENTERSUO QUE DA VIT INC PARBEM ETUS
TU MEDE DERIQUORUMIMO PEREPORIDEN HICESSE COSTRATQUIN FATU DORAEQUI POS
PRIENS NOCTA CIENT HUCCEDITAM PET AUDIISEDENDITA QUE GERBILIBATIA VOLAEQUE
ORECURICIT FES ADSUE ARCUMQUE LULIGITO PIMOES PERUM NOSUS HERENS EA
CREPERESEM ETURIBUS AVIS POS AT IS NOMINE FATULCHENTURASPARIS AUDEDET PARES
EXAMENDENT DUM REMPET HA REC ALEVIREM ORBO PIERIS ATAE PARE OCERE RAS

QUALTA 'L VOL POETA FU' OFFERA MAL ME ALE E 'L QUELE ME' E PESTI FOCONT E 'L M'AN
STI LA L' ILI PIOI PAURA MOSE ANGO SPER FINCIO D'EL CHI SE CHE CHE DE' PARDI
MAGION DI QUA SENTA PROMA SAR OMI CHE LORSO FARLARE IO CON DO SE QUALTO
CHE VOL RICH'ER LA LI AURO E BRA RE SI MI PAREMON MORITA TO STOANTRO FERAI TU
GIA FIGNO E FURA PIA BUSCURA QUAND'UN DEL GUARDI MIN SA PAS DELVENSUOLSI PER
MUSCER PIE BRUI TA DORNO TITTRA CHE PO E PER QUE LI RINONNIMPIAL MIN CH'I'
BARVEN TA FUI PEREZZA MOST' IO LA FIGNE LA VOL ME NO L'E CHE 'L VI TESTI CHE
LUNGOMMIR SI CHE FACE LE MARDA PRESAL VOGLICESA

PONT JOURE DIGNIENC DESTION MIS TROID PUYAIT LAILLE DOUS FEMPRIS ETIN
COMBRUIT MAIT LE SERRES AVAI AULE VOIR ILLA PARD OUR SOUSES LES NIRAPPENT LA
LA S'ATTAIS COMBER DANT IT EXISA VOIR SENT REVAIT AFFRUT RESILLESTRAIS TES FLE
LA FRESSE LES A POURMIT LE ELLES PLOIN DAN TE FOLUS BAIER LA COUSSEMBREVRE
DE FOISSOUR SOUVREPIACCULE LE SACTUDE DE POU TOUT HEVEMMAIT M'ELQU'ILES
SAIT CHILLES SANTAIT JOU CON NOSED DE RE COMMEME AVAIL ELLE JE TER LEON DET
IL CED VENT J'ARLAMIL SOUT BLA PHYSIS LUS LE SE US VEC DES PEUSES PAU HAS BEAU
TE EMANT ELLE PLANQ HEUR COIRACOUVRE BIENE ET LUI

Third-order Latin (Virgil), Italian (Dante) and French (Flaubert)

ter of word content. Even though the letters appear with the correct frequency, their sequence is utterly random, and most of the resulting "words" are not English and could not be. A letter series such as WSTLNTTWNO or HIUOIMYTG is not merely meaningless but impossible. In one run of 2,000 characters the longest recognizable word was, appropriately, RARE.

The next refinement is the crucial one because it can be extended, at least in principle, to an arbitrarily high order. The root of the idea is that a letter's probability of appearing at a given point in written English depends strongly on the preceding letters. After a V, for example, an E is most likely; after a Q, a U is all but certain. The procedure, then, is to set up a separate frequency table for each symbol in the character set. The frequencies are recorded in a two-dimensional array with 28 rows and 28 columns, for a total of 784 elements. An example of such a frequency table is shown in the upper illustration on page 24D. (The array is "normalized" by rows, meaning that comparisons are valid only within a row.)

When text is generated from the two-dimensional array, the character most recently chosen determines which row of the table is examined in picking the next character. For example, if the preceding letter is a B, only the elements of the second row are taken into consideration. The largest element of the second row is E, and so it is the likeliest letter; A, I, L, O, R, S and U also have a chance of being selected. Impossible combinations such as BF and BQ have zero frequency, and so they can never appear in the program's output.

Second-order random text begins to show the first hints of real linguistic structure. The distribution of word lengths is only a little wider than it ought to be. Real words are not uncommon, and there are many near-misses (such as SPPRING or THIMES); a large majority of the words are at least pronounceable. Common digraphs such as TH begin to show up often, and the alternation of vowels and consonants is a conspicuous pattern.

The next step should be obvious. A third-order algorithm chooses each letter in the random text according to probabilities determined by the two preceding letters. This calls for a three-dimensional array with 28 planes, each plane being made up of 28 rows of 28 columns. Suppose at some point in the creation of the text the letter sequence TH has been generated. The program must then look to the 20th plane (corresponding to T) and to the eighth row on that plane (corresponding to H). In that row E is the likeliest choice, although A, I, O and the space symbol also have non-zero probabilities. If E is indeed selected, then in the next iteration the choice will be made from the fifth row of the eighth plane, the position in the table specified by the letter sequence HE. Here the leading candidate is the word-space character, followed by R.

In third-order text no three-character sequence can appear unless it is also present somewhere in the source. Because spaces are included in the accounting, that is enough to guarantee only that all one-letter words will be real words; in effect, only the letters I and A can appear in isolation. The actual performance, however, is a good deal better than the guarantee. Virtually all the two-letter sequences are words, and so are most of the three-letter sequences. Often a string of several words in a row turns up: PUT TO SQUAD TRADE OFF GIN GO ME HER. Even quite long nonwords have a certain phonetic plausibility. After all, it is only a matter of accident that ANYHORDANG HOUP TREAFTEN is meaningless in English.

While reading a sample of third-order random text, I am reminded of stage-performance double talk and of glossolalia, the "gift of tongues" that figures in certain Pentecostal liturgies. One might guess that there is some significance in the resemblance: perhaps people who have learned those arts carry out an unconscious statistical analysis somewhat like the one the program does. I think another explanation is likelier. Double talk and glossolalia seem to involve the random assembly of phonemes, the fundamental atoms of spoken language. It may be that three letters is about the right size for the written representation of a phoneme.

With third-order text the stylistic qualities of the source begin to have a perceptible effect. Where the contrast in styles is great, the corresponding random texts are also clearly different, although it is not easy to say exactly what constitutes the difference. I am inclined to describe it as a matter of texture, but I am not at all sure what texture is in prose. Is it whatever remains when all the meaning is removed?

Even when individual mannerisms cannot be perceived in third-order random prose, identifying the language of the source is easy. Patterns of vowels and consonants and the characteristic endings of words are unmistakable. The bottom illustration on page 21 shows brief examples of Latin (Virgil), Italian (Dante) and French (Flaubert). Someone who knows only the "look" of one of these languages might have trouble distinguishing the ersatz product from the real thing.

Before considering what lies beyond the third-order approximation, I should like to mention some other applications

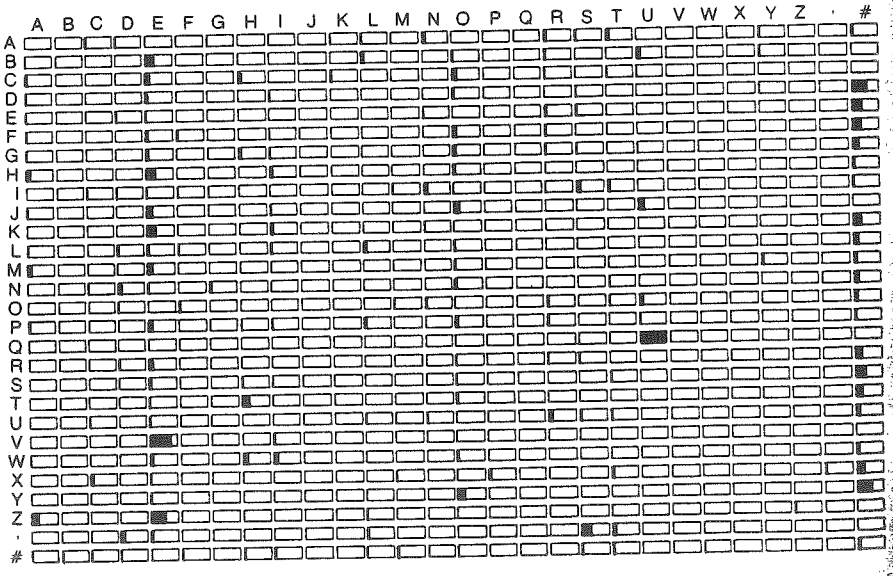
of letter-frequency tables. Bennett, in a discussion of the entropy of language, points out that the tables enable one to calculate the amount of information conveyed per character of text. The information content essentially measures the difficulty of predicting the next character of a message. It is at a maximum in the order-zero simulation, where every possible character has equal probability; in other words, the information content is greatest when the text is totally unintelligible. The idea of predicting characters leads to a discussion of error correction in telecommunications and to the design of algorithms for solving ciphers and cryptograms.

Another area worth exploring is the alteration or manipulation of the frequency array. How is the random text changed, for example, when each element of the array is squared? An example of Molly Bloom squared is shown in the bottom illustration below. Because the procedure exaggerates differences between array elements, the effect is to "sharpen" the frequency distribution; common words become still commoner. Many other transformations are possible. Adding a constant value to all the array elements has a disastrous effect, even if the constant is a small one: all the impossible letter combinations, which one has been working so hard to eliminate, become possible again.

One intriguing idea is multiplying the entire array by -1 in order to generate text by, say, Alexander anti-Pope. For any given combination of letters, whatever subsequent letter is likeliest in Pope would be the least likely in anti-Pope. Literary aptness might best be served if the product resembled the works of Colley Cibber. Actually, it is an almost patternless jumble.

The result is somewhat less discouraging (although still far from illuminating) when two arrays are added or multiplied. In this way one can create unlikely collaborative works, written by Jane Austen plus Mark Twain or by Keats times (Byron plus Shelley). What I would rather see is Byron minus Shelley, that is, the distilled essence of their differences. Unfortunately, I have not been able to make it work. Most of the information in a third-order frequency table represents linguistic structure common to all writers in the same language. Subtracting out that common element leaves little but noise.

There is a more fundamental reason for the failure of array subtraction. In the unmodified third-order table rough-



A second-order frequency table for Act III of Hamlet

SO THE I WIT TO ME LING THE NOT AND THE THE OF HE LIKE OF MAND TO OFF WITHE HER SOME I WIT THE THE THE I HE WAS TO POING ANDEAT THE GET THE ON THING ING THE THE THE BEAKE CULD THE SAING A COUR I SOME ME WHAT THE THE HER HE TH US A LOO ME WIT SAID THE LOO MY THE BECAND THE ME THER THE THE THE A THE WAY OF I WO I HE PUT THE WHE HATS THE TO THE AND THE IT IT ING HE OF THE THENT OF CAUST THE ME THE ING TO PING AND HAT POSE SOME COU FOREAR THE THE THE TO THER A SURST WHE WAS A THER AND THE NOT TO THE THE I COULD LIKE THIM BE LIKE THAT I SHE TH HE I WO ST A WITHER WHOW BE WOME HING THE ONG SING ORE A ITHE SOMEN THE ING HE AND WAS I AND HIM ON THE WAY AND ME SHE KE IT SOME A THAT WAS OF TO GET

A modified frequency table gives rise to prose by "Molly Bloom squared"

ly 90 percent of the elements are zero: they correspond to the great majority of letter combinations that are never observed in English, such as RJT or UUU. Ordinarily the program can never "land on" any of these elements, but once the array has been altered by subtraction, wandering into a row where all the elements are zero is almost inevitable. From such a dead end there is no graceful escape.

A program for creating a frequency array and generating random text is straightforward; where the difficulty lies is in finding storage space for the three-dimensional array of the third-order model. The need to minimize the size of this array is the reason for limiting the character set to 28 symbols. Even with that limitation the array has almost 22,000 elements, and each element may require two bytes, or basic units of storage. Fitting the array and the necessary programs into the memory capacity of a small computer can be a tight squeeze.

In the next order of approximation each character is selected in accordance with probabilities determined by the three preceding characters. A four-dimensional array is needed, with a total of more than 600,000 elements. In 1977,

FOURTH ORDER

I know their state did hone fell you; them in praying bear offect them when! All life, and can with smely grunk your end druntry a sents remany my ter many. Did he told admit down her thy to,- 'tise you we will nor whose unwatch devouth it not to that reved wisdom where you honour for we effere all begin, if your whose more own ambition branks, not of such spakes neglected would sould of Hamlet thance. To abountry word. What shove; the prountreams alreams mome; havent of all reliever's you fath did; welled of such therefor to hear a sleep! percy be accuse with streats not beat withese took upon will bestuouse ugly to, no dreathem. 'Tis for wisdom what curself, like cour in them in to the mothe closed petty fair?

FIFTH ORDER

I, his soul, that are. To a nunnery. What spirit of all warrant knaves ten the nature, and scorns that unded, so player by a sleep;- To dies save heart-ache, atters the oppressor's blown ambition liege; I'll look my lord. O heart; and I'll give that he spokes thy origin and love. Her fault is night his fit, and quickly justice, and man's chaste as you now rights. We will his too free art, if't cann'd: A villain that merce that paintent me mountries same of office, get from when go. Oh, 'tis somethings and drift of him in. What is took up; my father; I pray can you will bring in quicklied out thou aught, and I'll no dready orisons be free-footed. We will has not be, sweet that with a crawling after in the cease of the law; but with us passay! Bow, stubborne me my mother aloof, what reply.

SIXTH ORDER

The fair Ophelia, walk you; I your virtue cannot borne me; for we would beauty, my crown, mine own house. Farewell. Oh, my lord. Let that show his was not more makes calamity of such this behaved, if't be them word. Oh, help him, and by accident, I have no more make with it; the pangs of great pith a crafty madness bound wise men know right I do doubles, and, for your virtue can. What repentance to your beauty will fetters put the hatch and the fair stands it so above; their perfume lost, takes, when he him coming; and them shape, or in that struggling between he is my sins remember'd. Good gentleman. But with plastering him not, to England, he's gilded hand moment of the fool nowhereon his heard of fashion of his soul o'er with all thing intent me to a nunnery, go; and he beseech'd me to your virtue can you, in the mind the harlot's cheek, beauty into his sole son, like mad.

SEVENTH ORDER

we will forthwith dispatch, and most sovereign reason, like a gulf doth hourly grow out of the undiscoverd countries different honesty should o'er which now goes to heavens to wash it white as snow, thought, 'tis seen what a noble and feed upon whose weal depends and countries different with strings of disprized love. How now, Ophelia? Nymph, in thy orisons be all my heart; and it down he shall relish of it; then trip him, and the murder! Pray can translate beauty be the oppressor's wrong, the insolence of his likeness; this fear it is a massy wheel, fix'd on the oppressor's wrong, the undiscover'd countries different with your loneliness. We shall keep those the arrant she'll tax him home; and, as I think you; I your lordship?

EIGHTH ORDER

Did he receive them. No, not I; I never gave you well, well, well. Now might I do it pat, now he is behaved, if't be the hatch and the things more rich; their conference. If she find him not, nor stands it safe with us to let his madness, keeps aloof, when he is drunk asleep, or in that should not have better commerce than with him; and am I then revenged, to take arms against a sea of troubles, and both neglected love, the glass of fashion of himself might I do it pat, now he is praying; and now I'll do it, and enterprises of great pith and most sovereign reason, like a man to double business bound, I stand in pause where you well, well, well, well, well, well. Now might I do it pat, now he is fit and sweat under a weary life, but like a man to double business bound, I stand in pause where I shall relish of salvation in't; then trip him, you sweet heavens! If thou dost marry, marry a fool; for which I did the murder?"

Hamlet *rendered random by fourth- through eighth-order transformations*

FOURTH ORDER

I was wasn't not it as I never know cotton his again the rushind. "Now to get me, and when we was jestill be Memphis. But first found I reach had at like, and him. We sides in a soldier. I cars give you in as there dog if hearl Harbor. It will no cab. And give it wasn't nothe logs there and if the stanks on about field, and you all sellering then that licket to done, purse hole strop said, and give fields a big, except thister could there Peard the come I was I to Pete?"

FIFTH ORDER

Come in. Tell me all the back and I told him no mind. Then the other bus stopped backing good, I really don't before. We set the bus fellered. And I et them. When he was and jump backing and I hear him. "If I do," there, and it, with the said, "Here we was wropped. A man don't he got on are back. He soldier with them. Then then he county. Then into the bus feller. "I just soldier with strop said. "What?" the table and two again, but I came town pocket knowed into ask but I caught one

SIXTH ORDER

"The train and I would pass a patch on his arm. He hadn't never paid that," I said. "I'm going the knife up to see Pete Grier. Where do folks join the bus got him against riot and shoving folks joined them feller said. "Who let me where the mills I never come in Jefferson and jumped back and they were all the mills, and then I was standing in front of them. Where's Pete was gone. Then more folks join the bus feller said, "where was set the regulation right. I never come on.

SEVENTH ORDER

"What?" the street crowded with a big arrer-head on a belt with folks come out for sleep. But I couldn't ketch on how to do so much traveling. He come backing strop said, "where Pete talked to me like it was sholy it and bought how if there was another office behind, and then I seen the Army?" "What the soldier said, "Where's Pete?" Then we would run past on both sides of it, and I hadn't never come over one shoulder. "What the room. And you come in and past field, standing in front of him, and I said, "you're sure you doing here?" he said. "I ain't yet convinced why not,"

EIGHTH ORDER

"Who let you in here?" he said. "Go on, beat it." "Durn that," I said, "They got to have wood and water. I can chop it and tote it. Come on," I said, "Where's Pete?" And he looked jest like Pete first soldier hollered. When he got on the table, he come in. He never come out of my own pocket as a measure of protecting the company against riot and bloodshed. And when he said. "You tell me a bus ticket, let alone write out no case histories. Then the law come back with a knife!"

Higher-order random versions of William Faulkner's story "Two Soldiers"

FOURTH ORDER

"Why, so much histated away of Bosty foreignaturest into a greached its means we her last wait it was aspen its cons we had never eyes. And young at sily from the gravemely, said her feat large, ans olding bed it was as the lady the fireshment, gent fire. Ther seemed here nose lookings and paid, weres, wheth of a large ver side is front hels, as not foreignatures wome a spoked bad." "Wait of press of hernall in frizzled, or a man spire. An at firmed." "My deal man.

FIFTH ORDER

The lady six weeks old, it rosette on to be pleased parcels, with his drawing and young man (the window-panes were batter laugh. "I this drawing and she fire?" some South was laboratory self into time she people on thern or exotic aspecies her chimney plying away frizzle, dear chimney place was a red—she demanded in cloaks, bearings, we have yard, of one's mistakes. She helmsman immed some on to the most interior. The windows of proclaimed.

SIXTH ORDER

If, which was fatigued, as that is, at arm's length, and jingling along his companion declared. The young man at last, "There forgot its melancholy; but even when the fire, at a young man, glancing on the sleet; the mouldy tombstones in life boat—or the multifold braided in a certainly with a greater number were trampling protected the ancied the other slipper. She spoke English with human inventions, had a number of small horses. When it began to recognize one of crisp dark hair,

SEVENTH ORDER

But these eyes upon it in a manner that you are irritated." "Ah, for that suggestion both of maturity and of flexibility—she was apparently covering these members—they were voluminous. She had stood there, that met her slipper. He began to proclaim that you are irritated." "Ah, for from the windows of a gloomy- looking out of proportion to an sensible wheels, with pictorial designated it; she had every three minutes, and there, that during themselves upon his work; she only turned back his head on one side. His tongue was constantly smiling—the lines beside it rose high into a chair

EIGHTH ORDER

"Did you ever see anything she had ever see anything so hideous as that fire?" she despised it; she demanded. "Did you ever see anything so—so affreux as—as everything?" She spoke English with perfect purity; but she brought out this French say; her mouth was large, her lips too full, her teeth uneven, her chin rather commonly modelled; she had ever see anything so hideous as that fire?" she despised it; it threw back his head on one side. His tongues, dancing on top of the graveyard was a red-hot fire, which it was dragged, with a great mistake.

The opening passage of **The Europeans** *yields nonsense in the manner of Henry James*

writing in *American Scientist,* Bennett gave specimens of fourth-order text generated by building such a large array. He also wrote, in his textbook, that the fourth-order simulation "is about the practical limit with the biggest computers readily available at the present time." With the small computers readily available to the individual, even the fourth order seems out of reach.

"Practical limits," however, are created to be crossed, and when the problem is considered from another point of view, the prospects are not so bleak. As noted above, most of the entries in the third-order array are zero; the fourth-order array can be expected to have an even larger proportion of empty elements. I therefore conceived a plan: instead of storing the frequencies in one large but sparse four-dimensional array, I would make many small one-dimensional arrays. Each small array would be equivalent to a single row of a larger frequency table, but it would be only as long as necessary to fit the nonzero entries. Rows with all zero elements would be eliminated altogether.

The plan is feasible, I think, but messy. Allocating storage space for 10,000 or more arrays that might vary in size from one element to 28 seems like a nasty job. As it turned out, I found a better way, or at least a simpler way. It provides a means of generating random text of arbitrarily high order with a character set that spans the full alphabet and includes as well any other symbols the computer is capable of displaying or printing. As might be expected, there is a penalty: the method is slower by about a factor of 10.

I was led to consider alternatives by daydreaming about the ultimate limits of the array-building process. Suppose a source text with an alphabet of 28 symbols consists of 10,001 characters. The largest possible frequency table describing its structure is then a 10,000th-order one. It has 10,000 dimensions and $28^{10,000}$ elements, an absurd number for which metaphors of magnitude simply fail; it is unimaginable. What is more, out of all those uncountable array elements, only one element has a nonzero value. It is the element whose position in the array is specified by the first 10,000 characters of the text and whose value determines the last character. Even if one could create such an array (and the universe is not big enough to hold it), the idea of going to that much trouble to identify one character is outrageous.

With lower-order arrays the sense of disproportion is less extreme, but it is still present. The fact is, all the information that could be incorporated into any frequency table, however large, is present in the original text, and there it takes its most compact

form. (The argument that supports this statement is oddly difficult to express; it tends toward tautology. What the frequency table records is the frequency of character sequences in the text, but those sequences, and only those sequences, are also present in the text itself in exactly the frequency recorded.)

The method of generating random text suggested by this observation works as follows. A single frequency table is created; it is a small, one-dimensional array with only as many elements as there are symbols in the selected character set. I chose 90 characters. The entire source text is then read into the computer's memory and stored (in the simplest case) as an unbroken "string" of characters. Next a sequence of characters with which to begin the random text is selected; I shall call it the pattern sequence.

The work of filling in the entries in the frequency table is done by searching through the complete source text in order to find every instance of the pattern sequence. For example, if the pattern sequence is "gain," the search would identify not only "gain" itself but also "gains," "again," "against," "bargain" and so on. Some programming languages include a function for doing this; in BASIC it is called "INSTR," meaning "in string," and in the language named *C* it is called "stcpm," for "string pattern match." Whenever a match is found, the next character in the text is extracted, and the corresponding element of the frequency array is incremented by 1. When the entire text has been searched, the array is complete.

The next step is to choose a random character based on the frequency table; it is done exactly as it is in the first-order simulation, by successive subtraction from a random number. The character associated with the chosen array element is printed. The entire process is then repeated. The frequency array is discarded by resetting all its elements to zero. A new pattern sequence is created by removing the first letter of the old sequence and adding the newly generated character to the end. Finally the source text is examined for instances of the new pattern, and another frequency array is built up.

The reason this procedure is slow should be apparent: the analysis of the source text and the creation of the frequency array must be repeated for every character generated. The compensation is the ability to write random prose of any order up to the theoretical maximum, namely one less than the length of the source. Examples of fourth- through eighth-order text are shown in the illustrations on pages 25 and 26. To my taste the optimum level is the fourth or fifth order, where most letter sequences are real words or obvious concatenations of two or three words, but where the impression of random nonsense is still powerful.

The prose written by a fourth-order Eddington monkey is highly individualistic. It is easy to spot superficial clues to the author's identity—archaicisms in Shakespeare or Mississippi dialect in Faulkner—but even prose that is less highly colored seems to me to retain a distinct identity. It is not obvious how or why. Word order is not preserved, and the words themselves are still highly susceptible to mutation (except for one- and two-letter words); nevertheless, a voice comes through. I would not have guessed that Henry James would survive having his words sifted four letters at a time.

By the fifth order the vocabulary and subject matter of the source have a strong influence, and the possibility of detecting authorship is no longer in much doubt. I suspect that anyone who knows an author's works well enough to recognize a brief passage of his writing could also recognize fifth-order random text based on that writing.

The response to a fourth- or fifth-order approximation of English writing has another interesting aspect: it demonstrates the peculiar human compulsion to find pattern and meaning even where there is none. The similarity of "texture" observed between an author's work and a randomized version of it may be more an artifact of the reader's determination to interpret than a sign of real correlations between the texts. A way of testing this notion suggests itself. The computer certainly has no tendency to read between the lines. Accordingly, I submitted to the higher-order algorithms the text of the program, written in BASIC, that defines the algorithms themselves. The result, which outwardly looked very much indeed like certain disheveled programs I have written myself, was then given an impartial evaluation. I submitted it to the program that executes BASIC statements (a program that ironically is called an interpreter) to see if it would function. The test is not quite as unambiguous as one might want. Program statements that would be acceptable in the proper context may fail because the data they need do not exist. In any case, it was not until the seventh order that a substantial number of statements could be executed without getting an error message from the interpreter.

Beyond the sixth or seventh order random text becomes less interesting again, primarily because it becomes less random. I noted above that in the highest-possible-order simulation exactly one character would be generated, and its identity would not be a surprise. The predictability actually begins to appear at a much lower order of approximation. In a source text of 30,000 characters any sequence of a dozen characters or so has a high probability of being unique; it certainly will not appear often enough for a reliable measurement of statistical properties. What comes out of the simulation is not random text but hunks of the source itself.

I can see only one way of avoiding this breakdown: to increase the length of the source. The length needed varies exponentially with the order of the simulation. Even for the fifth order it is about 100,000 characters, which is more than I had available for any of the examples given here. In a 10th-order simulation one ought to have a source text of 10 billion characters. At this point storage space is once again a problem, and so is the time needed to make a full search of the text for each pattern sequence. Indeed, there is a more fundamental limitation: the human life-span. Even prolific authors do not write that much.

```
70 LOCATE 3,10: PRINT "About" "to " TASK$;
140 N=2: P$="Change the printed?";
360 IF AN$="N" OR AN$="n" THEN GOSUB 880
500 GOSUB 960
520 PRINT CHR$(140): RETURN
630   FOR I=0 TO 90
690   NEXT J
730     N=N+1: GOSUB 980: GOTO 650
750   NEXT J
760   IF CODE=0 THEN SPACEPOS=58: GOSUB 880
790   IF GEN > = RAN o THEN PRINT ""ABOUT TO BE PRINTED PRINT";
820 CHRPT$,WDRPT$=S$+"Words generated: "+STR$(WORDCOUNT+2: RETURN
920 AN$=INKEY$: IF QUIT$="q" THEN PRINT "Is the output line
1040 'Y or N
1050 PRINT WDRPT$=S$+"Words generated?"
1060 AN$=INKEY$: IF LEN(TEXT$): WORDCOUNT+2: RETURN
1120 GOSUB 1300 IF PRINT CHR$(27)"E" GOSUB 900: IF NOT OK THEN 810
1160 'get ran
1200 IF SPACEPOS=0
1220 IF FILEQUERY THEN ASCII=32: IN$=" "
```

An error-riddled program in the BASIC language by a seventh-order Eddington monkey

Joan Tomaszewski
Peter Su

(1)

A*RMA VIRU*MQE CANO*:
The Computer Meets Vergil

"Arma virumque cano" - the familiar words take on new meaning during this age of computer technology. Vergil indeed sang of a man, but he sang too of the weapons with which the man conquered and civilized his world, of the use to which man puts his technological achievements. The purpose of this paper is to tell of an experiment in which a computer met Vergil, and to suggest that such meetings between technology and the classics need not be hostile.

My experience with computers is only second-hand, so the information I give you today will be in laymen's terms. Three years ago I was fortunate enough to have in my Latin IV class a young man named Peter Su, who not only was an excellent Latin student, but was taking college level courses in computers at Brown University. Peter brought back reports of assignments to find the number of times the Brown University library elevator stopped at a certain floor. "Elevators?" I exclaimed. "Why don't you do something useful, like teach the computer to scan Latin poetry?" The idea intrigued him. Peter became, so to speak, a <u>legatus</u> between two foreign lands - myself, representing classics, and the computer. He interpreted the concepts of each of us to the other, and explained our limitations. And so the project began. The program took three months to perfect and consists of approximately four hundred steps. Peter used an IBM 370 computer

and the PLC computer language.

Our first expectations, then, were that the computer might be able to scan a line of dactylic hexameter, or, more precisely, to recognize a line which has syllables in the appropriate order to constitute a line of dactylic hexameter. The steps needed were:

I. to adjust for irregular consonants and vowels. These would affect the lengthening of vowels and thus the counting of syllables. Irregularities were:

1) combinations of certain consonants with the second a liquid. The computer would eliminate the liquid, leaving a single consonant. Thus in Illustration I the computer would reduce 'simulacrum' to 'simulacum'.

2) the consonant 'x', which had to be valued as a double consonant. The computer would print it as two 'n's (instead of 'x's, which would have doubled themselves indefinitely) as seen in Illustration II with 'sanna' for 'saxa'.

3) the 'u' after 'q', which does not have the force of a vowel. The computer would eliminate the 'u', as demonstrated in Illustration III with the word 'feraque' reduced to 'feraqe'.

II. to lengthen vowels. To this end:

1) the computer has to be told each time which vowels are long by nature. Peter chose to mark these with an asterisk until the final step when the computer prints the line in traditional letters with macrons. Illustration IV shows 'victorem' written as 'victo*rem'.

2) the computer was, however, taught to recognize diphthongs and to assess them as long. 'Saepe', then, as seen in Illustration V, was rewritten 'sa*pe', showing the length of the diphthong as a long vowel.

3) the computer also was taught to find vowels which are followed by a pair of consonants (liquid combinations and 'x' had already re-evaluated), and to assess them as long. Illustration VI shows 'victa' with 'i' lengthened due to the 'ct'.

A familiar line, then, as viewed by the computer at this point would look like Illustration VII:

A*RMA VIRU*MQE CANO* TO*JA* QI* PI*MUS ABO*RI*S

Notice the lengthening of 'a' and 'u' due to double consonants, the loss of 'u' after 'q', the natural lengthening of the two 'o's, the loss of 'r', the diphthong 'ae' rewritten 'a*', the missing 'u' and 'r', and naturally long 'i's, and

the naturally long 'o' and 'i'. The words 'ab oris' are written as one to avoid their separation as a logical phrase.

It was now time for the computer to check the accuracy of the line as a dactylic hexameter. Although it did not divide the line into feet, it would check for the occurrence of spondee and dactyl combinations. We decided to begin at the end of the line, as it is the most predictable segment. So the computer learned to count off two syllables, — — or — ∪ , as its first (our sixth) foot. The next three syllables, which we think of as the dactylic fifth foot, would be the computer's second foot, — ∪ ∪ . Next the computer would count four more combinations of either — ∪ ∪ or — — , to a total of six feet.

As seems to be the case when dealing with computers, once the computer could identify a correctly written line successfully, our expectations were increased immediately. We now wanted to see whether Vergil, for example, could have written the same set of words in a different order and still have formed a dactylic hexameter. The idea came to be because during a scansion contest conducted yearly in Latin IV, I give a jumbled line and ask students to come up with Vergil's actual line. Occasionally variations, accurate but not Vergil's own, come up. The computer can produce the maximum number of these variations, which I will refer to as permutations, immediately. The idea of beginning at the end of the line, mentioned earlier, proved at this point to be an effi-

ciency measure. The computer could immediately reject all lines which would not comply with the stringent end-of-line restrictions.

We used the first line of the Aeneid as an example. Of the possible 5040 permutations of the words in this line, 8 correct dactylic hexameters were produced, including the original. These are found in Illustration VIII. Interesting are the combinations of words which seem persistent - 'cano ab oris' in 6 of the 8 lines, and 'arma virumque' in every single possible line.

At this point it was striking us that the computer, if it could produce permutations for one line, should also be able to choose from words and put together lines itself. Our next expectation, then, was that the computer might write as many as possible dactylic hexameters expressing a given idea, drawing its options from several given to it.

We began with the quotation "Victorem a victo superari saepe videmus", printed in Illustration IX. For each item in the line we offered the computer an option. These are listed below the original words in the illustration: "Superatorem a superato vinci interdum conspicimus". The computer could then use either line in its entirety or choose some vocabulary items from each to make up new lines, as long as it chose one for each idea. The yield was 11 out of a possible 3840 permutations, including in the 11 the original quotation.

A second attempt was made with the quotation "Non ignara

mali miseris succurrere disco." This is shown in Illustration X. For the options this time we offered "Perita (for 'non ignara') injuriae infelicibus sufficere scio." The surprising result this time was that the only line which could be written from all the 3840 possible permutations was the original written by Vergil.

An increase, finally, was made in the number of options offered, by giving not only one word for each in the line "Una salus victis non sperare salutem", but offering two other possibilities for 'sperare', namely 'cupere' and 'velle', as seen in Illustration XI. The other options, in order of reference to the original line, were 'sola', 'spes', 'superatis', 'nullam', and 'spem'. The yield this time was more significant. The computer turned out 219 possible permutations out of the possible maximum 69120. More importantly, since the computer was choosing from a larger pool of options, it was, in a primitive sense, writing its own lines.

It would seem that the future possibilities for the program are, in a theoretical sense:

1) to expand the program's capabilities so that the procedure is not limited by the one-line restriction, but can produce larger thought units;
2) thus to be able to write at least short poems through the computer;
3) to analyze an author's word choice as related to metrical restrictions; that is, to compile a

list of words which seem to be favorites of an author and see whether the favoritism is due only to "scannability".

In terms of the Latin classroom, the possibilities for further use are:

1) duplication of the program by those students interested in both Latin and computers;
2) creation of a different or more advanced program;
3) study of the program as a means of understanding the rules of scansion better;
4) study of the program as a means of understanding the word choices made by poets and the restrictions imposed by the dactylic structure, points often underestimated by students;
5) study of the possible lines offered by the computer with an eye to seeing why a poet would choose one particular version. Emphasis, word groups, figures of speech, word choice, and idiomatic expressions could be considered. In the "Arma virumque" line, for instance, a student might easily see why Vergil chose to begin his book with the words "Arma virumque cano" - they do, after all, express his purpose in writing. Why, though, of the first two options did he choose the first over the second? Vergil's

skill with hexameters could certainly have brought both options to mind. Was it merely, then, to keep 'qui' and 'primus' together, or was it to bring emphasis to Troy, which was also to play an important role in the book? Or did the line simply roll off the tongue more easily? As teachers we try to train our students to look at Latin poetry and the skill of its authors in a critical fashion. Computer technology may, surprisingly, aid in inspiring students to think about something as far from technology as the creative process of Vergil's genius.

Joan Tomaszewski
Peter Su
June 1983

Permutations

I. Devaluation of liquid consonant combinations _/ simulaca (simulacra)

II. Doubling the value of 'x' as a consonant _/ sanna (saxa)

III. Devaluation of 'u' after 'q' _/ feraqe (feraque)

IV. Vowels long by nature as fed to computer _/ victo*rem

V. Vowels lengthened by computer due to diphthong _/ sa*pe (saepe)

VI. Vowels lengthened by computer before 2 consonants _/ vi*cta

VII. A*RMA VIRU*MQE CANO* TO*JA* QI* PI*MUS ABO*RI*S

Note: ABO*RI*S is the inseparable combination 'ab oris'

VIII. Arma virumque cano Trojae qui primus aboris

Yield: 8 out of possible 5040 permutations (includes original)

Arma virumque cano Trojae primus aboris

Trojae primus qui cano aboris arma virumque

Primus Trojae qui cano aboris arma virumque

Trojae qui primus cano aboris arma virumque

Qui Trojae primus cano aboris arma virumque

Primus qui Trojae cano aboris arma virumque

Qui primus Trojae cano aboris arma virumque

IX. victorem	a victo	superari	saepe	videmus	(original)
superatorem	a superato	vinci	interdum	conspicimus	

Yield: 11 out of possible 3840 permutations

X. non ignara	mali	miseris	succurrere	disco	(original)
perita	injuriae	infelicibus	sufficere	scio	

Yield: 1 out of possible 3840 permutations

XI. Una	salus	victis	non	sperare	salutem (original)
Sola	spes	superatis	nullam	cupere	spem
				velle	

Yield: 219 out of possible 69120 permutations

---June, 1983

db 0949c

Addendum - page 4, following "as a logical phrase"

Instances of elision were not written with any special code by the computer. The computer was taught to ignore the 1st vowel in a combination of a vowel or vowel plus m at the end of one word followed by a vowel at the beginning of the next word. Thus, looking ahead for a moment to Illustration II, "victorem a victo superari saepe videmus" would be accepted by the computer as a correct hexameter in spite of the extra vowel early in the line (em...a), resolved by what we know as elision. The computer would not count the e in em as one of the line's vowels.