THE MIND SYSTEM

Martin Kay
University of California, Irvine

The MIND system is a single computer program incorporating an extensible set of fundamental linguistic processors that can be combined on command to carry out a great variety of tasks from grammar testing to question-answering and language translation. The program is controlled from a graphic display console from which the user can specify the sequence of operations, modify rules, edit texts and monitor the details of each operation to any desired extent. Presently available processors include morphological and syntactic analyzers, a semantic file processor, a transformational component, a morphological synthesizer, and an interactive disambiguator.

MOTIVATION

For the most part, linguists are unaware of the importance that computers must one day have for their subject. The exact extent of the contribution that computer models of language acquisition, speech production, speech understanding, and the like will make to theoretical linguistics is uncertain; that it will

be considerable is hardly open to doubt.

To the extent that linguists attempt to produce formal descriptions of particular languages, the computer is an absolutely indispensable tool. A formal description can only be verified by checking it against actual cases. It must produce words and sentences that actually occur in the language, produce sets of sentences that native speakers accept as paraphrases of one another, answer questions correctly, or whatever. In other words, it must be possible to base a process or performance on the description and to have a speaker of the language judge it satisfactory.

One of the earliest lessons learned from computers was that human beings are curiously ill adapted to the business of describing processes accurately and in detail. When the notion of a computer had gained some currency, but there were still few machines available, it was thought that a person could write a program to carry out a complex process and that he would be able to put it to work as soon as he could get his hands on a machine. But, while the overall design of the program may well have been correct, untold numbers of minor adjustments had invariably to be made before it would produce correct results. It is now common knowledge that the time required to produce the first draft of a program is small compared to the time needed to test it, make corrections, test again, and so on to produce a satisfactory version.

If programs are complicated, formal descriptions of natural languages, or even of small parts of them, are more so. Modern linguists have been ingenious in designing economical notations for their descriptions so that a mistake in a single character can have widespread repercussions. Furthermore, if these descriptions are thought of as specifying processes, then these processes must often be nondeterministic. What this means is that the chain of events is not, in general, uniquely specified. Starting at a given point, there are typically several different directions in which the process might

continue and each of them must be pursued independently. Nondeterministic processes are fundamentally more difficult to specify than deterministic ones. Therefore, there are three options open to linguistics: It can restrict itself to theoretical and philosophical speculation about the form that linguistic descriptions should take and the consequences that this has for the human faculty of language; it can content itself with informal descriptions of particular languages as it has at times in the past; or it can accept the computer as a necessary tool of the trade. If linguistics is to make any contribution to the practical affairs of men, it is likely to be by showing how to make machines with which men will be able to communicate in everyday language. The utility of a machine with this capability would be obvious. Language translation, information storage and retrieval, the production and editing of text, and a host of other applications come readily to mind. Any such machine would necessarily incorporate a more or less complete formal description of at least one natural language. Whether machines will ever, in fact, be constructed to converse with men in their own language is open to doubt. It has even been questioned that it is a worthy goal to pursue. It is certain, though, that the success of the enterprise would depend on the success that linguists have in developing adequate formal descriptions of languages and that, in turn, rests on the use that linguists make of computers.

Linguistics is in a state of turmoil. There are a number of schools of thought, each with as many variants of the basic theoretical position as the school has adherents. Whatever seems most solid today is most likely to be overthrown tomorrow. The climate of opinion has recently become one where status is accorded only to contributions that touch the foundations of the science. Nevertheless, there is widespread, if often grudging, agreement on a number of broad issues such as: the study of the forms *of* words can be conducted independently of the study of sentences, and that it is useful to regard each sentence as having one or more syntactic structures; a syntactic structure has the shape of

a tree, etc. In other words, there is at least some measure of agreement on the headings under which the various phases of linguistic analysis should fall — phonetics, phonology, morphology, syntax, semantics. There is less agreement on just what each of these headings covers.

In this situation, the aspiring designer of a linguistic computer system is faced with a difficult set of choices. He may decide to wait until a clearer picture of linguistic theory begins to emerge: if he adopts this policy there is no knowing how long he will have to wait. He may decide to throw in his lot with one of the contending schools; this is a particularly hazardous policy, not only because of the restlessness of modern linguistic theories, but also because few, if any, of them are readily interpretable in terms of processes that can be carried out on a computer. The third alternative is to design a kit of sufficiently versatile computational tools to build computer systems that reflect one version or another of some prevailing theory (more or less liberally interpreted by a user of the tools).

This last solution is, in many ways, ideal and it is the one that was adopted in the design of the MIND system, the subject of this talk. The system designer that follows this course must attempt to reconcile three different, and often conflicting requirements. The tools he builds must be flexible enough to be usable by the advocates of many actual and potential linguistic theories. On the other hand, they must be so well fashioned for specific purposes that it will never be unduly burdensome for a computationally naive user to adapt them to his own view of linguistic theory. Finally, each tool in the kit must be designed so as to be compatible with the rest. Just as phonograph records must turn at the same speed if they are to be playable on generally available equipment, so standards must be established covering the way in which the data will be represented, on which various cooperating programs are to operate. Many attempts to construct packages of programs which serve the needs of a diverse community of users have foundered. Attempts to design coherent sets of programs

for analyzing statistical data are a case in point.

It goes without saying that the MIND system falls far short of the ideal of providing a linguistic computer system that can be all things to all men. Nevertheless, it has already been applied with some success to a considerable variety of linguistic tasks, and only the time and energy is wanting to apply it to a great many others.

THE OVERALL STRUCTURE OF THE SYSTEM

I began this talk by pointing out that formal descriptions of languages can only be made with the aid of a machine to verify that the description corresponds to the data. Computer programs written especially for this purpose are usually called "grammar testers," and the most notable of them was written by Joyce Friedman. Used as a tester of transformational grammars, the MIND system differs from the Friedman program mainly in its interactive facilities. The user can change any rule in the grammar at any time without disturbing the remainder, and can immediately see the effects of the change by applying the modified grammar to any basic sentence structure that takes his fancy.

The system can also be used to test grammars in a variety of different formalisms designed specifically for the task of analyzing, rather than generating sentences. An unusual feature of the system is that, despite the apparently different strategies involved, the same computer routines are used for generating and analyzing sentences. This is made possible by strictly adhering to a set of principles on which I will elaborate shortly, and which are the subject of another talk in this volume. By giving a command from his terminal, the user of the MIND system can link together several of its routines so as to make it carry out some more or less complex sequence of processes automatically. One such command causes the system to constitute itself as a question-answering machine, others as language

translators of various kinds, and so on.

The set of programs that make up the MIND system can be changed very readily when the need for new facilities arises, and as superior methods of performing some of the tasks become known. This is easy because each of the principal components of the system operates, as far as possible, independently of the remainder. The only major part of the system with which all the rest interact is the so-called "master scheduler" whose job is to dictate the overall sequence of events, to locate the input data for each routine, and to determine where the output of each is to be placed. The principal components of the system, as implemented on the IBM 360, Model 65, at the Rand Corporation are:

1. The Morphological Analyzer
2. The Syntactic Processor
3. The Disambiguator
4. The Semantic Processor
5. The Output Component

Earlier in this talk, I stressed the fact that cooperation among computer programs depends, more than anything else, on the maintenance of a set of standards to which the data that passes between them must adhere. In other words, there must be well-defined formats for any data that will be treated by more than one of the programs, but these formats must nevertheless be flexible enough to allow each user of the total system the freedom he needs. The problem is particularly delicate in the design of the MIND system because the user is free to choose any one of a large number of configurations of the overall system.

The output of one program may have to serve as input for a variety of other programs, depending on the configuration. For example, when the system is used as a tester of transformational grammars, the morphological analyzer reads a series of basic sentences from the user's terminal and stores them in the machine. The output of this process is passed to the syntactic processor which applies phrase-

structure and transformational rules to generate the surface forms of the sentences. These are then fed to the output program which displays them at the user's terminal. On the other hand, the system might be given a configuration corresponding to a simple translating machine in which the following was the typical sequence of events: a sentence to be translated is obtained by the morphological analyzer, either from the user's console or from a file of text. The morphological analyzer looks up each word in the text, making due provision for inflexional prefixes and suffixes, and for making changes in the basic form of a word that these sometimes entail. (The rules provided to the morphological analyzer would, for example, enable it correctly to associate the word "tries" with the dictionary entry of the word "try.") The output of this program, in which each word is accompanied by information about it obtained from the dictionary, now becomes the input to the syntactic processor which performs an analysis yielding one or more grammatical structures for the sentence. These are then passed to the disambiguator, whose job is to decide which grammatical structure is correct in the given context.

SYNTACTIC ANALYSIS AND THE CHART

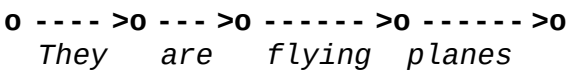
Many of the programs in the system refer to their own files whose format is not, therefore, constrained by other components. All programs are in contact with the master scheduler which decides when to call them into operation, but which passes virtually no information to them directly nor expects any from them. Information passes from one program to another through a single common data region called the "chart." The chart is a machine representation of a directed graph with labeled vertices and edges. The labels on the nodes are of minor significance, serving mainly as an aid in checking the operation of new programs. The labels on edges are, in general, complex. The chart is interpreted as a transition network, the edges emanating from a

given vertex being treated as mutually exclusive alternatives. A transition from one vertex, or state, to another corresponds approximately to left-to-right progress through a string. The chart as a whole can therefore be looked upon as a grammar which generates a set, possibly infinite in size, of strings of symbols.

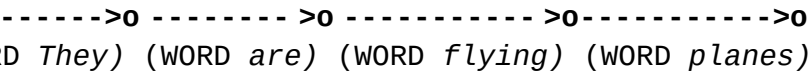
Consider the sentence:

They are flying planes

This can be represented by the following trivial finite-state grammar:



or, alternatively, by the following:



The additional structure in the second proposal has the advantage of distinguishing edges that represent words from others that the chart might also contain.

The program that reads a sentence from the input device leaves the chart looking approximately as in the second of the above diagrams. The job of the morphological analyzer is to look each word up in a dictionary and to add the information it finds about each at the appropriate place in the chart. After this has taken place, the chart might be expected to look somewhat like Figure 1.

This is in fact a considerable simplification, but it will serve to illustrate a number of points. First, none of the original edges have been changed. The morphological analyzer, like most other components of the system, restricts itself to adding new nodes and edges to the existing chart. This policy

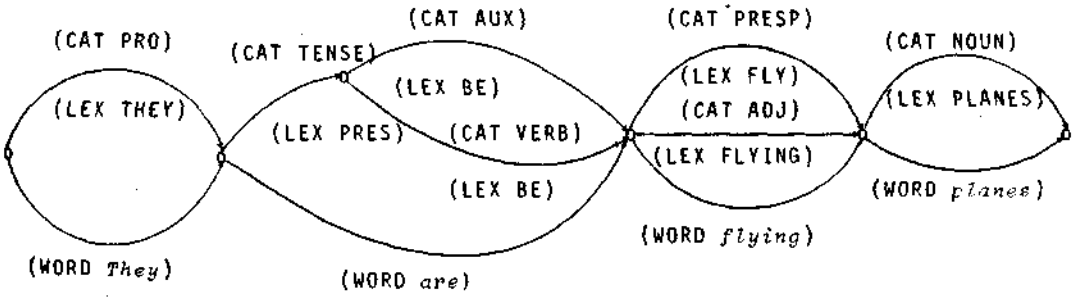
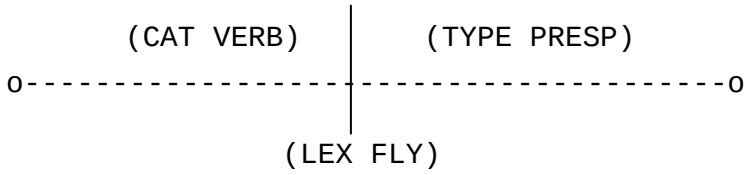


Figure 1

helps assure the relative independence of programs in the system. If the morphological analyzer were to delete existing material from the chart, then other programs would be committed to the kind of analysis of the sentence made by that program. As it is, another program can modify the analysis or ignore it because the original data is still available.

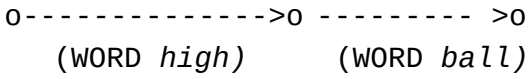
Each word in the initial sentence gave rise to a single edge joining a pair of vertices. The second vertex could be reached from the first in just one way, namely by following that edge. Successful morphological analysis results in at least one, and sometimes several new paths from the first to the second vertex. In the case of a syntactically unambiguous word, one new path is added; where there is ambiguity, a new path is added for each syntactic interpretation of the word. The original edges carried a label of the form (WORD x), where x is a word of the original sentence. The new edges have more complex labels. Each of them has, in fact, the structure of a tree with nodes labeled by lists of so-called "attribute-value" pairs. There is, for

example, an edge with the label:

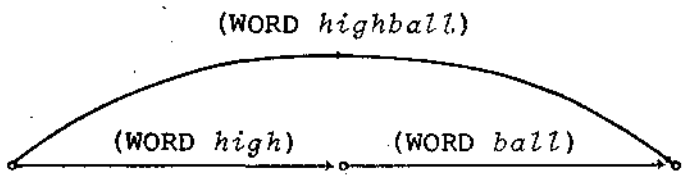


This records the fact that the lexical entry "flying" can be a present participle, that is, a verb of type "PRESP."

Suppose that a text contained a phrase like "high ball" which can, but need not, be interpreted idiomatically. In a sporting context, for example, it might be intended literally, whereas, where drinking was in question, it has an interpretation that is independent of the meanings of the individual words. The initial chart would contain the following segment:

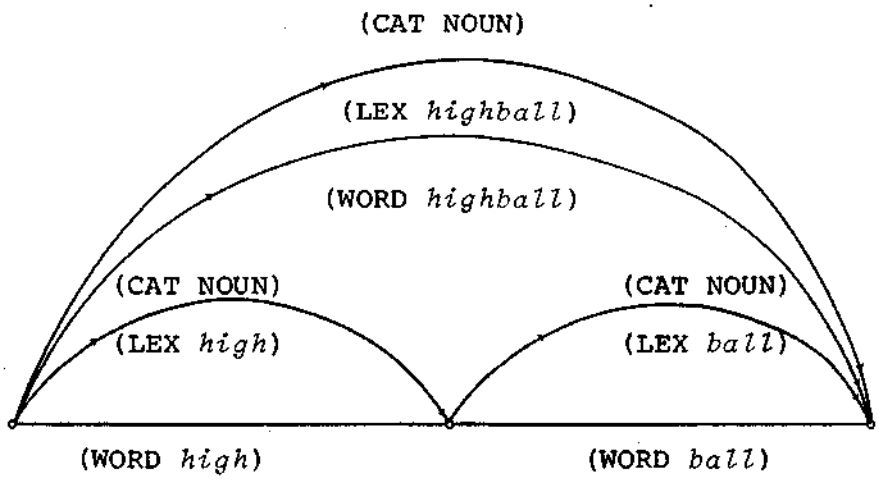


Perceiving that an idiom might be in question, the morphological analyzer begins by adding a new edge as follows:



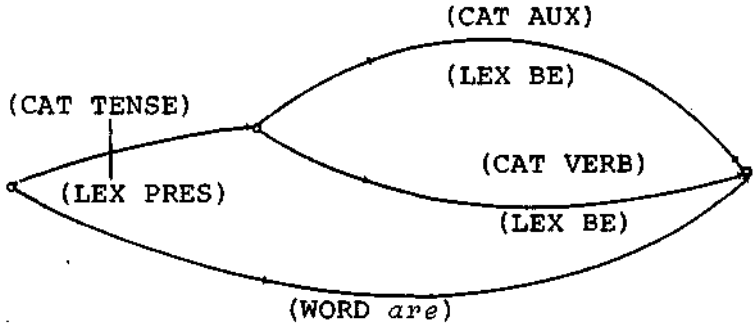
There are now two ways of traversing this section of the chart, one corresponding to the idiomatic, and one to the non-idiomatic interpretation of the phrase. The morphological analyzer now goes on to treat these two alternatives independently, adding

to the chart the dictionary information appropriate to each. The result might be somewhat as follows:



Initially, each edge represents a word, and the vertices represent the spaces between them. Once analysis begins this correspondence breaks down. An idiom is represented as a single word even though it contained a space in the original text. However, this corresponds to an intuitive view of what an idiom is — a word that contains spaces.

In an earlier example, I represented the result of applying morphological analysis to the word "are" as follows:



Each interpretation begins with an edge representing the tense of the verb and, since this is the same in both cases, a single edge serves for both. The two following edges represent "are" as an auxiliary and as a main verb respectively. My main concern here is not to argue for this particular analysis. Suffice to say that tense regularly appears separated from its associated verb in the deep structures of sentences. For the present, I am concerned only to point out that a word may be analyzed as a sequence of more than one segment if, for any reason that appears to be appropriate. This is the exact inverse of what is happening in the case of idioms.

Syntactic analysis is the process in which one or more tree structures are developed for a sentence, each corresponding to one way of breaking it down into meaningful parts. Each structure is based on exactly one path through the chart that results from morphological analysis. In other words, a subsidiary effect of syntactic analysis is to choose one of the alternative analyses of each word provided by the morphological processor.

In the course of the syntactic analysis, various hypotheses will be formed about the phraseological status of various parts of the sentence. Consider, for example, the sentence

Reading books can make this work

The first two words clearly constitute the subject, and the remainder the predicate of the sentence, but both parts can be interpreted in at least two ways, each corresponding to a different syntactic analysis. Thus, "books" is either a noun modified by the adjective "reading" or it is the object of the verb "read." This gives us a distinction like that between

The reading books can make this work

and

The reading of books can make this work

The predicate is also ambiguous, its two interpreta-

tions correspond approximately to

Reading books can achieve this work

and

Reading books can cause this to work

The sentence as a whole therefore has at least four interpretations.

The syntactic analyzer will use one of a large variety of strategies, each of which will result in the various interpretations of the various parts of a sentence being developed in a characteristic order. One will seek one interpretation of the first few words under which it could function as subject of the sentence, and then attempt to interpret the remainder as a predicate compatible with that view. It would then seek alternative compatible interpretations of the predicate before seeking new hypotheses about the subject. If new hypotheses about the subject are, in fact, found, it is clearly desirable that the work of analyzing the predicate should not be repeated. Another strategy would involve exploring all possible subjects first, deferring until later the search for compatible predicates. Those for which none are found are simply abandoned. The particular strategies that can be adopted in the MIND system are discussed in detail elsewhere; what is important here is the observation that, to operate efficiently, they all require some way of recording hypotheses about the phraseological status of parts of the sentence so that they will be available for use in constructing hypotheses about larger parts at some later time.

A hypothesis about the phraseological status of part of a sentence is an alternative interpretation of it differing in no important way from the alternative interpretations of individual words obtained from the dictionary. Indeed, it is formally indistinguishable from an idiom. So, for example, the chart for the sentence

Reading books can make that work.

after one hypothesis about the subject has been

developed, might be as follows:

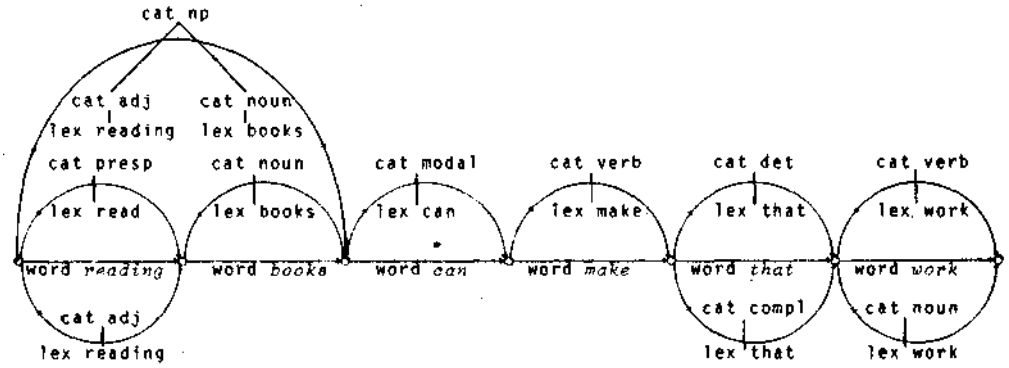


Figure 2

Suppose that the strategy calls for predicates compatible with this view of the subject to be sought next. The chart might then look like Figure 3. The next step is to seek alternative subjects, yielding the following Figure 4.

Finally, predicates must be sought to go with this subject. However, examination of the chart shows that the only candidates have already been found and recorded as alternative interpretations of the words "can make this work." In summary, phrases are simply ways of interpreting parts of a sentence other than as sequences of individual words.

The synthetic analysis of ordinary language can involve more than simply collecting words and

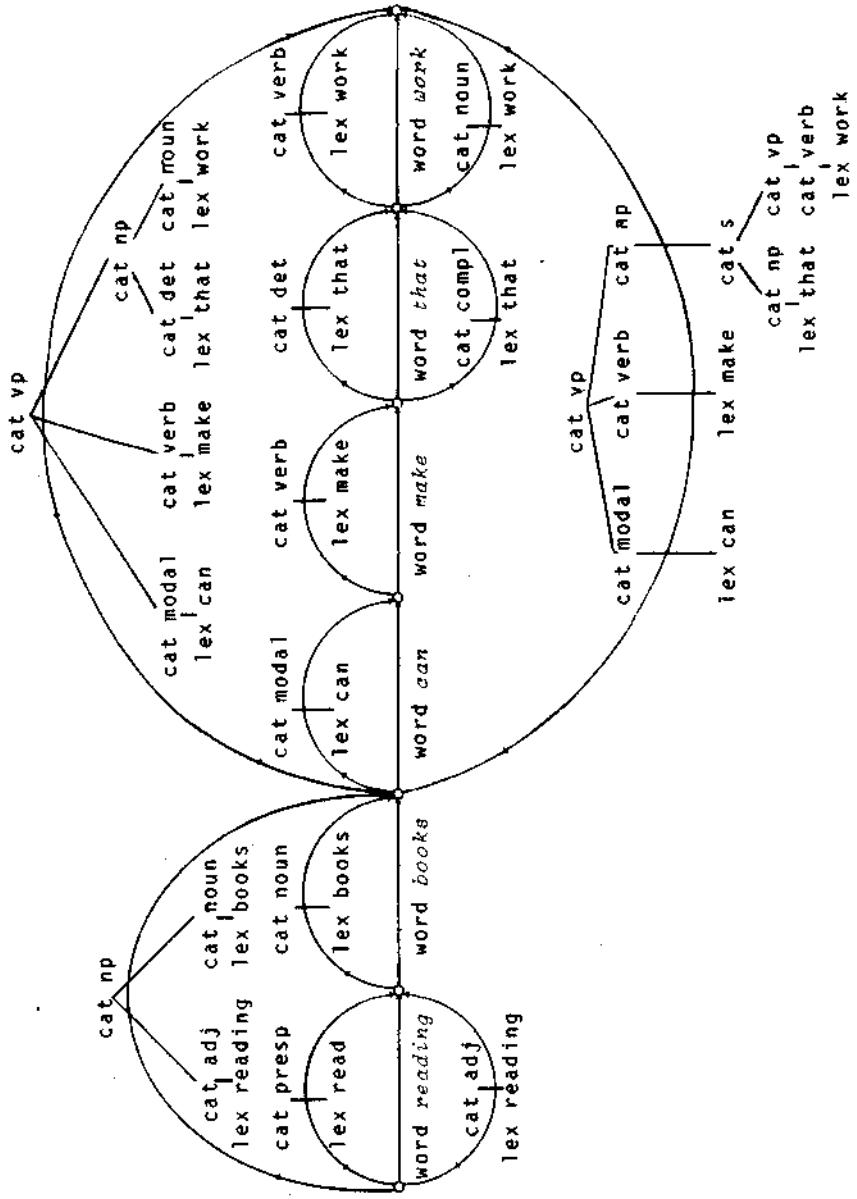

cat np
cat adj
lex reading
cat noun
lex books
cat vp
cat modal
lex can
cat verb
lex make
cat np
cat det
lex that
cat noun
lex work
cat np
cat adj
lex reading
cat presp
lex read
cat noun
lex books
cat modal
lex can
cat verb
lex make
cat det
lex that
cat verb
lex work
word reading
word books
word can
word make
word that
word work
cat vp
cat modal
lex can
cat verb
lex make
cat np
cat s
cat np
lex that
cat vp
cat verb
lex work

Figure 3

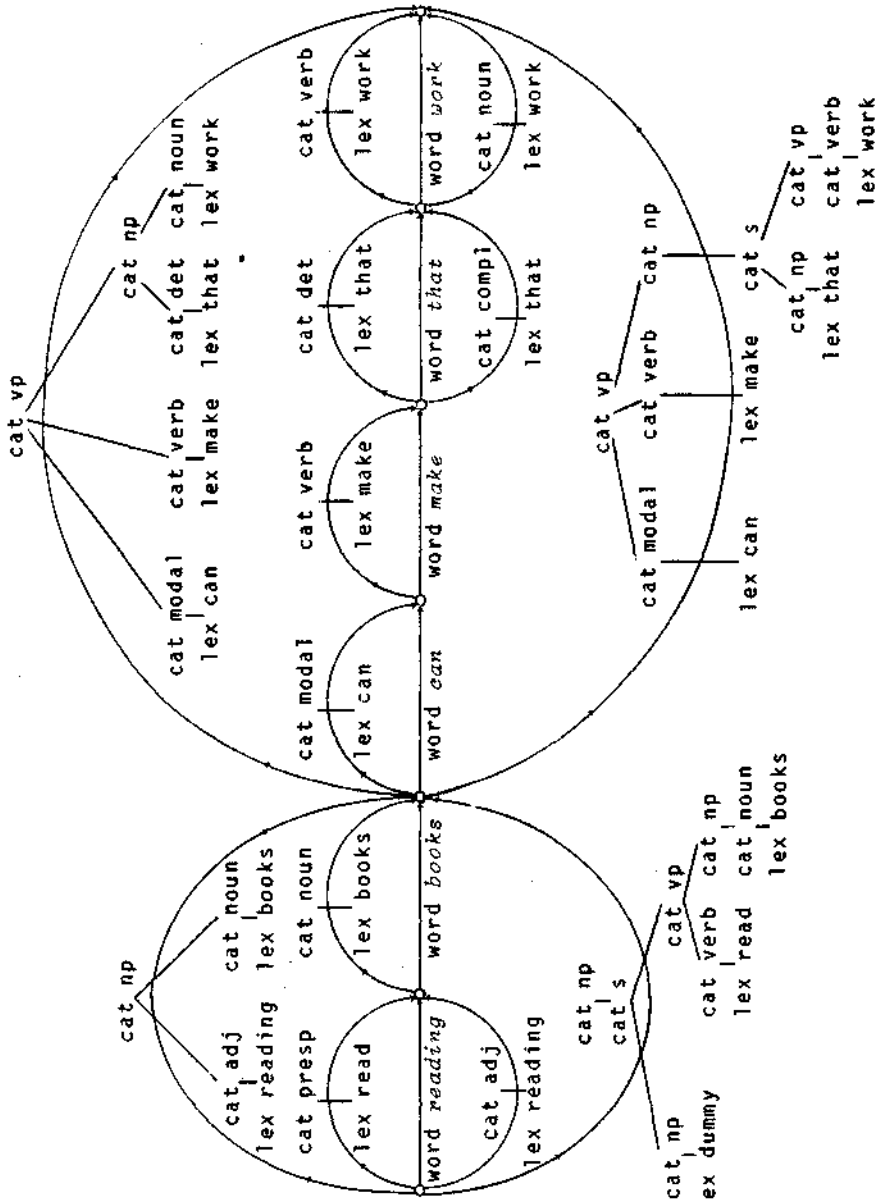

cat np
cat adj
lex reading
cat noun
lex books
word reading
word books
cat adj
lex reading
cat np
cat s
cat np
lex dummy
cat vp
cat verb
lex read
cat np
cat noun
lex books
cat vp
cat modal
lex can
cat verb
lex make
cat np
cat det
lex that
cat noun
lex work
word can
word make
word that
word work
cat modal
lex can
cat verb
lex make
cat det
lex that
cat compl
lex that
cat verb
lex work
cat noun
lex work
cat vp
cat modal
lex can
cat verb
lex make
cat np
cat s
cat np
lex that
cat vp
cat verb
lex work

Figure 4

and phrases to make larger phrases. If no more than this were done, then it would never be possible to associate one syntactic structure with more than one sentence. However, in modern linguistics, it is usually held that syntactic structures, in general, underlie a family of sentences which mean more or less the same and which are related in a systematic way. In other words, it is the job of syntactic analysis to reduce sentences to canonical forms each of which may underlie several actual sentences. For example, the sentences

John gave some flowers to Mary

and

John gave Mary some flowers

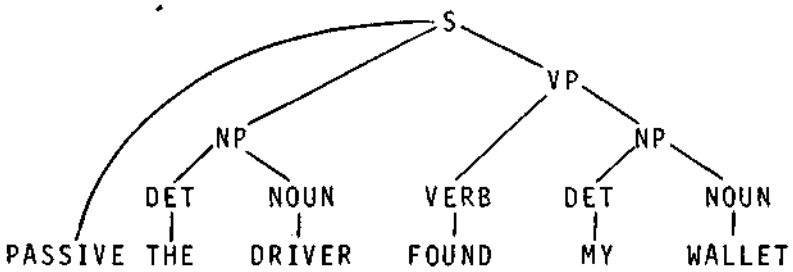
differ in no way that could be important for later processing. They are therefore assigned the same syntactic structure. The sentences

Mary was given some flowers by John
Some flowers were given to Mary by John
Some flowers were given Mary by John

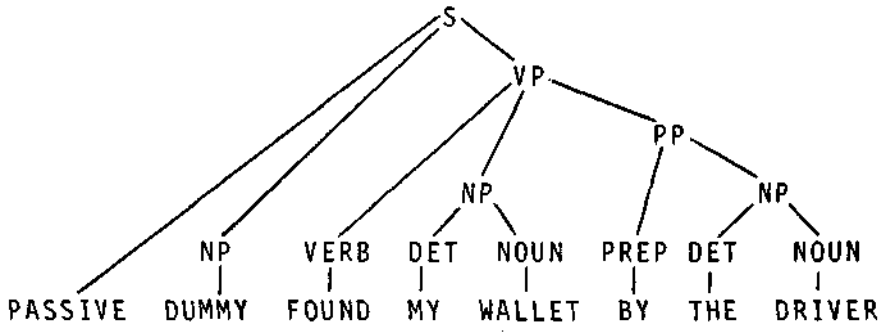
might or might not be assigned the same structure, depending on the theoretical views of the linguist and the overall aims of the project. They would certainly be given more similar structures than is suggested by the different sequences of words, and "John," for example, would doubtless be identified as subject in each case. Suppose that passives were to be given the same structures as the corresponding actives and, furthermore, that their common structure was to look approximately like a decomposition of the active sentence into phrases. One structure of the sentence:

My wallet was found by the driver

might therefore be something like the figure at the top of the following page.



Notice that there is also a second interpretation in which the phrase "by the driver" indicates the place where the wallet was found rather than the person who found it. In this case the structure would probably be more like:



What this means is that it must be possible to represent hypotheses about the structure of a sentence in which the elements making it up appear in various orders. A grammar rule must be able to establish a new order of elements for consideration by later rules without destroying the original order. Omitting irrelevant details, this can be done in the chart as follows in Figure 5.

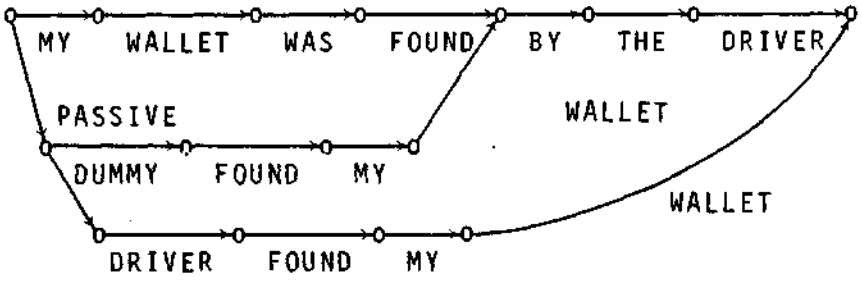


Figure 5

THE DISAMBIGUATOR

The MIND system contains a component called the "disambiguator" which is, at present, a fairly simple program but which can be expected to become considerably more elaborate in some applications of the system. In others, it will have little or nothing to do. The job of the disambiguator is to design and implement *a* strategy for removing the ambiguities that remain in a sentence after syntactic analysis. By saying that it designs a strategy, I mean simply that it chooses from a set of possibilities that may sometimes be quite large, the one that seems likely to involve the least work, and implements that.

Suppose that at a given stage in the disambiguation process *a* considerable number of interpretations of a sentence remain in play. The program attempts to compose a question about the sentence and the text that preceded it whose answer would effectively divide this number in as many parts of approximately equal size as possible. Suppose eight alternatives are still open and one question concerns only one

of them, a second question would eliminate either the first or the second group of four, and a third question would have four possible answers, each of which would eliminate all but two alternative analyses. The last question would be chosen, and the first would be considered least desirable.

How are the questions answered? Two possibilities are open, and only one of them has been explored in any depth. The questions can be directed to the semantic component of the system in the hope that an answer can be found based on the understanding that has been reached of the text up to this point. Alternatively, the question can be expressed in English and directed to the user of the system. An intriguing third possibility is to do both of these, always preferring the user's answer, but comparing it with the answer supplied by the semantic program so as to verify its operation.

Consider the sentence:

They filled the tank with gas

and suppose that the system is set up to translate into French with the help of a human collaborator. The entire text is displayed on the screen, sentence by sentence, for the human to read. When this one is reached, the system might ask:

DOES THE WORD "TANK" REFER TO

1. A MILITARY VEHICLE, OR
2. A VESSEL FOR FLUIDS?

If the user types a "1," the system will translate the word as "char d'assaut," otherwise it will translate it as "tanque." It might then go on to ask:

DOES "GAS" REFER TO

1. GASOLINE, OR
2. VAPOR?

Answer number 1 will lead to the translation "essence"

and number 2 to "gaz." A third question might be:

DOES THIS MEAN
1. "FILL WITH GAS," OR
2. "TANK WITH GAS"?

The correct answer to this is almost certainly number 1 which would result in "with" being translated as "de" rather than "avec." Finally, it might ask:

DOES "THEY" REFER TO

1. THE SOLDIERS
2. THE TANKS
3. THE SHELLS
4. THE ENEMY
...etc.

The potential answers to this question probably consists of nothing more than a list of recently used nouns. The object of the question is to determine whether "they" should be translated as "ils" or "elles." A better question would be:

WHAT DOES "THEY" REFER TO?

and this would work if the human collaborator could be counted upon to reply with a word that had actually been used in the text, or if failing this the disambiguator could recognize the answer as being a synonym of a word that had been used. The point is a delicate one because pronouns derive their gender not from the objects to which they refer, but from the words previously used to refer to those objects.

The disambiguator is a program that can be called into play by any other component in the system. Broadly speaking, it will be more effective the later it is called in the overall analysis process because more information will have been amassed to use in the construction of questions. The disambiguator will therefore be better able to profit from its ability to choose the question that will eliminate the greatest number of alternatives. Notice that,

although the word "tank" can function as a verb, this possibility was never raised by the disambiguator. This is because we imagine that the program was called after syntactic analysis, a process that eliminated this possibility. The third question — about what "with gas" modifies — arises only as a result of syntactic analysis and could not have been asked earlier.

The question of which other programs call the disambiguator, and for what purposes, involves an element of strategy. There is a trade off between calling it frequently and thereby foreclosing unproductive lines of analysis as soon as possible, and calling it rarely and late so as to minimize the total number of questions that need to be asked. Suppose that another of the sentences to be translated into French were:

He saw the girl with the telescope

If the disambiguator were called immediately after syntactic analysis, it would probably have to ask something like:

DOES THIS MEAN

1. "SAW WITH THE TELESCOPE," OR
2. "GIRL WITH THE TELESCOPE"?

But the syntactic ambiguity that this question is designed to resolve can, and indeed should, be preserved in the French. If each of the grammatical structures provided by the syntactic analyzer were pursued independently, it would emerge that there is at least one translation that can be generated from both of them. The question is simply whether the elimination of questions of this kind is worth the labor of following each analysis to a complete translation and then comparing the results. If the questions were being answered by another program, then it might be desirable to resolve this kind of ambiguity early; if they are being answered by a human, then it might be more important to minimize the number of questions asked. However, the user of the MIND system must decide these matters of policy.

SEMANTICS

The semantic component of the MIND system is in an early stage of development. It is here that the greatest development is to be expected. The effort that has gone into the system will be justified in large measure by its use as a test bed for semantic processors. Serious work on semantics need not wait for complete grammars or dictionaries to be written but it is made immeasurably easier by an environment in which fairly large and elaborate grammars and dictionaries can be processed easily.

The principal function of the semantic component is to mediate between the chart and a semantic file which contains the systems' knowledge about the outside world. New information arrives in the form of sentences whose syntactic structures are placed in the chart by the syntactic analyzer. The semantic routines must examine these structures and modify the contents of the semantic file to include the new information. If the system is to answer users' questions, then it will be up to the semantic processor to recognize which questions require an answer, seek the necessary information in the semantic file, and either output an answer directly, or, more probably, build a new syntactic structure which will be translated into an answer by the syntactic generator.

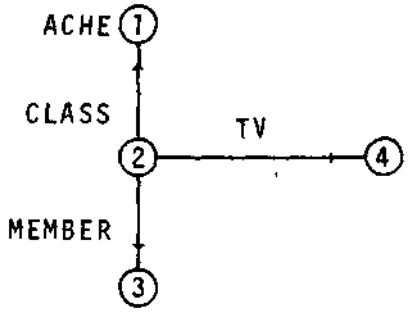
The semantic file, in the existing processor, is a computer implementation of a directed graph with labeled vertices and edges. The processor contains primitive functions that can add and delete vertices and edges. The vertices correspond to objects that the system knows about. In other words, they are the potential referents of linguistic expressions. If more than one vertex corresponds to the same object in the external world, it is presumably because the system does not know them to be the same. Some of the vertices correspond to propositions and edges connect these to other vertices representing objects implicated in them. Some of these propositions correspond to the beliefs of the system, whereas others figure only as terms in other propositions. Thus, for example, if it is part of the

system's knowledge that Bertrand Russell believed that God does not exist, then a proposition corresponding to the sentence "God does not exist" must be stored in the semantic file. However, this does not commit the system to any position on the existence of God. The vertices representing propositions the system is committed to are connected to a distinguished vertex, representing approximately "the Truth" by an edge with a special label. Edges represent relations and, whenever a particular relation is represented in the system, its inverse is also represented so that it is, in effect, possible to follow edges in both directions. Given a vertex representing a proposition, it is possible to discover if the system assents *to* it. It is therefore also possible to discover all the propositions that the system assents to.

There are as many approaches to the problems of semantics as there are people who have considered them. Two trends are, however, discernible. On the one hand, there are those who believe, with Chomsky and his followers, that the first requirement is for a universal semantic alphabet — a set of primitive objects or atoms — in terms of which meanings can be represented. On the other hand, there are those who hold that the meanings of words and phrases consist only of relations they contract with other words and phrases and that the search for any more fundamental kind of representation for them is futile. The second, so called "structuralist," position tends to be taken by the builders of computer models for reasons that are beyond our present scope. It is the position we took in developing the first semantic program for the MIND system.

Access to the semantic network is through vertices that are labeled with words of the language. By no means do all vertices carry such labels. Typically, a noun names a vertex corresponding to a class of objects which that noun names. The term "object" must, of course, be interpreted broadly. The word "ache" names a class of instances of "ache." A particular ache which might afflict a particular person on a particular occasion is represented by another

vertex which is not directly associated with a word in the language. The association between the two vertices is established somewhat as follows:



Vertex 1 — the only one with a label — represents the class of all aches. Vertex 3 represents the particular ache in question. The reason for separating these two needs no elaboration. Vertex 2 represents the proposition that the object represented by vertex 3 is a member of the class represented by vertex 1. Vertex 4 is the special point already mentioned to which all propositions are linked that count as true for the system. Vertex 2 is included in the network precisely because it may, in general, have differing statuses relative to the system's beliefs. For example, the network may also contain the information that someone does not believe that the object represented by vertex 3 is, in fact, an ache. The force of the labels "Class" and "Member" is evident. "TV" stands for "Truth Value" and will presumably be used only to label lines that terminate at vertex 4.

Consider, now, the sentence "John ran." I have shown numbers in the circles representing the vertices only to make them easier to refer to. Leaving aside the question of how the tense of the verb is to be represented, this might give rise to a structure of the following kind in the network.

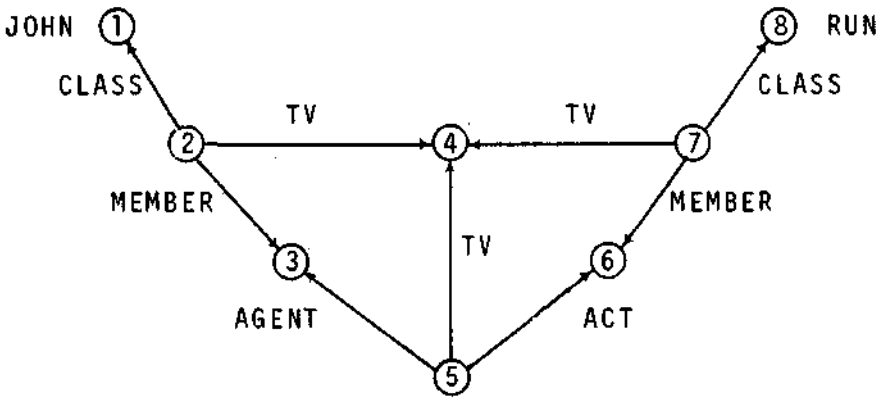


Figure 6

The main proposition is represented by vertex 5 which is marked as being among the beliefs of the system. Vertex 3 represents the particular person who is alleged to have run on this occasion. Vertex 1 represents the class of all people named "John." Notice that proper nouns are treated as essentially similar to common nouns, such differences as there are being treated as a matter of syntax. The reason is simply that there are few, if any, words that are true proper nouns in a semantic sense. "John" does not uniquely identify a person and therefore differs from a phrase like "the man" mainly in that it is not preceded by a definite article. The function of vertex 2 is the same as in the previous example.

Vertex 6 stands for the particular act of running in which John was involved and vertex 8 is the class of all acts of running. It must clearly be to vertex 8 that the word "run" is applied. It remains to argue for distinguishing vertices 5, 6, and 8. Intuitively, there is certainly a distinction between the class of acts or states that a verb names and the individual events and states that are members of that class. It is important to preserve this distinction in the network if only because of the possibility of such sentences as "Bill saw John

run" and "Brutus killed Caesar and Cassius saw it happen." What was seen, in each case, was an event like the one represented by vertex 6 and not a proposition such as vertex 5 represents. On the other hand, the sentence "Bill knew John ran" involves a reference to vertex 5 because it is the proposition, and not the event, that is believed.

I made no attempt to represent the tense of the verb in the example even though it is clearly more than a grammatical category. The reason is that, despite a great amount of effort that has been devoted to it, tense is very difficult to incorporate in a semantic model which attempts, as this one does, to keep track of the referents associated with a text. It makes no sense to have vertices with labels like "present" and "past" and to link them with propositions in some standard way. Clearly every event was present when it occurred and has been past ever since. An alternative that has been explored from time to time is to associate a specific time with each event. This is what would be required to give an accurate depiction of a set of events. But the information needed to do this is simply not obtainable from ordinary texts. The best that can be attempted — and even this is extremely difficult — is to establish a partial ordering over events. The difficulty comes from the fact that languages like English encourage great imprecision in recording even these partial orderings. Languages like Chinese treat them in an even more cavalier manner.

One of the most important questions that arises in designing a semantic file concerns the treatment of quantifiers. The reason they are so important is that sentences involving quantifiers tend to represent general statements which can be used in deducing facts from other facts. This is something that any semantic processor must be able to do because it is rare that the information it is seeking will be in the file in just the form required. Usually the correct answer to a question will be obtainable only by inference. If the system knows that John is Bill's father and Bill is Mary's father, and if it knows that the father of a person's father is that person's

grandfather, then it should be able to answer the question "Who is Mary's grandfather." This ability multiplies the potential utility of each piece of information in the system by an incalculable factor.

The sentence "All men are mortal" might be represented in a semantic network somewhat as follows:

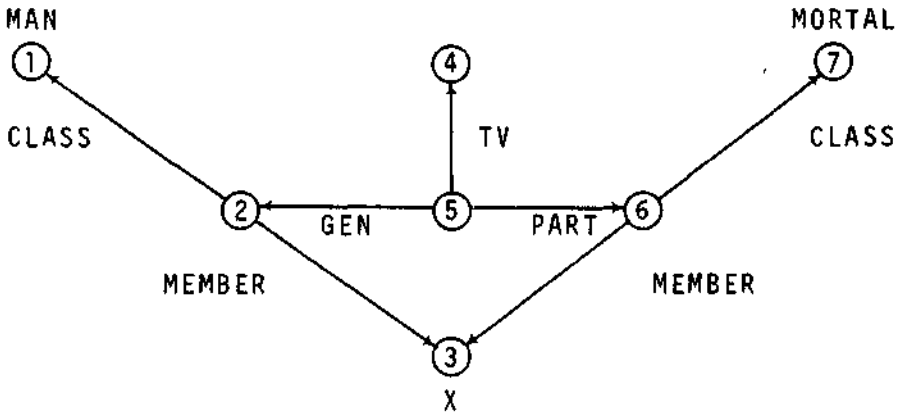


Figure 7

As before, vertex 4 represents the truth as the system sees it. Vertex 5 represents a proposition to which the system assents. The names "Gen" (for "General") and "Part" (for "Particular") have been chosen to represent two sides of the implication relation which, as usual in this system, must be factored into two parts so as to admit an intermediate propositional vertex. Proposition 5 is to the effect that the proposition represented by vertex 2 entails the one represented by vertex 6. Notice that neither of these is represented as being part of the program's system of beliefs. The program is committed to neither 2 nor 6 but only to the view that 2 cannot be true unless 6 is also.

Vertex 3 is labeled "x" to indicate that it is

of a special kind. It represents a so-called "free variable" which can be identified with any other vertex whatsoever and assertion 5 should still hold true. The universal quantifier of standard symbolic logic is not represented, it being understood that all free variables are bound by the universal quantifier. This is all very well until sentences must be considered which involve both the universal and the existential quantifier. How is the existential quantifier to be represented and what account is taken of the order of quantifiers? In other words, how is the distinction maintained between "Everybody loves someone" and "Someone is loved by everybody"? One of many possible answers is, by means of Skolem functions. In terms of a semantic network, a Skolem function is nothing more than a specially labeled link from *a* node representing a variable governed by the existential quantifier to any variables governed by the universal quantifier that would have preceded it in the standard notation of the predicate calculus. So, for example, "Everybody loves somebody" might be represented as follows:

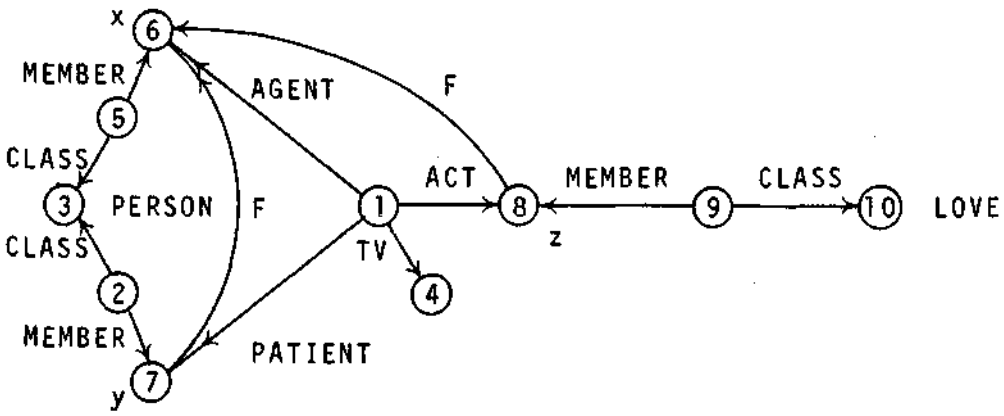


Figure 8

The principal proposition is represented by vertex 1 and x and y are variables governed by the universal and existential quantifiers respectively. The fact that there is an edge with the label "f" leaving vertex 7 shows that it is governed by the existential quantifier and that the vertex at the head of this edge corresponds to a preceding universal quantifier. Notice that vertex 8, representing the love that x has for y, is also represented as governed by the existential quantifier. Strictly speaking, the representation given for "John ran" should have been treated in the same manner because, according to the view I am taking here, almost every sentence describing an event involves such a quantifier. The sentence "Someone is loved by everybody" would be represented as follows:

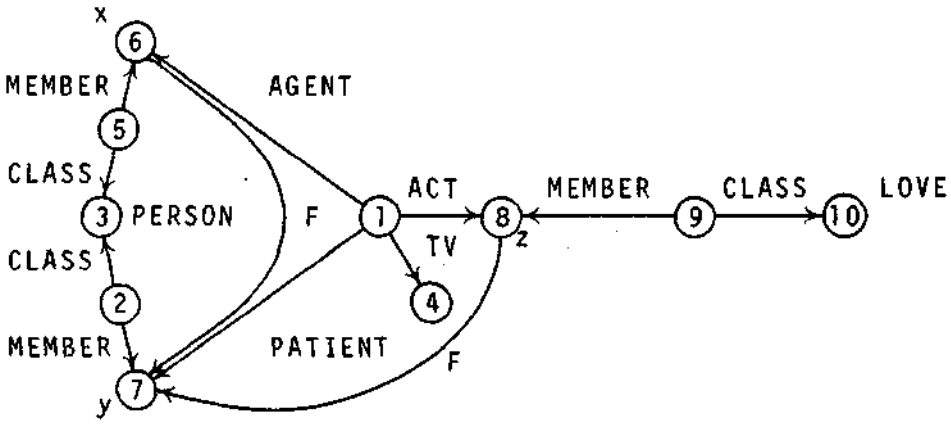


Figure 9

I make no particular claims for this method of expressing semantic relations; I simply want to illustrate the fact that a directed graph with labeled edges and vertices has a great amount of expressive power. In the final analysis, all that

I have done is to suggest a two-dimensional notation into which English sentences might be translated. The interest of a schema of this kind comes from the operations that can be performed upon it, and this is a matter about which we can claim to know only very little. A few things, at least, are clear. I have already mentioned the ability to make inferences. How this should be approached is still a matter of debate. An increasingly strong case is being built up against the view, which seemed promising until recently, that this part of a semantic processing should be treated as essentially the same as theorem proving in mathematics. A thoroughgoing attack on theorem proving, even if we did know how to conduct it, seems likely to result in *a* relatively poor model of how the mind works. One difference seems to be that mathematicians are typically concerned with constructing fairly long chains of argument on the basis of a small set of premises, whereas the everyday process of understanding usually involves short chains of inference based on a great number of premises. Furthermore, the logic of ordinary thought lacks the precision of mathematical argument. Experiments with so-called "fuzzy logic," in which propositions can be more or less true have been proposed to cope with this situation, but it is not clear that they do anything about the fundamental illogicality of everyday logic. The noun phrase "the set of all sets that are not members of themselves" has nothing anomalous about it in everyday English, but it must be regarded as meaningless in logic.

It also seems clear that language cannot be understood by a purely passive being. Almost all the mechanical language processors that have been built or projected have been intended to absorb anything they were told hungrily and uncritically. Weizenbaum's "Eliza" program was a notable exception and serves admirably to stress the importance of our expectations in interpreting what we are told. What we understand is, in large measure, what we expect to hear. Without any expectations, even from one sentence to the next, a machine has little hope of filling the gaps that ordinary language leaves

unfilled, or supplying the correct interpretation of ambiguous passages.

As I have said, the design of the MIND system contains no proposals on how these weighty questions are to be decided; it only provides a laboratory in which solutions can more easily be sought. The embryonic semantic component that now exists distinguishes the main semantic file from what is called "the discourse file."

THE OUTPUT COMPONENT

Relatively little needs to be said about the output component because, although it appears as a separate component to the user of the system, it is, in fact, the same program as is used for syntactic analysis. A separate paper in this volume argues the essential similarity of all kinds of syntactic processing, and the function of the output component is mainly syntactic. I have already explained how syntactic processes are carried out using a special data region called the "chart" which is also the only channel of communication among the various programs in the system. The chart provides a compact way of representing a large family of strings of labeled, oriented trees provided the members of the family typically have common substrings. The construction of well-formed sentences requires just such a data structure and the operations involved are similar to those required for analysis.

Just as in analysis, there are many formalisms and techniques that might be used in sentence generation. An obvious approach, and the one that has been most used in experiments with the MIND system so far, involves using a transformational component such as is proposed in standard transformational grammar. Transformational rules are essentially different from analysis rules because they are not, in general, reversible. In other words, it is not always possible to construct a sequence of transformational rules whose effect will be the inverse of

a given set. There are several reasons for this. One is that, whereas transformational rules each accept a single syntactic tree as input and deliver a single tree as output, syntactic analysis begins, not with a syntactic tree, but with a string of words. A syntactic tree is the end result of the process. This is not to say that, at a more fundamental level, the operations involved are not the same. A second reason is that the rules themselves are not sufficiently restricted for reversibility. They permit, for example, parts of a structure to be deleted without trace. It is true that injunctions against this practice have been proposed, but they have not been made precise and they are not generally assented to. In the MIND system what this means is that the same processor can be used, but a different compiler is required to translate rules from the external form in which the user writes them into the internal form required for processing.

A second important difference between transformational rules and those required for analysis concerns their ordering. Analysis rules — at least those that have so far been used in the MIND system — are either unordered or are self-ordering where, by self-ordering, I mean simply that it is one of the functions of *a* rule to determine what others will be tried after it. Standard transformational grammar, on the other hand, assumes that rules will be applied in a system of cycles which is the same for all grammars of this type, so that it does not need to be specified in the rules themselves. The syntactic component contains all the facilities required to insure that the rules are properly ordered; the way to invoke it is by means of a compiler that calls upon these facilities automatically as they are required. We have, for example, been working with a configuration of the MIND system in which the output component provides for a statement of the following form:

```
PERFORM n FOR S;
```

which causes the sequence of statements beginning with number n to be applied to each subtree of the

current tree whose root is labeled S. If this statement is itself numbered n, then the effect will be to cause it to be carried out recursively, thus producing the effect of the transformational cycle required in standard theory.

SUMMARY

The MIND system contains more facilities than I have described but there would be no point in continuing the recital. It is, in any case, essential in the system that facilities can come and go as new components are added and old ones replaced. My main aim in this talk has been to show something of what it takes to make computational linguistics more productive and to ease the performance of linguistic experiments involving the computer in important ways. It is not enough to furnish a battery of powerful subroutines for the linguist to incorporate into his programs. Linguistic data and linguistic formalisms are sufficiently complex to require the composition of new processes by the experimenter while seated at the console and a number of different languages and notational devices for different aspects of the problem. Furthermore, it must, to the extent possible, be aloof from sectarian differences among scientific schools. In this enterprise, the profit comes from providing a number of specialized languages in which to state linguistic facts and to decouple these, as far as possible from the programs that will carry the processes out. Experience with the MIND system has convinced its designers and those who have worked with it that there is a vantage point from which the contending linguistic theories show more similarities than differences but which is still close enough to reality to provide *a* useful basis for computer programming.