INFERENTIAL MEMORY AS THE BASIS OF MACHINES WHICH UNDERSTAND NATURAL LANGUAGE

by Robert K. Lindsay

Participants in the search for intelligent machines frequently disagree on a basic question of strategy in their quest. On the one hand there are those who believe that the major obstacles can be overcome by reliance on the computer's infallible memory, electronic speed, and arithmetic capabilities if these capacities are cleverly employed in sophisticated searching and statistical procedures. On the other hand there are those who feel that the problems of meaning and intuition must be somehow resolved before significant progress will be made, and that these problems are not solely a matter of speed and arithmetic. This issue will only be resolved by demonstration, and yet it is of some importance to decide how to allocate our efforts. This report takes the position that immediate, practical applications can derive from the former approach, but the major problems will be solved only by the latter. To mention a single example, the implementation of information retrieval techniques on present-day computers would be a large step forward, even though the techniques thus far considered have largely been conceptually trivial. The automation of libraries and scientific document files could immediately bring about great savings in valuable human time and effort, plus increased accuracy in literature searching. Kehl (1961) is now implementing a retrieval system which searches for certain combinations of key words in a large corpus and yields references to those documents which contain the proper combinations. Luhn (1958) has used a straightforward statistical procedure to extract key sentences from scientific articles, thus yielding useful abstracts of a sort. The use of these two techniques on a large scale would go a long way toward extracting us from the clutches of the information explosion which is so often discussed.

And yet it is quite clear that these techniques, whose primary advantages derive from using the great speed of the computer, will not produce intelligent machines, or even produce machines which do simple jobs with the intelligence displayed by a human clerk. For even an unintelligent human does more than count frequencies or search for key words. The human displays intelligent features which are generally summed up by saying that he *understands* the *meaning* of what he hears and reads.

The meaning of meaning and the meaning of understanding have never been adequately explicated when applied to human thought processes. How then can we hope to make them precise enough to enable us to build machines which understand meaning? Before attempting to answer this question, let us attempt to sharpen our intuitions by considering more specifically some examples of things which could be done by machines which understand but which would be beyond the capabilities of machines without this ability.

One of the major problems of the many encompassed by artificial intelligence is that of the mechanical translation of natural languages. Many of the early advocates of mechanical translation felt that high-quality translations could be produced by machines supplied with sufficiently detailed syntactic rules, a large dictionary, and sufficient speed to examine the context of ambiguous words for a few words in each direction. No doubt such machines will be able, when the syntactic rules are discovered, to produce fairly good translations, and yet it should be clear that such machines will never produce truly high-quality translations without the aid of pre-editing and postediting by human translators.

Here is one example of a difficulty. We wish to translate "The boy is in the house" and "The boy is in Paris" into French. In the first instance, the preposition "in" is rendered in French as "dans"; in the second sentence, the same preposition is rendered in French as "à." The human translator makes his decision by knowing that houses enclose people on all sides, while cities do not. This situation could, of course, be handled by marking all nouns with an indicator which tells the machine whether or not the thing denoted can enclose other things. But the hope that all such idiosyncrasies can be handled by such multiplication of stored details is futile. Bar-Hillel (1960) has given an even more perplexing example. We wish to translate the sentences "The pen is in the box" and "The box is in the pen" into French. There is no other context, and no other is needed by a human translator who knows that "pen" in the first sentence must denote a writing instrument and not a fenced enclosure, while the opposite is true in the second sentence. He can thus select the proper French equivalent for each, even though in French a single word does not suffice as it does in English. Once again we must increase the information stored in our

machine, this time indicating for each noun those things which it can enclose.

The problem of understanding may be rephrased to state that we must find ways of storing large amounts of such detailed knowledge while keeping the amount of memory capacity required within realizable limits.

Much of the literature on meaning has not been directly connected with this notion, but has been concerned with the problem of denotation: to what things does a symbol refer. Here, however, we are faced with the problem of what a *proposition* means. Osgood (1957) and Mowrer (1954) have attempted to extend notions of association and conditioning to include associations between groups of words rather than single words.

According to Osgood's theory, a word elicits associated internal responses. These responses can be described by their values along certain dimensions, such as active-inactive, good-bad, strong-weak. The meaning of a combination of words is an average of the component values for each of the words taken individually. For example, if "shy" is valued as mildly inactive, mildly bad, and mildly weak, and if "secretary" is valued as being very active, very good, and mildly weak, then "shy secretary" is valued as mildly active, mildly good, and mildly weak.

According to Mowrer's theory, sentences are temporal sequences of words and the internal responses to the first words are conditioned, in the classical sense, to the internal responses to the later words. Thus the sentence "Tom is a thief" conditions the notion of Tom to the notion of thief, where we use the loose term "notion" to indicate that it is not the words themselves, but the internal responses to them which become associated.

In both of these theories, the measure of meaning of a concatenation of words amounts to some sort of combination of the measures for the individual words. It appears that a useful theory must somehow make use of more complicated associative connections than those proposed by either of these two workers. For one thing, Osgood's scheme depends not at all on word order, only on which words are used; Mowrer's scheme depends only on word order, and not upon any other relations. To Osgood, "Tom hit Joe" would have the same meaning as "Joe hit Tom"; to Mowrer, "Tom is a thief" would have the same meaning as "Tom hit a thief."

Intuitively, a concept of meaning must include the notion of implication: what does a proposition imply. This does not mean, that is, imply, logical implication, but merely implication to the individual. Thus the meaning of a proposition is relative to the audience, and this probably is an unavoidable requirement.

Knowing more than one is told is a characteristic of human performance which is present in most behaviors which are called intelligent. We have

argued that this characteristic is necessary for machines which are to solve the real problems of information retrieval, language translation, and problem-solving. And furthermore, we must find efficient ways to store implications if we are to develop intelligent machines with finite memory capacities, that is, if we are to develop intelligent machines.

Examples of memory structures with these desired properties immediately come to mind from personal experience. They are often called mental pictures. Gelernter (1959*b*), for example, has developed a geometry machine whose basic source of intelligence lies in its ability to reject most of the formally possible sequences of proof steps because they "cannot possibly be correct." In effect, the machine constructs a diagram based upon the premises of the theorem to be proved. (Actually, the machine is supplied with such a diagram, although the task of constructing one is, while difficult, not taxing of memory and speed.) The implications of the premises are explicitly contained in this diagram, as are some nonimplications, but most nonimplications are not contained. The machine then merely rejects as possible subgoals (intermediate steps) all things which are not true in the diagram. For example, the premises "Triangle ABC," "AB = BC," "A, D, C collinear," and "AD = DC" are supplied in conjunction with coordinates for each point, such as A(0,0), B(5,5), C(10,0) and D(5,0) (see Fig. 1). The machine will never attempt to prove "triangle ABC congruent to triangle ABD" because this is not true in the diagram, as the machine can determine by calculating their respective perimeters. However, it might try to prove "angle DAB = angle DCB," which is implied by the premises. It may also try to prove "AD = DB" which is true in the diagram, but not implied by the premises. By supplying the machine with a more general diagram, such as by moving B to (5,15) (see Fig. 2), this last *cul de sac* could be avoided.

Such two-dimensional pictures have the properties we desire: they store implications and they do so in compact fashion. They also have a wide

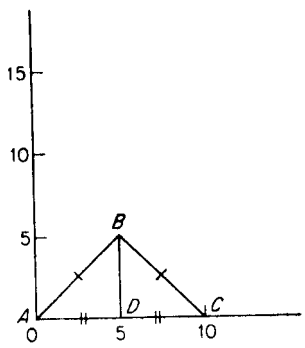

Figure 1.

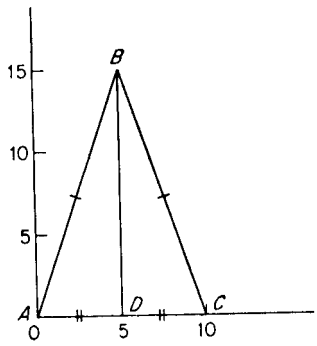

Figure 2.

range of application, few of which, aside from Gelernter's work, have been explored. One further example is provided by Venn diagrams, which are devices which aid logical reasoning. We may represent the proposition "all B are A" by two areas, one completely enclosed by the other, with appropriate labels. If we add a third area, C, according to the same rules to represent the proposition, "all C are B," the resulting diagram contains the implication that "all C are A," since area C must lie wholly within area A. Similar rules can be devised for other propositional forms, as elsewhere discussed by Lindsay (1961). Actually there are simpler schemes for such situations since not all of the properties of euclidian two-dimensional space are required, but, since such representations also handle many other situations, an intelligent machine would achieve some economy by employing such general-purpose representations wherever usable rather than devising special schemes for each case.

So far we have discussed only situations where few difficulties arise. Reasoning does not always obey the rules of logic and geometry, and we quickly encounter additional difficulties when we attempt to handle even simple situations. A program has been written to handle a different class of problems, and the difficulties will become clear as this program is described.

The program to be described parses sentences written in Basic English and makes inferences about kinship relations. To do this it constructs two types of complex structures in the computer memory, one corresponding to a sentence diagram of the sort produced by high-school students, the other corresponding to the familiar family tree. These are represented inside the computer by so-called *list structures*. A list structure is a form of associative memory, wherein each symbol is tagged by an indicator which tells the machine the location of a related symbol. So far this corresponds to the associative bonds which are the basic concept of stimulus-response psychology. However, each symbol may at times refer to a whole string of other, connected symbols, thus producing a hierarchical organization of memory associations. This feature provides much greater flexibility than either the single associations of stimulus-response psychology or the mediated associations which have recently been discussed and seem to be a first step in the direction of generalizing the limited stimulus-response schema.

The Sentence-parsing Program

The grammars of certain languages may be described by rules of replacement, which, if they satisfy certain conditions of simplicity, are called phrase structure rules (see Chomsky, 1957). For example, a simple grammar might consist of the following rules:

1. S $\leftrightarrow$ NP + Pred
2. S $\leftrightarrow$ NP + VP + NP
3. S $\leftrightarrow$ NP + V + NP
4. NP $\leftrightarrow$ They
5. NP $\leftrightarrow$ planes
6. NP $\leftrightarrow$ A + N
7. N $\leftrightarrow$ planes
8. N $\leftrightarrow$ man
9. A $\leftrightarrow$ the
10. A $\leftrightarrow$ flying
11. VP $\leftrightarrow$ Aux + V
12. V $\leftrightarrow$ are
13. V $\leftrightarrow$ flying
14. Aux $\leftrightarrow$ are
15. Pred $\leftrightarrow$ VP + NP + Ad
16. Ad $\leftrightarrow$ swiftly

These rules may be interpreted as, for example, the twelfth, "When V is encountered in a string of symbols, it may be replaced by 'are,'" or "when 'are' is encountered in a string of words it may be replaced by V." The former interpretation concerns the production of sentences, while the latter concerns the parsing of sentences. Thus we may produce a sentence by beginning with the symbol S and successively applying rules. For example: S $\rightarrow$ NP + VP + NP $\rightarrow$ NP + Aux + V + NP $\rightarrow$ They + Aux + V + NP $\rightarrow$ They + are + V + NP $\rightarrow$ They + are + flying + NP $\rightarrow$ They are flying planes.

Different sequences of rules produce different sentences. With the rules given above, certain sentences can be produced which are ungrammatical within English. For example, we could generate "The man are flying planes." A proper grammar (set of rules) for English would have to rule out such possibilities. This is generally accomplished both by defining rules more narrowly (assuring, for example, that subject agrees with verb) and by introducing certain metarules which specify which sequences of application are legitimate [for two methods of accomplishing this, see Chomsky (1957) and Pendergraft (1961)].

It is also clear that different sequences of rules may produce the same sentences. For example:[1] S $\rightarrow$ NP + V + NP $\rightarrow$ NP + V + A + N $\rightarrow$ They + V + A + N $\rightarrow$ They + are + A + N $\rightarrow$ They + are + flying + N $\rightarrow$ They are flying planes.

Consider now a straightforward parsing technique which might be applied to the sentence "They are flying planes swiftly," using the rules of our example. This sentence has a unique parsing which may be discovered as follows. Find each word or group of words which occurs in the sentence

[1] This example is due to Chomsky (1957).

on the right-hand side of one of the rules, and replace it by the symbol which appears on the left-hand side of the rule. Apply the same procedure to the resulting string of symbols. If a symbol or word appears on the right-hand side of more than one rule, form separate strings for each case. Continue until the sequence is reduced to the single symbol S, abandoning paths to which this procedure ceases to apply. Thus we have:

They are flying planes swiftly

1. NP + V + V + N + Ad
   can go no farther
2. NP + V + V + NP + Ad
   can go no farther
3. NP + V + A + N + Ad
   NP + V + NP + Ad
   S + Ad
   can go no farther
4. NP + V + A + NP + Ad
   can go no farther
5. NP + Aux + A + N + Ad
   NP + Aux + NP + Ad
   can go no farther
6. NP + Aux + A + NP + Ad
   can go no farther
7. NP + Aux + V + N + Ad
   NP + VP + N + Ad
   can go no farther
8. NP + Aux + V + NP + Ad
   NP + VP + NP + Ad

   8.1. S + Ad
   can go no farther

   8.2. NP + Pred
   S
   successful parsing

Even after a sufficient number of rules and metarules are supplied so as to eliminate ungrammatical sequences, there will remain, for natural languages, sentences which can be generated by two or more different production sequences. Conversely, any procedure which parses sentences should be able to discover all such sequences. The decision as to which parsing is correct depends upon a context larger than a single sentence and in many cases will also depend upon the meaning of the sentence, including a dependence upon what the various words denote.

However, even if we neglect the problem of selecting which legitimate parsing is correct in a given instance, the problem of discovering *any*

legitimate parsing is itself formidable when we deal with the tens of thousands of rules needed to describe a natural language. A complete set of rules is, in fact, so large that none has yet been devised for any natural language, although some have been under study for thousands of years. With even a moderately large number of rules, the parsing procedure described above will generate many possible branches, some of which may continue to be feasible for a long time. In order to discover any parsing in reasonable time and with reasonable effort, it is useful to employ some sort of selection procedure.

The procedure employed in the program to be described here is based upon two assumptions which are psychologically realistic. First, it is assumed that almost all sentences which will be encountered in actual text may be parsed by a procedure which proceeds from left to right, making decisions about the disposition of earlier phrases without considering the entire sentence. This reduces the number of rule combinations which must be searched. Secondly, it is assumed that a very limited amount of memory is available to remember intermediate results during the parsing of even extremely long sentences. This places severe restrictions upon which types of complexity will be analyzed and which types of syntactic structure will not be handled.

The final result of applying the parsing program to a sentence is an associatively organized memory whose structure reflects the interrelations among words, but does not give complete information as to which rules should be applied first to produce the sentence. The sentence diagram for our above example might be drawn as in Fig. 3. Each node in a sentence diagram corresponds to a substructure which has been constructed during the scanning of the sentence.

The sentence-parsing program is provided with a string of words as an input sentence. Each word may be found in a dictionary which indicates

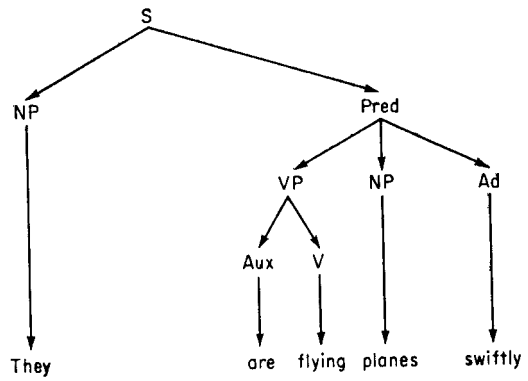

Figure 3.

a series of possible parts of speech which that word may serve as, the series being ordered so as to present the most frequent function first. The machine proceeds in a left-to-right fashion, assuming temporarily that the first word serves its most common function. To each part of speech there corresponds an associative structure which the machine forms. This structure is then temporarily held in memory and the next word is examined. If the structure so created requires the services of an additional word type, the machine continues to search for this type. If the next word serves the purpose, its structure is incorporated with that of the first word so that only a single structure name must be kept in rapid access memory, the remainder of the information being obtainable via the name. If the next word does not serve the desired purpose, its structure is stored separately, and the machine continues to look for words which will complete the structures of each of the words now held in memory. However, the number of structure names which can be held for rapid access is limited to a small total so that the machine must eventually begin to combine its substructures or else forget where it is. Frequently, the machine will be forced to complete a structure even though it has not found what it wants. This results in changing the part of speech designation for one or more words so that the entire sequence will now be compatible. The machine thus proceeds through the sentence, making temporary decisions, storing substructures in its limited rapid access memory, and revising its decisions only when forced to by lack of rapid access memory space or by complete incompatibility of substructures.

Loosely, the machine's behavior can be described as follows. The first word is "the." All right, now I need to find a nounlike word. The second word is "very," so now I need an adjective or adverb. The third word is "big," which is the adjective I needed, so combine these two words into the structure "very big." Now I need a nounlike word. The fourth word is "man," which is the noun needed. Now all words are combined into the structure "(the((very) big)) man." But now we have a subject, so look for a verb. The fifth word is "bit" which can act as the verb, so create the structure "((the((very)big))man) bit." Now another nounlike word or structure could serve as object. The sixth word is "the," so save it to modify a nounlike word; now we have two things saved, both looking for a nounlike word. The final word is "dog" which will serve both needed functions. We now have the complete sentence, whose structure is illustrated in Fig. 4.

In one sense this program is nothing but an algorithm which produces an output for every possible sequence of part-of-speech series inputs. However, if the program were written so as to merely check the input sequence and produce the corresponding output, a serious difficulty would arise. The size of the table required increases exponentially with sentence length,

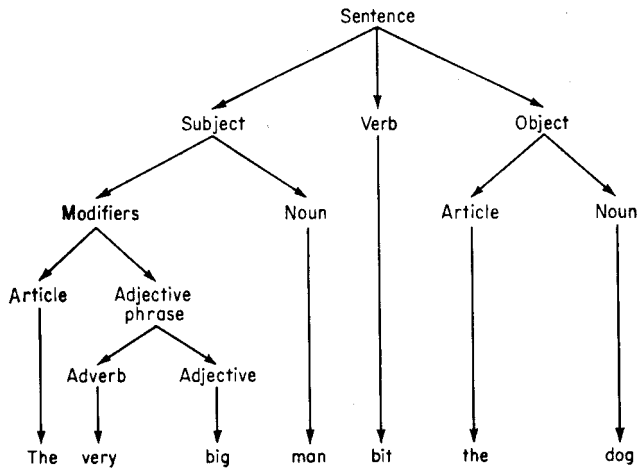

Figure 4.

thus an extremely large table for sentences of even moderate length would be needed. Although the program as it stands is limited in terms of the complexity of the sentences which it can handle, length alone does not contribute to complexity. For example, the program can handle a sentence such as "The big, black, ugly, ferocious, . . . , strong dog bit him," where the number of adjectives which may be inserted in the string is limited only by the memory capacity of the computer. This is possible because all of the adjectives are combined into a single substructure at every step, hence the rapid access memory is never exceeded. Further, the total memory requirements only increase linearly with sentence length. However, sentences which require the construction of an excessive number of substructures will cause the program to fail, even though these sentences are relatively short. Yngve (1961) has described a similar device for *producing* sentences, and argues that the limitations on complexity imposed by limits on rapid access memory capacity explain why certain constructions are not commonly used in natural English and hence are called ungrammatical.

The part-of-speech routines provide a finite set of processes which can handle an infinite number of sentences, in principle. They are superior to the table look-up method for the same reason that a program which computes e^x for any value of x is superior to a table of this function and a look-up program.

Obviously humans must employ some finite set of processes which are used to parse sentences, and obviously each word acts as a stimulus to elicit the corresponding processes. These processes, as in this program, are highly complex, and their decisions are contingent upon which other processes have been initiated before. It is quite consistent with our knowledge

of human thought processes to assume that the interaction takes place in the above-described manner, that is, through a small set of rapid access memory locations. However, the adult human undoubtedly has a larger set of processes, which effectively categorize words into more narrow categories than the few part-of-speech designations provided to our program, and these processes are no doubt much more complex.

Another psychologically tenable feature of this program is its left-to-right analysis. Although English grammar may conceivably be more readily analyzed in some other fashion, humans generally proceed from left to right, with only occasional reversals.

The limits of this program are not very well known. It will accept no words not included in Basic English, a system of grammar and a vocabulary of about 1700 English words which was defined by Ogden (1933). (Basic English is simplified English in the sense that anything which is good Basic is good English, but not vice versa.) The program will not accept certain punctuation marks, such as colons, and it does not distinguish between others, such as exclamation points and periods. Also, phrases and clauses in apposition must be indicated by dashes rather than commas. However, the program is not limited to single-clause sentences, nor must the input be a complete sentence. Thus, the program can handle many inputs which appear in actual writing but not in books on grammar. The program always makes a decision, and the result is always a complete structure containing all of the input.

The end result of the application of the program to a sentence is a structure which relates all the words of a sentence. This could be replaced logically by a set of descriptions listing all of these relations, but such a set would be far more elaborate and costly to memory. The syntactic meaning of the sentence is just this structure, wherein relations among words are implicit in its organization.

The Semantic-analysis Program

After the diagram of a sentence is constructed, the program attempts to deal with the meaning of some of the words. First, a list of all nouns is constructed. This list includes not only words which were used as subjects or objects, but also names used as possessive adjectives, such as "Bill's."

At this point, words cease to be considered solely by their syntactic-category membership. The sentence diagram is used as an information store which relates words. Subject-object combinations whose main verb is some form of "to be" are discovered. When such a combination is found, the words are marked "equivalent" by a cross-referencing scheme which indicates that both subject and object refer to the same thing or person. The modifiers of all equivalent nouns are then grouped together.

Next, a search is made for the eight words which Basic English provides to discuss kinship relations: "father," "mother," "brother," "sister," "offspring," "brother-in-law," "sister-in-law," and "married." If any of these relation words occurred in the sentence, their modifiers are examined to discover proper names which appeared as possessive adjectives or objects of a preposition, as would be the case if the original sentence contained phrases of the form "Jane's brother" or "the father of John." Each such proper name is paired with all others associated with the same occurrence of the relation word. By proceeding through the entire collection of words in this fashion, a list of elementary relations is formed. The items on this list are word triplets, two proper nouns, and a relation word which connects them.

Now the family tree is constructed. The computer memory is organized in an associative fashion again, with one computer storage location linked to others. The structure is isomorphic to diagrams such as given in the example below. Each "marriage" is represented by an association between the husband and wife, plus the name of a similar family unit for the parents of the husband, another for the parents of the wife, and the name of a list of offspring of this marriage. If the names of one of the partners, one of their parents, or some of their offspring are not given, places are reserved for these names should they occur in the future. The resulting tree is the same no matter whether the information was explicitly given in the text or merely implied.

By way of illustration, Fig. 5 depicts the memory structure for a simple family tree. The tree is composed of two basic family units, one formed by the marriage of A and B, and the other formed by the marriage of C and D. One of the offspring, E, of the first marriage is married to one of the offspring, F, of the second. It is evident that many relations are described by the tree given. However, it is important to note that the associations are one-way associations. This fact necessitates the addition of the name of the parent family unit at the end of each offspring unit. Thus, given A we may determine that E is one of his offspring by moving only in the direction of the arrows. Given E we may again trace the connection to A by moving only in the direction of the arrows, but this is true only because the family unit associated with E also contains the name of A's family unit. It follows that, given the fact that F (already located in the tree) and G (a name not previously encountered) are siblings, it is not sufficient to add G to the list of A-B offspring (dotted lines) but we must also copy the name of F's parental family unit into the newly constructed family unit of G (dashed lines).

The family tree, or trees, so constructed, are not erased after a single sentence is processed, but continue to grow as additional information is given throughout the passage.

The complexities and many small difficulties which are encountered in even this simple type of relation are indicative of the problem involved in the construction of semantic structures. More instructive, however, are the conceptual problems which arise in attempting to generalize this program to less strictly structured situations. Let us consider two of the most important problems.

It often happens, even when dealing with simple kinship relations, that the order of presentation of the input information has a crucial effect upon the efficiency of memory allocation. For example, if we are first told that X has offspring A, B, C, and D, we must construct an elaborate organization to handle this information, locations such as for the spouse of X being left blank. If we are then told that Y has offspring E, F, G, and H, we must construct another such structure, unrelated to the first. Finally, we may learn that B and H are brothers. This permits (neglecting such complications as multiple marriages) a collapsing of the two structures into a single organization which much more compactly represents the information

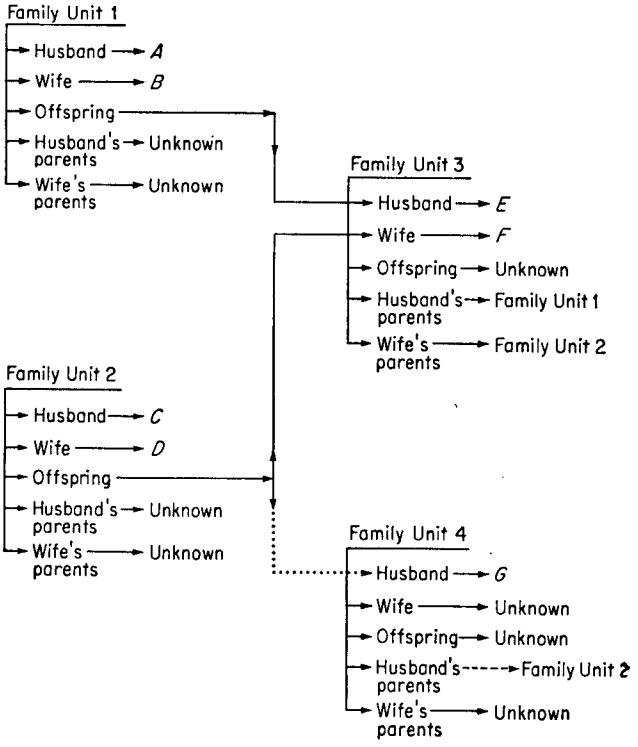

Figure 5.

implied. If we had been fortunate enough to have first learned of B's relation to H (or of X's relation to Y), we would have made much more efficient use of our memory capacity. In the program, the extra structures are "erased," that is, the memory used for them is returned to a common stockpile for use anywhere else it is needed. This is quite handily done with the easily altered computer memory, but a memory which is hard to erase, as the human memory presumably is, could be affected in important ways by such unhappy input sequences.

Nonetheless, an intelligent machine should have the property of being intelligent no matter what the order of its inputs. One aspect of the "aha" phenomenon is just that many formerly unrelated items of information are suddenly brought together by a single additional item, so that many implications suddenly leap out. Educators are beset by the problem of determining optimal orders of presentation of material, but, fortunately for the student, the human mind is capable of seeing connections under nonoptimal conditions.

An even more baffling problem is that of handling what has been called connotative meaning (Lindsay, 1961). Probably more often than not, a set of propositions which make some definite implications contains several subsets which alter the probabilities of other propositions without making any of them definite. Thus the statement that "George voted for Eisenhower and is opposed to medical care for the aged" makes it more likely that George is opposed to the United Nations, though only slightly so. It is quite clear that human cognitive organizations frequently take cognizance of these altered probabilities, perhaps to a greater extent than is reasonable. But how can such implicit connotations be intelligently and efficiently handled?

Let us consider a more concrete example arising in the context of the kinship-relation program. Consider the following sentence: "Joey was playing with his brother Bobby in their Aunt Jane's yard when their mother called them home." Certain definite information is given by this sentence, such as the fact that Joey and Bobby are brothers. Also, it is clear that Jane is either the sister or the sister-in-law of the children's mother, but it is not known definitely which is the case. If previous information has related, say, Joey to many others and Jane to many others, but has not related Joey's relations to Jane's relations, then the given sentence may imply a large number of things and remove the possibilities of a large number of other things. Still other possibilities depend upon knowing the exact relation between Joey's mother and Jane. The problem is to capture in the family-tree structure all of the definite implications, to eliminate all of the things definitely ruled out, indicate the altered probabilities of other relations, and still not make any *definite* assertions about the relation between Joey's mother and Jane.

The structure thus far described is unable to handle even this simple case, since the associations are either there or they are not, and only one connection is permissible. One solution to this problem is to construct several family-tree structures, one for each possibility. This corresponds, for example, to the situation in which a student will draw diagrams of an acute triangle, a right triangle, and an obtuse triangle corresponding to the possible cases for which he wishes to prove a theorem. This solution, however, will work well only when the number of alternatives is small and when the structures are themselves simple. In more complex situations this procedure is too taxing of memory capacity. It is desirable to include the uncertainty within a single structure.

In order to do this, we must allow multiple connections. Thus, in place of every association in our original format we must substitute a list of all the possibilities, and the process which retrieves information must recognize that only one of these can be correct. When nothing at all has been implied, the lists of possibilities will contain an "all" symbol indicating that all things are possible; when something definite is implied later, this "all" symbol is replaced by the proper connection; when several things have been implied, the universal symbol is replaced by a list of the remaining possibilities. We may also need to record a list of connections which are definitely impossible. When nothing has been implied, this list will contain a "none" symbol indicating that no things are impossible.

It is to be noted that a probabilistic connection of the sort frequently hypothesized by psychologists is not appropriate here. That is, we do not want a connection such that a given stimulus will sometimes evoke one association, sometimes another on a probability basis. In the above example, the reader knows *definitely* that either Jane is the sister of Joey's mother *or* is the sister of Joey's father, but not both; no reader would conclude half the time that Jane is the sister of Joey's mother and half the time that she is the sister of Joey's father, altering his decision from time to time.

But we are still faced with two problems. First, it is impossible, or at least impractical, to retain an extremely large number of possibilities; second, it is not clear how we should indicate that some possibilities are more probable than others. The first problem is perhaps solved by humans by not remembering all possibilities; thus humans are unable to remember all possible implications when the set of such possibilities is large. This will no doubt remain a problem for machines as well. We might solve the second problem by ordering the list of probabilities, placing the most likely alternatives first; or perhaps we might decide to associate probabilities with each alternative. In any event, the probabilities so established will determine the weight which is assigned to implications, but will *not* determine the implications which will hold to the exclusion of others. Finally,

we can imagine a situation in which the list of possibilities is truncated due to the limited computing capabilities of man or machine, and where subsequently all of the possibilities which remain are eliminated by further information. In this case, the machine, after all, will have to indicate that something is wrong and review previous inputs, this time reselecting possibilities in the light of knowledge of information to follow.

To complete our example, we may present the modified storage format (Fig. 6) which could be used to solve our sample problem.

Finally, we note that we have solved the problem of connotative meaning while retaining our basic device of storing *definite* implications implicitly, but we have resorted to storing *possible* implications explicitly. Techniques for avoiding this listing of possibilities would prove extremely valuable, since as we have seen, requirements on memory capacity increase rapidly when storage is explicit.

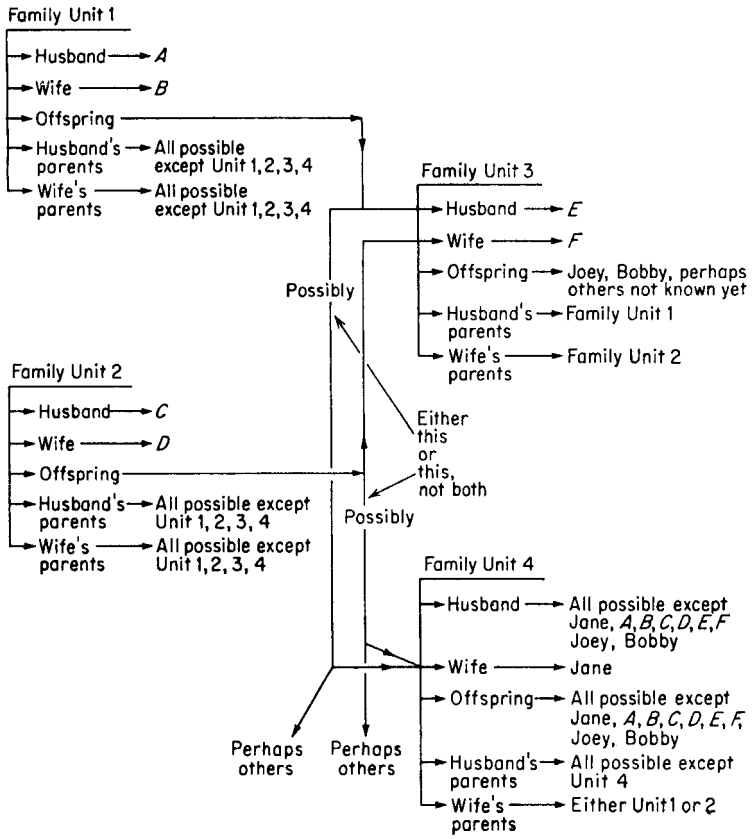

Figure 6.

Summary

It has been argued that the problem of meaning is of major importance in the study of the nature of intelligence, and that a useful definition of meaning must include not only denotation but connotation and implication as well. To handle these important questions it is necessary to study cognitive organizations which are more complex than those upon which most psychological theories are based. A central question is the storage of large numbers of interrelated propositions in a manner which efficiently uses memory capacity. Illustration of these points was given by reference to a computer program which stores syntactic relations and extracts and stores semantic implications of a very limited character. The illustrations put into concrete terms some of the problems which must be resolved before machines of formidable intelligence can be constructed.