A Procedural Model of Language Understanding

Terry Winograd
Massachusetts Institute of Technology

Much of the research on language is based on an attempt to separate it into distinct components—components that can then be studied independently. Modern syntactic theoreticians have been tremendously successful at setting up complex rules which describe in detail the possible orderings of syntactic constituents; at the same time other researchers are trying to define semantic relations and to model the cognitive structures underlying language use.

Most attempts to model language understanding on the computer have followed this strategy of dealing with a single component of language. They are constructed primarily as a syntactic program (Kuno, 1965), a model of semantic connections (Schank, 1971), or an attempt to model the memory structures (Quillian, 1967).

Question-answering systems have had to deal with the entire language process, but they have been severely limited in the breadth of their language ability. The only attempt to handle large portions of language data was the machine translation effort, and it soon became obvious that the methods were not up to the requirements of the task. Language translation could not be treated as a problem of rearranging syntactic structures and words, because attention to meaning was required even to achieve moderately acceptable results.

One basic limitation of those programs that have tried to handle the problems of meaning is that they have dealt almost exclusively with the understanding of single sentences, when in fact almost no human use of language takes place in such an artificial setting. We are always in a context, and in that context we make use of what has gone on to help interpret what is coming. Much of the structure of language comes from its being a process of communication between an intelligent speaker and hearer, occurring in a setting. The setting includes not only a physical situation and a topic of discourse, but also the knowledge each participant has about the world and the other's ideas.

This paper describes an attempt to explore the interconnections between the different types of knowledge required for language understanding. It is based on a computer program that "understands" language in a limited domain by including a model of the subject being talked about and a context of discourse. As an example of the interactions between the different sorts of knowledge the system must have, let us look first at the use of pronouns.

Our syntactic knowledge of English enables us to know that in the sentence "Arthur wants to see him," the word "him" must refer to someone other than Arthur (otherwise we would have used "himself"). In "Arthur wants somebody to see him," "him" might or might not refer to Arthur. The distribution of reflexive pronouns like "himself" depends in a complex way on the syntactic structure of the sentences in which they appear, and a language understander must have this knowledge. As a semantic fact, we know that "him" must refer to something which the speaker is characterizing as animate and male.

At another level, we know that the referent is likely to occur in the preceding sentence, or earlier in the sentence being interpreted, that it is more likely to refer to the topic or the subject of the previous sentence, and is much more likely to refer to a major constituent than to one deeply embedded in the structure. This type of heuristic knowledge about the organization of discourse also plays a part in our understanding.

Finally, there is a level based on knowledge of the world. In the sentence "Sam and Bill wanted to take the girls to the movies, but they didn't have any money," we understand "they" as referring to Sam and Bill. This doesn't involve syntactic or general semantic knowledge, but depends on our knowledge of our social culture. When someone takes someone else to the movies, it is the inviter who pays, and it is his or her financial situation that is relevant.

Whenever we look into realistic language use, these types of interaction play a large role, not only with pronouns, but in deciding on the structures of sentences and meanings of individual words as well. We assign different structures to sentences like "He gave the house plants to charity," and "He gave the boy plants to water," on the basis of our syntactic and semantic knowledge. Even the most common words have multiple meanings, and we must bring a variety of facts to bear in deciding, for example, the meaning of "had" in "Mary had a little lamb, but I preferred the baked lobster."

In discourse, people take advantage of a variety of mechanisms that depend on the existence of an intelligent hearer who will use all sorts of knowledge to fill in any necessary information.

In making a computer model of language use, this presents a serious problem. On the one hand, it is impossible to isolate one aspect of language from the others, or to separate a person's use of linguistic knowledge from his use of other knowledge. On the other hand, it is clearly folly at this point to think of giving the program all the knowledge a person brings into a conversation. In our program, we choose to resolve the dilemma by picking a tiny bit of the world to talk about. Within this mini-world, we can give the computer a deep kind of knowledge, including the equivalent of "Who would pay for a movie?"

The subject chosen was the world of a toy robot with a simple arm. It can manipulate toy blocks on a table containing simple objects like a box. In the course of a dialogue, it can be asked to manipulate the objects, doing such things as building stacks and putting things into the box. It can be questioned about the current configurations of blocks on the table, about the events that have gone on during the discussion, and to a limited extent about its reasoning. It can be told simple facts which are added to its store of knowledge for use in later reasoning. The conversation goes on within a dynamic framework—one in which the computer is an active participant, doing things to change his toy world, and discussing them.

The program was written in LISP on the PDP-10 ITS time-sharing system of the Artificial Intelligence Laboratory at MIT.* It displays a simulated robot world on a television screen and converses with a human on a teletype. It was not written for any particular use with a real robot and does not have a model of language based on peculiarities of the robot environment. Rather, it is precisely by limiting the subject matter to such a small area that we can address the general issues of how language is used in a framework of physical objects, events, and a continuing discourse.

The programs can be roughly divided into the three domains mentioned above: There is a syntactic parser which works with a large-scale grammar of English; there is a collection of semantic routines that embody the kind of knowledge needed to interpret the meanings of words and structures; and there is a cognitive deductive system for exploring the consequences of facts, making plans to carry out commands and finding the answers to questions. There is also a comparatively simple set of programs for generating appropriate English responses.

In designing these pieces, the main emphasis was on the interaction of the three domains. The form in which we want to state a syntactic theory or a type of deduction must take into account the fact that it is only a part of a larger system. One of the most useful organizing principles was the representation of much of the knowledge as procedures. Many other theories of language state their rules in a form modelled on the equations of mathematics or the rules of symbolic logic. These are static rules that do not explicitly describe the process involved in using them, but are instead manipulated by some sort of uniform deduction procedure. By writing special languages suited to the various types of knowledge (semantic, syntactic, deductive), we are able to preserve the simplicity of these systems. This is accomplished by putting the knowledge in the form of programs in which we can explicitly express the connections between the different parts of the system's knowledge, thus enriching their possibilities for interaction.

The rest of this chapter illustrates the capabilities of the system and explains in some detail how it works. The second section presents a sample of a dialogue with the program, along with comments describing some of the features being demonstrated. The third section gives a closer view of the program itself, and the final section discusses some of its basic limitations.

*For more details on the entire system, see Winograd (1972). The actual program is written in LISP, occupies about 100K of storage on the PDP-10, and operates in approximately real time, responding to a sentence in an average of 10 or 15 seconds.

Sample Dialogue

This section contains a sample dialogue with the language-understanding system. This system carries out actions and answers questions about a simple scene containing a table, hand, and box, and several blocks and pyramids. The numbered lines in lower case were typed in by a person, and the lines in upper case are the computer's response. Other lines are comments describing some of the features being exhibited.

The dialogue was carried out in real time with the response of the "robot" displayed on a television screen. Figures are included to show the state of the scene at various points.

1. *pick up a big red block.*

```
OK.
```

The system answers "OK" when it carries out a command. In order to pick up the red block, it had to clear it off by finding a space for the green one and moving the green one away. The initial situation is shown in Figure 4.1, and the result in Figure 4.2.

2. *grasp the pyramid.*

```
I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.
```

There are three pyramids in the scene, and the program knows that phrases beginning with "*the*" are intended to refer to a specific object the speaker has in mind.

3. *find a block which is taller than the one you are holding and put it into the box.*

The system has to figure out what is meant by "*one*" and "it," as well as using deduction to decide which block it is holding.

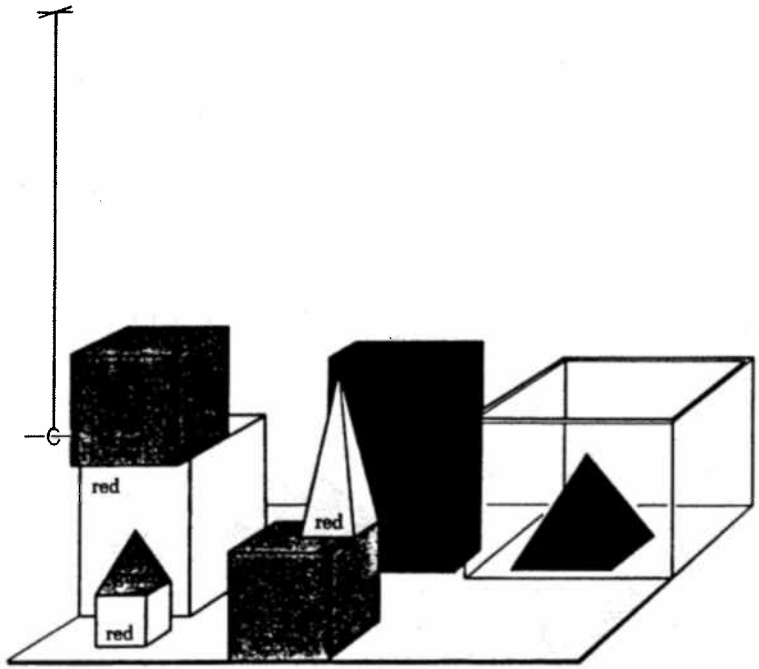

Figure 4.1 The Physical Environment of the Robot.

```
BY "IT," I ASSUME YOU MEAN THE BLOCK WHICH IS TALLER THAN
THE ONE I AM HOLDING.
```

There were two possible referents for "*it*": the one selected, and "*the one you are holding.*" The program has heuristics for choosing between them.

```
OK.
```

The hand has to set down the red block to pick up the taller blue one. Figure 4.3 shows the result of the system's manipulations so far.

4. *what does the box contain?*

```
THE BLUE PYRAMID AND THE BLUE BLOCK.
```

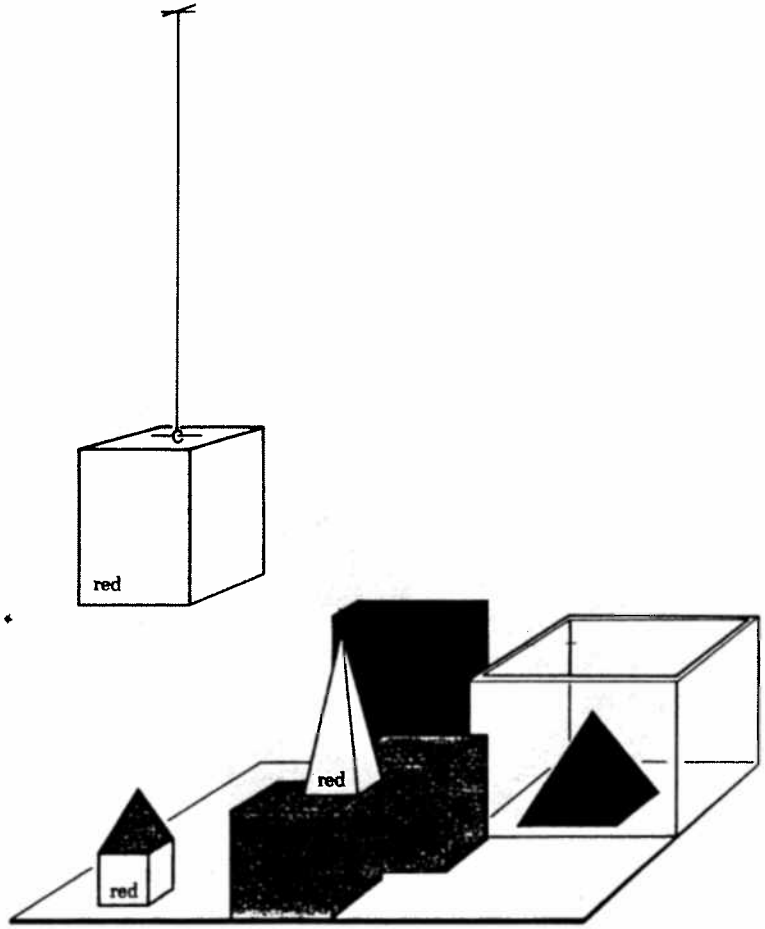

Figure 4.2 "Pick up a big red block."

5. *what is the pyramid supported by?*

Earlier the system responded to "*the pyramid*" by saying "I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN." Now, it assumes that we mean the one just mentioned.

```
THE BOX.
```

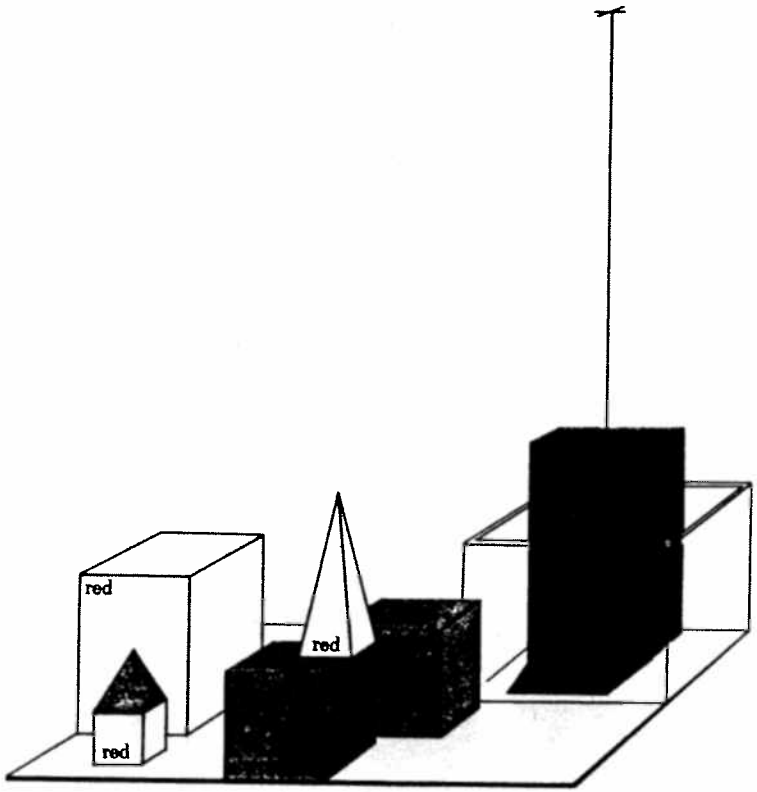

Figure 4.3 "Find a block which is taller than the one you are holding and put it into the box."

6. *how many blocks are not in the box?*

Negatives such as "not" and "no" are handled whether they appear in clauses or noun groups (such as "no boxes").

FOUR OF THEM.

7. *is at least one of them narrower then the one which i told you to pick up?*

The phrase "*at least one of them*" must be analyzed to refer back to the previous question, and "*the one which . . .*" must also refer to blocks. Noun groups can have complicated numbers such as "*at least one*," "*more than three*," "*exactly seven*," etc. The memory of past events is used to deduce which block is "*the one which I told you to pick up.*" Notice the complicated syntactic structure in which "*the one*" is actually the object of a clause embedded two levels deeper. The parser incorporates a grammar that handles this kind of complexity in a general way.

YES, THE RED CUBE.

8. *is it supported?*

YES, BY THE TABLE.

The system has heuristics about what information is relevant to a person who asks a question, even if he does not request that information specifically.

9. *can the table pick up blocks?*

NO.

The semantic rules include the knowledge that "*picking up*" must be done by an animate object, and that the table is not animate.

10. *can a pyramid be supported by a block?*

YES.

The deductive system finds an actual example, so it knows this is possible.

11. *can a pyramid support a pyramid?*

I DON'T KNOW.

The question makes sense, but there is no example in the scene. The system has no way of looking into the details of its programs, even though these ultimately define its capabilities.

12. *stack up two pyramids.*

I CAN'T.

The robot tries and fails.

13. the blue pyramid is mine.

```
I UNDERSTAND.
```

All of the information about colors, shapes, sizes and locations of the objects is known to the program initially. Possession is left as an unknown property of objects about which it can be told. It stores this information in its data base.

14. i own all blocks which are not red, but i don't own anything which supports a pyramid.

```
I UNDERSTAND.
```

The result of understanding these sentences is the creation of new procedures within the robot for deducing whether I own something.

15. do i own the box?

```
NO.
```

The box supports whatever it contains, including a pyramid.

16. do i own anything in the box?

```
YES, TWO THINGS: THE BLUE BLOCK AND THE BLUE PYRAMID.
```

17. will you please stack up both of the red blocks and either a green cube or a pyramid?

```
OK.
```

Logical connectives, such as "and," "or," "either," etc., are handled in both the grammar and semantics. A command is executed even if it looks like a question. The command is carried out by the following series of steps: It puts a green cube on the large red block (it chooses the green cube with nothing on it), then removes the small pyramid from the little red cube, so that it can use that cube to complete the stack as specified, resulting in what is described in Figure 4.4.

18. which cube is sitting on the table?

```
THE LARGE GREEN ONE WHICH SUPPORTS THE RED PYRAMID.
```

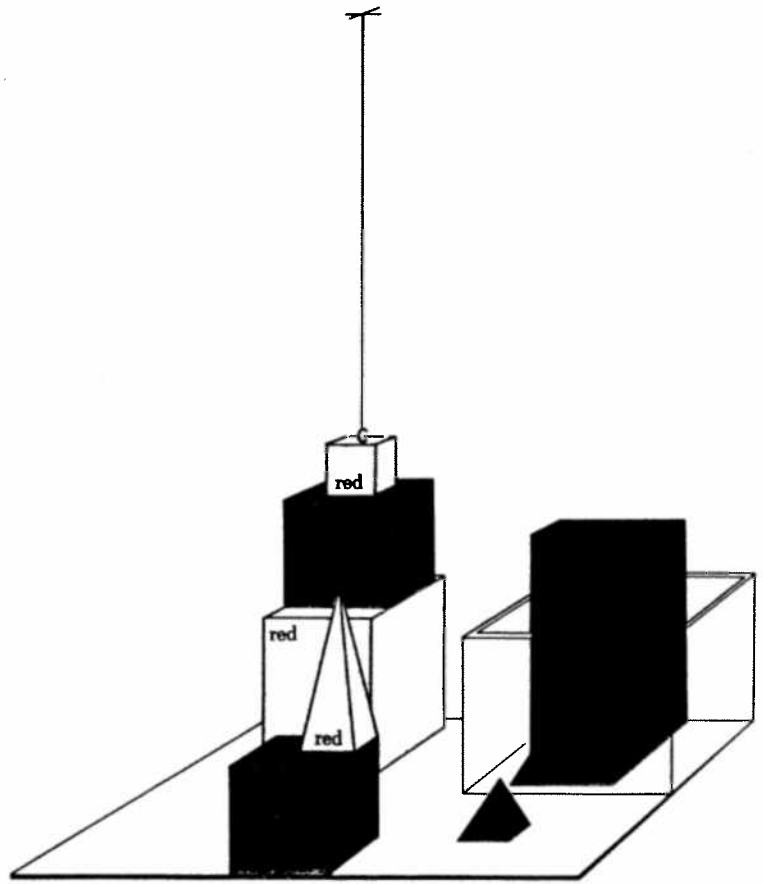

Figure 4.4 "Will you please stack up both of the red blocks and either a green cube or a pyramid."

Ordinarily the system would be satisfied with calling this "*a big green cube*," but since we asked "*which cube*" it has to be more specific in naming it. It has procedures for using substitute nouns (such as "one") and pronouns in answers to make the discourse more natural.

19. is there a large block behind a pyramid?

```
YES, THREE OF THEM: A LARGE RED ONE, A LARGE GREEN CUBE AND
THE BLUE ONE.
```

20. put a small one onto the green cube which supports a pyramid.

"*A small one*" must contrast with the person's earlier phrase "*a large block,*" so this means "*a small block.*"

```
OK.
```

21. put the littlest pyramid on top of it.

Words like "*littlest*" are not in the dictionary but are interpreted from the root forms like "little." The program compares the sizes of all the pyramids to carry out the action.

```
OK.
```

22. how many things are on top of green cubes?

```
I'M NOT SURE WHAT YOU MEAN BY "ON TOP OF" IN THE PHRASE
"ON TOP OF GREEN CUBES"
DO YOU MEAN:
1 - DIRECTLY ON THE SURFACE
2 - ANWHERE ON TOP OF?
```

In the previous sentence, the phrase "*on top of*" was interpreted without trouble because the context made it clear what was meant. In this one, the heuristics for disambiguation do not come up with a clearcut decision, so the system has to ask the user for more information. The paraphrases were included with the dictionary definition of "on." A request like this can also occur when the meaning of a pronoun such as "it" or "they" is unclear.

23. had you touched any pyramid before you put the green one on the little cube?

```
YES, THE GREEN ONE.
```

The system deduces from its memory what event is being referred to, and can relate other events to it in time. The analysis includes complex tenses of verb groups, and there are special facilities for keeping track of the various forms of irregular verbs like "have."

24. when did you pick it up?

```
WHILE I WAS STACKING UP THE RED CUBE, A LARGE RED BLOCK
AND A LARGE GREEN CUBE.
```

A time is described by naming the main goal which was being worked on.

25. why?

The system remembers immediately previous sentences to understand questions like this and the following ones.

```
TO GET RID OF IT.
```

By keeping track of selected parts of the original subgoal tree, the system has some understanding of its own motives.

26. why did you do that?

```
TO CLEAN OFF THE RED CUBE.
```

"*That*" refers to the action mentioned in the system's last answer.

27. why did you clear off that cube?

Here, "*that*" means the one just referred to.

```
TO PUT IT ON A LARGE GREEN CUBE.
```

28. why did you do that?

```
TO STACK UP THE RED CUBE, A LARGE RED BLOCK AND A LARGE GREEN
CUBE.
```

29. why did you do that?

```
BECAUSE YOU ASKED ME TO.
```

It remembers what it did, not how the request was worded.

30. how did you do it?

"It" refers to the action mentioned in the last question, rather than the answer.

```
BY PUTTING A LARGE RED BLOCK ON THE TABLE; THEN LETTING GO
OF IT; THEN PUTTING A LARGE GREEN CUBE ON IT; THEN LETTING GO
OF THAT CUBE; THEN PUTTING THE RED CUBE ON THAT CUBE; THEN
LETTING GO OF THAT CUBE.
```

This is a description of the top-level subgoals used to achieve the goal.

31. how many objects did you touch while you were doing it?

Time references can refer to a time span, using words like "while."

```
FOUR OF THEM.
```

32. what did the red cube support before you started to clean it off?

```
THE GREEN PYRAMID.
```

By remembering what has been moved, it is possible to deduce things about earlier states of the scene. We can use complex nestings of clauses like "*before you started to clean it off.*"

33. there were five blocks to the left of the box then.

"*Then*" must refer to the time mentioned in the last sentence.

```
NO, ONLY FOUR OF THEM: THE RED CUBE, TWO LARGE GREEN CUBES
AND A LARGE RED BLOCK.
```

Earlier in the dialogue, information about "owning" was accepted at face value. But the system knows that it has complete information about the basic properties of objects, including location. It checks any statements about these to see whether it agrees or disagrees.

34. put the blue pyramid on the block in the box.

This is syntactically ambiguous, but is understood without ambiguity since the parsing corresponding to "*put the blue pyramid which is on the block into the box*" does not make sense in the current scene, whereas "put *the blue pyramid onto the block which is in the box*" does.

```
OK.
```

35. is there anything which is bigger than every pyramid but is not as wide as the thing that supports it?

The system understands quantifiers such as "every," "any," "some," etc., and structures such as "bigger than", "as wide as," etc. Here "it" refers to a noun group within the same sentence but two levels up in the parsing tree.

```
YES, THE BLUE BLOCK.
```

36. does a steeple

```
SORRY, I DON'T KNOW THE WORD "STEEPLE".
```

The system interrupts immediately when it sees a word it doesn't know.

37. a "steeple" is a stack which contains two green cubes and a pyramid.

The quote marks indicate that we are defining a new word. The definition is added to the dictionary in a form compatible with all the words it already knows.

```
I UNDERSTAND.
```

38. are there any steeples now?

```
NO.
```

The new definition is used as a description to recognize steeples.

39. build one.

```
OK.
```

The description given in the definition is enough to deduce a plan for building new objects as well as recognizing them. The word *"one"* means *"a steeple,"* and the plan is carried out. First, one of the large green cubes is cleared off by removing the red cube and two pyramids. Then the two green cubes and the tall red pyramid are stacked up to fit the description.

40. call the biggest block "superblock."

OK.

We can also give objects names.

41. have you picked up superblock since we began?

YES.

42. why did you drop it?

BY "IT", I ASSUME YOU MEAN THE SUPERBLOCK.
I CAN'T EXPLAIN A NON-EXISTENT EVENT.

There is no record in memory of any block being dropped. The answer routines have special checks for this type of "loaded question" which presupposes things which aren't true.

43. is there anything to the right of the red pyramid?

YES, FOUR THINGS: THE BOX, THE RED CUBE, THE BLUE PYRAMID
AND THE BLOCK NAMED SUPERBLOCK.

Once an object has a name, the name is used to help identify it in responses.

44. thank you.

YOU'RE WELCOME!

The grammar is written as a procedure, which gives it the capability to carry out special sorts of parsing for idioms.

Details of the Model

In this section we go into some detail about the various parts of the language understanding program and its operation. This type of analysis is important for understanding any computer model, since the surface appearance of its output can be deceptive. A program may succeed by carefully choosing the problem it will attack, so that some simple special-purpose solution will work. ELIZA (Weizenbaum, 1964) and STUDENT (Bobrow, 1967) are examples of programs which give impressive performances owing to a severe and careful restriction of the kind of understanding they try to achieve. If a model is to be of broader significance, it must be designed to cover a large range of the things we mean when we talk of understanding. The principles should derive from an attempt to deal with the basic cognitive structures.

On the other hand, it is possible to devise abstract ideas of the logical structure of language—ideas which seem in theory to be applicable. Often, such systems, although interesting mathematically, are not valid as psychological models of human language, since they have not concerned themselves with the operational problems of a mental procedure. They often include types of representation and processes which are highly implausible, and which may be totally inapplicable in complex situations because their very nature implies astronomically large amounts of processing for certain kinds of computations. Transformational grammar and resolution theorem proving (Green, 1969) are examples of such approaches.

The Representation of Meaning

Our program makes use of a detailed world model, describing both the current state of the blocks world environment and its knowledge of procedures for changing that state and making deductions about it. This model is not in spatial or analog terms, but is a symbolic description, abstracting those aspects of the world which are relevant to the operations used in working with it and discussing it. First there is a data base of simple facts like those shown in Box 4.1, describing what is true at any particular time. There we see, for example, that B1 is a block, B1 is red, B2 supports B3, blue is a color, EVENT27 caused EVENT29, etc. The notation simply involves indicating relationships between objects by listing the name of the relation (such as IS or SUPPORT) followed by the things being related.* These include both concepts (like BLOCK or BLUE) and proper names of individual objects and events (indicated

*The fact that B1 is a block could be represented in more usual predicate notation as (BLOCK B1). We have chosen to associate with each object or concept a property describing its most relevant category for the purpose of generating an English phrase for it. Thus (IS B1 BLOCK) is used to describe B1 as a block. Similarly, properties like colors are represented (COLOR B1 BLUE) instead of (BLUE B1). This allows for more efficiency in the operation of the deduction system, without changing its logical characteristics.

Box 4.1 Typical Data Expressions.

```
(IS B1 BLOCK)
(IS B2 PYRAMID)
(AT B1 (LOCATION 100 100 0))
(SUPPORT B1 B2)
(CLEARTOP B2)
(MANIPULABLE B1)
(CONTAIN BOX1 B4)
(COLOR-OF B1 RED)
(SHAPE-OF B2 POINTED)
(IS BLUE COLOR)
(CAUSE EVENT27 EVENT29)
```

with numbers, like B1 and TABLE2).† The symbols used in these expressions represent the concepts (or conceptual categories) that form the vocabulary of the language user's cognitive model. A concept corresponds vaguely to what we might call a single meaning of a word, but the connection is more complex. Underlying the organization is a belief that meanings cannot be reduced to any set of pure "elements" or components from which everything else is built. Rather, a person categorizes his experience along lines which are relevant to the thought processes he will use, and his categorization is generally neither consistent, nor parsimonious, nor complete. A person may categorize a set of objects in his experience into, for example "chair," "stool," "bench," etc. If pushed, he cannot give an exact definition for any of these, and in naming some objects he will not be certain how to make the choice between them. This is even clearer if we consider words like "truth," "virtue," and "democracy." The meaning of any concept depends on its interconnection with all of the other concepts in the model.

Most formal approaches to language have avoided this characterization of meaning even though it seems close to our intuitions about how language is used. This is because the usual techniques of logic and mathematics are not easily applicable to such "holistic" models. With such a complex notion of "concept," we are unable to prove anything about meaning in the usual mathematical notion of proof. One important aspect of computational approaches to modelling cognitive processes is their ability to deal with this sort of formalism. Rather than trying to prove things about meaning we can design procedures which can operate with the model and simulate the processes involved in human use of meaning. The justification for the formalism is the degree to which succeeds in providing a model of understanding.

What is important then, is the part of the system's knowledge which involves the interconnections between the concepts. In our model, these are in the form of procedures written in the PLANNER language (Hewitt, 1971). For example, the concept CLEARTOP (which might be expressed in English by a phrase like "clear off") can be described by the procedure diagrammed in Figure 4.5. The model tells us that to clear off an object X, we start by checking to see whether X supports an object Y. If so, we GET-RID-OF Y, and go check again. When X does not support any object, we can assert that it is CLEARTOP. In this operational definition, we call on other concepts like GET-RID-OF and SUPPORT. Each of these in turn is a procedure, involving other concepts like PICKUP and GRASP. This representation is oriented to a model of deduction in which we try to satisfy some goal by setting up successive subgoals, which must be achieved in order to eventually satisfy the main goal. Looking at the flow chart for GRASP in Figure 4.6, we can see the steps the program would take if asked to grasp an object B1 while holding a different object B2. It would be called by setting up a goal of the form (GRASP B1), so when the GRASP program ran, X would represent the object B1. First it checks to see whether B1 is a manipulable object, since if not the effort must fail. Next it sees if it is already grasping B1, since this would satisfy the goal immediately. Then, it checks to see if it is holding an object other than B1, and if so tries to GET-RID-OF it. The program for GET-RID-OF tries to put the designated object on the table by calling a program for PUTON, which in turn looks for an empty location and calls PUT. PUT deduces where the hand must be moved and calls MOVEHAND. If we look at the set of currently active goals at this point, we get the stack in Box 4.2.

Notice that this subgoal structure provides the basis for asking "why" questions, as in sentences 25 through 29 of the dialog in Section 2. If asked *"Why did you put B2 on the table?,"* the program would look to the goal that called PUTON, and say *"To get rid of it."* If asked *"Why did you get rid of it?"* it would go up one more step to get *"To grasp B1."* (Actually, it would generate an English phrase describing the object B1 in terms of its shape, size, and color.) "How" questions are answered by looking at the set of subgoals called directly in achieving a goal, and generating descriptions of the actions involved.

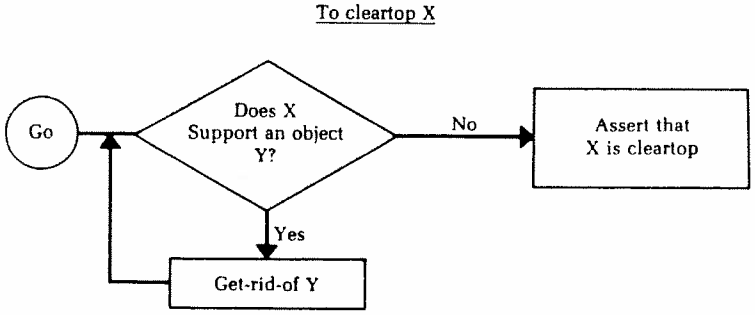

Figure 4.5 Procedural Description for the Concept CLEARTOP.

†The notation does not correspond exactly to that in the original program, as mnemonics have been used here to increase readability.

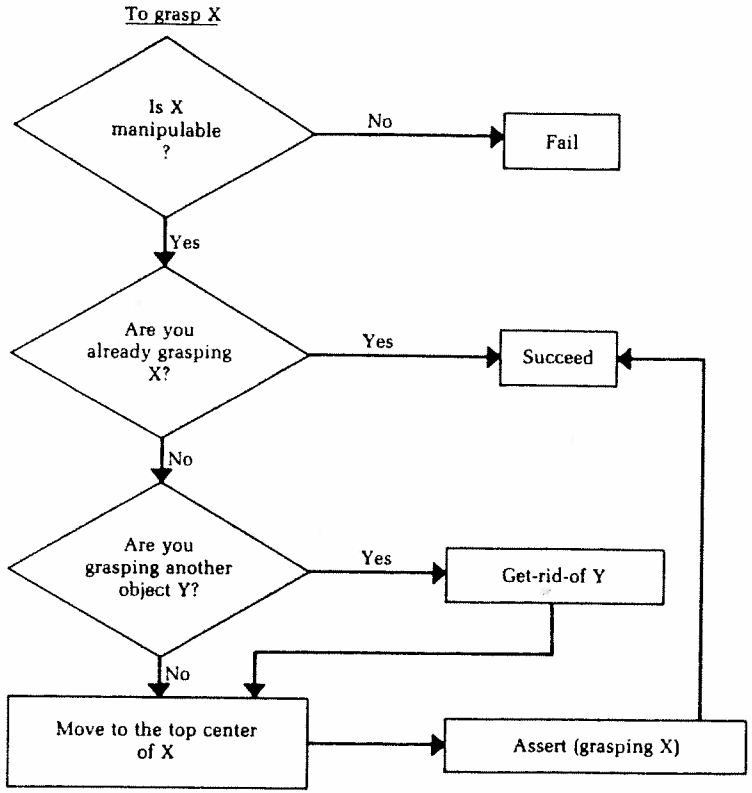

Figure 4.6 Procedural Description of GRASP.

These examples illustrate the use of procedural descriptions of concepts for carrying out commands, but they can also be applied to other aspects of language, such as questions and statements. One of the basic viewpoints underlying the model is that all language use can be thought of as a way of activating procedures within the hearer. We can think of any utterance as a program—one that indirectly causes a set of operations to be carried out within the hearer's cognitive system. This "program writing" is indirect in the sense that we are dealing with an intelligent interpreter, who may take a set of actions which are quite different from those the speaker intended. The exact form is determined by his knowledge of the world, his expectations about the person talking to him, his goals, etc. In this program we have a simple version of this process of interpretation as it takes place in the robot. Each sentence interpreted by the robot is converted to a set of instructions in PLANNER. The program that is created is then executed to achieve the desired effect. In some cases the

Box 4.2 Goal Stack.

```
(GRASP B1)
     (GET-RID-OF B2)
          (PUTON B2 TABLE1)
               (PUT B2 (453 201 0))
                    (MOVEHAND (553 301 100))
```

procedure invoked requires direct physical actions like the aforementioned. In others, it may be a search for some sort of information (perhaps to answer a question), whereas in others it is a procedure which stores away a new piece of knowledge or uses it to modify the knowledge it already has. Let us look at what the system would do with a simple description like "a red cube which supports a pyramid." The description will use concepts like BLOCK, RED, PYRAMID, and EQUIDIMENSIONAL—all parts of the system's underlying categorization of the world. The result can be represented in a flow chart like that of Figure 4.7. Note that this is a program for finding an object fitting the description. It would then be incorporated into a command for doing something with the object, a question asking something about it, or, if it appeared in a statement, it would become part of the program which was generated to represent the meaning for later use. Note that this bit of program could also be used as a test to see whether an object fit the description, if the first FIND instruction were told in advance to look only at that particular object.

At first glance, it seems that there is too much structure in this program, as we don't like to think of the meaning of a simple phrase as explicitly containing loops, conditional tests, and other programming details. The solution is to provide an internal language that contains the appropriate looping and checking as its primitives, and in which the representation of the process is as simple as the description. PLANNER provides these primitives in our system. The program described in Figure 4.7 would be written in PLANNER looking something like Box 4.3.* The loops of the flow chart are implicit in PLANNER's backtrack control structure. The description is evaluated by proceeding down the list until some goal fails, at which time the system backs up automatically to the last point where a decision was made, trying a different possibility. A decision can be made whenever a new object name or *variable* (indicated by the prefix ?) such as ?X1 or ?X2 appears. The variables are used by a pattern matcher. If they have already been assigned to a particular item, it checks to see whether the GOAL is true for that item. If not, it checks for all possible items which satisfy the GOAL, by choosing one, and then taking successive ones whenever backtracking occurs to that point. Thus, even the distinction between testing and choosing is implicit. Using other primitives of PLANNER, such as NOT and

*The system actually uses Micro-Planner, (Sussman et. al., 1970) a partial implementation of PLANNER. In this presentation we have slightly simplified the details of its syntax.

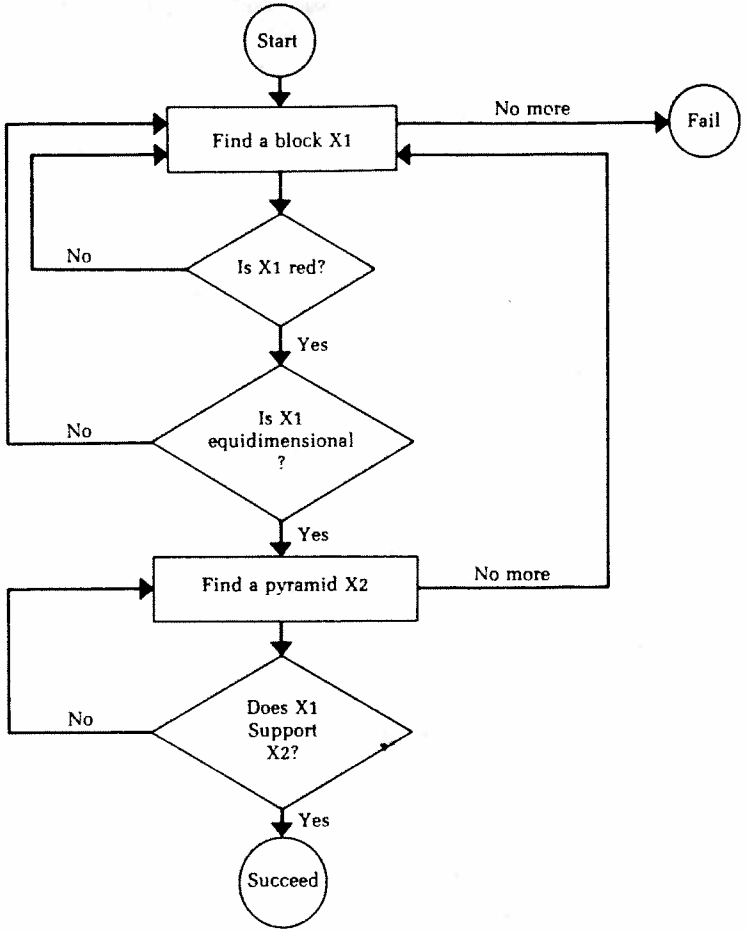

Figure 4.7 Procedural Representation of "a red cube which supports a pyramid."

Box 4.3 PLANNER Program for Description of "a red cube which supports a pyramid."

```
(GOAL (IS ?X1 BLOCK))
(GOAL (COLOR-OF ?X1 RED))
(GOAL (EQUIDIMENSIONAL ?X1))
(GOAL (IS ?X2 PYRAMID))
(GOAL (SUPPORT ?X1 ?X2))
```

FIND (which looks for a given number of objects fitting a description), we can write procedural representations for a variety of descriptions, as shown in Box 4.4.

Semantic Analysis

When we have decided how the system will represent meanings internally, we must deal with the way in which it creates a program when it is given an English input. There must be ways to interpret the meanings of individual words and the syntactic structures in which they occur. First, let us look at how we can define simple words like "cube", and "contain." The definitions in Box 4.5 are completely equivalent to those used in the program with a straightforward interpretation.* The first says that

Box 4.4 PLANNER Programs for some Quantified Modifiers describing the Object X1.

```
(GOAL (IS ?X2 PYRAMID))
(GOAL (SUPPORT ?X1 ?X2))
```
"which supports a pyramid"

****

```
(GOAL (SUPPORT ?X1 B3))
```
"which supports the pyramid'
B3 is the name of the object referred to by *"the pyramid"* which is determined earlier in the analysis

****

```
(FIND 3 ?X2 (GOAL (IS ?X2 PYRAMID))
            (GOAL (SUPPORT ?X1 ?X2)))
```
"which supports three pyramids"

****

```
(NOT (FIND ?X2 (GOAL (IS ?X2 PYRAMID))
               (GOAL (SUPPORT ?X1 ?X2)))
```
"which supports no pyramids"

****

```
(NOT (FIND ?X2 (GOAL (IS ?X2 PYRAMID))
               (NOT (GOAL (SUPPORT ?X1 ?X2)))))
```
"which supports every pyramid"

****

*Again, in comparing this with the details in Winograd (1972), note that some of the symbols have been replaced with more understandable mnemonic versions.

Box 4.5 Dictionary Definitions for "cube" and "contain."

```
(CUBE
   ((NOUN  (OBJECT
            ((MANIPULABLE RECTANGULAR)
                          ((IS ? BLOCK)
                           (EQUIDIMENSIONAL ?)))))))

(CONTAIN
   ((VERB ((TRANSITIVE (RELATION
                 (((CONTAINER)) ((PHYSICAL-OBJECT))
                          (CONTAIN #1 #2))
                 (((CONSTRUCT)) ((PHYSICAL-OBJECT))
                          (PART-OF #2 #1)))))))))
```

a cube is an object that is RECTANGULAR and MANIPULABLE, and can be recognized by the fact that it is a BLOCK and EQUIDIMENSIONAL. The first part of this definition is based on the use of semantic markers and provides for efficiency in choosing interpretations. By making a rough categorization of the objects in the model, the system can make quick checks to see whether certain combinations are ruled out by simple tests like "this meaning of the adjective applies only to words which represent physical objects." Chomsky's famous sentence "Colorless green ideas sleep furiously" would be eliminated easily by such markers. The system uses this information, for example, in answering question 9 in the dialogue, "*Can the table pick up blocks?*," as "*pick up*" demands a subject that is ANIMATE, whereas "*table*" has the marker INANIMATE. These markers are a useful but rough approximation to human deductions.

The definition for "contain" shows how they might be used to choose between possible word meanings. If applied to a CONTAINER and a PHYSICAL-OBJECT, as in "*The box contains three pyramids,*" the word implies the usual relationship we mean by CONTAIN. If instead, it applies to a CONSTRUCT (like "stack", "pile", or "row") and an object, the meaning is different. "*The stack contains a cube*" really means that a cube is PART of the stack, and the system will choose this meaning by noting that CONSTRUCT is one of the semantic markers of the word "stack" when it applies the definition.

One important aspect of these definitions is that although they look like static rule statements, they are actually calls to programs (OBJECT and RELATION) which do the appropriate checks and build the semantic structures. Once we get away from the simplest words, these programs need to be more flexible in what they look at. For example, in the robot world, the phrase "*pick up*" has different meanings depending on whether it refers to a single object or several. In sentence 1, the system interprets "*Pick up the big red block,*" by grasping it and raising the hand. If we said "*Pick up all of your toys,*" it would interpret "*pick up*" as meaning "*put away,*" and would pack them all into the box. The program for checking to see whether the object is singular or plural is simple, and any semantic system must have the flexibility to incorporate such things in the word definitions. We do this by having the definition of every word be a program which is called at an appropriate point in the analysis, and which can do arbitrary computations involving the sentence and the present physical situation.

This flexibility is even more important once we get beyond simple words. In defining words like "the," or "of," or "one" in "*Pick up a green one,*" we can hardly make a simple list of properties and descriptors as in Figure 4.12. The presence of "one" in a noun group must trigger a program which looks into the previous discourse to see what objects have been mentioned, and can apply various rules and heuristics to determine the appropriate reference. For example it must know that in the phrase "*a big red block and a little one,*" we are referring to "a little red block," not "a little big red block" or simply "a little block." This sort of knowledge is part of a semantic procedure attached to the word "one" in the dictionary.

Words like "the" are more complex. When we use a definite article like "the" or "that" in English, we have in mind a particular object or objects which we expect the hearer to know about. I can talk about "the moon" since there is only one moon we usually talk about. In the context of this article, I can talk about "the dialogue", and the reader will understand from the context which dialogue I mean. If I am beginning a conversation, I will say "Yesterday I met a strange man" even though I have a particular man in mind, since saying "Yesterday I met the strange man" would imply that the hearer already knows of him. Elsewhere, "the" is used to convey the information that the object being referred to is unique. If I write "The reason I wrote this paper was . . .", it implies that there was a single reason, whereas "A reason I wrote this paper was . . ." implies that there were others. In generic statements, "the" may be used to refer to a whole class, as in "The albatross is a strange bird." This is a quite different use from the single referent of "The albatross just ate your lunch."

A model of language use must be able to account for the role this type of knowledge plays in understanding. In the procedural model, it is a part of the process of interpretation for the structure in which the relevant word is embedded. The different possibilities for the meaning of "the" are procedures which check various facts about the context, then prescribe actions such as "Look for a unique object in the data base which fits this description." or "Assert that the object being described is unique as far as the speaker is concerned." The program incorporates a variety of heuristics for deciding what part of the context is relevant. For example, it keeps track of when in the dialogue something has been mentioned. In sentence 2 of the dialogue, "*Grasp the pyramid*" is rejected since there is no particular pyramid which the system can see as distinguished. However, in sentence 5 it accepts the question "*What is the pyramid supported by?*" since in the answer to sentence 4 it mentioned a particular pyramid.

This type of knowledge plays a large part in understanding the things that hold a discourse together, such as pronouns, adverbs like "then", and "there", substitute

nouns such as "one", phrases beginning with "that", and ellipses. The system is structured in such a way that the heuristics for handling mechanisms like these can be expressed as procedures in a straightforward way.

The Role of Syntax

In describing the process of semantic interpretation, we stated that part of the relevant input was the syntactic structure of the sentence. In order to provide this, the program contains a parser and a fairly comprehensive grammar of English.* The approach to syntax is based on a belief that the form of syntactic analysis must be useable by a realistic semantic system, and the emphasis of the resulting grammar differs in several ways from traditional transformational approaches.

First, it is organized around looking for syntactic units which play a primary role in determining meaning. A sentence such as "The three big red dogs ate a raw steak" will be parsed to generate the structure in Figure 4.8. The noun groups (NG) correspond to descriptions of objects, whereas the clause is a description of a relation or event. The semantic programs are organized into groups of procedures, each of which is used for interpreting a certain type of unit.

For each unit, there is a syntactic program (written in a language called PROGRAMMAR, especially designed for the purpose) which operates on the input string to see whether it could represent a unit of that type. In doing this, it will call on other such syntactic programs (and possibly on itself recursively). It embodies a description of the possible orderings of words and other units, for example, the scheme for a noun group, as shown in Figure 4.9. The presence of an asterisk after a symbol means that that function can be filled more than once. The figure shows that we have a determiner (such as "the") followed by an ordinal (such as "first"), then a number ("three") followed by one or more adjectives ("big," "red") followed by one or more nouns being used as classifiers ("fire hydrant") followed by a noun ("covers") followed by qualifying phrases which are preposition groups or clauses

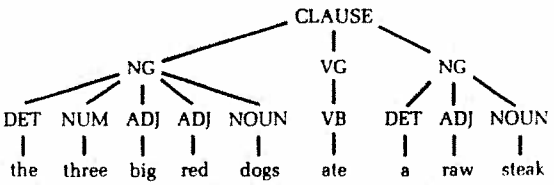

Figure 4.8 Syntactic Parse Tree.

*It is of course impossible to provide a complete grammar of English, and often difficult to evaluate a partial one. The dialogue of Section 2 gives a sample of the constructions which can be handled, and does not make use of specially included patterns. Winograd (1972) gives a full description of the grammar used.

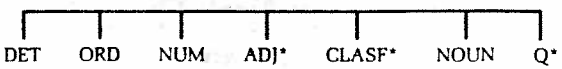

Figure 4.9 Structure of Noun Groups.

("without handles" "which you can find"). Of course many of the elements are optional, and there are restriction relations between the various possibilities. If we choose an indefinite determiner such as "a," we cannot have an ordinal and number, as in the illegal string "a first three big red fire hydrant covers without handles you can find." The grammar must be able to express these rules in a way which is not simply an ad hoc set of statements. Our grammar takes advantage of some of the ideas of Systemic Grammar (Halliday, 1971).

Systemic theory views a syntactic structure as being made up of units, each of which can be characterized in terms of the *features* describing its form, and the *functions* it fills in a larger structure or discourse. In the sentence in Figure 4.8, the noun group "three big red dogs" can be described as exhibiting features such as DETERMINED, INDEFINITE, PLURAL, etc. It serves the function SUBJECT in the clause of which it is a part, and various discourse functions, such as THEME as well. It in turn is made up of other units – the individual words – which fill functions in the noun group, such as DETERMINER and HEAD. A grammar must include a specification of the possible features a unit can have, and the relation of these to both the functions it can play, and the functions and constituents it controls.

These features are not haphazard bits of information we might choose to notice about units, but form a highly structured system (hence the name Systemic Grammar). As an example, we can look at a few of the features for the CLAUSE in Figure 4.10. The vertical lines represent sets from which a single feature must be selected and horizontal lines indicate logical dependency. Thus, we must first choose whether the clause is MAJOR – which corresponds to the function of serving as an independent sentence – or SECONDARY, which corresponds to the various functions a clause can serve as a constituent of another unit (for example as a QUALIFIER in the noun group "the ball *which is on the table*"). If a clause is MAJOR, it is either DECLARATIVE ("She went"), IMPERATIVE ("Go"), or INTERROGATIVE ("Did she go?"). If it is INTERROGATIVE, there is a further choice between YES-NO ("Did she go?") and WH- ("Where did she go?").

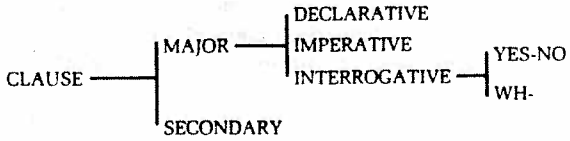

Figure 4.10 Simple System Network for Clauses.

It is important to note that these features are syntactic, not semantic. They do not represent the use of a sentence as a question, statement, or command, but are rather a characterization of its internal structure – which words follow in what order. A DECLARATIVE can be used as a question by giving it a rising intonation, or even as a command, as in "You're going to give that to me," spoken in an appropriate tone. A question may be used as a polite form of a command, as in "Can you give me a match?," and so on. Any language understander must know the conventions of the language for interpreting such utterances in addition to its simpler forms of syntactic knowledge. To do this, it must have a way to state things like "If something is syntactically a question but involves an event which the hearer could cause in the immediate future, it may be intended as a request." Syntactic features are therefore basic to the description of the semantic rules. The actual features in a comprehensive grammar are related in a more complex way than the simple example of Figure 4.10, but the basic ideas of logical dependency are the same.

In the foregoing we stated that there is a choice between certain features, and that depending on the selection made from one set, we must then choose between certain others. In doing this we are not postulating a psychological model for the order of making choices. The networks are an abstract characterization of the possibilities, and form only a part of a grammar. In addition we need realization and interpretation rules. Realization rules describe how a given set of choices would be expressed in the form of surface syntactic structures, whereas interpretation rules describe how a string of words is analyzed to find its constituents and their features.

Our grammar is an interpretation grammar for accepting grammatical sentences. It differs from more usual grammars by being written explicitly in the form of a program. Ordinarily, grammars are stated in the form of rules, which are applied in the framework of a special interpretation process. This may be very complex in some cases (such as transformational grammars) with separate phases, special "traffic rules" for applying the other rules in the right order, cycles of application, and other sorts of constraints. In our system, the sequence of the actions is represented explicitly in the set of rules. The process of understanding an utterance is basic to the organization of the grammar.*

In saying that grammars are programs, it is important to separate the procedural aspect from the details usually associated with programming. If we say to a linguist "Here is a grammar of English," he can rightfully object if it begins "Take the contents of location 177 and put them into register 2, adding the index . . ." The formalization of the syntax should include only those operations and concepts that are relevant to linguistic analysis, and should not be burdened with paraphernalia needed for programming details. Our model is based on the belief that the basic ideas of programming such as procedure and subprocedure, iteration, recursion, etc. are central to all cognitive processes, and in particular to the theory of language. What is needed is a formalism for describing syntactic processes. Our grammar is written in a language which was designed specifically for the purpose. It is a system built in LISP, called PROGRAMMAR, and its primitive operations are those involving the building of syntactic structures, and the generation of systemic descriptions of their parts.

*For a discussion of the psycholinguistic relevance of such interpretive grammars see Kaplan (1971). He describes a similar formulation of procedural grammar, represented as a transition network.

The set of typical grammar rules shown in Box 4.6 would be expressed in PROGRAMMAR by the program diagrammed in Figure 4.11. For such a simplified bit of grammar, there isn't much difference between the two formulations, except that the PROGRAMMAR representation is more explicit in describing the flow of control. When we try to deal with more complex parts of syntax, the ability to specify procedures becomes more important. For example the word "and" can be associated with a program that can be diagrammed as shown in Figure 4.12. Given the sentence "The giraffe ate the apples and peaches", it would first encounter "and" after parsing the noun "apples." It would then try to parse a second noun, and would succeed, resulting in the structure shown in Figure 4.13. If we had the sentence "The giraffe ate the apples and drank the vodka," the parser would have to try several different things. The "and" appears at a point which represents boundaries between several units. It is after the noun "apples," and the NP, "the apples." It is also after the entire VP "ate the apples." The parser, however, cannot find a noun or NP beginning with the following word "drank". It therefore tries to parse a VP and would successfully find "drank the vodka". A CONJOINED VP would be created, producing the final result shown in Figure 4.14. Of course the use of conjunctions is more complex than this, and the actual program must take into account such things as lists and branched structures in addition to the problems of backing up if a wrong possibility has been tried. But the basic operation of "look for another one like the one you just found" seems both practical and intuitively plausible as a description of how conjunction works. The ability to write the rules as procedures leaves us the flexibility to extend and refine it.

Viewing "and" as a special program that interrupts the normal parsing sequence also gives us a sort of explanation for some puzzling syntactic facts. The statement "I saw Ed with Steve" has a corresponding question, "Whom did you see Ed with?" But "I saw Ed and Steve" cannot be turned into "Whom did you see Ed and?" The "and" program cannot be called when there is no input for it to work with.

Box 4.6 Simple Grammar in Replacement Rule Form.

```
 S → NP VP
NP → DETERMINER NOUN
VP → VERB/TRANSITIVE NP
VP → VERB/INTRANSITIVE
```

Program Organization

So far, we have described how three different types of knowledge are represented and used. There is the data base of assertions and PLANNER procedures which represent the knowledge of the physical world; there are semantic analysis programs

which know about such problems as reference; and there is a grammar which determines the syntactic structure. The most important element, however, is the interaction between these components. Language cannot be reduced into separate areas such as "syntax, semantics, and pragmatics" in hopes that by understanding each of them separately, we have understood the whole. The key to the function of language as a means of communication is in the way these areas interact.

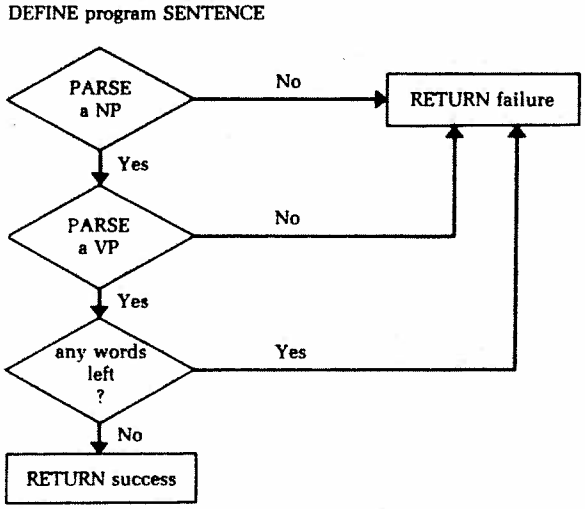

Figure 4.11 PROGRAMMAR Grammar From Winograd, T., "Understanding Natural Language." *Cognitive Psychology*, 3:1–191. Copyright © by Academic Press.

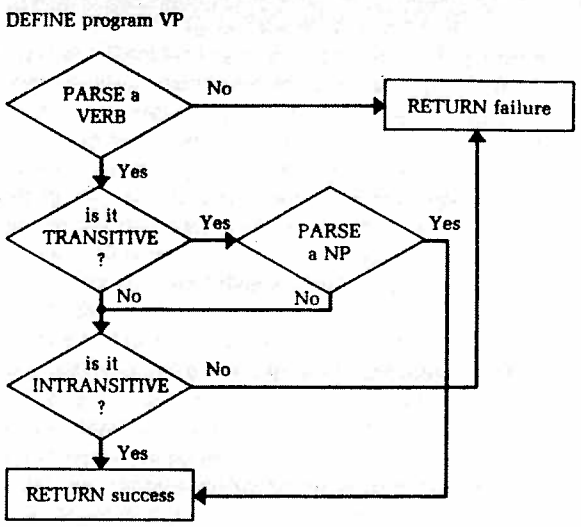

Figure 4.11 *(continued)*

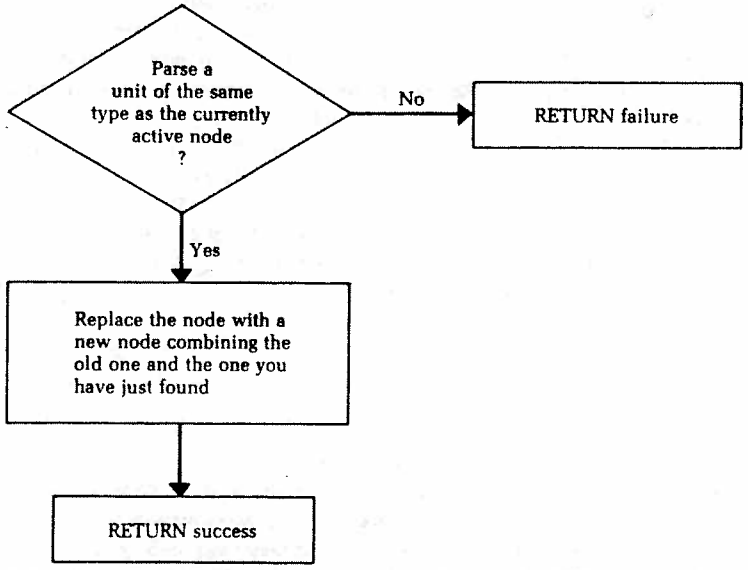

Figure 4.12 Conjunction Program. From Winograd, T., "Understanding Natural Language." *Cognitive Psychology*, 3:1–191. Copyright © by Academic Press.

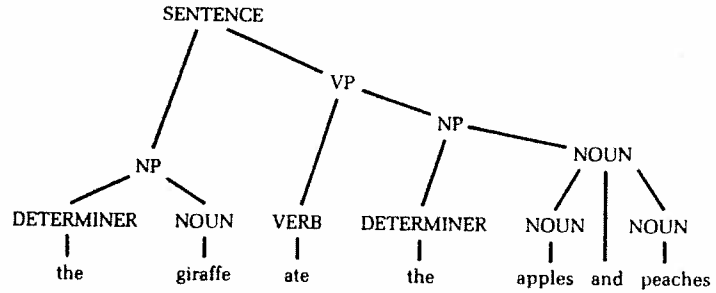

Figure 4.13 Conjoined Noun Structure. From Winograd, T., "Understanding Natural Language." *Cognitive Psychology*, 3:1–191. Copyright © by Academic Press.

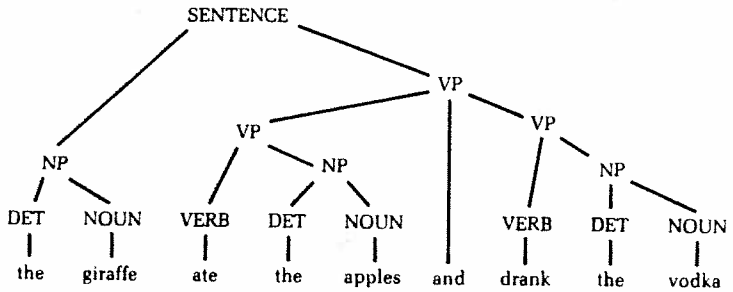

Figure 4.14 Conjoined VP Structure. From Winograd, T., "Understanding Natural Language." *Cognitive Psychology*, 3:1–191. Copyright © by Academic Press.

Our program does not operate by first parsing a sentence, then doing semantic analysis, and finally by using deduction to produce a response. These three activities go on concurrently throughout the understanding of a sentence. As soon as a piece of syntactic structure begins to take shape, a semantic program is called to see whether it might make sense, and the resultant answer can direct the parsing. In deciding whether it makes sense, the semantic routine may call deductive processes and ask questions about the real world. As an example, in sentence 36 of the dialogue ("*Put the blue pyramid on the block in the box*"), the parser first comes up with "the blue pyramid on the block" as a candidate for a noun group. At this point, semantic analysis is begun, and since "the" is definite, a check is made in the data base for the object being referred to. When no such object is found, the parsing is redirected to find the noun group "the blue pyramid." It will then go on to find "on the block in the box" as a single phrase indicating a location. In other examples the system of semantic markers may reject a possible interpretation on the basis of conflicting category information. Thus, there is a continuing interplay between the different sorts of analysis, with the results of one affecting the others.

The procedure as a whole operates in a left to right direction through the sentence. It does not carry along multiple possibilities for the syntactic analysis, but instead has ways of going back and doing something different if it runs into trouble. It does not use the general backup mechanism of PLANNER, but decides what to do on the basis of exactly what sort of problem arose. In the sentences like those of the dialogue, very little backup is ever used, since the combination of syntactic and semantic information usually guides the parser quite efficiently.

Limitations of the Approach

The program we are describing does not purport to be a point by point model of psychological processes at a detailed level. Rather, it is an attempt to show how a general view of language can really be filled in with enough detail to provide a working model. The importance from a psychological point of view is the approach to language as a process which can be modeled within the context of a procedural description of cognitive processes. Rather than trying to attach psychological meaning to isolated components into which language has been divided for abstract study, it attempts to relate the various types of knowledge and procedures involved in intelligent language use.

Looking into the specific capabilities of the system, we can find many places where the details seem inadequate, or whole areas are missing. The program does not attempt to handle hypothetical or counterfactual statements; it only accepts a limited range of declarative information, it cannot talk about verbal acts, and the treatment of "the" is not as general as the description above, and so on. These deficiencies, however, seem to be more a matter of what has been tackled so far, rather than calling into question the underlying model. Looking deeper, we can find two basic ways in which it seems an inadequate model of human language use. The first is the way in which the process is directed, and the second is concerned with the interaction of the context of the conversation and the understanding of its content.

We can think of a program for understanding a sentence as having two kinds of operations—coming up with possible interpretations, and choosing between them. Of course, these are not separate psychologically, but in the organization of computer programs, the work is divided up.

In our program, the syntactic analysis is in charge of coming up with possibilities. The basic operation requires that we find a syntactically acceptable phrase, and then do a semantic interpretation on it to decide whether to continue along that line of parsing. Other programs such as Schank (1971) and Quillian (1967) use the semantic information contained in the definitions of the words to provide an initial set of possibilities, then use syntactic information in a secondary way to check whether the hypothesized underlying semantic structure is in accord with the arrangement of the words.

By observing human language use, it seems clear that no single approach is really correct. On the one hand, people are able to interpret utterances which are not syntactically well formed, and can even assign meanings to collections of words without use of syntax. The list "skid, crash, hospital" presents a certain image, even though two of the words are both nouns and verbs and there are no explicit syntactic connections. It is therefore wrong to insist that some sort of complete parsing is a prerequisite to semantic analysis.

On the other hand, people are able to interpret sentences syntactically even when they do not know the meanings of the individual words. Most of our vocabulary (beyond a certain age) is learned by hearing sentences in which unfamiliar words appear in syntactically well-defined positions. We process the sentence without knowing any category information for the words, and in fact use the results of that processing to discover the semantic meaning. In addition, much of our normal conversation is made up of sentences like "Then the other one did the same thing to it" in which the words taken individually do not provide clues to enable us to determine the conceptual structure without a complete syntactic analysis.

What really seems to be going on is a coordinated process in which a variety of syntactic and semantic information can be relevant, and in which the hearer takes advantage of whatever is more useful in understanding a given part of a sentence. Our system models this coordination in its order of doing things, by carrying on all of the different levels of analysis concurrently, although it does not model it in the control structure.

Much remains to be done in understanding how to write computer programs in which a number of concurrent processes are working in a coordinated fashion without being under the primary hierarchical control of one of them. A language model able to implement the sort of "heterarchy" found in biological systems (like the coordination between different systems of an organism) will be much closer to a valid psychological theory.

The second basic shortcoming is in not dealing with all the implications of viewing language as a process of communication between two intelligent people. A human language user is always engaged in a process of trying to understand the world around him, including the person he is talking to. He is actively constructing models and hypotheses, and he makes use of them in the process of language understanding. As an example, let us consider again the use of pronouns. In Section 1, we described some of the knowledge involved in choosing referents. It included syntax, semantic categories, and heuristics about the structure of discourse.

But all of these heuristics are really only a rough approximation to what is really going on. The reason that the focus of the previous sentence is more likely to be the referent of "it" is because a person generally has a continuity in his conversation, which comes from talking about a particular object or event. The focus (or subject) is more likely just because that is the thing he is talking about, and he is likely to go on talking about it. Certain combinations of conceptual category markers are more plausible than others because the speaker is probably talking about the real world, where certain types of events are more sensible than others. If we prefix almost any sentence with "I just had the craziest dream . . ." the whole system of plausible conceptual relations is turned topsy-turvy.

If someone says "I dropped a bottle of Coke on the table and it broke," there are two obvious interpretations. The semantic categories and the syntactic heuristics make it slightly more plausible that it was the bottle that broke. But consider what would happen if we heard "Where is the tool box? I dropped a bottle of coke on the table and it broke" or, "Where is the furniture polish? I dropped a bottle of coke on the table and it broke." The referent is now perfectly clear—only because we have a model of what is reasonable in the world, and what a person is likely to say. We know that there is nothing in the tool box to help fix a broken coke bottle and that nobody would be likely to try fixing one. It would be silly to polish a table that just got broken, while it would be logical to polish one that just had a strong corrosive spilled on it. Of course, all this must be combined with deductions based on other common sense knowledge, such as the fact that when a bottle containing a liquid breaks, the liquid in it spills.

Even more important, we try to understand what the speaker is "getting at." We assume that there is a meaningful connection between his sentences, and that his description of what happened is probably intended as an explanation for why he wants the polish or toolbox. More subtle deductions are implied here as well. It is possible that he broke the table and fixed it, and now wants the polish to cover the repair marks. If this were the case, he would almost surely have mentioned the repair to allow us to follow that chain of logic.

Our system makes only the most primitive use of this sort of deduction. Since it keeps track of when things have been mentioned, it can check a possible interpretation of a question to see whether the asker could answer it himself from his previous sentences. If so, it assumes that he probably means something else. We could characterize this as containing two sorts of knowledge. First, it assumes that a person asks questions for the purpose of getting information he doesn't already have, and second, it has a very primitive model of what information he has on the basis of what he has said. A realistic view of language must have a complex model of this type, and the heuristics in our system touch only the tiniest bit of the relevant knowledge.

It is important to recognize that this sort of interaction does not occur only with pronouns and explicit discourse features, but in every part of the understanding process. In choosing between alternative syntactic structures for a sentence, or picking between multiple meanings of words, we continually use this sort of higher level deduction. We are always basing our understanding on the answer to questions like "Which interpretation would make sense given what I already know?" and "What is he trying to communicate?"

Any attempt to model human language with simple semantic rules and heuristics like those described above is a bit like an attempt to model the behavior of a complex system by using unrelated mathematical formulas whose results are a general

approximation to its output. The results may be of interest, and the resulting equations may have a high correlation with what is going on, but it is not a model in the true sense of reflecting the underlying process.

It seems likely that more advanced computational models will move towards overcoming these deficiencies. As we learn more about the organization of large complex systems, we may well be able to model language in ways which are more complete, clearer, and closer to psychological reality.