Peter Szolovits

OWL

A System for Building Expert Problem Solving Systems Involving Verbal Reasoning

Contents

Course 6.871 Notes

Introduction

The MACSYMA system for algebraic manipulation has found wide application in applied mathematics. A key point in its success is the design philosophy for choosing what mathematical knowledge should be built into the system and how this should be done. Experiments showed that a system which provided only list or string structure operations and a control structure for mathematics problem solving was of very little value. For one thing, getting the right computer representations of mathematical expressions was essentail if one wanted to build more than a toy system. Once these were found, they constituted an important part of the knowledge of algebraic manipulation to be delivered to the user. For another, the amount of mathematics which had to be coded into the computer was so large that no one user could reach critical mass. It was necessary for a user to build on the work of others; he could not define all of his own conventions. An important problem was thus to deliver such operations as factoring, simplification, and power series manipulation in a form which many different users could adapt to their individual needs. This was done by putting the system into the field and reacting to user suggestions. In this way it was possible to gradually build up a larger and larger base of generally acceptable conventions. A third point is that the implications to the system design of facts like the commutability of plus are so great that the system must be built assuming them to be true. Such facts are built into the heart of every module of the MACSYMA system, not accessed from a data base. Since at the present time it would be difficult to imagine a compiler smart enough to make the same module constructions given a table of such facts, building the facts in is required.

An important thing to note is that although the number of data representation conventions in MACSYMA is large, it is much smaller than the number of algorithms, and even smaller than the number of problem situations for which the system might be useful. However, once the data representation conventions are established, the basic functions such as simplification can be provided and any number of new modules can be added which communicate through these standard representations.

The language of MACSYMA is the language of the elementary functions of mathematics. Anyone who can represent his problem so that the solution involves storing, retrieving, and manipulating elementary functions may find MACSYMA useful to him. Unfortunately, it has not been possible to represent many problems in business, law, and medicine in this way. Although some of these problems may not even possess well defined solutions, it is clear that, for practical purposes, many of them do. Although we do not know exactly how medical, legal, and business experts solve these problems (just as we did not know how applied mathematicians simplified expressions in practice and they were unable to tell us), we do know that these experts explain their problem solving process in terms of manipulations on facts expressed in English, not mathematics. If we are to proceed as we did with MACSYMA, we must develop conventions for good data structures capable of expressing these facts. The reader may think our project too ambitious. He must remember that we seek only a structure which a user may usefully map his problem into. Since much has been done in medicine and business by mapping problems into operations on sets and numbers, we can expect

further success using structures inspired by natural language, even though they lack the full power of the English language. Also, we must remember that we are primarily interested in expert problem solving, where terms are rather more narrow and well defined.

We insist that our new language, OWL, be similar to English for several reasons. First, the language will be very large. It will have a big "grammar" compared to a programming language, but it will also have a much larger body of "facts", such as which actions take objects. If we can make the "facts" of the OWL language the same or similar to those of English, then the user will essentially have only the basic grammar rules of OWL to learn. In addition, it will be easier for us to translate from English to OWL automatically. English also provides us with a source of useful concepts. The concepts and structure of the English language are not chosen at random. If business men make the distinction between a function and a process, then they must find that distinction useful. The use of English as a guide in the definition of the data structures of OWL is, in fact, the major feature of the OWL system. In the conclusion to this monograph we will attempt to evaluate its effectiveness.

There are many more questions which must be raised. It seems most natural to do this during the course of the presentation.

Chapter 1

The OWL data structure and the OWL data base.

Subsets and characteristics

A basic notion in OWL (and in other languages such as SIR and MERLIN) is that one set of objects contains another as a subset. For example, the set of all fruit contains the set of all apples. We express this as (KIND FRUIT APPLE). The set SOMETHING is defined to contain everything, and thus everything is a KIND of SOMETHING.

We allow abstract concepts like QUESTION, CUSTOMER, and WANT in OWL, but we will start our explanation with physical objects and their properties. The essence of this structure of objects and their properties is given in Figure 1.1. Physical objects may be assigned a wide variety of characteristics, as we will discuss later. Nominal characteristics are those like color which take their values, which can be expressed as nouns, from a well defined set. Ordinal

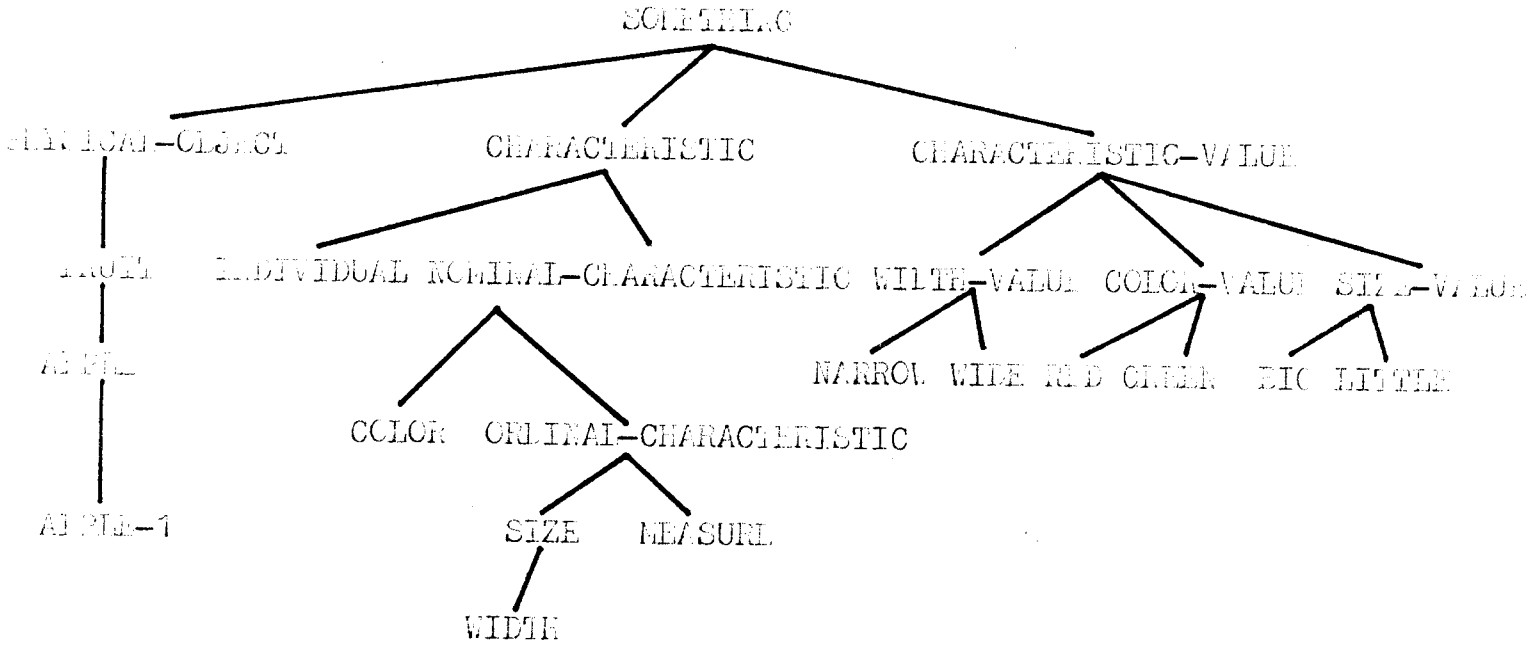


Figure 1.1

Subset relations among objects and characteristics

characteristics are nominal characteristics whose values possess a natural ordering. Note that we have defined RED to be a COLOR-VALUE rather than a COLOR. Later we will attempt to convince the reader that this is the better choice.

To state that FRUIT has the characteristic COLOR we write (COLOR FRUIT COLOR-VALUE). We think of this expression as standing for the set of all applications of COLOR to FRUIT and saying that each application specifies a COLOR-VALUE. Note, that since APPLE is a KIND of FRUIT and RED is a KIND of COLOR-VALUE, that (COLOR APPLE RED) is an instantiation of (COLOR FRUIT COLOR-VALUE). In other words, the set (COLOR APPLE RED) is a subset of the set (COLOR FRUIT COLOR-VALUE). The concept APPLE-1 stands for a subset of APPLE. It may in fact be a set of one member, but we still treat it as a set. The fact that it has one member can be marked with the characteristic INDIVIDUAL: (INDIVIDUAL APPLE-1). Note that because of the subset relation between APPLE and APPLE-1, stating (COLOR APPLE RED) implies (COLOR APPLE-1 RED). At this point the reader may be thinking that APPLE-1 is probably red mixed with yellow, green and brown. We can express this thought in OWL, because it is a sentence in English. The OWL system, however can't visualize APPLE-1 as the reader may be able to do. It is the object of our experimentation to test the value of using only verbal descriptions in expert problem solving, coupling them of course with such numerical techniques as already exist.

Note that the characteristic INDIVIDUAL does not take a third argument. There are many other characteristics of this type, such as "current" and "possible". For subset statements and characteristic statements which take a third argument, the third argument is optional.

Thus, we can refer to a subset of apples as (KIND APPLE) and we can refer to the color of APPLE-1, without mentioning that color, as (COLOR APPLE-1).

A MEASURE is a characteristic of an ordinal characteristic. For example we might state (SIZE APPLE-1 BIG). Since there is no absolute notion of BIG, such a statement must be given a non-absolute interpretation. We can be more precise by giving a measure of the size of APPLE-1. The value of a measure must be a number. A characteristic of a measure is its UNIT-OF-MEASURE. For example,
(UNIT-OF-MEASURE (DIAMETER APPLE-1) 3) INCH) (MEASURE
Here MEASURE does not modify DIAMETER, but the set of applications of DIAMETER to APPLE-1. We will take "MEASURE" by itself to stand for all of the applications of MEASURE. Applying these ideas we could rephrase the statement above as

(KIND MEASURE INCH-MEASURE)

(KIND-OF-MEASURE INCH-MEASURE INCH)

(INCH-MEASURE (DIAMETER APPLE-1) 3)

Definition of an arch

We can introduce a number of additional ideas as well as allow comparison of OWL with the data structure of Winston by defining an ARCH as shown in Figure 1.2.

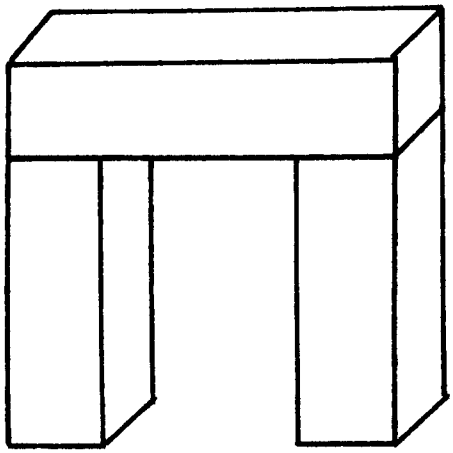


Figure 1.2

(KIND STRUCTURE ARCH)

We will need the concept of PART, which has its intuitive meaning. We also take (KIND PART TOP), i.e., the set of all PARTs contains the set of all TOPs. Winston terms a rectangular solid a BLOCK. In order to modify a given expression more than once it will be convenient to give it a name. This name will be a meta-linguistic device and is introduced with the operator TOKEN. Thus (TOKEN (KIND BLOCK) B1) names this specification of a subset of blocks B1.

Now we are ready to start the definition of the arch. We have

(PART ARCH (TOKEN (KIND BLOCK) B1))
(QUANTITY B1 TWO)
(TOP ARCH (KIND BLOCK LINTEL))
(QUANTITY LINTEL ONE)

The values of QUANTITY are ONE, TWO, FEW, SEVERAL, and MANY. Note that the QUANTITY is taken to be the number in any specific arch instance.

Next we want to say that the lintel is on the other two blocks. We define words like ON, AT, and NEAR to be LOCATION-RELATION's. The application of a LOCATION-RELATION to a physical object specifies a part of space known as a LOCATION. The POSITION of a physical object has as value a LOCATION. For example,

(POSITION LINTEL (ON B1)).

In these conventions we are following McDermott. The use of the words position and location is motivated by the difference between a "scenic location" and a "scenic position". We can talk about the size of location but not about the size of a position.

In order to complete our definition we must introduce the notion of a PROCEDURE. For OWL, a procedure does not necessarily imply a series of actions, except in the sense of the purposeful maintenance of a condition, as in "He stood at attention for two hours". In OWL, procedures correspond to specific meanings of verbs, although there are many procedures without corresponding verb meaning because no single word exists for that procedure in English. Frequently there is a principal object to which the procedure is applied, corresponding to the direct object of a sentence in English. For example, we specify the set of applications of the procedure HIT to a subset of BALL by (HIT (KIND BALL)). This can be understood to mean that (KIND BALL) is in the state of being hit. Thus a procedure is similar to a one argument characteristic. Procedures have characteristics, just as other OWL concepts do. The most important characteristics applicable to procedures are the SEMANTIC-CASE's. Of these, the most important is the AGENT. The AGENT performs or causes the indicated action. For example,

we might have (AGENT (HIT (KIND BALL)) JOHN).

Let us return now to our definition of an arch. So far we have

(KIND STRUCTURE ARCH)
(PART ARCH (TOKEN (KIND BLOCK) B1))
(TOP ARCH (KIND BLOCK LINTEL)
(QUANTITY B1 TWO)
(QUANTITY LINTEL ONE)
(POSITION LINTEL (ON B1))

We add

(STAND B1)
(LIE LINTEL)
(AGENT (SUPPORT LINTEL) B1)
(NOT (ABUT B1))

This says that any object specified by B1 is in the state of standing, the lintel is in the state of lying, the objects specified by B1 support the lintel and are not in the state of abutting. Note that whether stand and lie are procedures or characteristics is a moot point here because of the uniform treatment. SUPPORT is tested as a procedure not because there is action, but because the notion that the objects in B1 are active while the LINTEL is passive is useful in determining which blocks can be placed first in constructing an arch. Contrast

(POSITION B1 (UNDER LINTEL))

with (AGENT (SUPPORT LINTEL) B1). It is the use of the SUPPORT relation

which determines its form.

We have only to declare that one of the two supporting blocks is to the left of the other in order to finish our ARCH definition. Note that this is the first point where we distinguish between the two supports. When using this structure for identifying a possible arch, it is necessary to find candidates for both supports before the left-right relationship can be tested. This definition facilitates the formation of the set of candidates based on common properties. The complete definition is given in Figure 1.3.

```
(KIND STRUCTURE ARCH)
(PART ARCH (TOKEN (KIND BLOCK) B1))
(TOP ARCH (KIND BLOCK LINTEL))
(QUANTITY B1 TWO)
(QUANTITY LINTEL ONE)
(POSITION LINTEL (ON B1))
(STAND B1)
(LIE LINTEL)
(AGENT (SUPPORT LINTEL) B1)
(NOT (ABUT B1))
(POSITION (TOKEN (KIND B1) LEFT-SUPPORT)
        (LEFT-OF  (TOKEN (KIND B1) RIGHT-SUPPORT)))
(QUANTITY LEFT-SUPPORT ONE)
(QUANTITY RIGHT-SUPPORT ONE)
```

Figure 1.3

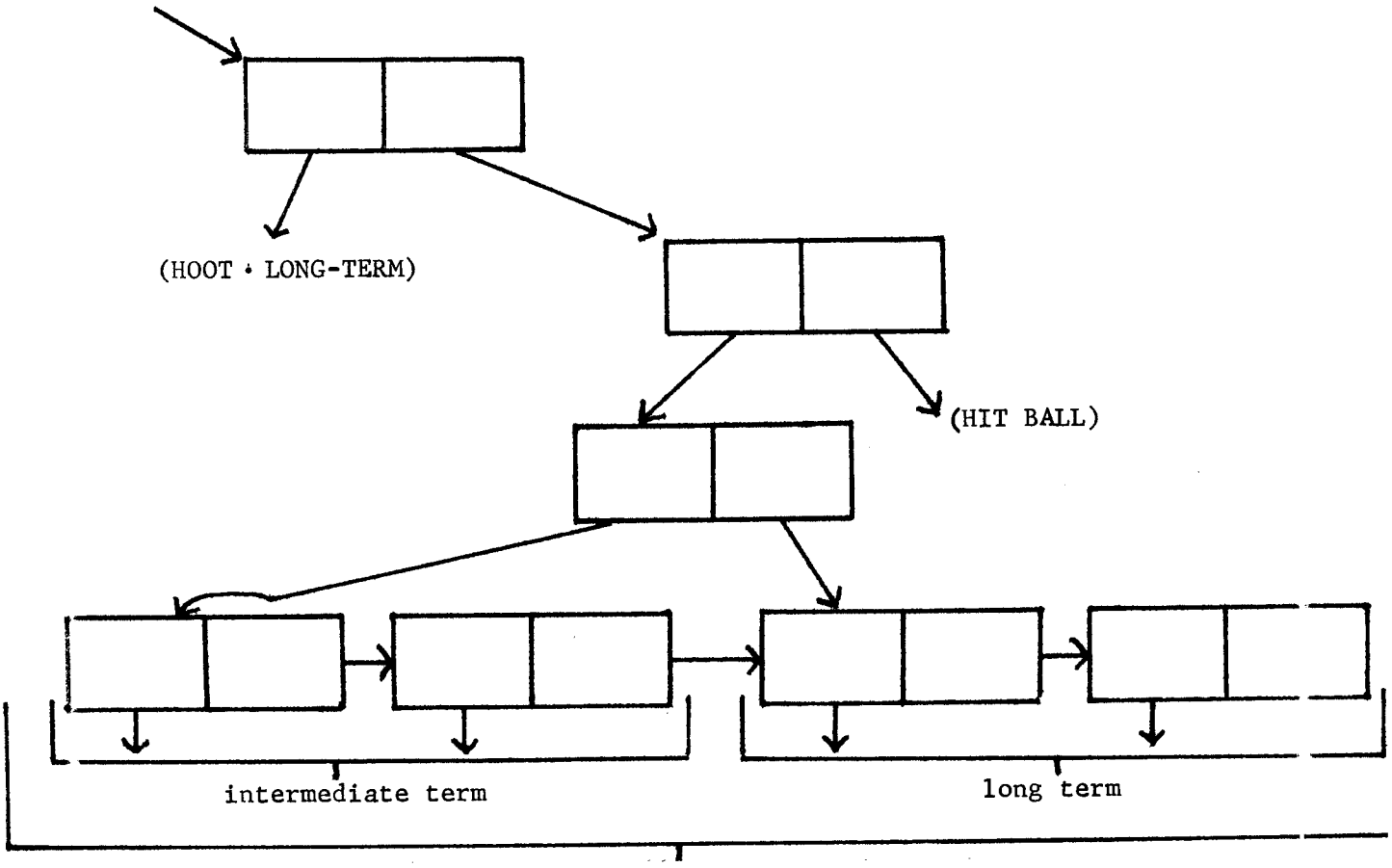
Definition of an ARCH

The OWL data base

In the previous section we have defined an arch, but we have not described how this definition is read into the machine and stored there, or what functions are available for accessing it. A complete list of the data base handling functions is given in Appendix A. This section is a primer in their use.

The data base of OWL consists of a short term memory, an intermediate term memory, and a long term memory. There are no contexts as in CONNIVER. While it is essential to group and classify facts, the CONNIVER context mechanism is too crude. In fact the CONNIVER data retrieval facility is a general purpose mechanism which screens the user from its workings and actually inhibits the use of special knowledge in order to speed retrieval. The emphasis in OWL is on the procedural specification of data retrieval, using knowledge of the semantic structure of the data base.

Short term memory is simply a set of LISP free variables, such as GOAL-LIST, to be described in the next chapter. The intermediate term and long term memories are structurally identical except for markers indicating the relevant memory. They both contain atoms and non-atomic data items. The form of the non-atomic structure is shown in Figure 1.4.



Back pointers to all items containing this item as a top level element not in first position.
Figure 1.4

The long term memory element: (HIT BALL)

Basically, OWL data is like LISP data except that every list and atom has a pointer to the front of every maximal list in which it appears. By a maximal list we mean one that is not the CDR of another list. Lists cannot appear in the first position of a list. In order to take the CAR, CDR, CADR, etc. of an OWL item the functions $CAR, $CDR, $CADR, etc. must be used. These functions first take the CDDR of the item and then perform the normal LISP functions.

The function which reads definitions into long term memory is called LEARN (the corresponding intermediate term memory function is called ABSORB). Any name defined by TOKEN is local to the LEARN. Inside the machine the name is replaced with the designated OWL item. When an application of KIND has a special argument, the second argument also becomes a global variable. To learn a slightly modified definition of our arch we would write

```
(LEARN (KIND STRUCTURE ARCH)
       (PART ARCH SUPPORT-N)
       (TOP ARCH (KIND BLOCK LINTEL))
       (QUANTITY SUPPORT-N TWO)
       (QUANTITY LINTEL ONE)
       (POSITION LINTEL (ON SUPPORT-N))
       (STAND SUPPORT-N)
       (LIE LINTEL)
       (AGENT (SUPPORT LINTEL) (KIND BLOCK SUPPORT-N))
       (NOT (ABUT SUPPORT-N))
       (POSITION (TOKEN (KIND SUPPORT-N) LEFT-SUPPORT)
         (LEFT-OF (TOKEN (KIND SUPPORT-N) RIGHT-SUPPORT)))
       (QUANTITY LEFT-SUPPORT ONE)
```

(QUANTITY RIGHT-SUPPORT ONE))

As in LISP, atoms are represented uniquely in OWL. The real power of the OWL data structure is that within one LEARN all lists which are EQUAL are represented by a unique OWL item, unless they are distinguished with TOKEN as are LEFT-SUPPORT and RIGHT-SUPPORT above. Thus an OWL item can be treated much like a LISP atom (which is also unique) with the uses of the item forming its property list.

Development of the meaning of items through the contexts in which they occur seems to be a powerful idea for solving the representation problems we face in OWL. For example, consider the statement (AGENT (SUPPORT LINTEL) (KIND BLOCK SUPPORT-N)). We use the name SUPPORT-N rather than SUPPORT because SUPPORT has been used for the procedure name. Words in English often have several meanings and thus we run out of words in OWL where, as in other programming, things are simplified by not using the same identifier for two different purposes.) From the property list of SUPPORT-N, we find that a SUPPORT-N is a kind of BLOCK, but then from the property list of (KIND BLOCK SUPPORT-N) we find that this expression is the value of the AGENT characteristic of (SUPPORT LINTEL). This is conventionally interpreted in OWL to mean that one type of SUPPORT-N is a block which is distinguished from other blocks precisely because it supports a LINTEL, Similarly, by this rule we find that a LINTEL has been defined as a BLOCK which is the TOP of an ARCH.

In a typical use of the OWL system the user would load his permanent definitions into the system from a file containing a series of

LEARN's, thus putting them in long term memory. He would then commence execution (as described in the next chapter). The OWL interpreter and his own hand coded LISP functions would add new facts to intermediate term memory using the function ADD. Occasionally, new facts would be LEARNed. Nothing would ever be delted! Old facts must be recognized as obsolete by their context. Eventually, intermediate term memory would fill up. Using the LISP function (WIPEOUT), the user can cause everything in intermediate term memory to be garbage collected. Thus, when memory becomes full, it is up to the user to decide what to LEARN. He can cause everything else to be forgotten.

Suppose X has as LISP value the OWL item (KIND APPLE) and Y has the LISP value RED. Executing the LISP call (ADD 'COLOR X Y) would add the OWL item (COLOR (KIND APPLE) 'RED) to intermediate term memory and return that item as its value. Thus ($CADR (ADD 'COLOR X Y)) would be (KIND APPLE). At this point executing (FINDVAL X 'COLOR) would return (COLOR (KIND APPLE) RED). (FINDVAL (KIND APPLE) 'COLOR) will find nothing unless the quoted KIND expression is the OWL item in the X COLOR expression. Notice that the entire COLOR item is returned, not just RED. This is because the item may in turn be modified. For example, if we execute (ADD 'NOT (FINDVAL X 'COLOR)) then subsequently (FINDVAL X 'COLOR) will return (COLOR (KIND APPLE) RED) as before but a further check, i.e., (FINDVAL (FINDVAL X 'COLOR) 'NOT) yields (NOT (COLOR (KIND APPLE) RED)).

It should be clear from the above discussion, that a search through OWL memory is made by tracing down pointers. The proper order in which to search for properties such as NOT, which must be checked,

are not decided by the OWL data base functions. It is a premise of the OWL system design that even if there are 10,000 items in memory representing a particular situation, a good model of the world implicit in LISP and OWL procedures will allow any particular fact to be located quite quickly.

The function call (FINDVAL X Y) returns the item which has Y in first position and X in second position. The third position of the item may be missing or contain anything. Often the user will know that there is only one such item in memory. If there is, in fact, more than one, OWL will return the first one it comes to. To get all such items one uses the function call (FINDVALS X Y). The call (FINDOBJS X Y) finds all OWL items with Y in first position and X in third position. (FINDALL X Y) finds all with Y in first position and X in any other position and (FIND-SERVES-AS-ARGUMENT X) finds all items with X in any position except the first.

Definition of the procedure PUT-ON-TOP-OF

We have mentioned procedures, but we have not yet shown how to define one in OWL. There are two forms of procedure in OWL; the METHOD, and the SCENARIO. We will defer discussion of SCENARIO's until the next chapter. Here, we will demonstrate the definition of METHOD using the blocks world of Winograd.

Figure 1.5 shows a scene familiar to M.I.T. Artificial Intelligence Laboratory workers.

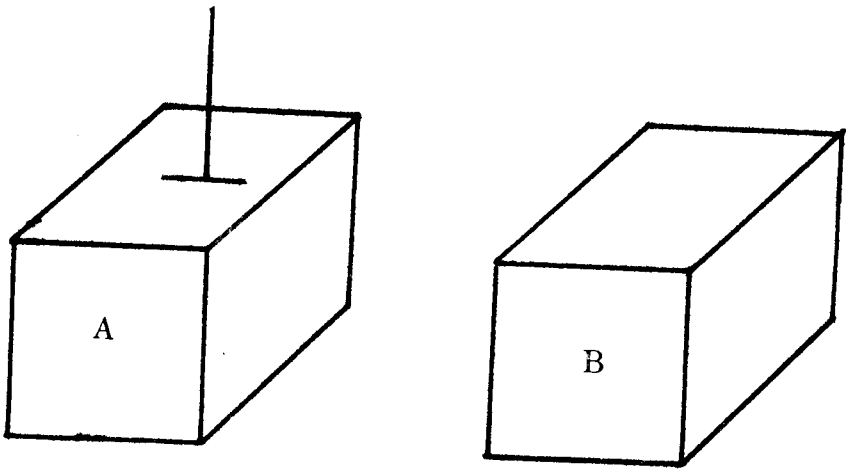


Figure 1.5

A typical blocks world scene.

A simulated hand is preparing to put block A on top of block B. In order to do this the machine must go through a well defined series of steps, which can be conditional on the details of the situation). We will define this series of steps with the METHOD procedure PUT-ON-TOP-OF. OWL keeps separate pointers to procedure definitions. We tell OWL about a definition with DEFINE.

```
(LEARN (DEFINE PROCEDURE
        (PUT-ON-TOP-OF BLOCK-1))
```

Here we have used the short hand BLOCK-1 for (TOKEN (KIND BLOCK) BLOCK-1). This is expanded by LEARN on input. As the OBJECT semantic-case of PUT-ON-TOP-OF, the OWL item bound to BLOCK-1 on execution will specify the object to be moved. The use of (KIND BLOCK) means that the OBJECT

must be a kind of block or the semantic type checking in the OWL interpreter will reject the argument in the call. Of course, procedures must have more than one argument. Additional arguments are specified in the definition by modifying the definition with semantic-cases and in the call by modifying the call with corresponding semantic cases.

A user must use the cases provided by the OWL system. However, he can define subcases of the given cases using KIND. OWL will use a user defined procedure to understand English and to answer questions involving pattern matching rather than deduction. In order for this to work well, the user procedure must use the objects specified by each semantic-case according to the guidelines to be given below.

PUT-ON-TOP-OF takes three arguments in addition to the OBJECT : who can do it, which hand to use, and where to put it.

```
(LEARN (DEFINE PROCEDURE
        (PUT-ON-TOP-OF BLOCK-1))
(AGENT (PUT-ON-TOP-OF BLOCK-1) PERSON-1)
(INSTRUMENT (PUT-ON-TOP-OF BLOCK-1) HAND-1)
(PART AGENT HAND-1)
(SPECIFIC-POSITION (PUT-ON-TOP-OF BLOCK-1)
        (ON-TOP-OF BLOCK-2))
```

(PART AGENT HAND-1) specifies that the hand used must be a part of the agent. Note we have written AGENT, not PERSON-1. When type checking occurs, the properties of an object are also checked. Thus the hand specified must be a part of the agent. When the agent is bound we find

that PERSON-1 has no properties, but when the instrument is bound we find that HAND-1 has the property (PART AGENT HAND-1). Note that we could have defined this as (PART PERSON-1 INSTRUMENT) or (PART PERSON-1 HAND-1) depending on the evaluation we feel should take place when that case is assigned. Whenever the name of a semantic case occurs in other than first position, the interpreter looks up the binding to make the check. How this is done will be described in the next chapter. When the procedure is called, all the arguments provided in the call are type checked and bound. Unless specifically indicated, only the OBJECT is required. If other arguments are missing, the interpreter will proceed and bind them the best it can (when and if they are needed).

A very important idea in Micro-PLANNER is pattern- directed invocation. This is used to find a procedure which produces a class of results of which a result currently needed might be a member. In OWL the choice of procedures will be under the direct control of the interpreter, but it needs specification of results in order to guide its choice. Thus we will state for PUT-ON-TOP-OF

```
(PRINCIPAL-RESULT (PUT-ON-TOP-OF BLOCK-1)
        (POSITION OBJECT SPECIFIC-POSITION))
```

The complete definition of PUT-ON-TOP-OF is shown in Figure 1.6.

```
(LEARN (DEFINE PROCEDURE (PUT-ON-TOP-OF BLOCK-1))
       (AGENT (PUT-ON-TOP-OF BLOCK-1) PERSON-1)
       (INSTRUMENT (PUT-ON-TOP-OF BLOCK-1) HAND-1)
       (PART AGENT HAND-1)
       (SPECIFIC-POSITION (PUT-ON-TOP-OF BLOCK-1)
                                  (ON-TOP-OF BLOCK-2))
       (PRINCIPAL-RESULT (PUT-ON-TOP-OF BLOCK-1)
              (POSITION OBJECT SPECIFIC-POSITION))
       (METHOD (PUT-ON-TOP-OF BLOCK-1) (FIND SPACE-1))
       (POSITION SPACE-1 SPECIFIC-POSITION)
       (BENEFICIARY SPACE-1 OBJECT)
       (THEN (FIND SPACE-1 ) (GRASP OBJECT))
       (THEN (GRASP OBJECT)
              (MOVE (INSTRUMENT-1 (GRASP OBJECT))))
       (DESTINATION (MOVE INSTRUMENT-1) POSITION-1)
       (RESULT (MOVE INSTRUMENT-1)
              (POSITION OBJECT SPECIFIC-POSITION))
       (THEN (MOVE INSTRUMENT-1) (LET-GO-OF OBJECT))
       (Y-COORDINATE POSITION-1
              (PLUS 2
                 (Y-COORDINATE (POSITION (OBJECT (FIND SPACE-1))))
                    (MEASURE (HEIGHT OBJECT))))
       (X-COORDINATE POSITION-1
              (X-COORDINATE (POSITION (OBJECT (FIND SPACE-1))))))
```

Figure 1.6

Definition of PUT-ON-TOP-OF

A METHOD of a procedure is a state transition network. The arcs leaving each node are indicated by THEN. Although we don't need it here, the value of THEN can be an AND, OR, XOR, or conditional statement. (In reading the METHOD, don't forget that within a LEARN, identical list structure is translated into a single OWL item.) Because OWL is derived from English, we can read the METHOD for PUT-ON-TOP-OF out in words. Find space on top of block-2 for block-1; then grasp block-1: then move the hand used to grasp block-1 to a position the y-coordinate of which is the plus of 2, the y-coordinate of the position of the space which was found, and the measure of the height of block-1, and the x-coordinate of which is the x-coordinate of the position of the space which was found; then let go of block-1. The result of moving the hand used to grasp block-1 is that the position of block-1 is on top of block-2. The statement (MOVE (INSTRUMENT-1 (GRASP OBJECT))) needs some explanation. First INSTRUMENT-1 is just a short hand for (TOKEN (INSTRUMENT (GRASP OBJECT))INSTRUMENT-1). The expression (INSTRUMENT (GRASP OBJECT)) means by convention find the most recent (GRASP OBJECT) OWL item and return the instrument of it. This instantiation will occur when the interpreter executes the second step of the procedure. It turns out to be convenient to reference the arguments of previous events as we are doing here.

The statement (RESULT (MOVE INSTRUMENT-1) (POSITION OBJECT SPECIFIC-POSITION)) says that after the hand is moved as indicated, block-1 will be on block-2. Note, that because the procedure PUT-ON-TOP-OF is the one which is planning the move, it is the one which knows what result will be at the PUT-ON-TOP-OF level and must be noted before LET-GO-OF is called because, otherwise, LET-GO-OF will refuse to release

```
block-1.
```

Prerequisites and TESTS

Figure 1.7 shows the definition of GRASP.

```
(LEARN (DEFINE PROCEDURE (GRASP BLOCK-1))
 (AGENT (GRASP BLOCK-1) PERSON-1)
 (INSTRUMENT (GRASP BLOCK-1) HAND-1)
 (PART AGENT HAND-1)
 (PRINCIPAL-RESULT (GRASP BLOCK-1) (HOLD OBJECT))
 (PREREQUISITE (GRASP BLOCK-1) (AND
        (POSITION INSTRUMENT (ON-TOP-OF OBJECT))
        (NOT (POSITION BLOCK-2 (ON-TOP-OF OBJECT))))))
```

One notes immediately that the procedure has no method. There will be a limit to the level of detail to which we want to go. By leaving out the method, we state that if all conditions for the procedure have been satisfied, the principal result can be declared. Before execution of the METHOD of a procedure can be started, its PREREQUISITE's must be satisfied. Since it is possible to place all manner of restrictions on the arguments of a PROCEDURE, one might wonder why PREREQUISITE's are needed. For example, instead of defining GRASP as we have, we could have stated

```
(LEARN (DEFINE PROCEDURE (GRASP BLOCK-1))
   (NOT (POSITION BLOCK-2 (ON-TOP-OF BLOCK-1)))
   (AGENT (GRASP BLOCK-1) PERSON-1)
   (INSTRUMENT (GRASP BLOCK-1) HAND-1)
   (PART AGENT HAND-1)
```

(POSITION HAND-1 (ON-TOP-OF OBJECT))

(PRINCIPAL-RESULT (GRASP BLOCK-1) (HOLD OBJECT))).

This way the procedure cannot even be invoked unless block-1 has no other block on top of itt and the hand is on top of block-1. The point of PREREQUISITE's is that we may want to try to actively satisfy them if all the argument binding goes through all right. In doing this we are following the work of Sussman.

A SPACE is like a physical object except that there is nothing in it. Its size and position are described as if it were a physical object. It is convenient to define a TEST for (OCCUPY SPACE). If the conditions given by the TEST are satisfied, the condition (OCCUPY SPACE) holds. (AGENT (OCCUPY SPACE-1) BLOCK-1) is tested by comparing the coordinates of SPACE-1 and BLOCK-1.

Definition of Characteristics

Some words in English, such as bachelor, can be defined rather well in terms of other words, while others, such as smell, cannot. In expert systems we well be dealing with a larger proportion of defined words and in particular with defined characteristics. For example, it is convenient to think of the profit of a company or the fever of a patient as characteristics useful in diagnosing them. In OWL, we define the TEST of a characteristic in a way analogous to the TEST of a procedure, except that a characteristic can have a VALUE. For example, a definition of fever is given below.

(LEARN (DEFINE CHARACTERISTIC (FEVER HUMAN-1))

```
(TEST (FEVER HUMAN-1) (GREATER-THAN
(ORAL-TEMPERTURE HUMAN-1) 98.6)))

(LEARN (DEFINE CHARACTERISTIC
                (ORAL-TEMPERTURE HUMAN-1))
(VALUE (ORAL-TEMPERATURE HUMAN-1)
       (MEASURE TEMPERTURE HUMAN-1)))
(POSITION (TEMPERATURE HUMAN-1) (IN MOUTH-1))
(PART HUMAN-1 MOUTH-1))
```

Semantic cases for arguments to procedures

For specifying arguments to procedures, OWL allows the following semantic cases. The OBJECT is the most important and the AGENT is the second most important. For alternative suggestions about cases, see Fillmore, Celie-Murcia, Chafe, or Drake. We give sentences to help the reader understand the semantic functions of each case.

OBJECT: The object is the most important thing.

I sneezed.

Give a ball to John.

Empty the boat of water.

The ball lay in the corner.

Make a statue from clay.

Notice the ball was red.

The bill came to us.

AGENT: The agent is responsible for the action.

I hit the ball.

I lay down before dinner.

For certain intransitive actions, the choice between agent and object constructions is not clear cut. The user can use his judgment, remembering that the AGENT should be used when it is useful to treat the item as if it was responsible. Of course, an item can be both the AGENT and the OBJECT. This might require a reflexive pronoun in English, or it might be implied as in "I lay down before dinner".

SOURCE: The source describes a former position or state. When speaking of physical objects with the verbs give, take, put, send, receive, lose, emit, use, and combine, the source is the previous location of the OBJECT. For the verbs experience, reach, and hit, it is the location of the AGENT. For the verbs make, form, break, and change, it is the previous state of the OBJECT. For the verbs get-out and knock-out, it is something the OBJECT is constrained by.

Take a block from the box.

See the parade from the hill.

Make a toy from old milk cartons.

Get fat from over eating.

Knock the leg out from under the table.

DESTINATION: The destination is to the future as the source is to the past. As does the source, it takes subcases which emphasize constraints.

Give a ball to John.

Play John a game of tennis.

Go home.

TRAJECTORY: This is the trajectory taken by the action.

Run across the street to the store.

Run across.

Run out.

Run around the house.

Come by way of the Panama Canal.

DIRECTION:

Head south across the desert to Mexico.

TARGET:

Shoot the gun at the rabbit.

Throw the ball at John.

SPECIFIC-LOCATION: Some verbs like ride take a location which is quite specific to the meaning of the verb.

Ride in a dog sled in Alaska.

There are only certain things which can be ridden in or places where a car can be kept.

INSTRUMENT: This is something used as a tool in the procedure which is left over afterward.

Cut the butter with a knife.

CO-AGENT: This is someone who has a responsibility level equal to that of the agent for some part of the action.

I played tennis with John.

SPECIFIC-RAW-MATERIAL: This is something used and consumed in the procedure.

Bake a cake with powdered eggs.

BENEFICIARY:

John is on a ladder. Answer the phone for him.

The beneficiary receives the benefit of the procedure.

OBJECT-DESCRIBED: This is used for mental and communication procedures.

Talk about a bear.

Go on at length about a bear.

Think about that.

EXCHANGE: This is something received in exchange.

Trade seeds for food.

Pay five dollars for a hat.

RATE:

Rent a room at three dollars a day.

Walk at a fast pace.

DETAIL-OF-METHOD: This is advice on how to do the method which can only be understood properly by storing information about it.

Match it in size.

Buy the dress on credit.

Work it out in detail.

Talk in a whisper.

Get the television on trial.

The METHOD-OF-TRAVEL is a subcase of the DETAIL-OF-METHOD.

Go by train.

Arrive by mail.

Each of these semantic cases except the OBJECT, INSTRUMENT, TARGET and DETAIL-OF-METHOD can also be used to modify objects.

A painting by Durer. AGENT

The man from Havana. SOURCE

A letter to my parents. DESTINATION

A path through the woods. TRAJECTORY

The road south. DIRECTION

The leaves on the tree. SPECIFIC-LOCATION

The soft drink with sugar. SPECIFIC-RAW-MATERIAL

The cookie for John. BENEFICIARY

The book about a bear. OBJECT-DESCRIBED

The money for our food. EXCHANGE

A room at three dollars a day. RATE

Advice to the OWL Interpreter

In the next chapter we will see how the OWL interpreter interprets OWL procedures. As the procedures are relatively abstract, it is best to think of them more as plans, with the interpreter filling in the details in an intelligent manner. It is sometimes convenient to give the interpreter advice on how to do this, and also to prescribe when, where, and for how long the procedure should be carried out. Note that, in general, advice requires more understanding on the part of the interpreter than argument matching.

METHOD-OF-ACCOMPLISHMENT: This is a method of accomplishing at least part of the procedure.

Answer her with a look.

Reward him with a kick.

Finish it with a flick of his wrist.

Do it by tying the magnet to the string.

MANNER: This is a special constraint, attitude, or mannerism the agent should follow while executing the procedure.

Hit the ball with great force.

Discipline her with mercy.

Do it with the arguments reversed.

Do it with one hand behind your back.

COMPARISON: This is another object or action from which the interpreter should be able to get a useful analogy for doing the procedure.

Run like a deer.

Answer like you know what you are talking about.

POSITION:

Manufacture that item stateside.

Type that out at home.

Remain in that building on the ninth floor at the end of the hall.

Paint the boat from stem to stern.

TIME:

Hit the ball from morning to night.

Hit the ball in 20 minutes.

Hit the ball tomorrow.

Hit the ball during the morning.

QUANTITY:

Print them out by the hundreds.

DURATION:

Try to solve that for 5 minutes.

Run for 100 yards.

In milking the cow, use both hands for the first buckets.

The duration may be measured in time or in some repetitive aspect of the procedure.

DEGREE-OF-COMPLETION:

Finish that at least half way.

Don't partly finish the work.

REPETITION-RATE:

Run twice a week.

Run weekly.

Run every week.

FREQUENCY:

Never do that.

Do that frequently.

Always do that.

Properties of sets and quantification

Consider the sentence:

Two of the three eggs are brown.

This would be represented in OWL as

(COLOR (KIND (KIND EGG)) BROWN)
(QUANTITY (KIND (KIND EGG)) TWO)
(MEASURE (QUANTITY (KIND EGG)) 3)
(*SELECTION* (KIND EGG) THE)

Suppose the sentence is given to OWL and then we ask

Is one of the three eggs white?

(WHETHER (COLOR (KIND (KIND EGG)) WHITE))
(QUANTITY (KIND (KIND EGG)) ONE)
(MEASURE (QUANTITY (KIND EGG)) 3)
(*SELECTION* (KIND EGG) THE)

The first step in answering this question is to identify the (KIND EGG) set with the one above. The question then is whether there is a one egg subset of that set which is white. There are two ways which OWL can go about trying to answer this question. The first, known as the extrinsic method, is to form each of the one egg subsets of this set and then try to prove it white. Since nothing is known about an individual egg this approach gets us nowhere. The best we can do is to observe that since a

subset of two is known to be brown, the individual egg is probably brown (with probability 2/3). The second method, known as the intrinsic method, is what we want. This method would reason about the sets rather than enumerating the three egg set. Noting that a subset of two has been declared brown, the heuristic that if one is left out it is probably different would be employed. Checking the other possible colors for eggs yields only white, and thus we conclude that it is probably white.

It is frequently the case that OWL must engage in the intelligent manipulation, rather than the execution, of its statements. For this reason it is more important that the statements are easy to match as patterns and to reason about than that they can be executed by a simple interpreter. Also, OWL often receives statements where the scope of the quantifier is ambiguous: We do not want the notation of OWL to force an early commitment to alternatives; hence, we do not show the scope of quantifiers. Ambiguities must be resolved by adding additional properties. Let us give the properties of sets, and then we can show some examples.

The properties of sets useful in dealing with sets and subsets are:

PROPERTY	VALUES
ORDINAL	FIRST, SECOND,...,NEXT,LAST
FRACTION	FEW MOST
QUANTITY	ONE TWO FEW SEVERAL MOST

SELECTION EACH ALL ANY NO THE A SOME ENOUGH

OTHER

ONLY

SIMILARITY SAME SIMILAR DIFFERENT

Consider the sentence:

No two stores sell the same kind of items.

In OWL this would be

(AGENT (SELL (KIND ITEM)) (KIND STORE))
(QUANTITY (KIND STORE) TWO)
(SIMILARITY (KIND ITEM) SAME)
(SELECTION (KIND STORE) NO)
(*SELECTION* (KIND ITEM) THE)

Upon finding the similarity property the OWL interpreter must know to consider a plural subject. Next, consider

The President has been married since 1945.

Here, it is not clear whether it is the current president who has been married since 1945, or whoever has been president. This would be ambiguous in OWL also. It can be cleared up by adding to the description.

The current President has been married since 1945.

President Nixon has been married snce 1945.

A common operation in programming is to specify that an operation be done for each member of a set.

Print all the colors of each big rock.

In OWL this would be

```
(PRINT (COLOR (KIND ROCK)))
(SELECTION (COLOR (KIND ROCK)) ALL)
(DEMONSTRATIVE (COLOR (KIND ROCK)) THE)
(SELECTION (KIND ROCK) EACH)
(SIZE (KIND ROCK) BIG)
```

Finally, consider:

Every boy either loves Santa or hates him.

This translates to

```
(AGENT (XOR (LOVE SANTA) (HATE SANTA)) (KIND BOY))
        (SELECTION (KIND BOY) EACH)
```

Comparison

Comparison allows the illustration of the modification of the values of characteristics. In OWL we do not consider

John is fatter than Bob.

to be equivalent to

Bob is thinner than John.

Rather, we represent these

(FATNESS JOHN FAT-1)

(FATNESS BOB FAT-2)

(GREATER FAT-1 FAT-2) and

(FATNESS BOB THIN-1)

(FATNESS JOHN THIN-2)

(GREATER THIN-1 THIN-2).

The sentence John is fatter. would become

(FATNESS JOHN FAT-1)

(GREATER FAT-1)

The sentence

Mary is as red as a beet. would become

(COLOR MARY RED-1)

(COLOR BEET-1 RED-1)

The sentence

The men all approve from the oldest to the youngest.

would become

(AGENT (APPROVE) MAN-1)

(SELECTION MAN-1 ALL-1)

(EXTENT ALL-1 OLD-1 YOUNG-1)

(GREATEST OLD-1)

(GREATEST YOUNG-1)

Frames

We will designate a frame to be the contents of a single LEARN. Since buy, sell, and pay all describe the same underlying action from different points of view it is interesting to define a TEST for them in the same frame. This way they can share parts in common. Notice that the destination of sell is the agent of buy. By sharing this part in common this transformation can be made to fall out naturally. The definition is given in Figure 1.7.

```
(LEARN (DEFINE PROCEDURE (BUY-1 PHYSICAL-OBJECT)) (AGENT BUY-1 (KIND
PERSON BUYER)) (SOURCE BUY-1 SELLER) (EXCHANGE BUY-1 MONEY-1) (EXAMPLE
MONEY-1 CREDIT-1) (EXAMPLE MONEY-1 CASH-1) (DETAIL-OF-METHOD BUY-1 (ON-1
CREDIT-1)) (PREREQUISITE BUY-1 (AND-1 (OWN-1 PHYSICAL-OBJECT-1)
        (POSITION MONEY-1 (AT BUYER)))) (AGENT OWN-1 SELLER) (PRINCIPAL-
RESULT BUY-1 (AND-2 (OWN-2 PHYSICAL-OBJECT-2)
        (POSITION MONEY-1 (AT SELLER))) (AGENT OWN-2 BUYER) (DEFINE
PROCEDURE (SELL-1 PHYSICAL-OBJECT-1)) (AGENT SELL-1 (KIND PERSON
SELLER)) (DESTINATION SELL-1 BUYER) (EXCHANGE SELL-1 MONEY-1) (DETAIL-
OF-METHOD SELL-1 ON-1) (PREREQUISITE SELL-1 AND-1) (PRINCIPAL-RESULT
SELL-1 AND-2) (DEFINE PROCEDURE (PAY-1 MONEY-1)) (AGENT PAY-1 (KIND
PERSON)) (DESTINATION PAY-1 SELLER) (EXCHANGE PAY-1 PHYSICAL-OBJECT-1)
(PREREQUISITE PAY-1 AND-1) (PRINCIPAL-RESULT PAY-1 AND-2))
```

Chapter 2

The OWL control structure, self awareness, and explanation

Susie Software is a system that we are currently developing in OWL which will be expert in writing blocks programs. Since techniques for writing programs in the blocks world have been explored by Sussman and Fahlman, we are able to concentrate on the issues of dialogue and the underlying OWL system.

Figure 2.1 shows a hypothetical dialogue which Susie Software will be able to have with a user. At this moment (April 1974) only the first few lines of dialogue are operational, but this will let us explain quite a bit about the OWL system and the OWL interpreter.

A fundamental principle of OWL is that the entire console session is a procedure defined in OWL by the subsystem builder. Any line the user types at the console which is not defined as a next step in that procedure is rejected. (Whenever this happens, OWL will try to give a prompting comment to get the user "back on track.") It is only by having this strong model of the conversation that the OWL system is able to understand user sentences.

Hello, I am Susie. What is your name?

* Bill.
* I need a program to manipulate blocks written.

I can help you solve the problem of writing a program which builds a scene using the blocks from another scene.

* That is what I want.

Ok. I need a description of the initial and final scenes.

* The final scene contains 3 blocks: B1, B2, and B3.

Ok.

* What else do you want to know about the final scene?

What are the positions of B1, B2 and B3 with respect to each other?

* B1 is on top of B2 and B3 is on top of B1.

Ok.

* Initially, all of them are on the ground.

Do you mean in the initial scene?

* Yes.
* Can you write the program from that?

I have enough information to start writing the program. I can't determine all of the information which I will need until I write the program.

* Ok.
* Start writing it.

Do you want me to decide implementation issues such as what hand to grasp a block with?

* Yes, if I can ask you about them later?

You can. Is there a block on top of B2 in the initial scene?

* The initial scene only contains B1, B2 and B3.

I see. I have finished the program. Do you want me to describe the procedure?

* Yes.

First I move B2 with my left hand, then I put B1 on top of B2 with my left hand, then I put B3 on top of B1 with my left hand.

* How did you decide to use your left hand to move B2?

I needed to choose a hand with which to grasp B2. Both of my hands are free at that point in the problem.. You had told me to decide the implementation issues. I chose the first hand which I thought of.

Figure 2.1

A hypothetical dialogue with Susie Software.

```
(LEARN (DEFINE PROCEDURE (SUSIES-TOP-LEVEL-PROCEDURE))
       (AGENT (SUSIES-TOP-LEVEL-PROCEDURE) SUSIE)
       (METHOD (SUSIES-TOP-LEVEL-PROCEDURE)
               (HAVE CONSOLE-SESSION))
       (PURPOSE (HAVE CONSOLE-SESSION)
                (OR (ASK-FOR (CHARACTERISTIC AGENT))
                    (HELP (SOLVE (PROBLEM
                                  (WRITE PROGRAM-1))))))
       (AGENT (SOLVE (PROBLEM (WRITE PROGRAM1))) CO-AGENT)
       (AGENT (ASK-FOR (CHARACTERISTIC AGENT)) CO-AGENT)
       (QUANTITY SCENE-1 ONE)
       (PURPOSE PROGRAM-1 (BUILD SCENE-1))
       (METHOD-OF-ACCOMPLISHMENT (BUILD SCENE-1) (USE BLOCK-1))
       (QUANTITY SCENE-2 ONE)
       (SOURCE BLOCK-1 SCENE-2))


(LEARN (DEFINE PROCEDURE
               (HAVE (KIND SESSION CONSOLE-SESSION)))
       (AGENT (HAVE CONSOLE-SESSION) VERBALIZER-1)
       (CO-AGENT (HAVE CONSOLE-SESSION)
                 (KIND PERSON USER))
       (MUST USER)
       (PURPOSE (HAVE CONSOLE-SESSION)
                (OR (HELP (SOLVE PROBLEM-1))
                    (ASK-FOR (CHARACTERISTIC AGENT))))
       (AGENT (SOLVE PROBLEM-1) CO-AGENT)
       (AGENT (ASK-FOR (CHARACTERISTIC AGENT)) CO-AGENT)
       (SCENARIO (HAVE CONSOLE-SESSION) (TALK-1))
       (PURPOSE TALK-1
                (AND (TELL (LOCAL-QUOTE (NAME AGENT)))
                     (ASK-FOR (LOCAL-QUOTE
                                (NAME CO-AGENT)))))
       (DESTINATION (ASK-FOR (LOCAL-QUOTE (NAME
                                           CO-AGENT)))
                    CO-AGENT)
       (THEN TALK-1 (TALK-2))
       (THEN TALK-2 (TELL GOODBYE))
       (AGENT (TELL GOODBYE) CO-AGENT))
```

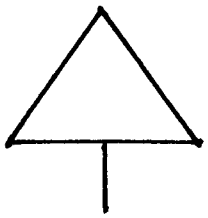
Figure 2.2

```
(LEARN (DEFINE PROCEDURE (TELL SOMETHING))
       (DESTINATION (TELL SOMETHING) PERSON-1)
       (AGENT (TELL SOMETHING) PERSON-2)
       (METHOD (TELL SOMETHING)
               (XOR (IMPLIES (NOT (KIND
                                   OBJECT
                                   'PROCEDURE))
                             (SAY (LOCAL-QUOTE
                                   (IS
                                    OBJECT
                                    (VALUE OBJECT)))))
                    (IMPLIES (KIND OBJECT 'PROCEDURE)
                             (SAY OBJECT)))))

(LEARN (DEFINE PROCEDURE (ASK-FOR SOMETHING))
       (AGENT (ASK-FOR SOMETHING) PERSON-1)
       (DESTINATION (ASK-FOR SOMETHING) PERSON-2)
       (SUB-PROCEDURE (SCENARIO (ASK-FOR SOMETHING)
                                (ASK-AND-ANSWER OBJECT))
                      (ASK-AND-ANSWER SOMETHING-1))
       (AGENT (PERTAIN) SOMETHING-1)
       (DESTINATION (PERTAIN) OBJECT))

(LEARN (DEFINE PROCEDURE (ASK-AND-ANSWER SOMETHING))
       (METHOD (ASK-AND-ANSWER SOMETHING) (ASK OBJECT))
       (THEN (ASK OBJECT) (SAY (KIND SOMETHING ANSWER)))
       (AGENT (SAY ANSWER) DESTINATION)
       (DESTINATION ANSWER OBJECT)
       (THEN (SAY ANSWER)
             (UNDERSTAND (IS (OBJECT (SAY ANSWER))))))
```

Figure 2.3

The top level procedures for Susie Software are shown in Figure 2.2. Executing the LISP call (PERSONIFY 'SUSIE) will cause OWL to look for what Susie does and, since she does just one thing, to do it. The METHOD of Susie's procedure is simply to have a console session. As can be seen in the bottom of Figure 2.2, the procedure (HAVE (KIND SESSION CONSOLE-SESSION)) does not have a METHOD, but rather a SCENARIO. A SCENARIO differs from a METHOD in that it is not strictly procedural; it may have various subprocedures, but these are not organized into a whole. In order to give a SCENARIO direction, we must give it a PURPOSE.

In order to explain our notion of PURPOSE we will use an example from Goldstein. Figure 2.4 shows a line drawing, a program to draw it, and an interpretation of that program.



Column 1	Column 2	Column 3
FORWARD 10	DRAW SIDE	
RIGHT 120	SETUP FOR SIDE	
FORWARD 10	DRAW SIDE	DRAW TOP
RIGHT 120	SETUP FOR BOTTOM	
FORWARD 10	DRAW BOTTOM	
RIGHT 180		
PEN UP		
FORWARD 5	SETUP FOR TRUNK	DRAW TRUNK
RIGHT 90		
PEN DOWN		
FORWARD 10	DRAW TRUNK	

Figure 2.4

A tree and procedures for drawing it

at three levels of abstraction

Note that the procedure in column 1 has been described in column 2 by another procedure. Whereas in MICRO-PLANNER and CONNIVER as well as in OWL METHODS we describe a procedure by the pattern of its result, here we are describing a procedure by anotner procedure at a higher level of abstraction, which in turn is described by yet a higher level procedure. There is a many to one mapping between the steps of a lower level procedure and the next higher one. This use of a procedural description of the lower level procedure is important for understanding it, but in this simple example the result can be described non-procedurally as shown. It is well known, however, that it is very difficult to describe some things non-procedurally, and in this case a procedure is also needed for describing the result of the lower level procedure.

Returning to OWL, we will define a higher level procedural description of a procedure P to be a PURPOSE of P. A SCENARIO is a collection of subprocedures and associated knowledge which lacks a procedural structure. This loose body of facts can be used in partial fulfilment of many different PURPOSE's. By mapping the individual steps of the SCENARIO into the steps of some PURPOSE, we give procedural structure to the SCENARIO.

Want, Events and Conditions

It seems funny to talk about a purpose unless you want something. Figure 2.5 shows increasing degrees of precision in specifying a want. (This classification results from Charniak and discussion with Gretchen Brown).

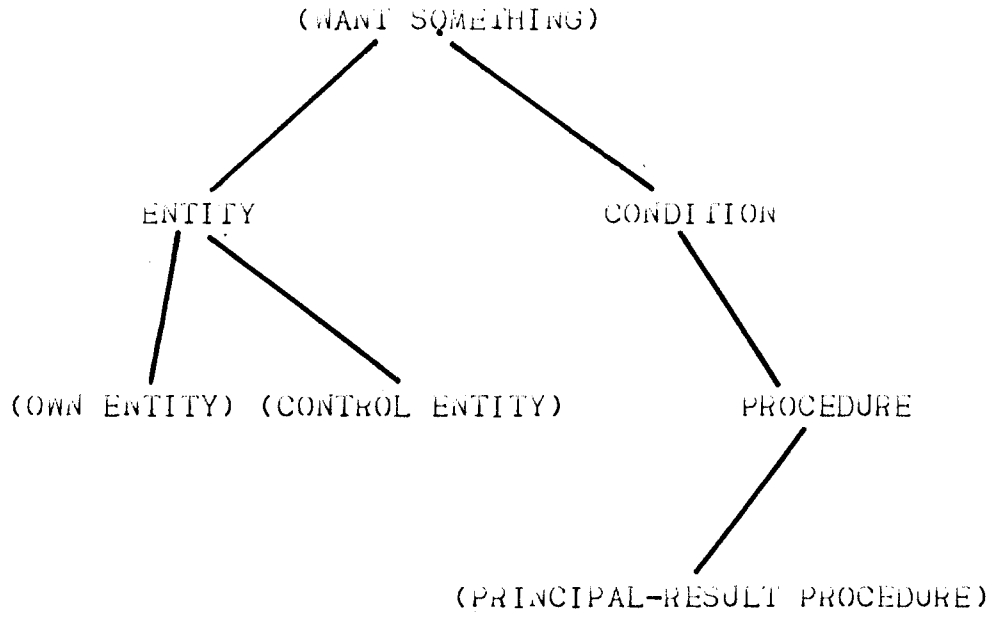


Figure 2.5

Specifying a want.

If we want something, it could be an entity like a ball or a company or a condition like brown hair, to be going to the

circus, or MUMPS. If we want an entity, it means we want to have it, but this could be either to own it or merely to control it as in:

My father gave me the car for tonight.

Tip has the ball.

Many people feel we should write (WANT (HAVE BALL)), not (WANT BALL). It's true that, at a deeper semantic level, (WANT SOMETHING) means that we want either a condition to obtain that does not now obtain or an event to happen. By syntactic convention, "to have" can usually be left out; thus, (WANT BALL) does, at the deeper level, mean (WANT (HAVE BALL)). Although we have no definitive argument, we feel that HAVE should not be gratuitously inserted into the ONL representation. The knowledge of the deeper interpretation should be an integral part of the interpreter's functioning. In order to work well, the interpreter must be able to think at a high level of abstraction, and this implies having a good model of the world built into it, not into a canonicalized representation.

Everything we know about a thing which is important in characterizing it will be referred to as its PROPERTY's. If we think of a PROPERTY as being true for all time, we will call it a CHARACTERISTIC. Otherwise, it will be known as a CONDITION.

The ONL PROCEDURE's we have been defining might have been called (will be called) ACTIVITY's. An instantiation of an

ACTIVITY is also an ACTIVITY because instantiation is a multi-level process. However, just as we have INDIVIDUAL dogs, we have INDIVIDUAL ACTIVITY's, known as EVENT's.

The ACTIVITY's in which a thing has participated aren't thought of as PROPERTY's except insofar as they are habitual or typical or otherwise important in characterizing that thing. For example,

We sell items to Sears.

gives an activity. We might also consider it a property of us or Sears, although the syntactic subject is emphasized as the choice for the property.

Sears bought a battery from us yesterday.

describes an event. It probably would not be a property.

Cows eat grass.

is an activity and a characteristic of cows.

I have the mumps.

is a condition.

People were smaller in the old days.

is a condition.

We used to sell items to Sears.

is an activity and, as a habitual one, is probably a condition of "us".

We sell an apple to Sears every Monday.

is an activity which is like the last example except that it suggests a set of events.

Whenever the OWL interpreter begins executing a procedure, it establishes an event. The interpreter will not initiate an event unless it wants the event to happen, and since it is driven solely by reason, it will not want any event to happen which is not a subgoal of an event already initiated. The interpreter does an

(ADD 'SET-GOAL-FOR new-event on-going-event).

The interpreter also keeps a list *GOAL-LIST* of all the on-going events.

Now let us see how the interpreter would use the notions of purpose and of wanting a goal in order to answer a question about its actions. Suppose block B is on top of block A and the interpreter tries to execute (GRASP A). It will note the prerequisite of GRASP that A must have nothing on it and want to get (POSITION B (OFF-OF A)). Following Sussman's work it will discover that this instantiates the pattern of the principal result of CLEAR-OFF. (CLEAR-OFF A) will be established as a subgoal of (GRASP A) with a PURPOSE of (GET (POSITION B (OFF-OF A))). Now suppose we ask the system,

Why did you grasp B?

It can answer

I wanted to clear off A.

or

I wanted to get B off of A.

The method of answering is to give the next higher level goal, possibly reinterpreted in terms of its purpose. Note how "GET" is used to make the purpose a procedural statement.

Since the console session is interactive, not everything that happens is a goal of the expert system. The user types in statements from the console. To understand these statements the expert system establishes expectations. Whenever execution reaches a point where the next step is for the user to say something, the system establishes an expectation which it puts on *EXPECTATION-LIST*. When the user statement is matched to an expectation, the interpreter declares

(SET-EXPECTATION-FOR user-statement expecting-event).

Sometimes the expectation is found only after the user statement is in hand.

To see exactly what happens, let us return to the example of Susie Software. (HAVE CONSOLE-SESSION) is called with a purpose of helping the user to solve problems of writing programs to build a scene using blocks from another scene. When the interpreter initiates a new event, it makes all of the arguments which are explicitly passed in properties of that event. Those which are free at one level are looked up by tracing back up the goal tree, following the SET-GOAL-FOR chain. The AGENT of the top level event is SUSIE. Since SUSIE will do the thinking, it will only be necessary to remind the AGENT when the user says a

line of console input. In order to achieve the subgoal (HAVE CONSOLE-SESSION), the interpreter looks for a PROCEDURE of the form (HAVE X) where CONSOLE-SESSION is an instantiation of X. If it finds one it sets up a new event and binds the arguments to the event, making sure each argument passed instantiates the value which that argument can have as declared in the procedure. Next, as discussed above, it checks the prerequisites and satisfies them if necessary. It then begins to carry out the method or scenario. A procedure (TALK) is given as the value of the SCENARIO for (HAVE CONSOLE-SESSION). When a value of a SCENARIO is given, it is defined to be the procedure to follow in order to initiate action on the scenario. Although it is not done here, various sub-procedures of the scenario could also be given.

The procedure TALK is in effect a special OWL interpreter which interprets its PURPOSE, but with the important property that it has recourse to user suggestions. When it gets stuck, it knows that a statement of user input can be obtained whenever it types a * on the console. Since OWL is a system for building expert problem solving systems, not for shooting the bull with the user, it wants to keep the discussion focused on the purpose if possible. Therefore, TALK checks the purpose to see if there is an obvious first step to take without getting user input. The purpose passed to TALK is

(AND (TELL (LOCAL-QUOTE (NAME AGENT)))

(ASK-FOR (LOCAL-QUOTE (NAME CO-AGENT)))).

This is an AND, both parts of which have to be done. The agent of both parts is SUSIE. Therefore TALK sets this up as an event and starts the sub-goal (TELL (LOCAL-QUOTE (NAME AGENT))). LOCAL-QUOTE means to evaluate the arguments of its argument, but not to apply the function. Thus, the AGENT is evaluated to SUSIE, and we get (TELL (NAME SUSIE)).

The procedure TELL is shown in Figure 2.3. We see that the method is an XOR, or exclusive or. The interpreter tries its arguments in turn until one is successful. The first argument says that if (NOT (KIND OBJECT 'PROCEDURE)) is true then do (SAY (LOCAL-QUOTE (IS OBJECT (VALUE OBJECT)))). To see whether (NOT (KIND OBJECT 'PROCEDURE)) is true, the interpreter hands it to the OWL system function THINK-THAT.

In programming languages other than OWL, a formal environment mechanism is maintained so that the truth of a statement which is, for example, true at one point in time and not at another can be tested at a point during execution, without the testing process needing to know the semantics of the individual statement being tested. In OWL this is not done. Since OWL is a very high level problem solving language, it is often the case that the truth of a statement cannot be absolutely established, only strongly supported, with the data at hand. What level of evidence is required before the interpreter should

proceed as if the statement is true would seem to depend on global considerations such as the other available alternatives, the resources available to find a solution, the seriousness of the problem being solved, the possibility of reviewing the solution later and changing it, the types of discrepancies in evidence which have been occurring in this environment, and (as mentioned by Carbonell and Thompson) the system's willingness to take lack of knowledge as a proof of negation in a certain topic or its willingness to answer by analogy. Because of these factors, it might be that the same proposition would be taken as true at one point and false at another. In order to cut down on the duplication of effort in investigating a proposition more than once, THINK-THAT builds a case for why the proposition is true or false and returns this case as its result. A function TRUEP is then used to examine the case and make the final decision. If the same proposition has to be investigated a second time, the old case is available as a starting point for THINK-THAT. Note that, as pointed out by Srinivasan, if a proposition is not true, it may be desirable to use the case returned by THINK-THAT as a guide for making it true. This is done by OWL in the satisfaction of PREREQUISITE's.

Another reason for not establishing an environment mechanism such as the CONTEXT mechanism in CONNIVER is that it is very hard to establish a filing system for large chunks of

information before one knows the use of the information. Since it is harder to predict the use of information in a problem solving situation than in most current data processing applications, a CONTEXT mechanism is less useful. Also, as originally conceived, the natural structuring of contexts was tied to the flow of control, but it is not clear that the order in which information is learned is the best way to structure it. For example, consider the process of medical diagnosis. Since a patient's fever and other symptoms may vary from day to day, it is useful to take a series of CONTEXT's through time. A fever can then be high in one of these and low in another without contradiction. Because of the combinatorial nature of medical diagnosis, it is necessary to hypothesize a diagnosis based on a few symptoms and then attempt to verify or refute it. Once a hypothesis is made, all deductions are done assuming it to be true and thus new contexts must be started for each time interval. The situation is as shown in Figure 2.6. The intersection of the various assumptions divide

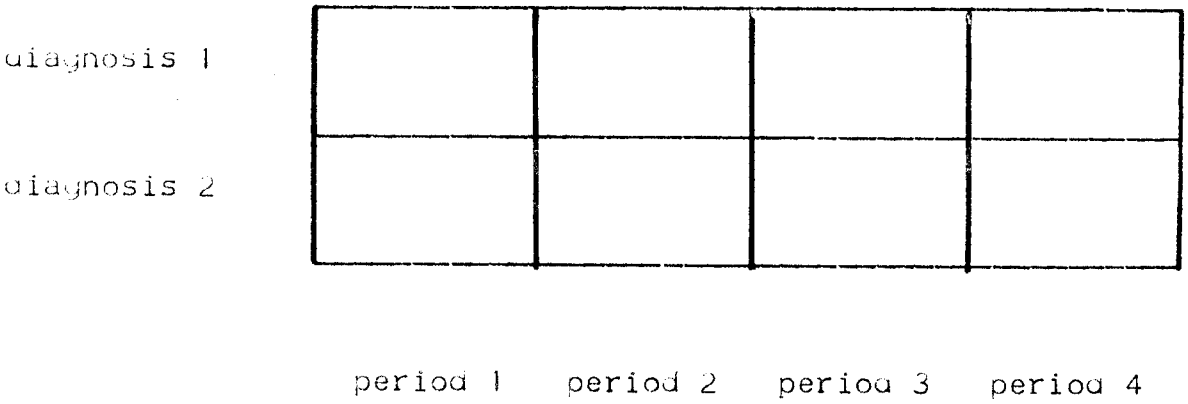


Figure 2.6

the data base into small packets. Suppose that the hypothesis of diagnosis 2 turns out to be false, we will then want to abandon that assumption and its associated CONTEXT's. However, many of the facts we learned in the investigation of diagnosis 2 may be true in spite of the fact the hypothesis is false. In order to save those when we abandon the CONTEXT's, we must examine each fact and place it in an appropriate higher CONTEXT, which is contrary to the very spirit of the context mechanism.

We are fortunate that a number of sophisticated programs have been written in the CONNIVER language. We find, in fact, that in the programs of Sussman and Fahlman, once-created CONTEXT's do not dictate the subsequent flow of program control. For example, there is no direct back-up through previous contexts. Therefore, the notion that we are somehow "in" the

context in which evaluation is done becomes awkward. A program like BUILD or HACKER wants to back off and contemplate its past. We also find that the average CONTEXT contains only a small packet of facts and that if any account is to be made of their validity under changing circumstances, an additional mechanism may be required (belief-rings in McDermott), which allows the explicit statement of their origin. This is precisely what our record of events in intermediate term memory provides us. (We assume that if there are bad misconceptions in the frames of long term memory, the subsystem builder must be called back to pay for his lies). The only data stored in intermediate term memory are events and their properties, which of course, include the results of events. The two most important dimensions for structuring events are taken to be the flow of control and the flow of time. Consequently, as mentioned before, all events are linked by SET-GOAL-FOR and/or SET-EXPECTATION-FOR properties. Furthermore, they are chained together in order of their start and finish times by START-TEMPORAL-SUCCESSOR and FINISH-TEMPORAL-SUCCESSOR. THINK-THAT must use these relations in deciding what facts in the data base apply to the current question.

We have digressed from Susie Software and the TELL procedure of Figure 2.3. Since (NAME SUSIE) is not a kind of procedure, (LOCAL-QUOTE (IS OBJECT (VALUE OBJECT))) is evaluated to (IS (NAME SUSIE) WORD-SUSIE), and this is passed to the OWL

system function SAY, which is responsible for turning it into "My name is Susie." Completion of the SAY completes the TELL and the interpreter begins work on the second clause of

```
(AND (TELL (LOCAL-QUOTE (NAME AGENT)))
     (ASK-FOR (LOCAL-QUOTE (NAME CO-AGENT))))
```

The procedure ASK-FOR has a scenario which will call ASK-AND-ANSWER, the function that asks single questions. (Although we also allow as subprocedures of the scenario additional ASK-AND-ANSWER's whose subject pertains to the question.) In evaluating the argument (LOCAL-QUOTE (NAME CO-AGENT)) The CO-AGENT is evaluated. The interpreter finds that the CO-AGENT has never been bound. On finding this, the interpreter goes back up through the SET-GOAL-FOR's looking at the procedure corresponding to each event until it comes to one which has a CO-AGENT case specified. This will be (HAVE CONSOLE-SESSION). Here we see that the CO-AGENT must be a person who will be called a USER by virtue of his being the CO-AGENT of (HAVE CONSOLE-SESSION). We see further that USER has the characteristic MUST. This means that if the scenario of (HAVE CONSOLE-SESSION) is in progress, then a specific CO-AGENT exists in the real world, although we may not know anything about him. Having discovered all this, the interpreter generates the OWL item (KIND USER) and binds the CO-AGENT of the (HAVE CONSOLE-SESSION) event to it. It thus has to

interpret (ASK-AND-ANSWER (KIND USER))).

The METHOD for ASK-AND-ANSWER something is to first ask the question, so (WHAT (NAME (KIND USER))) is given to SAY. SAY substitutes YOU for the CO-AGENT of (HAVE CONSOLE-SESSION). Following the THEN pointer, the interpreter sees that the next step is for the user to say something, so it puts this step on the expectation list and reads an input statement from the console. In this case, the user's statement is WORD-BILL, and the interpreter tries to ATTRIBUTE this to one of the expectations. It notes that the expectation (SAY (KIND SOMETHING ANSWER)) is part of the plan (ASK-AND-ANSWER (NAME (KIND USER))). Since WORD-BILL is a name value, a match is found. ATTRIBUTE sets up

(IS WORD-BILL (KIND ANSWER))
(DESTINATION (KIND ANSWER) (NAME (KIND USER)))

It then returns (KIND ANSWER). TALK declares

(SET-EXPECTATION-FOR (SAY WORD-BILL)
(ASK-AND-ANSWER (NAME (KIND USER))))

and the interpreter follows the THEN pointer in ASK-AND-ANSWER to (UNDERSTAND (IS (OBJECT (SAY ANSWER)))). To evaluate the argument of UNDERSTAND, the interpreter finds the event (SAY

WORD-BILL) attributed to (SAY ANSWER) and takes its OBJECT, WORD-BILL. From above, the value of IS applied to WORD-BILL IS (KIND ANSWER). (UNDERSTAND (KIND ANSWER)) merely declares (NAME (KIND USER) WORD-BILL) as its PRINCIPAL-RESULT.

At this point ASK-AND-ANSWER and TALK are both complete. Control passes back to HAVE-CONSOLE-SESSION, where the second TALK is executed. This TALK has the purpose

```
(OR (ASK-FOR (CHARACTERISTIC AGENT))
    (HELP(SOLVE (PROBLEM(WRITE PROGRAM-1)))))
    (PURPOSE PROGRAM-1 (BUILD SCENE-1))
    (METHOD-OF-ACCOMPLISHMENT (BUILD SCENE-1) (USE BLOCK-1))
    (QUANTITY SCENE-1 ONE)
    (SOURCE BLOCK-1 SCENE-2)
    (QUANTITY SCENE-2 ONE)
    (AGENT (ASK-FOR (CHARACTERISTIC USER)) CO-AGENT)
    (AGENT (SOLVE (PROBLEM (WRITE PROGRAM-1))) CO-AGENT)
```

Since the top level connective is an OR, TALK does not know how to proceed. It therefore requests a user input statement, which it hopes will prove useful in deciding what to do. The user statement obtained is

```
(SAY (NEED (WRITE PROGRAM-1)))
(AGENT (NEED (WRITE PROGRAM-1)) CO-AGENT)
(QUANTITY PROGRAM-1 ONE)
(PURPOSE PROGRAM-1 (MANIPULATE BLOCK-1)).
```

As before TALK first refers to the expectation list, but now this is empty. It must therefore resort to the CLUE's associated with its SCENARIO. Each SCENARIO can have CLUE's associated with it. These clues suggest actions which the interpreter could take in the context of that SCENARIO. The clues are patterns which indicate it might be advantageous to attempt execution of some subprocedure of the SCENARIO.

```
(MATCH-NAME (DO SOMETHING) DO-SOMETHING)
(BENEFICIARY (CLUE (SCENARIO (TALK))
                   (OR (TELL
                         (WANT
                          (OR
                            (DO SOMETHING)
                            (HELP
                             (DO SOMETHING)))))
                       (ASK-IF
                        (CAN
                         (HELP
                          (DO SOMETHING))))))
             (NEGOTIATE-1 (HELP DO-SOMETHING)))
(PREREQUISITE NEGOTIATE-1
              (AND (AGENT (TELL
                            (WANT
                             DO-SOMETHING))
                          USER)
                   (AGENT (TELL
                            (WANT
                             (HELP
                              DO-SOMETHING)))
                          USER)))
(METHOD NEGOTIATE-1
```

```
(XOR (IMPLIES (KIND
                (GOAL AGENT)
                OBJECT)
              (AND
                (TELL OK)
                (BEGIN OBJECT)))
     (IMPLIES (SUGGEST-AND-ACCEPT
                (GOAL AGENT))
              (BEGIN
                (OBJECT
                  (SUGGEST-AND-ACCEPT
                    (GOAL AGENT)))))))
(KIND OBJECT (GOAL AGENT))
(SELECTIVITY (GOAL AGENT) ANY)
(CURRENT (GOAL AGENT))
```

This says that "do something" is a clue for negotiating to help do it. The prerequisites for negotiation are that the user wants to do it and that he wants us to help do it. If the thing he wants to do is an instantiation of one of our current goals, we say "OK" and begin helping him do it. Otherwise, if we can suggest, and he accepts, a current goal of the agent which is an instantiation of what the user wants to do, we begin helping him do that.

A number of things need to be pointed out. First, note that the form of matching here is to look for an occurrence of the clue in the input. If this is found, then the more general pattern match involving the procedure arguments and prerequisites is attempted. If this is successful, then the METHOD is attempted. This is the form of matching rule used by Moses in his integration program. Second, note the form of the method. The XOR says to try its arguments in order until one succeeds. The second argument

```
(IMPLIES (SUGGEST-AND-ACCEPT (GOAL AGENT))
         (BEGIN (OBJECT (SUGGEST-AND-ACCEPT
                         (GOAL AGENT)))))
(KIND OBJECT (GOAL AGENT))
(SELECTIVITY (GOAL AGENT) ANY)
```

contains the expression (SELECTIVITY (GOAL AGENT) ANY) which says to match this expression for any goal of the agent. The interpreter could do this by picking one and, if that didn't succeed, backing up to try another. This would then be the infamous pure back up mechanism of MICRO-PLANNER!

In CONNIVER Sussman and McDermott suggested that this backup mechanism be replaced by a programmer-specified procedure. With this we heartily agree. However, their idea of including the programmer's search ideas as part of the pattern using the "possibilities list" and "try next" has two disadvantages. First, it makes the pattern difficult to handle when we want to manipulate it, rather than execute it. Second, it implicitly asserts that the primary method of reacting to the failure of a possibility is a more intelligent selection from the remaining possibilities on the list. In fact, a more global reaction is often required. For example, as Sussman says, if a robot has selected the next block to add to a construction and finds the block not, the response may be to find a procedure to cope with

not blocks, rather than to select an alternative block. Similarly, in Sussman's example of conflicting goals to be discussed in the next section, the solution is to reorder actions at a higher goal based on a global understanding of the difficulty. For these reasons we feel it is better to give the interpreter the pattern as it is, and then advise it separately on how to match it, rather than to embed the advice in the pattern.

Finally, observe that this clue and its procedure deal with user statements about how to execute the purpose, rather than user statements which are mapped into elements of the purpose. That is, in the dialogue at this point we are discussing what we are going to do, not doing it.

Let us see in a little more detail how the matching will proceed on this example. (TELL (NEED (WRITE PROGRAM-1))) will match the clue (TELL (WANT (DO SOMETHING))), the event (NEGOTIATE (HELP (WRITE PROGRAM-1))) is set up as a subevent of TALK. To make this match the interpreter assumes that if a person needs something he wants it. DO-SOMETHING is set to (WRITE PROGRAM-1) by the match. Next the prerequisites are checked. THINK-THAT finds the first one satisfied by the same logic used in matching the clue. The second one is true if we assume that expressing a want to Susie is equivalent to expressing a want for help on it. We will make this assumption - if we did not, then TALK would have to ask the user.

The prerequisites satisfied, we enter the METHOD. We attempt to find a current goal of Susie which (NEED (WRITE PROGRAM-1)) instantiates. Unfortunately, this fails because "manipulate blocks" is not an instantiation of "build scene". The interpreter passes to the second IMPLIES. Now the interpreter tries to instantiate (MANIPULATE BLOCK-1) with (BUILD SCENE-1). Looking at the frame for (MANIPULATE BLOCK-1) the interpreter finds:

```
(DEFINE PROCEDURE (MANIPULATE BLOCK-1))
(AGENT (MANIPULATE BLOCK-1)
       (OR PERSON-1 PROGRAM-1))
(EXAMPLE (SCENARIO (MANIPULATE BLOCK-1))
         (OR (PUT-IN BLOCK-2) (BUILD SCENE-1)))
(SPECIFIC-POSITION (PUT-IN BLOCK-1) BOX-1)
(METHOD-OF-ACCOMPLISHMENT (SCENARIO (MANIPULATE BLOCK-1))
                  (OR (MOVE HAND-1)
                      (PICK-UP BLOCK-1)
                      (PUT-ON-TOP-OF BLOCK-1)))
(PART AGENT HAND-1)
(AGENT (HOLD BLOCK-1) HAND-1)
```

Since (BUILD SCENE-1) is an example of the scenario for (MANIPULATE BLOCK-1), the match is accomplished. The function

SUGGEST-AND-ACCEPT then suggests this alternative to the user as shown in the dialogue and sets up an expectation for an acceptance. When this is received, BEGIN sets up the appropriate EVENT.

Protocol Analysis

We have been speaking of the interpreter as if it were a single unit. Actually it is divided into modules, as shown in Fig. 2.7.

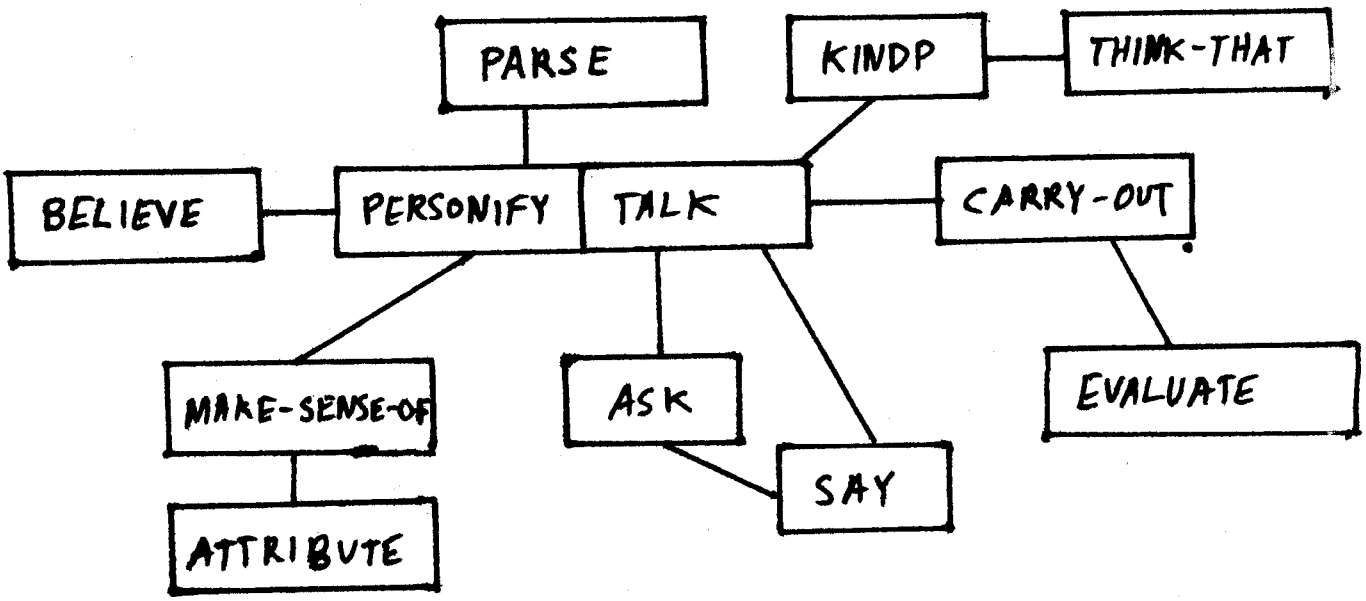


Fig. 2.7

Modules of the OWL interpreter.

All of these modules are written in LISP. The LISP call and return mechanism is used to communicate control information only. The modules find all of their data either as free variables in short term memory or as OWL items in intermediate or long term memory. The function which actually interprets a METHOD or SCENARIO (roughly corresponding to APPLY in LISP) is CARRY-OUT. The data to CARRY-OUT is referred to as a PLAN. Suppose CARRY-OUT were given the PLAN

(PUT-ON-TOP-OF B1)
(SPECIFIC-POSITION (PUT-ON-TOP-OF B1) (ON-TOP-OF B2))

It would look for a procedure (PUT-ON-TOP-OF X), where B1 is a kind of X. Finding this, it would establish a new event and bind the SPECIFIC-POSITION. Next, it would check the prerequisites of the procedure selected.

If no procedure for doing the plan can be found or if no procedure can be found for the plan for which the arguments can be bound or if a prerequisite is not satisfied or if the method of the selected procedure involves a subgoal, CARRY-OUT will return control to PERSONIFY-TALK. PERSONIFY-TALK will then attempt to decide what to do next.

For example, consider the problem discussed by Sussman,

```
(GET (AND (POSITION B1 (ON-TOP-OF B2))
          (POSITION B2 (ON-TOP-OF B3)))).
```

Given such a goal, the interpreter should attempt to plan which argument of the AND should be done first. Following Fahlman, the interpreter would recognize that the positions of objects are a clue to the use of SUPPORT-relation planning. We might have for example

```
(DEFINE PROCEDURE (PLAN (GET (AND POSITION-1))))
(PRINCIPAL-RESULT (PLAN (GET (AND POSITION-1)))
                  (GET (AND KIND-2)))
(SELECTIVITY KIND-1 ANY)
(QUANTITY KIND-1 ONE)
(AGENT (SUPPORT (OBJECT (KIND-2 POSITION-1)))
       (OBJECT KIND-1))
(NOT (SUPPORT (OBJECT (KIND-2 POSITION-1))))
(OTHER (KIND-2 POSITION-1) KIND-1)
(MENTAL-POSITION (SUPPORT (OBJECT
                           (KIND-2 POSITION-2)))
                 (IN (OBJECT OBJECT)))
(THEN (GET (AND KIND-2))
      (GET (KIND-1 POSITION-1)))
```

This says that the principal result of planning to get the AND of some positions is to get a position whose object does not support, as specified by the AND of positions given, the object of any of the other positions after getting the other positions. (Note how OTHER is used to achieve a mutually exclusive partition.)

When CARRY-OUT passed control to PERSONIFY-TALK signaling a new subgoal and the PLAN was

```
(GET (AND (POSITION B1 (ON-TOP-OF B2))
          (POSITION B2 (ON-TOP-OF B3))))
```

then PERSONIFY-TALK would see the AND and attempt to call CARRY-OUT with

```
(PLAN (GET (AND (POSITION  B1  (ON-TOP-OF  B2))
                (POSITION  B2  (ON-TOP-OF  B3))))).
```

Using the procedure above CARRY-OUT would get as the principal result of the PLAN event

```
(GET (POSITION B2 (ON-TOP-OF B3)))
(THEN (GET (POSITION B2 (ON-TOP-OF B3)))
```

(GET POSITION B1 (ON-TOP-OF B2))))

PERSONIFY-TALK would then call CARRY-OUT with this.

Suppose, though, that the above procedure for planning positions were not known. Then CARRY-OUT would have to signal PERSONIFY-TALK that it could not find a procedure. In this case PERSONIFY-TALK would give CARRY-OUT the original AND plan and CARRY-OUT would take the subgoals of the AND in the order written. First (POSITION B1 (ON-TOP-OF B2)) would be obtained, and then (PUT-ON-TOP-OF B2) would be called with the purpose (GET (POSITION B2 (ON-TOP-OF B3))). This would result in a subgoal of (GRASP B2). Now (POSITION B1 (ON-TOP-OF B2)) violates the prerequisite of (GRASP B2). Thus (GET (POSITION B1 (OFF-OF B2))) is passed to PERSONIFY-TALK. At this point the tree of subgoals is as shown in Fig. 2.8.

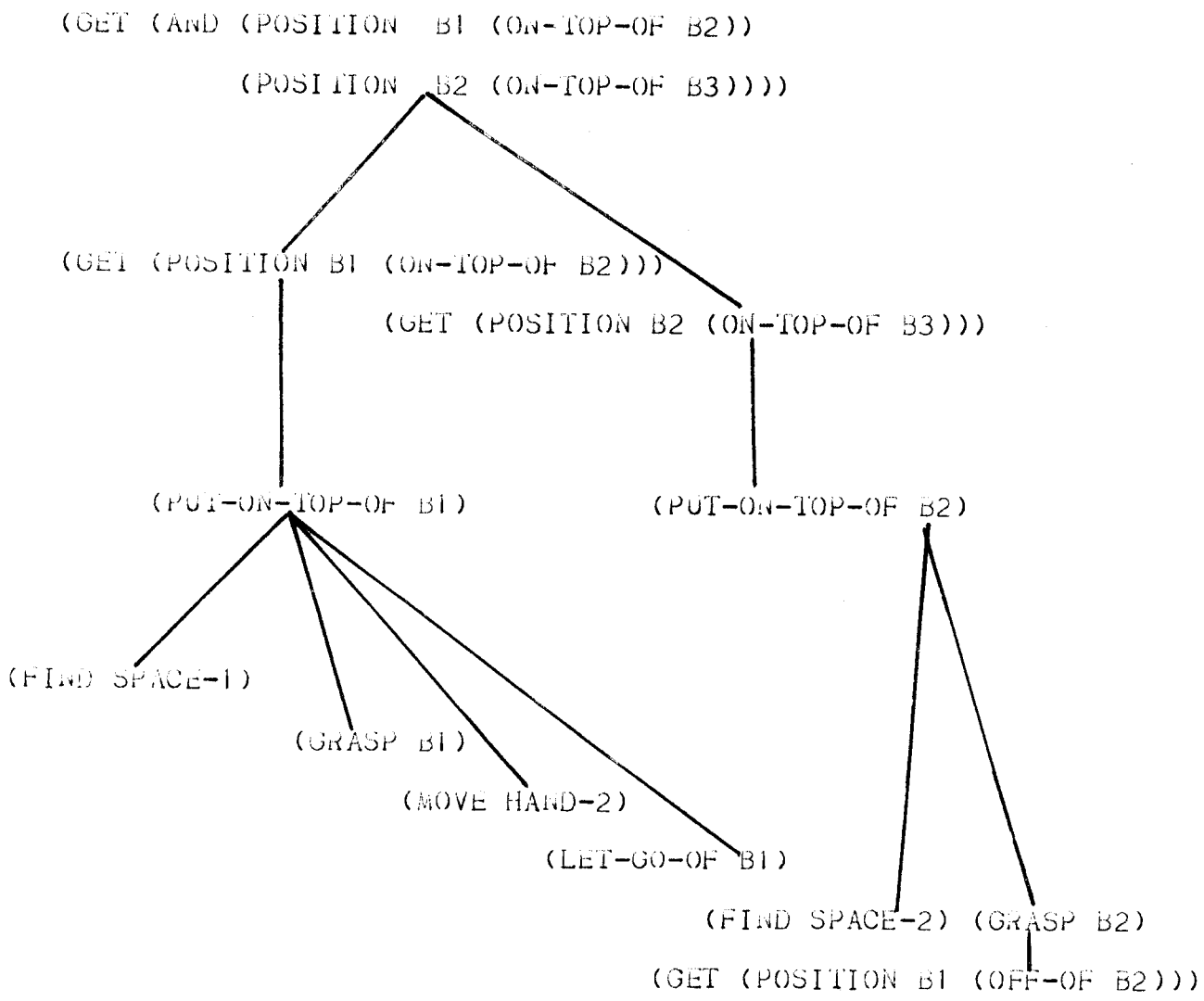


Fig. 2.8

Goal tree for building a three block stack.

Before moving B1, PERSONIFY-TALK will check to see if the current position is of any importance. It will look for the event which established the current position. It will find that (POSITION B1 (ON-TOP-OF B2)) is the PRINCIPAL-RESULT of (PUT-ON-TOP-OF B1), which was done for the PURPOSE of establishing the position. It may be difficult in general to establish when a position can be changed, but certainly one established "on purpose" has to be looked at closely. PERSONIFY-TALK will give CARRY-OUT

```
(RESOLVE CONFLICT-1)
(MENTAL-POSITION
 CONFLICT-1
 (BETWEEN
  (AND
   (GET (POSITION B1 (ON-TOP-OF B2)))
   (GET (POSITION B1 (OFF-OF B2))))))
```

where the two GET's refer to the events. As pointed out by Sussman, there are a number of cases which must be considered by (RESOLVE CONFLICT). Here we will only develop the case in point. CARRY-OUT will find the definition

```
(DEFINE PROCEDURE (RESOLVE CONFLICT-1))
(MENTAL-POSITION CONFLICT-1
                 (BETWEEN (AND EVENT-1 EVENT-2)))
(MATCH-NAME EVENT-4 E4)
```

```
(AND (KIND (PREREQUISITE (SUPERIOR EVENT-2))
           (OBJECT EVENT-2))
     (SUBGOAL EVENT-4 EVENT-1)
     (SUBGOAL EVENT-4 EVENT-2))
(SELECTIVITY EVENT-4 ANY)
(KIND (GET (AND CONDITION-1)) EVENT-4)
(START EVENT-2 (AFTER (START EVENT-1)))
(MATCH-NAME EVENT-2 E2)
(PRINCIPLE-RESULT (RESOLVE CONFLICT-1)
                  (PROCEDURE E4
                             (GET
                              (OBJECT EVENT-5))))
(THEN (GET (OBJECT EVENT-5))
      (GET (AND CONDITION-3)))
(OTHER (OBJECT EVENT-5) CONDITION-3)
(SELECTIVITY EVENT-5 THE)
(SUBGOAL EVENT-5 E2)
(SUPERIOR EVENT-5 E4))
```

This procedure will resolve the conflict between event-1 and event-2, where event-2 has as its object a prerequisite of its superior and event-2 occurs after event-1 and both are subgoals of an event E4, which is of the form of getting the AND of some conditions. To resolve this conflict, we first get the condition which is the object of event-5 (which is an inferior of E4 and has event-2 as a subgoal) and then get the other conditions of E4. With this procedure applied to the conflict at hand a procedure for

```
(AND  (POSITION   B1  (ON-TOP-OF B2))
      (POSITION   B2  (ON-TOP-OF B3)))
```

will be generated. PERSONIFY-TALK can then call CARRY-OUT on this PLAN again. It is not necessary to undo (POSITION B1 (ON-TOP-OF B2)), since the satisfaction of the prerequisites will

take care of that.

Another interesting example of protocol analysis is found in the thesis of Mark. This involves tracing back through previous steps of a simulation. Suppose a variable value is too high. Then if that value is evaluated as a sum, one of the inputs must be too high. Tracing back in this way Mark's program looks for a result caused by a decision rule which the user could change to produce a better result

CHAPTER 3

Understanding English

Augmented State Transition Network Pars i_

Recent progress in computer parsing of English, such as the work of Winograd and Woods, gives hope that the syntactic difficulties in handling natural language will soon be solved. In the OWL system our interest is almost exclusively with typed conversation. This is easier to handle than speech or literary prose, and the remaining syntactic difficulties appear to be conjunction, adverb placement, and ellipsis.

In building the OWL parser, we have followed Woods in using an augmented state transition network. Using a similar paradigm, we have faced the same design decisions as he. However, we have followed a somewhat different philosophy, which has caused us to make quite a different set of choices. As our philosophy is based on as yet unproven assumptions about the phrases to be parsed, we cannot yet say that our methods are better than his, but at least we will get some different test experience.

Appendix B gives an extensive discussion of the OWL grammar. As most of this is no different than other systems, we will not go through it in detail here. Rather, we want to explain the particular parsing philosophy of OWL.

A basic tenet of Woods, Winograd, and OWL is that not all of the facts of English syntax are best handled with the same mechanism. Woods and OWL identify five different mechanisms:

a) The part of speech and properties of individual words are found by ad hoc routines.

b) The language is taken to contain a limited number of phrase types (groups).

In OWL these are:

1) clause
2) noun group
3) verb group
4) preposition group
5) question group
6) adverb group
7) adjective group

The basic facts of word order for each of these groups are represented by a state transition network.

c) Additional syntactic facts and the semantics of these groups are handled by functions associated with the arcs of the state transition network.

d) The basic recursive process by which these word groups are assembled is implemented in an interpreter.

e) Conjunction is handled by a separate algorithm superimposed on the basic interpreter.

Some illustrative state transition networks are shown in Fig. 3.1. The parser uses these networks as plans just as the OWL interpreter uses plans.

Suppose we give the parser the sentence:

The man sold a big red apple.

The parser always starts out trying to complete a top level network, which, for illustration, we will take to be the major clause.

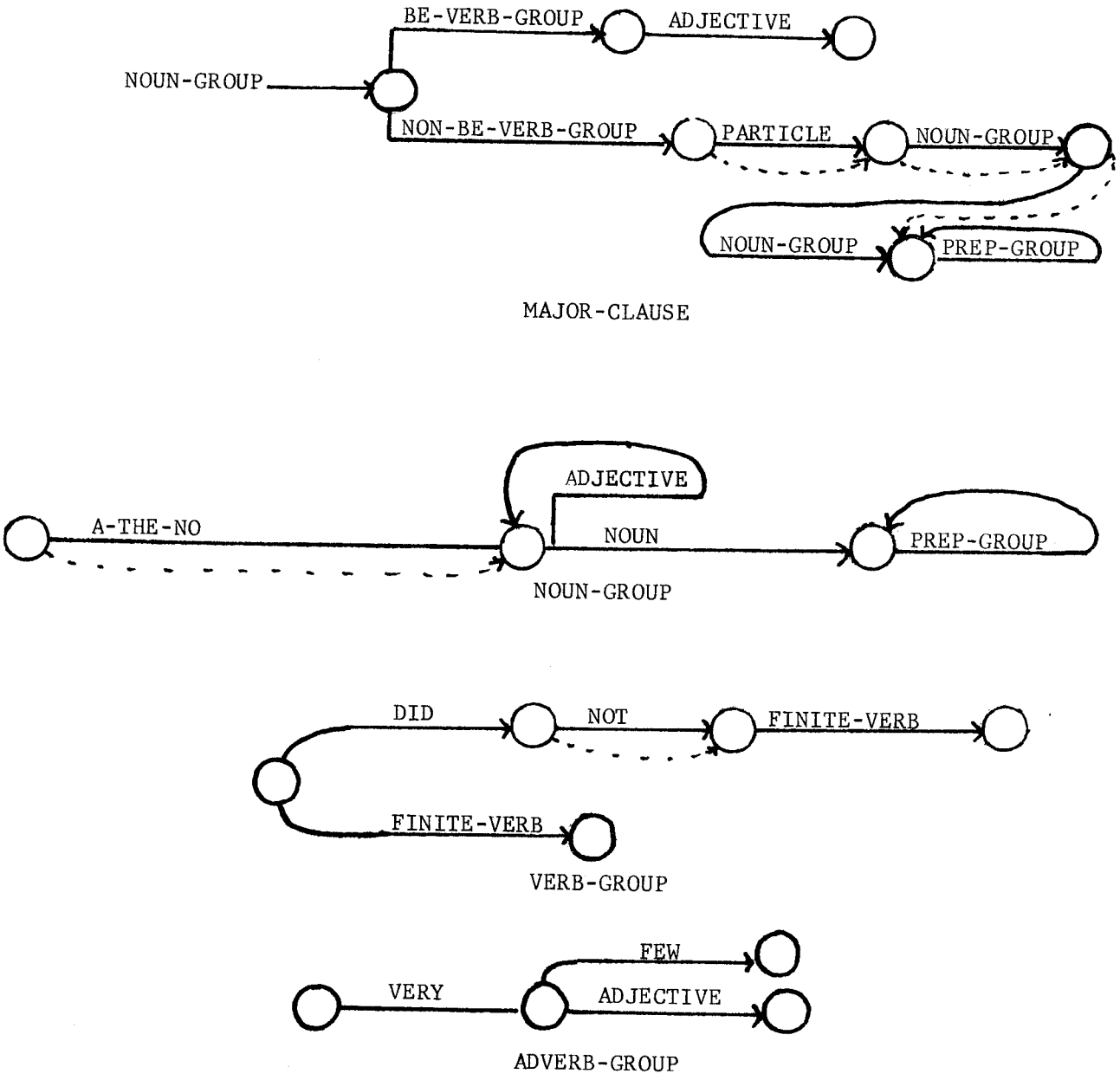


Fig. 3.1

Some illustrative state transition networks.

The parser thus assumes the initial major clause network state and looks at the first word of the input sentence, which is the. It now attempts to pursue one or both of two paths.

a) It attempts to use the as the next arc of MAJOR-CLAUSE.

b) It attempts to start a new group with the.

To add the to the major clause, the parser gets all the's parts of speech. (We have extended the notion of part of speech to include all of the categories which show up on the arcs of the state transition network.) The part of speech of the is A-THE-NO. Since A-THE-NO is not a kind of noun group, the parser is not able to add the to the major clause. The parser then goes on to consider the second possibility - starting a new group. Every part of speech has associated with it the group it can start. (Things are arranged so that there is only one.) The parser finds that the can start a noun group as an A-THE-NO. Before starting the noun group, the parser checks to see whether, in fact, a noun group could fit into the major clause at this point. Since one could, the parser attempts to start it. Each arc of the transition network has a function associated with it. This function makes additional checks (beyond the simple checking of parts of speech) and builds the OWL structure. It takes as arguments the OWL structure at the current node and the constituent (word or group) just found. Applying the function associated with the A-THE-NO arc, the parser gets (SELECTIVITY NOUN-GROUP-1 THE). The dashed arc paralleling the A-THE-NO arc in the noun group network means that a noun group can also be started directly with an adjective.

The second word big is an adjective. The parser considers starting a noun group with it but doesn't, because a noun group is not needed as the next constituent in the current network. However, the parser is able to add the adjective to the current network. Since the noun is not known yet, the adjective is just stored by forming the OWL structure:

(UNUSED-ADJECTIVE NOUN-GROUP-1 WORD-BIG)

The next word man is a noun. When the parser adds it to the noun group, it has enough information to construct a set of meanings for the noun group. For each meaning of the noun, the parser goes through any UNUSED-ADJECTIVE's (in the reverse order in which they were found). It attempts to modify each meaning of the noun with each meaning of each adjective. The parser succeeds in producing a meaning for the noun group only if some modification is permitted by the user's OWL world model. Because of the limited world of a particular problem solving system, it is not common to have many meanings for a noun group. The parser takes a branch for each meaning. In deciding what noun groups make sense, the parser can use any information in the data base as long as it is not derived from the major clause being parsed. For the big man, the parser will end up with one meaning, namely:

(SIZE MAN-1 BIG)
(SELECTIVITY MAN-1 THE).

The next word sold is a verb. It will not fit onto the noun group, nor will it start a group which will fit onto the noun group.

Now, at each node of the noun group, there is a function which gives a list of groups found if the function can be applied successfully to the OWL structure that has reached that node. Applying that function, the parser is successful in finding:

(SIZE MAN-1 BIG)
(SELECTIVITY MAN-1 THE)
(SYNTACTIC-TYPE MAN-1 NOUN-GROUP).

It then sees that this noun group satisfies the first arc of MAJOR-CLAUSE. However, just as the parser must wait for the noun in a noun group before constructing the meaning, it must wait until it is past the surface direct object, if any, of a clause. So the OWL structure is now:

(UNUSED-SUBJECT MAJOR-CLAUSE-1 MAN-1)
(SIZE MAN-1 BIG)
(SELECTIVITY MAN-1 THE)
(SYNTACTIC-TYPE MAN-1 NOUN-GROUP).

At this point the word sold can be used to start and, in fact, constitute a verb group. The verb group formed is a kind of NON-BE-VERB-GROUP. After integration into the major clause, we have:

(UNUSED-SUBJECT MAJOR-CLAUSE-1 MAN-1)
(SIZE MAN-1 BIG)
(SELECTIVITY MAN-1 THE)
(SYNTACTIC-TYPE MAN-1 NOUN-GROUP)

(MAIN-VERB MAJOR-CLAUSE-1 SELL-1)

(PAST SELL-1)

(TYPE SELL-1 FINITE-VERB-GROUP)

The next word, a, is not a particle nor does it start a group which leads to a particle. However, the parser is able to continue by following the dashed arc to the next node, since it can successfully apply the function on this arc to the OWL expression at the previous node. (In this case the function succeeds and does nothing.) Now, the word a can start a noun group. This noun group is found just as was the first one. Again, the adjectives are applied to the noun from right to left, because the natural order of adjectives in English is thought by some to reflect the criterion that those adjectives closer to the noun are more restrictive.

At this point, there are no more words in the sentence, so the completion function for the current node will be applied to the current OWL structure.

(UNUSED-SUBJECT MAJOR-CLAUSE-1 MAN-1)

(SIZE MAN-1 BIG)

(SELECTIVITY MAN-1 THE)

(SYNTACTIC-TYPE MAN-1 NOUN-GROUP)

(MAIN-VERB MAJOR-CLAUSE-1 SELL-1)

(PAST SELL-1)

(TYPE SELL-1 FINITE-VERB-GROUP)

(UNUSED-FIRST-OBJECT MAJOR-CLAUSE-1 APPLE-1)

(COLOR APPLE-1 RED)

(SIZE APPLE-1 BIG)

(SELECTIVITY APPLE-1 A)

(SYNTACTIC-TYPE APPLE-1 NOUN-GROUP).

The location of the completion function in the network implies that the OWL structure must contain a subject, an active or passive verb, possibly a particle, and possibly a direct object. The function finds a subject, active verb, and direct object. From this it knows that the direct object must be the OBJECT semantic case of the verb and that the subject must be the AGENT. Checking the user's world model of SELL, it finds a meaning of sell that conforms to this pattern and returns:

(SELL APPLE-1)

(COLOR APPLE-1 RED)

(SIZE APPLE-1 BIG)

(SELECTIVITY APPLE-1 A)

(AGENT (SELL APPLE-1) MAN-1)

(SIZE MAN-1 BIG)

(SELECTIVITY MAN-1 THE).

(PAST (SELL APPLE-1))

This completes the parse. The reader should note that the parser will find all parses, branching in its search to follow every possibility. Incomplete parses are not cut off on the basis of low probability, because we want the subsystem builder to know that he is getting all reasonable interpretations of the user's input statement. There are several reasons why this can be done without combinatorial explosion. First, the limited nature of the user's world means that

semantic checks rule out many paths. Second, the parser and networks are written to postpone decisions as long as possible, which results in better decisions using more information. We saw this illustrated above, where assembly of the clause was delayed until the direct object was found. Third, as we will see below, it is possible to save constituents so that if they are needed by another branch they do not have to be recomputed. Finally, it is our informal observation (which we must still prove) that when the available local syntactic and semantic constraints are used, most false branches die before they go very far. So-called "garden path" sentences do not occur often in discourse.

Consider the sentences:

The man sold very few red apples.
The man sold a few very red apples.

Referring to the adverb state transition network, we see that very can produce either an adjective or a quantity. When very is reached in the second sentence, we can use an adverb but not a quantity. This is not a problem because very has only one part of speech and thus must be used as an adverb (and, because adverb groups are cheap to produce, we always form them).

Next, consider the sentences:

I went down the path for bicycles.

(GO I)

(PAST (GO I))

(TRAJECTORY (GO I) (DOWN PATH-1))

(BENEFICIARY PATH-1 BICYCLE-1)

(SELECTIVITY PATH-1 THE)

I went down the path for lunch.

(GO I)

(PAST (GO I))

(TRAJECTORY (GO I) (DOWN PATH-1))

(SELECTIVITY PATH-1 THE)

(PURPOSE (GO I) (GET LUNCH-1))

(AGENT (GET LUNCH-1) I)

For is one of the most difficult prepositions to handle semantically. But, beyond that, notice that in the second sentence the preposition group for lunch will be parsed twice. As shown in the first sentence, it might modify path. Thus it must be parsed as a possible modifier of path. When this fails, down the path is returned to the clause level and for lunch is considered again as a modifier of the clause. For this reason, we save every completed group, and we make the rule that once a group is started no information from the current major clause except what is inside that group can be used to control its formation. This way a group is equally valid no matter what prompted its formation.

Whenever we are ready to start a group, we first look to see if all of the parsings for that type of group and beginning at that point in

the sentence are already known. Notice that this means that in the sentence

I didn't sell no apples.

the double negative cannot be used to stop the noun group no apples once we have started it. If we did that, then in

I went down no path for no particular reason.

we would stop no particular reason when considering it as a possible modifier of path. It would then be unavailable for modifying went, since the fact that a noun group was once started at that point in the sentence is later taken as evidence that all possible noun groups that could ever start there have already been formed and saved away. Since double negatives, improper person-number, and the like will not occur that often, not using these criteria to block parsing paths is no great loss. It also seems that people parse sentences with these mistakes anyway. Since we check the results at the higher level, it actually seems preferable to be able to let such usage through as long as a combinatorial explosion doesn't result.

We face a new difficulty in the sentence:

The apple the man sold Bob was red.

After finding the apple, the parser encounters another the. This doesn't post-modify a noun group and, although it can start a noun

group, a noun group can't post-modify a noun group either. However, a noun group can start a relative clause which post-modifies a noun group. The parser knows about this situation as a special case and can produce the relative clause as one possible parse of this construction.

In the sentence

He gave the dog the man was with the bone.

the object of with is the dog. The reading "the man was with the bone" might fail because people are "at" bones not "with" them. The grammar must be written to return

(UNUSED-SUBJECT RELATIVE-CLAUSE-1 MAN-1)
(MAIN-VERB RELATIVE-CLAUSE-1 BE)
(PAST BE)
(SELECTIVITY MAN-1 THE)
(UNUSED-PREPOSITION RELATIVE-CLAUSE-1 WITH)

as a modifier of the dog. The dog is then modified to form

(SYNTACTIC-TYPE DOG-1 NOUN-GROUP)
(SELECTIVITY DOG-1 THE)
(POSITION MAN-1 (WITH DOG-1))
(SELECTIVITY MAN-1 THE).

Let us consider an example involving multiple semantic interpretations. The sentence

He looked up the pipe.

has two readings. One is to look up a pipe in a listing of pipes; the other is to physically look through it. We get these two readings by treating the preposition up either as a particle as in look up or to mark the trajectory as in up the pipe. Since we want to get both readings, unless semantics eliminates one or both, we must always try prepositions both ways. This is an example of a situation where a word can both fit into the current group and start a new group, depending on the part of speech taken. In the phrase The big fat man's store, we see that in an expanded grammar a noun group can in fact modify another noun group, and that, because adjectives could go with either one, we have a combinatorially explosive situation. Our approach will be to put together whatever combinations make semantic sense. We must hope that phrases like Wood's

U.S. naval vessel acquisition cost estimates

port main gear door rear book operating spring strut plunger

do not occur too often or, if they do, that semantics will eliminate many possibilities. If this doesn't hold, we must face the fact that U.S. can modify naval or vessel or, possibly, estimates without changing the semantic interpretation. This would suggest delaying the placement of U.S. until the phrase is used.

In the sentences

The city people like is Philadelphia.

The city people like Philadelphia.

this same approach of delaying the decision on city people could be taken. But here two really different syntactic constructions are involved, and the OWL approach would be to branch. Similarly, in

The last day he sold apples.
The last day he sold apples was Friday.

we would branch.

Now, let us contrast our approach with that of Winograd for the sentence:

How much did we sell to Sears?

When Winograd's verb group expert finds the did, it will scan over the we and locate the sell, getting did sell. Bookkeeping will allow the we to be picked out as the subject. In our approach, the verb expert finds did and later sell; it is at the clause level that the combination is made. We move systematically from left to right -- there is no looking ahead or picking apart.

Finally, consider

It is not possible to drive a nail with a fish.

We will not accept drive a nail with a fish because not only is a fish not a hammer, but we do not have access to the not possible since it is

part of the same major clause. After the parser is finished, however, we could apply a follow-up program to attach with a fish using the phrase it is not possible as context.

Some aspects of meaning

Using the parser described above we can locate possible syntactic units. The problem is to construct the appropriate OWL for these units. In this section we want to discuss some of the assumptions underlying our method of doing this.

There is a general recognition that language can be described at various levels such as the phonemic and the lexemic. For example, in Stratificational grammar Lamb shows six strata: hyprophenemic, phonemic, morphemic, lexemic, sememic, and hyersememic. The nature of the mapping between levels appears to be that small sets of elements at one level map into small sets of elements at the next. Thus any one element might be used in several ways; it is the set of elements we are looking for. In our opinion these sets are easy to find because they are often physically adjacent or at least adjacent once some parse structure has been determined. We associate with each word the sets it occurs in and then eliminate by checking neighbors similar to Waltz. Everyone is familiar with the idea that some strings of words form idioms. What we are suggesting is that this is a frequent phenomenon rather than the exception in the translation process.

The above point of view will come out in our classification of the roles prepositions can play in a sentence. As an example of an idiomatic unit consider the sentence:

I put the cake ontop of the jar.

We will take ontop of to be a unit; we will not consider of to flag the genitive here. We might contrast this with:

I put the cake on the top of the jar.

Other examples are left-of, right-of, east-of, south-of, and west-of.

Occurrence of an idiomatic expression can be fortuitious, causing a slight difficulty in reading.

I took the cake out of the oven.

I emptied the bag I threw out of groceries.

We note seven roles prepositions can play in a clause:

A) Prepositions can be used as particles to select a meaning of a verb or of an adjective with be. Consider first the selection of the meaning of verbs other than be.

I threw up.

I threw my dinner up.

I threw up my dinner.

This use accounts for the ambiguity in the sentence:

I looked up the pipe.

One reading is to take up as particle, meaning to look it up in a reference book. The other use of up, covered in G), indicates the TRAJECTORY where one looked. These two uses of up split when we replace the pipe with it. For the first we say:

I looked it up.

For the second we say:

I looked up it.

With the pronoun, the sentence is harder to understand because of the reference problem. Our experience indicates that, as new uncertainties appear, the rules of English force the removal of other uncertainties to keep the processing load in control.

Note the different use of the reflexive in

They threw a smokescreen around themselves.

They expected a smokescreen around them.

We will handle this by making around a particle in the first sentence and not in the second. The idea is that throwing a smokescreen around something is a different procedure than throwing it between two things, for example. One does not have different procedures for expecting a smokescreen. We then make the rule that a particle takes the reflexive. We can think of a verb as having many different meanings. For example

I shot the rabbit.

I shot the picture.

I shot the breeze.

are all different meanings of shoot. In this case, the direct object selects a meaning of shoot (except that the second sentence allows two of them). The particle also selects meanings of the verb, although the object many resolve it further.

I changed into dry clothes.

I changed into a prince.

If we define a particular procedure put-on-top-of which is more specific than put, then we may want to make on-top-of a particle in

Put block A on top of block B.

so that the parser can find our procedure easily. Prepositions can also be used in conjunction with be and an adjective:

I am mad at myself.

Music is popular with teenagers.

Elephants are fond of peanuts.

Childern are wild about candy.

The heat was hard on this eskimo dogs.

The paper was full of water.

I am sold on exercise.

He is good with his hands.

B) Prepositions can also select a meaning of noun. One way to think of this class of expressions is that it is analogous to location expressions in the physical world, but, because the expressions are abstract, no particular preposition is required on semantic grounds. Therefore, an idiomatic choice is made.

He was on time.

He was in time.

He got the television on trial.

He was under investigation.

We saw the color TV show in black and white.

That is beyond question.

We talked in a whisper.

We talked at length.

C) Prepositions can indicate time and position. If a prepositional phrase indicates time or position then the preposition is not just a flag, but contributes to the meaning. Since many prepositions can indicate time and position as well as other things, it is necessary to recognize time and position expressions by the semantics of the expression in the context.

We sold apples in 1972.

We sold apples before 1972.

We sold apples during 1972.

We sold apples in Boston.

D) In giving a trajectory, the preposition can stand alone or with an object, and it gives semantic memory.

He ran across.

He came over.

He ran in.

He ran into the house.

E) With can flag a second subject or direct object. It may be that no particular distinction specific to the procedure can be made between the two subjects or objects. There is only a general subordinance.

Bob and I went to the store.

I went to the store with Bob.

The ice cream was eaten with the cake.

I ate the ice cream with the cake.

F) By is used in the expressions: by himself, by herself, etc.

G) Of can flag the genitive.

The running of the deer was graceful.

The story of the bear was interesting.

That boot of mine is missing.

H) To introduces a clause. It can flag the PURPOSE when it is expressed as a clause,

I went home to get a book.

I) Finally, prepositions can flag semantic cases. By "flag" we mean that the preposition serves solely as a marker. As Fillmore postulated, we can define cases so that there is only one preposition for those not involved with time and space.

Case	Preposition
AGENT	by
QUANTITY	by
TARGET	at
RATE	at
BENEFICIARY	for
EXCHANGE	for
DURATION	for
PURPOSE	for
INSTRUMENT	with
SPECIFIC-RAW-MATERIAL	with

METHOD-OF-ACCOMPLISHMENT	with
MANNER	with
SOURCE	from
DESTINATION	to
COMPARISON	like
OBJECT-DESCRIBED	about

The DETAIL-OF-METHOD case is marked by preposition use B), that is, it is idiomatic.

I bought it on credit.

The remaining cases are marked by prepositions in the earlier uses. In particular, it seems well to point out that away-from, out-of, off-of, out-from-under, into, onto, over, and under mark the TRAJECTORY.

Nouns derived from verbs

The word sale is derived from sell. It is important to know this because it allows the parser to handle sale by looking at the OWL definition of sell. The user of prepositions is demonstrated by

We sold a battery to Sears from the warehouse in '72.

Our sale of a battery to Sears from the warehouse in '72 was profitable.

The battery's sale by us to Sears was profitable.

The rule is that all of the cases are flagged by their normal prepositions. The possessive plays the role of the surface subject and of flags the direct object.

There are many other verbs from which nouns are derived; some examples are example, infect, propose, and insist. Two interesting nouns are referral and reference, which are derived from different meanings of refer.

I referred Bob to the dictionary.

My referral of Bob to the dictionary was unusual.

I referred to the dictionary.

My reference to the dictionary was unusual.

Consider the procedure selling. We may want to speak about the all sellings or a specific instance of selling. We may also want to speak about principal results of selling in general or a specific principal result. The four possibilities are:

Selling requires a certain kind of personality.

I gave him a hard sell.

Sales is where the money is.

I got three sales yesterday.

Let us try the same thing for the verb employ.

Employing requires a certain kind of personality.

* I gave him a hard employ.

Employment is where the money is.

I got three employees yesterday.

We can see that the verbs do not make their derived nouns in a completely predictable way. The noun derived from the agent, however, is easy to predict, it marked by er, as in seller or employer.

Adjective

If a man says

I want my house painted.

then he can mean either that he wants the act of painting performed on his house, ot that he wants a house that has painting already done to it. We will distinguish This is OWL as

(WANT (PAINT HOUSE-1))

(WANT HOUSE-2)

(PAST (PAINT HOUSE-2)).

Thus, we will define the adjective painted in terms of the past of the verb.

When someone is certain, trivial, fun, or simple to please, these adjectives refer to the speaker's opinion. When someone is able, happy, or slow to please, these adjectives refer to his characteristics. Note that these are distinguished by

John was fun to please,

It was fun for me to please him.

John was able to please,

* It was able for me to please him.

Alternate Surface Level Constructions.

These last sentences bring us to the discussion of different representations in English for OWL expressions which differ only in ways to subtle for us to do much about at the present. Consider

I gave the ball to John.

I gave John the ball.

By moving John into the position normally held by the OBJECT, it acquires overtones of the OBJECT, that is, of being the most important noun group in the sentence. We could indicate this in the OWL structure by adding an additional property, but we would probably not know what to do with it. We also have the passives

John was given the ball by me.

The ball was given to John by me

? The ball was given John by me.

Again all of these sentences would be represented the same in OWL. Note, that in order to distinguish the first one from the third we need to know what can be given to what. At a slightly greater difference in meaning than the above we have

I smeared paint on the wall.

I smeared the wall with paint.

I shot the gun at the rabbit.

I shot the rabbit with the gun.

We can handle these with two meanings of smear or shoot in the same frame, similar to the way in which buy and sell are handled.

For so called ergative verbs like open, the surface subject can be the AGENT, the INSTRUMENT if the AGENT is missing, or the OBJECT if the AGENT and INSTRUMENT are both missing.

John opened the door with a stick.

A stick opened the door.

The door opened.

Here again, by going into subject position the noun groups take on overtones of its primary occupant, the AGENT. We would, however, use the same OWL form.

In the case of benefit we have only an object and a source. The source can go to subject position.

John benefits from the will.

The will benefits John.

Verbs which take a plural subject often take different forms.

John and Jim met.

John met with Jim.

John met Jim.

In these three sentences the properties of AGENT are shifting increasingly away from Jim, yet we are uncertain of any specific change in Jim's role in the meeting procedure. In OWL, the user might have different procedures for these or he might not.

A more complex problem is

I smelled the rose.

The rose smelled good to me.

I caught a whiff of a good rose smell.

We smelled the same smell.

As additional data let us look at

I ran the machine

The machine ran.

That is okay by me.

That seems okay to me.

The rose looks okay to me.

It seems reasonable to handle run exactly like open, that is, with just one meaning. However, with smell there seems to be more of a distinction between someone smelling a rose and a rose smelling. The smeller of a rose is not responsible for the rose smelling the way the runner of a machine is responsible, unless we think the rose has no smell without a smeller! The third sentence would tend to indicate that the smell is there whether the smeller gets it or not.

We will side aganist Bishop Berkeley and maintain that the rose creates a perception whether anyone is there to get it or not. Words like <u>good</u> and <u>okay</u> give the opinion of the perceiver. The parallel analysis of <u>seem</u> would then be that the AGENT generates a preception

which gives the perceiver a certain opinion. The perceiver can be mentioned as the destmation of the preceptions. Advancing this solution is a demonstration of our desive to give OWL a linquistic base.

Have

There are several meanings of have illustrated by the sentences below.

(KID PART LEG)

Bob has a leg. (LEG BOB)

(KIND CHARACTERISTIC FEVER)

Bob has a fever. (FEVER BOB)

Bob has a wife, Sally. (WIFE BOB SALLY)

Bob has an ability to run. (PURPOSE (ABILITY BOB) RUN)

Bob has a car. (as a possession) (AGENT (OWN CAR) BOB)

Bob has a car. (at his disposal) (AGENT (CONTROL CAR) BOB)

(KIND PROCEDURE RUN)

Bob is having a run. (RUN BOB)

(KIND PROCEDURE MEAL)

Bob is having a meal. (MEAL BOB)

Bob has rats. (in his house) (RAT BOB)

It is not always possible to disambiguate have, so HAVE itself also exists in OWL. Note that something is not at a person, rather it is with him and he has it. The reading (RAT BOB) may seem odd at first glance, but a person seems to aquire as characteristics anything associated with him that he cannnot control.

The genitive

The genitive flags the agent or the object, where we understand the object to be the second position of an OWL expression.

The hunting of the dogs was objectionable. (AGENT or OBJECT)

The dog's hunting was objectionable. (AGENT)

The branch of the tree fell down.

(KIND PART BRANCH)

(BRANCH TREE-1)

Half of the apples were rotten

(FRACTION APPLE-1 HALF)

Five of the apples were rotten

(MEASURE (QUANTITY APPLE-1) 5)

A few of the apples were rotten

(QUANTITY APPLE-1 FEW)

I burned a cord of wood.

(KIND QUANTITY CORD)

(CORD WOOD-1)

I bought two cartons of milk,

(KIND QUANTITY CARTON)

(QUANTITY (CARTON MILK) TWO)

I saw a flock of sheep.

(KIND GROUP FLOCK)

(FLOCK SHEEP)

The biggest of the cakes was too small.

(SELECTION (KIND CAKE-1) THE)

(SELECTION (KIND (KIND CAKE-1)) BIG-1)

(GREATEST BIG-1)

What kind of chemical is that.

(IS THAT (WHAT (KING CHEMICAL)))

A way of finding out is to ask John.

(IS (WAY (FIND OUT)) (ASK JOHN))

The color of this ball is red.

(COLOR BALL-1 RED)

The state of Texas is very big.

(KIND CHARACTRISTIC STATES)

(STATE TEXAS)

The story of his adventures was exciting.

(STORY ADVENTURE-1)

He got the idea of copying birds.

(IDEA (COPY BIRD-1))

I deprived Bob of his rations.

(AGENT (DEPRIVE (RATION BOB)) I)

This and That

This and these are taken to be THE plus a POSITION of here; that and those are taken to be THE plus a POSITION of there.

Relations Between Events

We have now described objects and events. It remains to analyze the relations between events. The relations within a single sentence

are given below. The relations within a paragraph must await a future memo.

The connective gives the order of the arguments and can negate one or both of them.

IMPLIES (E1, E2)

If it looks like a rose then it is a rose.	E1,E2
It is a rose if it looks like a rose.	E2,E1
It is a rose unless it looks like a rose.	E2,E1
It looks like a rose therefore it is a rose.	E1,E2

DID-NOT-BLOCK (E1,E2)

He was fat, nevertheless, he played well.	E1,E2
He was fat, yet he played well.	E1,E2
He played well, even though he was fat.	E2,E1

He played well <u>although</u> he was fat.	E2,E1
He played well <u>despite</u> being fat.	E2,E1
I was tired; <u>however</u>, I kept working.	E,E2
I will have fun <u>whether or not</u> I go to the movies.	E2,or E2,E1

SET-GOAL-FOR (E1,E2)

I eat candy <u>because</u> I like it.	E2,E1
I like candy <u>so</u> I eat it.	E1,E2
I like candy, <u>thus</u> I eat it.	E1,E2
I like candy, <u>consequently</u>, I eat it.	E1,E2
I like candy, <u>hence</u>, I eat it.	E1,E2
I like candy, <u>accordingly</u>, I eat it.	E1,E2
I eat candy <u>rather than</u> strave.	E2, E1
I eat candy <u>because of</u> my appetite.	E2,E1

XOR (E1,E2)

I can go to Boston or I can go to New York.	E1,E2
Either I can go to Boston or I can go to New York.	E1,E2

AND (E1,E2)

I eat and I sleep.	E1,E2
I didn't eat, rather, I kept working.	E1,E2
Boys like girls, conversely, girls like boys.	E1,E2
Boys like girls, similarly, roosters like hens.	E1,E2

Building a clause

In the first part of this chapter we showed how the OWL parser would locate the verb group, particle, and noun groups of a clause. By looking at the user's OWL world model it can put these together to get the OWL meaning of the clause. The strategy for any sentence is to first find the main verb, the particle, and the OBJECT. To do this we discover how many noun groups occur immediately after the verb without prepositions. The number will be between zero and three.

The knife cut beautifully.

I hit a ball.

I hit Bill a ball.

I hit Bill a ball the day before yesterday.

Having done this we can locate the particle if any. (Remember that the parser will attempt clause constructions for both particle and non-particle parts of speech of each preposition, but that since noun groups are found only once this doesn't make much extra work.) If we have a particle, we get all the meanings of the verb and discard those which don't take that particle.

Next, we find the OBJECT. If the verb is passive, the OBJECT is the noun group in subject position. We next check our verb meanings to see which ones take the subject as object. For each one that does we have found a possible meaning of the clause. The parser branches and performs the remaining steps on each of them in turn. If the verb is active and there is no noun group after the verb, then we consult the verb to see if an object is required. If it is, as for open, then the subject must be the OBJECT and we check it as before. If the OBJECT is not required, as for cut, then the verb will list those cases that can appear in subject position. For cut these are the AGENT, INSTRUMENT, and OBJECT. We start a parsing for each of these if the subject passes their respective tests. If there are one or more noun groups after the verb, then we must be aware that the last one could represent a time. If there are three, it definitely does. Otherwise, a time noun group will probably fail the OBJECT test for the verb meanings, and then we will know. Having assigned the OBJECT, we attempt to assign the rest of the noun groups we have found so far. It should be possible to assign all of them uniquely. Next, we find and assign the remaining noun groups. As each noun group is found, we know from the preposition what the possible cases are. We try each of these and, if the noun group only fits the verb meaning in one case, the assignment is made. If it fits more than one, it is saved along with a list of the possibilities. Every time a case is filled, it is removed from the possibilities list of every unassigned noun group, as only space and time can be filled by more than one noun group not coordinated by conjunction.

I went home for food.

*I went home for food to get a book.

I went home for mother to get a book.

I went home for mother and to get a book.

The phrase for mother could mean either in mother's place (BENEFICIARY) or to get mother (PURPOSE). Since to get a book is a purpose the uncertainty is resolved.

Note that the sentences

I shot the rabbit with the gun.

I shot the gun at the rabbit.

force us to have two meanings of shoot (SHOOT RABBIT-1) and (SHOOT GUN-1) in order to use the method of analysis given above. We treat these two meanings like buy and sell - which also describe a similar situation from different viewpoints - and put the same objects in different cases.

Consider the sentences:

Bill and Tom met.

Bill met with Tom.

Bill met Tom.

Tom would be one of the AGENT's in the first. In the second, Tom could be a CO-AGENT if the user's MEET procedure has one; otherwise, the parser could make it an AGENT. In the third, Tom would be the OBJECT.

The word order in questions and relative clauses can make it necessary to wait longer before resolving the assignment of cases.

What store did it? (AGENT)

What store did we sell to Sears? (OBJECT)

What store did we sell to Sears from? (SOURCE)

The store we sold to Sears from is closed now.

In the last sentence, the relative clause is passed up to the noun group store with an open preposition from. Store is then plugged in at the noun group level and the case assignment of the relative clause completed.

In the sentences

We expected John to run for President.

We asked for Bob to be admitted.

we have a phenomenon known as subject raising. The parser can tell from the specific verb whether to expect this. The required OWL forms are

(EXPECT (RUN JOHN))

(ASK (ADMIT BOB)).

The OBJECT is an activity. A similar thing happens in

I saw Mary in the park with Bill.

when we take the reading

(SEE (POSITION MARY (WITH BILL))).

Also interesting is

We elected John chairman.

(ELECT (CHAIRMAN JOHN))

Occasionally, modifiers of a unit are displaced from it.

We have enough beer in the cooler to have a party.

(AGENT (HAVE BEER-1) WE)
(POSITION BEER-1 (IN COOLER-1))
(SELECTION BEER-1 ENOUGH-1)
(PURPOSE ENOUGH-1 (PARTY WE))

How many houses has John sold that are big?

(WHAT(QUANTITY HOUSE-1))
(SIZE HOUSE-1 BIG)
(AGENT (SELL HOUSE-1) JOHN)
(PAST (SELL HOUSE-1))

We have not yet made a thorough analysis of this problem, but it appears that the word enough or the question form of a sentence or some other clue will be available and that there is no harm in constructing the

clause before the displaced modifier is found. Thus we need not delay in expectation of it.

Building the clause - a slightly deeper lo

Suppose a user has added the following frame to his world model:

```
(LEARN (DEFINE PROCEDURE
            (FARM (KIND LAND FARM-LAND)))
   (AGENT (FARM FARM-LAND) (KIND PERSON FARMER))
   (SUB-PROCEDURE (SCENARIO (FARM FARM-LAND))
                        (GROW CROP))
   (DEFINE PROCEDURE (GROW PLANT))
   (AGENT (GROW PLANT) (KIND PERSON GROWER))
   (KIND PLANT ROSE)
   (KIND GROUP CROP)
   (CROP PLANT) )
```

The system then receives the sentence (suggested by Andee Rubin):

A farmer grows roses.

The system discovers two meanings of grow, (GROW PLANT) and (GROW CROP). Taking (GROW PLANT), it would find that a rose is a kind of a plant and so the OBJECT would check. Checking the AGENT, it would find that the agent of GROW must be a person. Since it is known that a farmer is a kind of person, the match can be made. However, the fact that the person was referred to as a farmer was not used. The match is less than

perfect unless the use of farmer is explained by the situation.

Next, considering (GROW CROP) the parser would note that crop is a kind of group and that we can thus have the paraphrases

He grows a crop of roses.

He grows a crop.

He grows roses.

Since (GROW CROP) is found to be a sub-procedure of the scenario of farming, the agent is a farmer, which is a perfect match.

At this point the parser has two questions:

a) It is a crop of roses?

b) If not, why is he called a farmer?

Sometimes a notion of typical examples is useful. Suppose the user's world model contained the following kind of information.

```
(LEARN
   (POSITION PHYSICAL-OBJECT-1
         (LOCATION-RELATION PHYSICAL-OBJECT-2))
    (TYPICAL (GREATER (SIZE PHYSICAL-OBJECT-2)
                      (SIZE PHYSICAL-OBJECT-2)))
    (KIND LOCATION-RELATION BY) )
```

```
(LEARN (KIND PHYSICAL-OBJECT BALL)
   (GREATEST (SIZE BALL-2))
   (EXAMPLE BALL-1 BEACH-BALL)
   (TYPICAL (SIZE BALL-2))
   (EXAMPLE BALL-2 SOCCOR-BALL)
   (LEAST (SIZE BALL-3))
   (EXAMPLE BALL-3 GOLF-BALL) )
```

Given the sentence

A banana is under a table by a ball.

The parser will attempt to modify table with by a ball. It sees that an object is typically said to be by a larger object. Checking its examples of tables and balls, it sees that only a very small table would be smaller than a beach ball. Modifying banana with by a ball works nicely on the other hand.

As a final example consider the structure.

```
(LEARN (DEFINE PROCEDURE
          (FLAP-WINGS-FLY ANIMAL-1))
(KIND PART WING)
(WING ANIMAL-1)
(TYPICAL (KIND ANIMAL-1))
(EXAMPLE (KIND ANIMAL-1) BIRD)
(NOT (CAN (FLAP-WINGS-FLY BIRD-1))) (HEALTH BIRD-1 SICK)
(NOT (CAN(FLAP-WINGS-FLY PENGUIN)))
```

```
(DEFINE CHARACTERISTIC (ANTARCTIC SOMETHING-1))
(TEST (ANTARCTIC SOMETHING-1)
(PERTAIN SOMETHING-1 ANTARCTICA))
(KIND LAND-AREA ANTARCTICA)
(HABITAT BIRD LAND-AREA)
(HABITAT SEA-GULL SEA-SHORE)
(PART ANTARCTICA SEA-SHORE-1)
(KIND BIRD PENGUIN)
(KIND BIRD SEA-GULL) )
```

The parser is given the sentence

Fred is an antartic bird who cannot fly.

The parser finds that antarctica bird means the bird pertains to Antartica. Antartica is seen as a land area, and the habitat of a bird is a land area, so here we have a match. Evaluating who cannot fly the parser notices that the typical example of flap-wings-fly is in fact a bird. Thus, the situation must be a-typical in some way. A penguin is located as a particular antarctic bird who cannot fly. At this point the parser has the questions

a) Is Fred's habitat Antarctica?

b) Is he a penguin?

c) If not, is he a sick seagull?

The OWL system is not intended for reading stories. We expect to answer such questions from the context of the console session or by asking the user directly.

user directly.

Chapter 4

Analogies and the Representation of Expert Domains

A model of the physical world

In the previous chapters we have presented a number of ideas without being very specific about the meaning of abstract concepts such as store. Our goal in defining the OWL notation is to make possible the construction of large systems. We want to provide conventions and make simplifying assumptions whenever possible in order to reduce the compexity of the system description. Of course, if we go too far, the resulting systems will not be able to represent important subtleties of the expert problem solving environment. On the other hand, if we attempt to include too many fine points, or leave too many decisions to the subsystem builder, we can expect that no subsystem will be successfully completed. It is with this in mind that we introduce the sylized OWL concept of the physical world.

In OWL we define the physical world to be made up of PHYSICAL-OBJECT's, each of which has PROPERTY's. TIME advances in the physical world, and over time PHYSICAL-OBJECT's come into existence or cease to exist; also, their properties change. Basically, a physical object is something made of matter, what Shank would call a "picture producer", we include air, and, as was mentioned earlier, a smell. The properties of physical objects fall into two classes; those which they have intrinsically, such as COLOR, and POSITION, PART, and CONSTRAIN, which they have with respect to other physical objects. A SCENE is a collection of physical objects and their properties. At any point in

time the physical world is completely described by listing all of the objects in it and their properties.

As we move from one point in time to the next most things in the physical world stay the same. This makes it highly desirable to describe a new state of the world in terms of the CHANGE from a previous state. It is not clear what the best terms are for expressing these CHANGE's. We could, of course, have only an operator which says that a certain property of a given object changed from one value to another. This approach, however, corresponds too closely to computer micro-code. The level of detail is too great to make it easy to store, retrieve, and manipulate such descriptions although it may be fine for executing them. Since it is primarily the predictive power of our model of the physical world which interests us, manipulation of the descriptions is much more important than execution. We will, therefore, seek a more aggregate description of each.

This need for prediction leads us to include the notion of AGENT of a CHANGE in our model. Even if we do not model the motivations of the AGENT, observed sequences of acts by the same AGENT give us predictive power. We will in fact allow some modeling of the agent's motivations and PURPOSE but this model will lie outside the physical world.

Analogies to the physical world

There are many concepts such as idea, plan, and family which will not appear in the physical world. We will allow additional concepts in OWL, but only if they are defined in terms of a world which is an anology to the physical world. Shank and others have called attention to the fact that some verbs seem to have meaning on a physical, social, or mental level. The meanings are related by analogy. For example, we can say

He came to a tree.

He came from a good family.

He came to a conclusion.

By way of demonstration, we will construct three analogies to the physical world. In OWL we allow the user to define whatever additional analogy worlds he wishes. The most complete analogy to the physical world is one which involves primarily social concepts. The purpose of the model is understanding any AGENT which has a stimulus response type of behavior. We will call this the system world. The principal entities in the system world are shown in Fig. 4.1. It should be obvious that doctors, lawyers, business men, and programmers are concerned with this system world. The principal use of OWL will be the manipulation of the descriptions of systems.

As one can see in Fig. 4.1, there are many types of systems. Animals can create organizations by mutually agreeing to respond in certain ways. The concepts for describing this, such as responsibility, fall into the system world.

```
SOMETHING
        REACTIVE-COMPLEX
          DISEASE
          SYSTEM
                MACHINE
                GOAL-DIRECTED-SYSTEM
                  SEARCH-ORIENTED-COMPUTER-PROGRAM
                  EMOTIONAL-SYSTEM
                        SOCIAL-CLASS
                          WORKING-CLASS
                        MARKET
                        PERSON
                        ORGANIZATION
                          NATION
                          ENTERPRISE
                            STORE
                          DIVISION
                          FAMILY
                          CHURCH
                          POLITICAL-PARTY
                        INTERACTION
                          DISCUSSION
                            ARGUMENT
                            NEGOTIATION
                          COMPETITION
                            GAME
                            FIGHT
                                WAR
  SOMETHING
        SOCIAL-CONVENTION
                RESPONSIBILITY
                AUTHORITY
                OWNERSHIP
                CONTROL
                AGREEMENT
                  CONTRACT
                RULE
```

Fig. 4.1

Entities in the system world.

Properties of systems are shown in Fig. 4.2. Fig. 4.3 shows typical social locations. Corresponding to PART in the physical world, one can be PART of an organization. A kind of a PART is a MEMBER. An organization may have a TOP, a BOTTOM, and a HEAD.

The essence of this world is the recognition of systems and in particular organizations. Organizations establish the conventions of ownership, control, responsibility, authority. Organizations make rules and individuals make agreements. By having this model we can understand the organizational constraints placed on AGENT's, and the types of AGENT's which can be controlled.

```
(CHARACTERISTIC GOAL-DIRECTED-SYSTEM)
        (RELATIVE PERSON)
        (CUSTOMER ENTERPRISE)
        (SUPPLIER ENTERPRISE)
        (POSITION SOCIAL-CONVENTION (AT EMOTIONAL-SYSTEM))
        (SOCIAL-POSITION PERSON SOCIAL-LOCATION)
        (NOMINAL-CHARACTERISTIC GOAL-DIRECTED-SYSTEM)
                (BUSINESS ENTERPRISE)
                (OCCUPATION PERSON)
                (ORDINAL-CHARACTERISTIC GOAL-DIRECTED-SYSTEM)
                        (ORGANIZATION-SIZE ORGANIZATION)
                        (SUCCESS GOAL-DIRECTED-SYSTEM)
```

Fig. 4.2

Properties of Systems

```
SOCIAL-LOCATION
        (ON TRIAL)
        (UNDER INVESTIGATION)
        (IN SURGERY)
        (ON SOCIAL-SCENE)
        (IN ORGANIZATION)
        (IN SOCIAL-CLASS)
        (LOCATION-RELATION HOME)
        (LOCATION-RELATION WORK)
        (ON THE-SPOT)
        (IN TROUBLE)
```

Fig. 4.3

Social locations

Plans and Actions

We have modeled the physical world where directly observable changes take place. In order to better predict these changes we have introduced the notion of AGENT. In the world of systems we can describe all possible agents, and the organizational constraints placed upon them. In order to better predict how these agents will act, we need the notion that some of these agents, namely animals and organizations, carry out actions in accordance with plans. Entities in the world of plans and actions are shown in Fig. 4.4. Animals and origanizations have plans which are made up of goals as explained in Chapter 2. A goal can be any action, but there are some actions which make sense only

```
SOMETHING
     PLAN
      INTENTION
      ALTERNATIVE
     GOAL
      (MAINTAIN CONDITION)
      (INSURE EVENT)
      (LIMIT QUANTITY)
      (TRICK GOAL-DIRECTED-SYSTEM)
     MISTAKE
       TROUBLE
       RESOURCE
        INFORMATION
     ACTION
```

Fig. 4.4

Entities in the world of plans and actions

in the context of plans. Using resources, agents take actions. These are either in accordance with plans or they are mistakes. Characteristics of these entities are shown in Fig. 4.5. The position of a plan is given by its degree of completion.

We will call our final model the mental world. The principal entities in the mental world are shown in Fig. 4.6 Characteristics of mental entities are shown in Fig. 4.7. It is in the mental world that the plans are formulated.

These four models give a surprisingly complete model of the world. To see this we must describe the activities which take place in each model.

```
SOMETHING
      (DEGREE-OF-COMPLETION PLAN)
      (PURPOSE PLAN)
      (RESULT ACTION)
      (NECESSITY ACTION)
      (DIFFICULTY ACTION)
      (COMPLEXITY ACTION)
      (PROBABILTY RESULT)
      (READINESS RESOURCE)
```

Fig. 4.5

Characteristics in the world of plans and actions

```
something
      mental-entity
      situation
      problem
      question
      answer
      suggestion
      offer
      order
      concept
      fact
      idea
      decision
      choice
      reason
      view
        opinion
      view-point
      frame-of-reference
      mind
      attention
      thinker
```

Fig. 4.6

Entities in the mental world

```
(WISDOM THINKER)
(KNOWLEDGE THINKER)
(BELIEF THINKER)
(DOUBT THINKER)
(ASSUMPTION THINKER)
(SUPPOSITION THINKER)
(CONCLUSION THINKER)
(NAME SOMETHING)
(POSITION MENTAL-ENIETY MENTAL-LOCATION)
(INTELLIGENCE THINKER)

MENTAL-LOCATION
   (IN MIND)
      (IN (PART MIND))
   (ON TONGUE)
   (ON (TIP TONGUE))
   (IN HEAD)
   (ON SOLID-SURFACE)
```

Fig. 4.7

Characteristics in the mental world

Simple activities in the physical world

As we have defined it, all activities in the physical world must ultimately be definable in terms of the creation and destruction of physical-objects or a change in their properties. Thus, it would seem that the only three activites; create, change, and destroy would be needed. The use of a small number of basic activites is in fact recommended by Shank, who claims there are only six needed for the physical world: MOVE, INGEST, PTRANS, PROPEL, GRASP, and EXPEL. Discussing an example in the mental world he states "What is important here is that we need only this one inference rule for MTRANS in order to answer such a question regardless of how the information was MTRANSed. Thus, if Mary had "read" X rather than being 'told' it, we would still have MTRANS and thus would require no new rules. Since there are thousands of verbs and only fourteen ACTs for which inference rules need be written, this amounts to a tremendous saving and is probably quite a bit more like the way people operate."

Our notion in OWL that inference is often made by matching what would be at the basic level a complex pattern of highly specialized facts, does not fit with Shank's view. Inference rules for a single ACT would only be able to distinguish the highly specialized facts if the conceptual data structure were very extensive, thus making the rules very difficult to write. In Shank's notation the sentence "The fox tricked the duck". would be broken down into many sub-actions. Answering the question "Is the fox reliable." the system would have to recognize these subactions as an instance of trickery and say no.

A further difficulty with Shank's approach is that it makes it harder to use different representations of the same situation in different problem solving tasks. This has been quite useful, however, in problem solving programs.

While we dislike Shank's scheme we hesitate to give up the ability to classify actions in terms of what are frequently their most important properties. In order to base OWL on English, we employ what Miller calls the menthod of incomplete definitions. Suppose we want to express a change of position with CHANGE. We might say

John changed the position of the chair from in front of the table to near the window.

To focus only on the new position we can replace changed the position of with put.

John put the chair near the window.

Note that what happens here is similar to buy-sell-pay. The DESTINATION of (CHANGE POSITION) is the SPECIFIC-LOCATION of PUT, the OBJECT of POSITION is the OBJECT of PUT, and PUT does not take a case corresponding to the SOURCE of (CHANGE POSITION). Next, suppose we don't even want to mention the new position, only that a new position has been given. We can replace PUT with POSITION.

John positioned the chair.

If we add near the window to this sentence it will not indicate whether the chair was there before or not. It gives the POSITION of the act not the SPECIFIC-POSITION. Note that POSITION has a connotation of precision which is missing from PUT. Thus PUT is an incomplete definition of POSITION. Just as it is useful to know that a dog is an animal, it is useful to know that put is a change of position. By substituting this heirarchy for Shank's primitive ACT's we can make deductions with the simple verbs when that is appropriate and still retain the overtones of a word like POSITION.

As Charniak has observed, the answer to the question

Where is Spot?

can be

At the window.

or

With Bob.

or

Bob has him.

but not

At Bob.

When the position of an object is given as a person, the question immediately arises as to whether the person controls it. The default assumption is that he does. The verb give lets us make this specific.

I put the chair in Bob's control.

I gave the chair to Bob.

I put my dog in Bob's care.

I gave Bob my dog to care for.

The DESTINATION of give is AGENT of control. (Remember there is no ownership in the physical world.)

If we want to concentrate on the original position, we use GET. Here the problem of control arises in a different form.

Same took the chair from Bob.

Sam got the chair from Bob.

Note that take does not imply that Bob gave the chair to Sam. For this meaning we must use get.

It is also possible to concentrate on the motion, rather than the initial and final positions. To do this we use move.

I moved the chair from in front of the table across the rug to the window.

When using move, we can also include a trajectory but we cannot give a location for the destination. To indicate that the agent did not move with the object we replace move with send. As with give and get, the destination of send can receive. Bring and Take are used when the AGENT goes too. Fig. 4.8 shows the heirarchy we have built up so far. We

now turn to a more comprehensive approach.

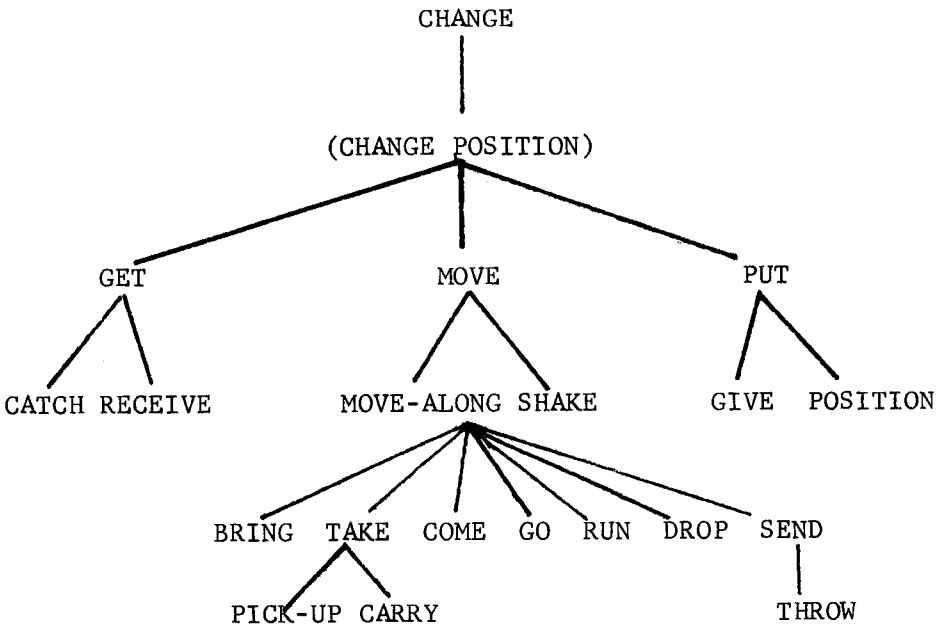


Fig. 4.8

Verbs involving change of position

BASIC ENGLISH

OPERATIONS ETC.

100

COME
GET
GIVE
GO
KEEP
LET
MAKE
PUT
SEEM
TAKE
BE
DO
HAVE
SAY
SEE
SEND
MAY
WILL
ABOUT
ACROSS
AFTER
AGAINST
AMONG
AT
BEFORE
BETWEEN
BY
DOWN
FROM
IN
OFF
ON
OVER
THROUGH
TO
UNDER
UP
WITH
AS
FOR
OF
TILL
THAN
A
THE

THINGS

400 General

ACCOUNT
ACT
ADDITION
ADJUSTMENT
ADVERTISEMENT
AGREEMENT
AIR
AMOUNT
AMUSEMENT
ANIMAL
ANSWER
APPARATUS
APPROVAL
ARGUMENT
ART
ATTACK
ATTEMPT
ATTENTION
ATTRACTION
AUTHORITY
BACK
BALANCE
BASE
BEHAVIOUR
BELIEF
BIRTH
BIT
BITE
BLOOD
BLOW
BODY
BRASS
BREAD
BREATH
BROTHER
BUILDING
BURN
BURST
BUSINESS
BUTTER
CANVAS
CARE
CAUSE
CHALK
CHANCE
EDUCATION
EFFECT
END
ERROR
EVENT
EXAMPLE
EXCHANGE
EXISTENCE
EXPANSION
EXPERIENCE
EXPERT
FACT
FALL
FAMILY
FATHER
FEAR
FEELING
FICTION
FIELD
FIGHT
FIRE
FLAME
FLIGHT
FLOWER
FOLD
FOOD
FORCE
FORM
FRIEND
FRONT
FRUIT
GLASS
GOLD
GOVERNMENT
GRAIN
GRASS
GRIP
GROUP
GROWTH
GUIDE
HARBOUR
HARMONY
HATE
HEARING
HEAT
METAL
MIDDLE
MILK
MIND
MINE
MINUTE
MIST
MONEY
MONTH
MORNING
MOTHER
MOTION
MOUNTAIN
MOVE
MUSIC
NAME
NATION
NEED
NEWS
NIGHT
NOISE
NOTE
NUMBER
OBSERVATION
OFFER
OIL
OPERATION
OPINION
ORDER
ORGANIZATION
ORNAMENT
OWNER
PAGE
PAIN
PAINT
PAPER
PART
PASTE
PAYMENT
PEACE
PERSON
PLACE
PLANT
PLAY
PLEASURE
SENSE
SERVANT
SEX
SHADE
SHAKE
SHAME
SHOCK
SIDE
SIGN
SILK
SILVER
SISTER
SIZE
SKY
SLEEP
SLIP
SLOPE
SMASH
SMELL
SMILE
SMOKE
SNEEZE
SNOW
SOAP
SOCIETY
SON
SONG
SORT
SOUND
SOUP
SPACE
STAGE
START
STATEMENT
STEAM
STEEL
STEP
STITCH
STONE
STOP
STORY
STRETCH
STRUCTURE
SUBSTANCE
SUGAR

200 Pictured

ANGLE
ANT
APPLE
ARCH
ARM
ARMY
BABY
BAG
BALL
BAND
BASIN
BASKET
BATH
BED
BEE
BELL
BERRY
BIRD
BLADE
BOARD
BOAT
BONE
BOOK
BOOT
BOTTLE
BOX
BOY
BRAIN
BRAKE
BRANCH
BRICK
BRIDGE
BRUSH
BUCKET
BULB
BUTTON
CAKE
CAMERA
CARD
CARRIAGE
CART
CAT
CHAIN
CHEESE
CHEST
KNEE
KNIFE
KNOT
LEAF
LEG
LIBRARY
LINE
LIP
LOCK
MAP
MATCH
MONKEY
MOON
MOUTH
MUSCLE
NAIL
NECK
NEEDLE
NERVE
NET
NOSE
NUT
OFFICE
ORANGE
OVEN
PARCEL
PEN
PENCIL
PICTURE
PIG
PIN
PIPE
PLANE
PLATE
PLOUGH
POCKET
POT
POTATO
PRISON
PUMP
RAIL
RAT
RECEIPT
RING
ROD

QUALITIES

100 General

ABLE
ACID
ANGRY
AUTOMATIC
BEAUTIFUL
BLACK
BOILING
BRIGHT
BROKEN
BROWN
CHEAP
CHEMICAL
CHIEF
CLEAN
CLEAR
COMMON
COMPLEX
CONSCIOUS
CUT
DEEP
DEPENDENT
EARLY
ELASTIC
ELECTRIC
EQUAL
FAT
FERTILE
FIRST
FIXED
FLAT
FREE
FREQUENT
FULL
GENERAL
GOOD
GREAT
GREY
HANGING
HAPPY
HARD
HEALTHY
HIGH
HOLLOW
IMPORTANT
KIND

50 Opposites

AWAKE
BAD
BENT
BITTER
BLUE
CERTAIN
COLD
COMPLETE
CRUEL
DARK
DEAD
DEAR
DELICATE
DIFFERENT
DIRTY
DRY
FALSE
FEEBLE
FEMALE
FOOLISH
FUTURE
GREEN
ILL
LAST
LATE
LEFT
LOOSE
LOUD
LOW
MIXED
NARROW
OLD
OPPOSITE
PUBLIC
ROUGH
SAD
SAFE
SECRET
SHORT
SHUT
SIMPLE
SLOW
SMALL
SOFT
SOLID

EXAMPLES OF WORD ORDER

THE
CAMERA
MAN
WHO
MADE
AN
ATTEMPT
TO
TAKE
A
MOVING
PICTURE
OF
THE
SOCIETY
WOMEN
BEFORE
THEY
GOT
THEIR
HATS
OFF
DID
NOT
GET
OFF
THE
SHIP
TILL
HE
WAS
QUESTIONED
BY
THE
POLICE

WE
WILL
GIVE
SIMPLE
RULES
TO
YOU
NOW

ALL
ANY
EVERY
NO
OTHER
SOME
LITTLE
MUCH
SUCH
THAT
THIS
I
HE
YOU
WHO
AND
BECAUSE
BUT
OR
IF
THOUGH
WHILE
HOW
WHEN
WHERE
WHY
AGAIN
EVER
FAR
FORWARD
HERE
NEAR
NOW
OUT
STILL
THEN
THERE
TOGETHER
WELL
ALMOST
ENOUGH
EVEN
NOT
ONLY
QUITE
SO
VERY
TOMORROW
YESTERDAY
NORTH
SOUTH
EAST
WEST
PLEASE
YES

CHANGE
CLOTH
COAL
COLOUR
COMFORT
COMMITTEE
COMPANY
COMPARISON
COMPETITION
CONDITION
CONNECTION
CONTROL
COOK
COPPER
COPY
CORK
COTTON
COUGH
COUNTRY
COVER
CRACK
CREDIT
CRIME
CRUSH
CRY
CURRENT
CURVE
DAMAGE
DANGER
DAUGHTER
DAY
DEATH
DEBT
DECISION
DEGREE
DESIGN
DESIRE
DESTRUCTION
DETAIL
DEVELOPMENT
DIGESTION
DIRECTION
DISCOVERY
DISCUSSION
DISEASE
DISGUST
DISTANCE
DISTRIBUTION
DIVISION
DOUBT
DRINK
DRIVING
DUST
EARTH
EDGE
HELP
HISTORY
HOLE
HOPE
HOUR
HUMOUR
ICE
IDEA
IMPULSE
INCREASE
INDUSTRY
INK
INSECT
INSTRUMENT
INSURANCE
INTEREST
INVENTION
IRON
JELLY
JOIN
JOURNEY
JUDGE
JUMP
KICK
KISS
KNOWLEDGE
LAND
LANGUAGE
LAUGH
LAW
LEAD
LEARNING
LEATHER
LETTER
LEVEL
LIFT
LIGHT
LIMIT
LINEN
LIQUID
LIST
LOOK
LOSS
LOVE
MACHINE
MAN
MANAGER
MARK
MARKET
MASS
MEAL
MEASURE
MEAT
MEETING
MEMORY
POINT
POISON
POLISH
PORTER
POSITION
POWDER
POWER
PRICE
PRINT
PROCESS
PRODUCE
PROFIT
PROPERTY
PROSE
PROTEST
PULL
PUNISHMENT
PURPOSE
PUSH
QUALITY
QUESTION
RAIN
RANGE
RATE
RAY
REACTION
READING
REASON
RECORD
REGRET
RELATION
RELIGION
REPRESENTATIVE
REQUEST
RESPECT
REST
REWARD
RHYTHM
RICE
RIVER
ROAD
ROLL
ROOM
RUB
RULE
RUN
SALT
SAND
SCALE
SCIENCE
SEA
SEAT
SECRETARY
SELECTION
SELF
SUGGESTION
SUMMER
SUPPORT
SURPRISE
SWIM
SYSTEM
TALK
TASTE
TAX
TEACHING
TENDENCY
TEST
THEORY
THING
THOUGHT
THUNDER
TIME
TIN
TOP
TOUCH
TRADE
TRANSPORT
TRICK
TROUBLE
TURN
TWIST
UNIT
USE
VALUE
VERSE
VESSEL
VIEW
VOICE
WALK
WAR
WASH
WASTE
WATER
WAVE
WAX
WAY
WEATHER
WEEK
WEIGHT
WIND
WINE
WINTER
WOMAN
WOOD
WOOL
WORD
WORK
WOUND
WRITING
YEAR
CHIN
CHURCH
CIRCLE
CLOCK
CLOUD
COAT
COLLAR
COMB
CORD
COW
CUP
CURTAIN
CUSHION
DOG
DOOR
DRAIN
DRAWER
DRESS
DROP
EAR
EGG
ENGINE
EYE
FACE
FARM
FEATHER
FINGER
FISH
FLAG
FLOOR
FLY
FOOT
FORK
FOWL
FRAME
GARDEN
GIRL
GLOVE
GOAT
GUN
HAIR
HAMMER
HAND
HAT
HEAD
HEART
HOOK
HORN
HORSE
HOSPITAL
HOUSE
ISLAND
JEWEL
KETTLE
KEY
ROOF
ROOT
SAIL
SCHOOL
SCISSORS
SCREW
SEED
SHEEP
SHELF
SHIP
SHIRT
SHOE
SKIN
SKIRT
SNAKE
SOCK
SPADE
SPONGE
SPOON
SPRING
SQUARE
STAMP
STAR
STATION
STEM
STICK
STOCKING
STOMACH
STORE
STOVE
TABLE
TAIL
THREAD
THROAT
THUMB
TICKET
TOE
TONGUE
TOOTH
TOWN
TRAIN
TRAY
TREE
TROUSERS
UMBRELLA
WALL
WATCH
WHEEL
WHIP
WHISTLE
WINDOW
WING
WIRE
WORM
LIKE
LIVING
LONG
MALE
MARRIED
MATERIAL
MEDICAL
MILITARY
NATURAL
NECESSARY
NEW
NORMAL
OPEN
PARALLEL
PAST
PHYSICAL
POLITICAL
POOR
POSSIBLE
PRESENT
PRIVATE
PROBABLE
QUICK
QUIET
READY
RED
REGULAR
RESPONSIBLE
RIGHT
ROUND
SAME
SECOND
SEPARATE
SERIOUS
SHARP
SMOOTH
STICKY
STIFF
STRAIGHT
STRONG
SUDDEN
SWEET
TALL
THICK
TIGHT
TIRED
TRUE
VIOLENT
WAITING
WARM
WET
WIDE
WISE
YELLOW
YOUNG
SPECIAL
STRANGE
THIN
WHITE
WRONG

NO 'VERBS'

IT
IS
POSSIBLE
TO
GET
ALL
THESE
WORDS
ON
THE
BACK
OF
A
BIT
OF
NOTEPAPER
BECAUSE
THERE
ARE
NO
'VERBS'
IN
BASIC
ENGLISH

A
WEEK
OR
TWO
WITH
THE
RULES
AND
THE
SPECIAL
RECORDS
GIVES
COMPLETE
KNOWLEDGE
OF
THE
SYSTEM
FOR
READING
OR
WRITING

RULES

ADDITON OF 'S'
TO THINGS WHEN
THERE IS
MORE THAN ONE

ENDINGS
IN 'ER,' 'ING,' 'ED'
FROM 300 NAMES
OF THINGS

'LY' FORMS
FROM
QUALITIES

DEGREE
WITH
'MORE' AND 'MOST'

QUESTIONS
BY CHANGE OF
ORDER,
AND 'DO'

FORM-CHANGES IN
NAMES OF ACTS,
AND 'THAT,' 'THIS,'
'I,' 'HE,' 'YOU,'
'WHO,' AS IN
NORMAL ENGLISH

MEASURES
NUMBERS
DAYS, MONTHS
AND THE
INTERNATIONAL
WORDS
IN ENGLISH
FORM

THE
ORTHOLOGICAL
INSTITUTE
LONDON

Basic English

There are a great many verbs in English, however, many of these seem so obscure that it is improbable that a theory of the language would hinge on their treatment. In an attempt to get a small set adequate for the development of a theory we have turned to Basic English.

Basic English is a subset of English developed by Ogden in the 1930's as a proposed international language. Ogden sought to simpify things by using only the verbs: come, get, give, go, keep, let, make, put, seem, take, be, do, have, say, see, send, may, and will. He then included many nominals derived from verbs, which have been taken as verbs for this data. In his selection of words Ogden used criteria very suitable to our purposes here. As he says (p. 11) "For all practical purposes, there are objects which we wish to talk about, the operations which we perform on them, and the directions in which we operate. When the most necessary names, the most fundamental operation-words, and the essential directives have been determined, it can be shown that a verb is primarily a symbolic device for telescoping an operation and an object or a direction (enter for go into). Sometimes an operation-word, a directive, and a name are thus telescoped, as in the word disembark (get, off, a ship)."

It was Ogden's hypothesis that the telescoped words could be removed. Ogden gave 850 words in his basic list and indicated that this should be extended by about 150 words for a given field. We have added the verbs which arose in the protocols of two managers asking questions

about production data, and a few others in order to flesh out the picture which was being formed.

We have classified these verbs in a heirarchy along the lines just explained. Each of them applies to one or more of the physical, system, plans and actions, and mental worlds. Some have meanings which carry over by analogy from one world to the next. Our top level physical world actions are physically-orient, constrain, notice, remain, change, destroy, consume, devide, make, emit, bear, combine, and touch. It is interesting to compare our list with Shank's.

```
ACTIVITY
  PHYSICALLY-ORIENT
        STAND
        SIT
        LIE
        POINT
  CONSTRAIN
        SUPPORT
        CONNECT
        SURROUND
        CONTAIN
        COVER
  NOTICE
        LISTEN-TO
        LOOK-AT
        SMELL
        TOUCH
        TASTE
  REMAIN
        LIVE
        SLEEP
  CHANGE
        (CHANGE POSITION)
          (GET PHYSICAL-OBJECT)
                CATCH
                RECEIVE
                PUT
                        SET
                        PLACE
                        POSITION
                        EMBED
                        DISTRIBUTE
                        DISSOLVE
                        EXCHANGE
                        GIVE
                MOVE
                        SHAKE
                        SHIVER
                        MOVE-ALONG
                          COME
                            ARRIVE
                            ENTER
                          GO
                            LEAVE
                            DEPART
                            DISAPPEAR
                          RIDE
                          RUN
                          WALK
                          FLY

                          SWIM
                          ROLL
                          FALL
                          JUMP
                          SLIP
```

DROP
FORCE
  PUSH
  PULL
ACCOMPANY
SEND
  KNOCK
  KICK
  THROW
TAKE
  PICK-UP
  CARRY
BRING

(CHANGE DIRECTION)
  TURN
(CHANGE FORM)
  FORM
  TWIST
  CRACK
  BURST
  EXPAND
BECOME
  GROW
  INCREASE
  DECREASE
  FLUCTUATE
(CHANGE STATE)
  FREEZE
  DIE
  COOK
  BAKE
  BURN
  WAKE-UP
PROCESS
  KILL
  RUB
    POLISH
  FOLD
  HEAT
  MARK
DAMAGE
  BREAK
    SMASH
  CRUSH
  BITE
DESTROY
CONSUME
  EAT
  DRINK
  SWALLOW
DIVIDE
MAKE
  (MAKE SOUND)
    COUGH
    SNEEZE
    CRY

```
                ROAR
                LAUGH
                TALK
                SPEAK
        PRODUCE
                FLOWER
                MANUFACTURE
                BUILD
                PRINT
EMIT
        SMOKE
        STREAM
BEAR
COMBINE
        ADD
TOUCH
        HIT
        STEP-ON
```

Fig. 4.9

Activities in the physical world.

```
ACTIVITY
     CONTRAIN
         SUPPORT-FINANCIALLY
                     LIKE
                     PREFER
                     LOVE
                     WANT
                     HOPE
                     DISLIKE
                     MIND

     CHANGE
         (CHANGE POSITION)
                          (GET SOLIAL-ENTITY)
                              (BUY OWNERSHIP)
                              (RECIVE OWNERSHIP)
                          (GIVE SOCIAL-ENTITY)
                              LET
                              PROVIDE
                              CONTRIBUTE
                              SUPPLY
                              SELL
                              TRADE
                          MOVE
                             MOVE-ALONG
                                  COME
                                  GO
                                     (LEAVE SOCIAL-SCENE)
                                  SLIP-SOCIALLY
                                  DROP-OUT
                                  RUN-FROM-RESPONSIBILITY
                                  RISE-SOCIALLY
                                  FALL-SOCIALLY
                                  FORCE
                                     PUSH
                                     PULL
                                     RUN
                                     DRIVE
                                     RIDE
                                     LEAD
                                     DIRECT
                                     GUIDE
                                     DELIVER
          (CHANGE FORM)
                          (EXPAND SYSTEM)
                          SPECIALIZE
          BECOME
                        SUCCEED
                        MATURE
                        FAIL

                    CONTROL
                          MANAGE
```

CHANGE-MOOD
REWARD
PUNISH
AMUSE
COMFORT
DISGUST
HUMOR
SHOCK
PLEASE
SURPRISE
TROUBLE
ALIENATE
INTEREST
ATTRACT
FLATTER
(BREAK ORGANIZATION)
(SMASH ORGANIZATION)

DISBAND
DISCHARGE
(DIVIDE ORGANIZATION)
MAKE
(MAKE RULE)
RULE
BEFRIEND
ORGANIZE
FOUND
MERGE
COMPLETE
FIGHT
MEET

Fig. 4.10

Activities in the world of systems