Technical Report 291

Assimilation of New Information by a Natural Language Understanding System

Drew V. Mcdermott

MIT Artificial Intelligence Laboratory

This blank page was inserted to preserve pagination.

ASSIMILATION OF NEW INFORMATION BY A NATURAL LANGUAGE-UNDERSTANDING SYSTEM

Drew Vincent McDermott

Artificial Intelligence Laboratory

Massachusetts Institute of Technology

Cambridge, Massachusetts 02139

Abstract

This work describes a program, called TOPLE, which uses a procedural model of the world to understand simple declarative sentences. It accepts sentences in a modified predicate calculus symbolism, and uses plausible reasoning to visualize scenes, resolve ambiguous pronoun and noun phrase references, explain events, and make conditional predictions. Because it does plausible deduction, with tentative conclusions, it must contain a formalism for describing its reasons for its conclusions and what the alternatives are. When an inconsistency is detected in its world model, it uses its recorded information to resolve it, one way or another. It uses simulation techniques to make deductions about other creatures' motivation and behavior, assuming they are goal-directed beings like itself.

This report reproduces a thesis of the same title submitted to the Department of Electrical Engineering, Massachusetts Institute of Technology, in partial fulfillment of the requirements for the Degrees of Bachelor of Science and Master of Science.

Acknowledgments

This work would not have been possible without extensive discussions with workers at the MIT A.I. Laboratory, especially Gerry Sussman, Scott Fahlman, Terry Winograd, Andee Rubin, and Eugene Charniak. If I had listened to them more, my thesis would be better. My entire approach was influenced (some might say perverted) by some of the philosophical ideas about intelligence that are to be picked up around here. Many of these are traceable to Marvin Minsky and Carl Hewitt. I would also like to thank Allison Platt for drafting the illustrations, and Tomas Lozano for final editing.

Table of Contents

Illustrations

I. Introduction

This is a report on an approach to the solution of a very small part of the problem of telling computers things in English. This is a difficult problem because language is a complex symbolic system which is ultimately as deep as what it represents--the world. To some degree, the loose syntactic regularities in utterances determine their meaning, but there is no complete analysis of inessential syntactic and lexical ambiguities that is not a theory of linguistic understanding to some extent.

In perceiving speech, as in perceiving anything, plausible reasoning must come into play. If seeing were a matter of building up descriptions of lines, regions, objects, and scenes, step by step, a hierarchy of rules could be applied to a visual input until a useful description emerged. (Cf. Waltz, 1972) However, technology will probably never be good enough to find an edge in a scene independent of context, so application of the rules to real-world data results in finding impossible or ambiguous structures. Then one must give consideration to various heuristics which suggest ways of restoring missed lines, askew vertices, etc.

Linguistic utterances are at least as bad as visual snapshots. The phonemes we think are there can be found only by

re-analyzing input in the light of a preliminary analysis that suggests what might have been said. In speech, ambiguity is a more pervasive problem than in vision; even written language, whose characters are relatively simple to recognize, is full of syntactic and semantic ambiguities. Parsers can grind out hundreds of arrangements of sentences of moderate length (Coles, 1968), even when these sentences appear unambiguous to a human being.

In vision, ambiguities are resolved by answering the question, "What is most likely to be out there, given what I know is there and what has been there up until now?" Inessential linguistic ambiguity (i.e., ambiguities which are not noticed in context by humans) can be reduced by considering the question, "What is the current speaker most likely to have said, given the possible alternative interpretations and the overall situation, especially the conversation, so far?" This question involves several sub-questions:

1. What are the speaker's overall and immediate goals? Does he wish to inform, discover, supervise, cajole, flatter, amuse, threaten, or kill time? Is he now telling, asking, commanding, responding, clarifying, or emoting?

2. What linguistic devices are being used to further a speaker's intentions? Which of several interpretations is most likely purely on syntactic grounds? For example, can syntactic rules (such as restrictions on reflexives) be used in determining possible pronoun referents?

3. In considering a speaker's immediate goals, what is the local discourse context? Did someone just ask a question that he might be answering? Did he just say something that seems to require explanation? In general, can I expect his speech to fall into a predictable frame because of some linguistic institution?

4. Of the possible meanings of a sentence, which is most likely on logical grounds? For declaratives, which interpretation is easiest to believe? For imperatives, which is easiest to carry out? For questions, which is easiest to answer in an informative way?

5. What are the relative states of knowledge of the speaker and hearer? Which interpretations of a statement are known by the speaker but not believed (by the speaker) to be known by the hearer? Which possible meanings of a question are unknown by the speaker, but are believed (by the speaker) to be known by his hearer?

Of these questions, I have chosen to work on (4), and only on declaratives. This choice was made for two reasons:

1. With a declarative-assimilator alone, the most straightforward, neutral linguistic scenario can be explored: a speaker honestly informing a hearer about a situation. With this restriction, the overriding consideration in choosing the correct interpretation of each sentence is, can it be believed with minimum effort? A full simulation would require other machinery, for considering the longer-range motives of the narrator; for questioning whether he believes an interpretation; and whether he believes the hearer believes it; etc. But ignoring these questions is not immediately fatal.

2. The plausible-reasoning machinery which assimilates beliefs into a world model would seem to underlie the other aspects of linguistic understanding, and, indeed, of all understanding. Until a representation and dynamic world model have been built which are capable of understanding simple actions, motivations, and belief systems, it would seem to be futile to consider questions of linguistic goals, methods, and knowledge.

This paper reports on a program (named TOPLE, for "top level") that attempts to understand new sentences about a simple world by using a set of programs which embody a logical model of that world. It includes an understanding of the sorts of causal progressions and assumed presuppositions that are a large part of the unspoken, but important, contents of speech. It does not deal directly with English sentences, but interprets simple semantic structures such as might be produced by a natural-language parser. Ambiguities are preserved in this notation, explicitly or by use of ambiguous constructions in the semantic language (SL). Input is in the form of paragraphs, groups of SL formulae which are to be understood as a unit. The program digests these by deciding which interpretation is most likely. It adds assertions in the unambiguous internal language to its world model. It reports on the assumptions it makes and some of the difficulties it encounters.

The program's model of the world is represented by sets of items and methods in the CONNIVER programming language (McDermott and Sussman, 1972). This language makes it easy to construct alternative models of the world, each representing a separate conjecture about its static and dynamic structure. It also makes it relatively simple to write programs to consider the alternatives in complex, interdependent ways.

The program TOPLE operates as follows. Each formula of the

SL causes a tree of hypothetical worlds to be created and pondered, one for each interpretation of it. This tree is built by a function named PLAUSIBLE? which calls methods to turn SL formulae into assertions. Assertions are modeled as items with property-list structures that relate them to other beliefs in ways TOPLE's subsystems can utilize. Such methods and their subroutines attempt to fit what they are told into what they know with as few conflicts as possible. If a formula cannot be understood by itself, the tree of possible worlds is preserved while succeeding SL formulae are read, which hopefully resolve the problems and point to a single interpretation.

The overall static structure of TOPLE is illustrated by Fig. 1. The center of the system is marked "WORLD MODEL"; this is a set of if-needed methods and items which embody TOPLE's knowledge of the world. (See next chapter.) They are called, as described in Chapter III, by PLAUSIBLE?, either to deduce consequences of already-known items, or to make changes to the set of items. The methods call the functions PLAUSIBLE?, ASSUME, and DOUBT to effect needed dependent changes. The boxes marked "if-added methods" and "ring reconstructors" contain programs with additional knowledge about making additions to or deletions from the world model. These are explained in Chapter III.

If-needed methods can run in two modes; when making data base changes (BELIEVE+ mode), calls to DOUBT and ASSUME are allowed;

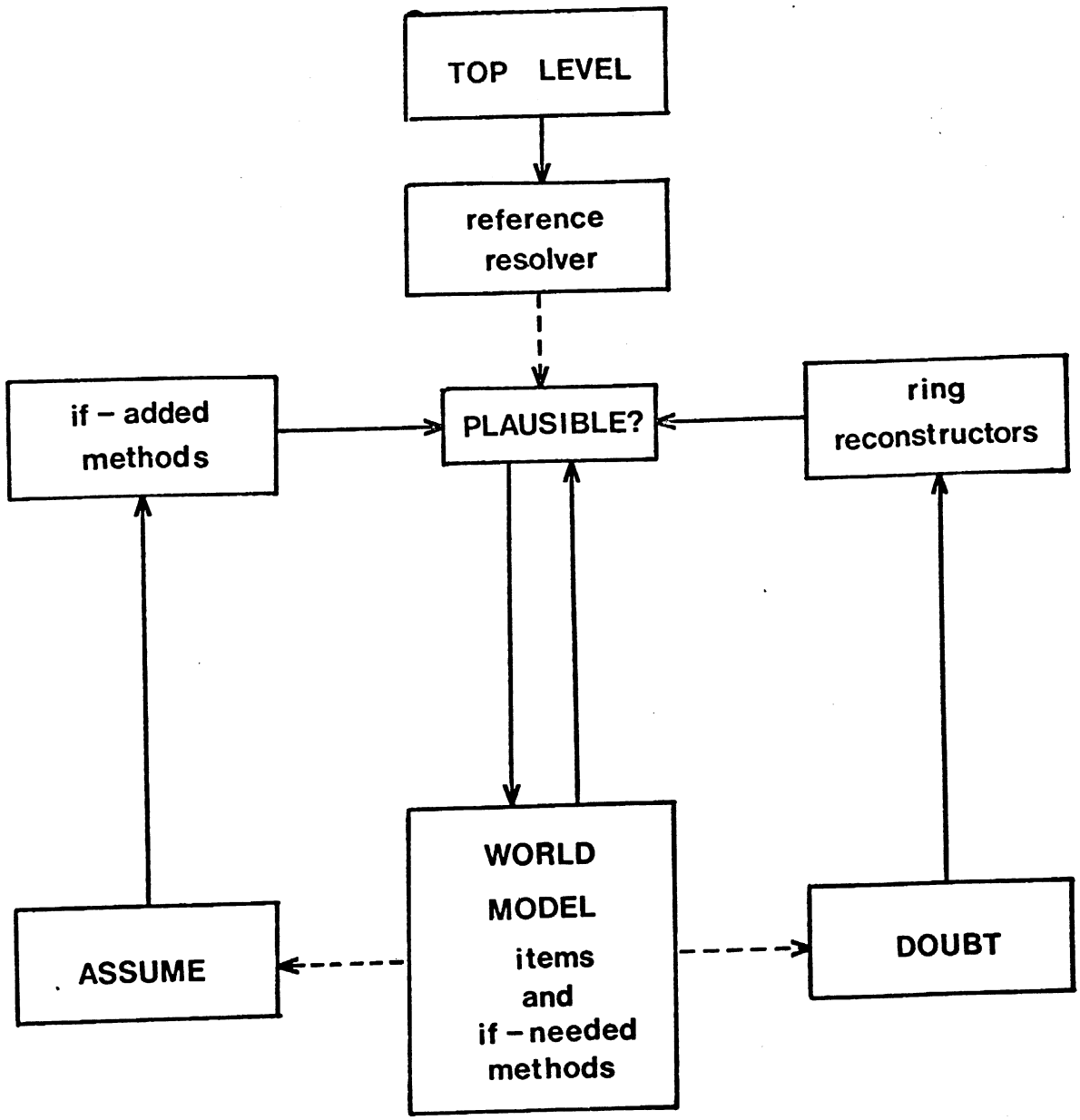

TOP LEVEL
reference resolver
if – added methods
PLAUSIBLE?
ring reconstructors
WORLD MODEL
items and if – needed methods
ASSUME
DOUBT
means "calls"
means "calls to change model" (BELIEVE+ mode)

figure 1

otherwise (DEDUCE mode), only additions of deduced items are allowed.

Fig. 1. illustrates only the static configuration of TOPLE. The context structure it builds as it runs will be described later.

Although, in some sense, question-answering ability is a measure of understanding, TOPLE does not answer questions. The user must interrogate the data base "by hand," or keep track of all the assumptions the program has made.

The world that TOPLE lives in is almost as simple as the BLOCKS world of Winograd's (1971) language understander, but involves more complex notions of process and action. The world is that of monkey and experimenter in a single room. The monkey (named Spiro) is capable of several actions, and driven by simple motives of hunger, thirst, and curiosity. Wolfgang, the experimenter, can do similar things, and his motivation is assumed to be to see what Spiro will do in response to his actions. The contents and characteristics of the room are not as capturable as in the BLOCKS world; the program tries to make allowances for sloppiness and incompleteness in describing the layout of the room. The program listens to a present-tense account of goings-on in this room, and attempts to understand why what happens happens, and what can be expected as the story progresses. It tells us at the end of every paragraph what new

assertions it has assumed as a result of hearing it.

As an example, consider the following "protocol" of a conversation with TOPLE. In each case, I give a paragraph in English, followed by the equivalent SL formulae. After each formula of a paragraph, I have sandwiched in a description of the actions and assumptions the machine takes. (The bracketed numbers identify such actions and assumptions. They are not generated by TOPLE.)

"The banana is under the table, by the ball."

```
(AT (THE BAN1 (IS BAN1 BANANA))
    (PLACE (THE TAB1 (IS TAB1 TABLE) UNDER)))
```

The SL term (THE var -predicate-formulae-) is intended to refer to the object such that all the predicate formulae are true if the name of the object is substituted for var in all of them. The program finds the references first and attempts to resolve them. In fact, it knows of no banana or table, so its response is to let BAN1 and TAB1 name two new objects. The program then files away

```
(AT BAN1 (PLACE TAB1 UNDER))          [1]
(AT TAB1 (PLACE FLOOR1 ON))           [2]
```

as its response. The latter statement is an <u>assumption</u> made by the spatial reasoning routines when they hear of something

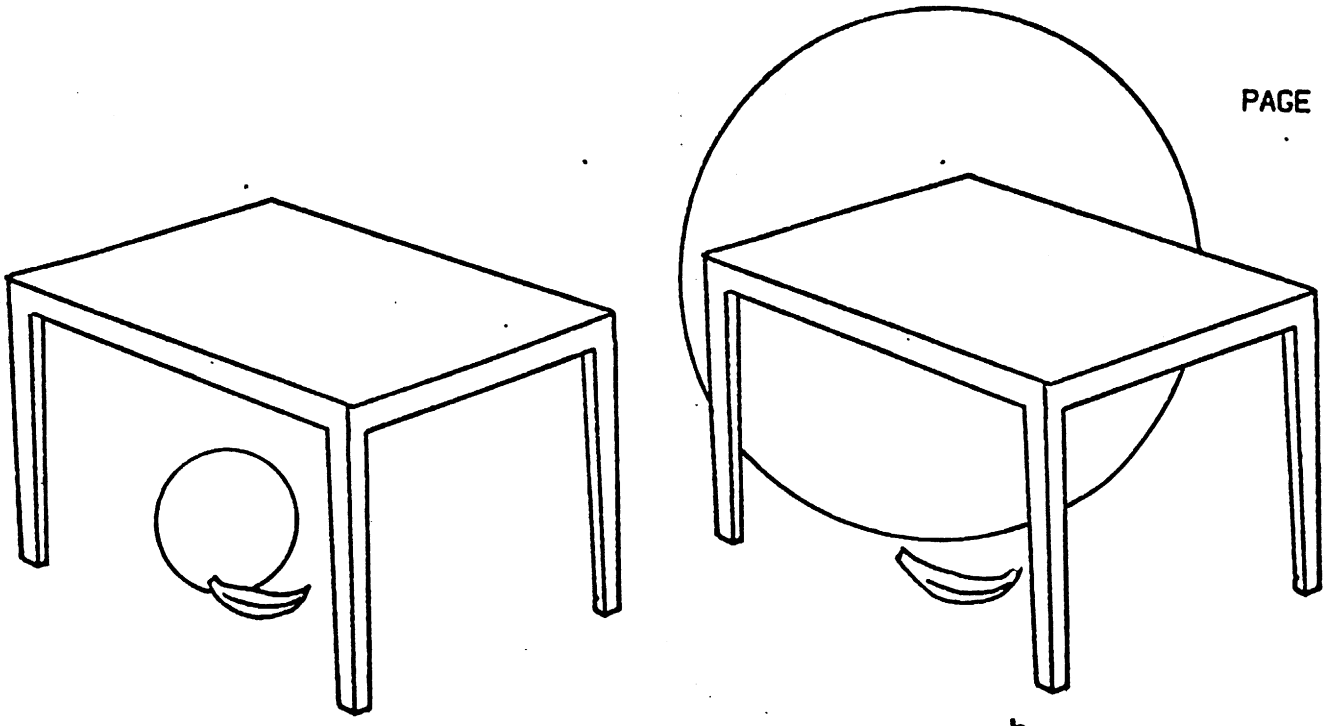
without knowing its supports.

(AT BAN1 (PLACE (THE BALL1 (IS BALL1 BALL)) BY))

The second formula of the paragraph uses BAN1 as well; hence, "BAN1" can be said to be "bound globally," rather than, e.g., in the first formula. The reference is resolved as the others were, and

(AT BAN1 (PLACE BALL1 BY)) [3]
(AT BALL1 (PLACE TAB1 UNDER)) [4]

become its new "visualization" of the situation. This system of beliefs corresponds to accepting Fig. 2(a) rather than Fig. 2(b), or some representation of indifference or uncertainty between them. The motivation for forcing a choice is important and will be dealt with; this particular choice is based on its belief that the object sizes of Fig. 2(a) are more likely.



a b

figure 2

"The monkey goes over to the table and picks up the banana."

```
(AND
    (GO (THE MONK1 (IS MONK1 MONKEY))
        (PLACE (THE TAB2 (IS TAB2 TABLE))))
    (PICK-UP MONK1 (THE BAN2 (IS BAN2 BANANA))))
```

The two SLF's representing the clauses of this paragraph are not unambiguously related, as they were before. AND is an SL operator, which has (among other meanings) the force of a claim that its arguments are "simultaneously" true, or that they are temporally (and possibly causally) related. The first embodies the meaning of "and" in, "He made brownies and baked a cake," implying at least logical simultaneity of actions; the second, that in, "He stuck his finger in and wiggled it around." (Other meanings of "and" are beyond the reach of the program so far.)

When the program encounters such an ambiguity, it is capable of splitting into two or more parallel investigations to decide which interpretation is most likely. However, in the case of AND, there is a nicer way to resolve the ambiguity, at least in

simple cases. This is to hand the two clauses of the conjunction to the plausibility checkers in order, and allow the second one to be believed in whatever situation the first one leaves. For simple actions, this reduces to assuming AND means "first one, then the other"; for conditions, that it means "both now." This interpretation would result in a misreading of "He made brownies and baked a cake," but not for most action sequences. (It is also arguable that people are prejudiced toward the interpretation of this sentence that the brownies were made first.)

This way of interpreting AND is an instance of TOPLE's general inclination not to investigate unlikely alternatives at all unless some feature of the situation or difficulty in assimilating the first choice arises.

In the case of AND, both alternatives cause it first to try to believe the first formula. The reference to MONK1 is resolved as a reference to Spiro, the only monkey the program knows about. Similarly, TAB2 becomes TAB1, the same table as before.

Consequently, TOPLE must accept (GO SPIRO (PLACE TAB1)). To do so, it must establish that the monkey can and wants to perform this action, its model of the monkey including its properties as a goal-seeking creature as well as as a physical object. It does not doubt that Spiro can go the table, but is uncertain whether he does it out of interest in the table, the ball, or the banana

that are in that spot. There is no reason it can imagine for Spiro being interested in tables, but balls are fun to play with and bananas can be eaten. The uncertainty between these two is reflected in the fact that it generates two alternative world views:

In situation 0:
(WANT SPIRO (HOLD SPIRO BAN1)) [5]
(WANT SPIRO (EAT SPIRO BAN1)) [6]
(HUNGRY SPIRO) [7]
(GO SPIRO (PLACE TAB1)) [8]

In situation 1:
(AT SPIRO (PLACE TAB1)) [9]
(PICK-UP SPIRO BAN1) [10]

In situation 2:
(HOLD SPIRO BAN1) [11]

In situation 3:
(EAT SPIRO BAN1) [12]

In situation 4:
BAN1 ceases to exist [13]

OR

In situation 0:
(WANT SPIRO (HOLD SPIRO BALL1)) [14]
(WANT SPIRO (PLAY SPIRO BALL1)) [15]
(GO SPIRO (PLACE TAB1)) [16]

In situation 5:
(AT SPIRO (PLACE TAB1)) [17]

In situation 6:
(PLAY SPIRO BALL1) [18]
(PICK-UP SPIRO BALL1) [19]

In situation 7:
(HOLD SPIRO BALL1) [20]

Each situation referred to here is a separate state of the world, which represents a state incrementally different from its predecessor as a result of some action. These two sequences of situations are mutually exclusive; situations 1 and 5 are both possible successors to state 0. (Notice that there are two different versions of state 0 the system keeps in mind.) The sequences are generated by simulating Spiro with the goals corresponding to his respective WANTs. "(HUNGRY SPIRO) in situation 0" or "(WANT SPIRO (PLAY SPIRO BALL1)) in situation 0" are the crucial assumptions shown--they were introduced to explain the monkey's behavior.

The program is now half way through the paragraph, with two completely independent world-views to choose from. The first seems most likely, since only hunger need be assumed, but the program suspends judgment.

Now it accepts the sequential interpretation of "and" and attempts to believe (PICK-UP SPIRO BAN1) (it resolves the reference in favor of a banana it knows rather than a new one) in the state produced by the previous action. It fits best with the supposition that the monkey was interested in the banana, since it confirms the sequence of [5]-[13]. Notice that the program has only been told about events through assumption [11]; the beliefs about situations 3 and 4 are predictions. Beliefs [14]-[20] have been collected as garbage and are no longer accessible.

Now TOPLE informs its interlocutor of its assumptions and awaits the next paragraph.

"He eats it."

(EAT (HE M1) (IT T1))

(HE X) is treated as though it were (THE X (REFERRED-TO X) (SEX X MALE)); (IT X), as though it were (THE X (REFERRED-TO X) (IS X INANIMATE)). Here, (HE M1) is obviously Spiro. IT could be several things. The top level program keeps track of referred-to objects, and considers each in reverse chronological order of reference. So, the first object considered is BAN1. (EAT SPIRO BAN1) is so easy to believe (since it was predicted), that the other alternatives are not even considered. States 3 and 4 are accepted as reality.

Note that at this point the banana ceases to exist. The place assertions about it are removed from the data base. (This may cause other assumptions to be made; see Chapter IV.) TOPLE does not know about peels.

"The experimenter attaches a bunch of bananas to the ceiling."

```
(ATTACH (THE E1 (IS E1 EXPERIMENTER))
        (A BUNCH1 (IS BUNCH1 (GROUP-OF BANANA)))
        (THE C1 (IS C1 CEILING)))
```

The program knows of one experimenter, Wolfgang. It assumes C1 means the only ceiling--CEILING1. (A more sophisticated reference resolver could understand "the ceiling" when discussing a room without having to have a ceiling previously in mind.) (GROUP-OF class) means "a group all of whose members are in class." Here we have an indefinite reference, which the program takes to be to an object with no important properties not deducible from those given from now on. Hence, BUNCH1 is a new object. The program assumes

```
In situation 4:
        (ATTACH WOLFGANG BUNCH1 CEILING1)          [21]

In situation 8:
        (AT BUNCH1 (PLACE CEILING1 SUSPENDED))     [22]
```

(I am numbering situations chronologically in order of creation; situation 8 is to be understood as a successor to 4 in this notation even though they are not consecutively numbered.)

When TOPLE hears of the actions of the experimenter, it does not try to simulate him, because his motives are not intelligible to it. The experimenter presumably does things to test the monkey; that is, he wants to find out something about him. TOPLE cannot model states of knowledge different from its own, so it cannot think about "not knowing" or "wanting to know."

However, there is one simple consequence of an action by Wolfgang; it is likely to affect what Spiro does. The program checks Spiro's motivations again and ponders the consequences of them. It knows him to be hungry, since a single banana is not enough to satisfy him, so it runs a new simulation of him planning the conquest of this bunch, with his actions being transformed into predictions (or alternative possible futures) in the "real world."

The obvious general prediction is that Spiro will try to get as close to the bunch of bananas as he can. There are two methods available to him. One is to go to the closest spot to the bananas (under them) and jump for them; the other is to find something to climb onto that will bring them within reach. The table is the obvious candidate for platform in the second strategy, but unfortunately it is too heavy for the monkey to move; it must be under the bananas already for it to be useful. If it is, the machine predicts Spiro will climb it (since he is already standing next to it), and reach for the bananas; whether he will make it or not is unclear. If it is in the wrong place, Spiro will go to the bananas and jump for them. Thus one version of the world has the bananas in the same place as the table; the other, in a different one.

These two incompatible assumptions are represented by

```
(AT TAB1
```

(PLACE FLOOR1 ON (PLACE BUNCH1 UNDER))) [23]

(AT TAB1 (PLACE FLOOR1 ON MOD1)) [24]

(This notation expresses in some fashion that (PLACE FLOOR1 ON) is to be intersected with any places used as modifiers after "ON." However, the notation is asymmetrical because the support relation of the floor is a more important aspect of "at-ness" than the precise location on the floor. Half-baked reasons for this notation will be given much later. (See Chapter IV.) The new modifier MOD1 is recorded as incompatible with (PLACE BUNCH1 UNDER).)

The two (incompatible) predictions that are generated are as follows:

In situation 9:
(CLIMB SPIRO TAB1) [25]

In situation 10:
(AT SPIRO
(PLACE TAB1 ON
(PLACE BUNCH1 BY))) [26]

In situation 11:
(GRAB SPIRO BUNCH1) [27]

In situation 12:
(HOLD SPIRO BUNCH1) [28]

In situation 13:
(EAT SPIRO BUNCH1) [29]

In situation 14:
BUNCH1 ceases to exist [30]

OR

In situation 15:
(GO SPIRO
(PLACE FLOOR1 ON
(PLACE BUNCH1 UNDER))) [31]

In situation 16:
(AT SPIRO
(PLACE FLOOR1 ON
(PLACE BUNCH1 UNDER))) [32]

In situation 17:
(JUMP-UP SPIRO) [33]

In situation 18:
(AT SPIRO (PLACE BUNCH1 BY)) [34]

In situation 19:
(GRAB SPIRO BUNCH1) [35]

In situation 20:
(HOLD SPIRO BUNCH1) [36]

In situation 21:
(FALL SPIRO) [37]

In situation 22:
(AT SPIRO (PLACE FLOOR1 ON)) [38]

In situation 23:
(EAT SPIRO BUNCH1) [39]

In situation 24:
BUNCH1 ceases to exist [40]

These two sequences, [25]-[30] and [31]-[40], are mutually exclusive; states 9 and 15 are both possible successors to situation 8. Which prediction comes true will decide between them.

"Spiro goes over to them and looks up."

```
(AND (GO SPIRO (PLACE (THEM G1)))
     (LOOK SPIRO (PLACE SPIRO OVER)))
```

(THEM X) means (THE X (REFERRED-TO X) (IS X GROUP)), and can only mean BUNCH1. Now (GO SPIRO (PLACE BUNCH1)) would seem to confirm the second alternative prediction. However, that prediction involved the belief that (GO SPIRO (PLACE FLOOR1 ON (PLACE BUNCH1 UNDER))). However, the human speaker has not been so precise; he is allowing context to supply the floor which presumably the monkey will not leave without special mention being made of it. In fact, the usual interpretation of "go" entails that it occurs entirely on the same horizontal plane that it starts on; and that a creature can only go to objects he can have "immediate control" (Charniak, 1972) over when he arrives. This interpretation is handled by a CONNIVER program that amends accounts of going to take such things into account. However, having to make assumptions like that makes TOPLE a little nervous, so it checks the alternatives.

One is afforded by the interpretation of "go" to mean "made it to a destination, by whatever means." (For example, "Kissinger went to Peking.") This meaning causes TOPLE to try to believe (ACHIEVE SPIRO (AT SPIRO (PLACE BUNCH1))). In this case, however, believing this helps in no way to decide between the two alternative predictions. Both predict he will try to do this;

but both cannot be believed because of the incompatible assumptions.

Consequently, the machine tries again at believing (GO SPIRO (PLACE FLOOR1 ON (PLACE BUNCH1 UNDER))). After some trying, it decides this is indeed a confirmation of the prediction based on the table's not being under the bananas; that prediction and its underlying assumption are accepted. State 16 becomes the current situation.

The second clause of the AND causes the assumption:

In situation 16:
(LOOK-AT SPIRO BUNCH1) [41]

There are no relevant prerequisites for (LOOK SPIRO (PLACE SPIRO OVER)). (There would be for, e.g., (LOOK SPIRO (PLACE BOX17 IN)).) The interesting part comes in deciding why it was done. The program (like most people) would always rather express this as LOOK-AT or LOOK-FOR, since these concepts include purpose as well as physical actions. However, because TOPLE cannot model another creature's knowing or not knowing the location of something, LOOK-FOR, which implies an intention to learn it, is not understood. Here, some tedious processing leads it to accept (LOOK-AT SPIRO BUNCH1), an action explainable in terms of his wanting the bunch.

There is still part of the original prediction pending

(namely, that Spiro will jump for the bunch of bananas), and it would ordinarily become the currently accepted future. The next statement, however, was that Spiro would look up. Does this violate the "jumping" prediction? In general, it should not; the jumping could be the next action after looking up. However, the look-up action is strictly incompatible with jumping (or so TOPLE) believes, so it reconsiders the consequences of Spiro's current goal, wanting to have the bananas. Since nothing has changed, the same prediction is regenerated. (The assumption that the table is in the wrong place is now in the general data base, so the possibility of climbing on it does not need to be explored.) This is, strictly speaking, a new prediction, but I will not bother to copy it down with new situation numbers. No mechanisms are built in for deciding that a prediction has been around too long possibly to be correct.

Again, since the second clause required the world to be in the state created by the first, the "simultaneous" interpretation of "and" is not explored.

"Wolfgang places a box in the corner."

```
(PUT WOLFGANG (A BOX1 (IS BOX1 BOX))
              (PLACE (THE CR1 (IS CR1 CORNER) IN)))
```

The current version of the program is aware that a corner is not really a kind of object at all. When it attempts to resolve

a reference to one, it checks to see in what context the phrase "the corner" occurs. If it does not see a PLACE-context, it becomes confused and gives up trying to understand. (For example, it will not understand "He picked up the corner," or "He turned the corner.") It knows that corners are really just subplaces of floors. So it turns the phrase (PLACE (THE CR1 (IS CR1 CORNER))) into

```
(PLACE
   (THE FLOOR2 (IS FLOOR2 FLOOR))
   ON
   (PLACE (THE WALL1 (IS WALL1 WALL)) BY)
   (PLACE (THE WALL2 (IS WALL2 (OTHER WALL))) BY)).
```

That is, a corner is the place on a floor near two walls. Therefore, the correct assumption is:

```
In situation 25:
        (PUT WOLFGANG
             BOX1
             (PLACE FLOOR1 ON
                    (PLACE WALL1 BY)
                    (PLACE WALL2 BY)))                [42]

In situation 26:
       (AT BOX1
           (PLACE FLOOR1 ON
                  (PLACE WALL1 BY)
                  (PLACE WALL2 BY)))                  [43]
```

As usual, the experimenter is assumed to be interested in the monkey's actions, so TOPLE expects Spiro to do something new. He knows he still wants the bananas, so it reasserts his hunger, as

it did when the bananas were first attached to the ceiling. This time, the methods that simulate his climbing approach to the problem have something new to work on, namely BOX1. The program knows boxes are almost as good as tables to use as platforms, and are light enough to be movable. So it predicts the following sequence of events:

```
In situation 27:
        (GO SPIRO
            (PLACE FLOOR1 ON
                   (PLACE WALL1 BY)
                   (PLACE WALL2 BY)))                    [44]

In situation 28:
        (AT SPIRO
            (PLACE FLOOR1 ON
                    (PLACE WALL1 BY)
                    (PLACE WALL2 BY))                    [45]

In situation 29:
        (GRAB SPIRO BOX1)                                [46]

In situation 30:
        (HOLD SPIRO BOX1)                                [47]

In situation 31:
        (GO SPIRO
            (PLACE FLOOR1 ON
                        (PLACE BUNCH1 UNDER)))           [48]

In situation 32:
        (AT BOX1
            (PLACE FLOOR1 ON
                   (PLACE BUNCH1 UNDER)))                [49]

In situation 33:
        (CLIMB SPIRO BOX1)                               [50]
```

In situation 34:
(AT SPIRO
(PLACE BOX1 ON
(PLACE BUNCH1 BY))) [51]

In situation 35:
(GRAB SPIRO BUNCH1) [52]

In situation 36:
(HOLD SPIRO BUNCH1) [53]

In situation 37:
(EAT SPIRO BUNCH1) [54]

In situation 38:
BUNCH1 ceases to exist [55]

And, after all of this, Spiro will no longer be hungry.

Notice that, as with [41], [44] contradicts the current JUMP-UP prediction. However, this time re-predicting causes the JUMP-UP and its sequel ([33] - [40]) to be supplanted, and they vanish.

"Spiro goes over to the box, drags it back to the bananas, and gets them."

(AND
(GO SPIRO (PLACE (THE BOX2 (IS BOX2 BOX))))
...)

Not the same as the prediction [44], but it <u>implies</u> [44]. When the new features of a predicted situation imply or are implied by what happens, the predicted state is identified with the actual state that occurs. In a case like this, the actual statement is believed.

```
(DRAG SPIRO (IT T2)
      (PLACE (THE BUNCH2 (GROUP-OF BANANA))))
```

After easy reference resolving, this is translated into the operations [46] - [48], with substitution as before of (PLACE FLOOR1 ON (PLACE BUNCH1 UNDER)) for (PLACE BUNCH1).

```
(ACHIEVE SPIRO (HOLD SPIRO (THEM G2)))
```

To believe an "ACHIEVE" statement, the machine must explain why the achievement is attempted (an easy task by now), and be reasonably sure of how. (As suggested before, a reference to an achievement, like a reference to anything else, must suggest something known clearly to both speaker and hearer.) In this case, there is only one possible future, leading to a prediction that the monkey will climb BOX1 and grab the bananas. This is accepted as fact.

"He eats the bananas and goes to sleep."

```
(AND (EAT (HE M2)
          (THE BUNCH4 (IS BUNCH4 (GROUP-OF BANANA))))
     (GO M2 (PLACE SLEEP)))
```

The eating statement requires nothing new to be believed, as it was predicted. The second is treated as a special construction by the GO-understanding routine, that is translated into (SLEEP SPIRO). This interpretation makes sense in light of Spiro's little meal, since eating is quite capable of making him sleepy.

In situation 38:
(SLEEP SPIRO) [56]

At the end, the program believes Spiro is no longer hungry, and leaves him peacefully sleeping. (Somewhat naively, it leaves him on top of the box.)

Relation to Other Work

Few other workers have touched on exactly the problem I am interested in, namely, maintaining a coherent (not just consistent) model of the world. Other question-answering systems have been told to think about new things they are given, but not in the way I have described. For example, SIR (Raphael, 1964) could detect simple inconsistency between some statements and its data base, and refuse to believe them. Green's (1969b) QA3 question-answerer tests for contradiction or tautology in new clauses before storing them. Winograd's (1971) system treats some declarative sentences as questions, which it tries to answer, but does not store as new information if it fails; others are taken at face value without any checking.

All these systems react to contradiction by (at most) rejecting the latest contradictory datum. A closer match to my approach is that of Colby and Smith (1969). They are attempting

to model belief systems in which the machine maintains information regarding the credibility of each piece of knowledge it has and each person it talks to. Their work differs from mine in several ways. First, all knowledge is in the form of simple syllogisms, regarding (static) class inclusion. Second, credibility is computed numerically on the basis of a simple function of the foundation and consistency of a proposition. Since I am interested in debugging a data base, my system requires symbolic information regarding just what the relation is between one statement and the rest of the data; hence, I store a summary of TOPLE's reasons for a given change to its world model, but nothing so concise as a single number. Third, Colby is primarily studying belief, whereas I am interested in linguistic communication. My program will always try to assume that incredibilities are due to its lack of understanding, and attempt to correct it, rather than mark its interlocutor as an incredible source of information.

Abelson and his associates (Abelson and Carroll, 1965; Abelson and Reich, 1969; Abelson, 1973) have studied belief systems from a point of view different from mine; like Colby, they are more interested in simulating belief than in including belief in a simulation of the world. Nevertheless, some of their ideas seem exactly right:

> ...In our opinion what is lacking [in current

> 'understanding programs'] is not primarily in the area of clever syntactic or even semantic analysis of the input process. Rather, the most difficult problems lie in what has been called the 'pragmatics' of language -- the penetration beyond literal decoding of sentences to the construction of plausible implications from these sentences. (Abelson and Reich, 1969, p. 641.)

However, their "implicational molecule" model of the process of imagining unspoken consequences of sentences is much more psychological than logical. (I have used an analogical structure, the "belief ring," to record the relations between beliefs; see Chapter III.)

In his later writings, Abelson (1973) has recognized the necessity for a much deeper understanding on the part of a belief system than he had previously allowed. His most recent paper, a proposal for a belief system to understand process and causality, shows a remarkable convergence to ideas of procedure and scenario (in Abelson's terms, "script") that have developed at the M.I.T. A.I. Laboratory (Winograd, 1971, and personal communications; Hewitt, 1972). This paper also introduces a new term, "knowledge system," which describes what I am trying to develop better than "belief system." A knowledge system describes what everyone believes. Howevever, it seems very unlikely to me that this terminological division indicates any real difference of method in the two types of device, but only perhaps that the "pleasure principle" influences which beliefs are held more than the "reality principle" which governs simple real-world knowledge.

The system which is closest to mine is that of Charniak (1972). Indeed, it is fair to say that TOPLE is a "Charniak-type" understanding system. Many of the elements of his model have been incorporated into mine: procedural embedding of common-sense knowledge, use of a "base-routine" for each important word, plausibility analysis for reference resolution, forcing deductions from what is heard before being asked questions about it, and pattern-directed updating of the state of the world model. In a lot of cases, I made different design decisions from Charniak, sometimes for a reason, sometimes to be different, sometimes because CONNIVER forces different prejudices from Micro-PLANNER, the language Charniak had in mind. For example, my approach to reference resolution is what Charniak calls the "breadth-first method." One reason for this is that the restriction method he favors is difficult to implement in CONNIVER. (It is equally true, however, that CONNIVER was designed to make such methods harder to use; cf. Sussman and McDermott, 1972.)

The principal differences between his work and mine are as follows. First, my work is less ambitious than his. I am interested in creating a working system, so I limit my world much more than to the children story domain. This means that in many cases where Charniak can't see the trees, I have missed the forest. Second, much less of the knowledge in my system has been

encoded in the form of pattern-sensitive "demons." I give reasons for this in the fifth chapter of the thesis, which deals with how TOPLE talks about the future. However, one reason is that I am interested in complementing Charniak's work as well as implementing it; I believe enough has been speculated about how demons might work (e.g., in Meyer, 1972), and that it is time to provide them with an environment in which to operate. Consequently, nothing in TOPLE prohibits use of demons, but there are none in use. Finally, I believe an important omission in Charniak's work is on debugging a data base. He discusses several good heuristics for jumping to a reasonable conclusion, but leaves out a model of how false conclusions are to be undone. It seems to me that the essence of common-sense, plausible reasoning is to be able to correct the inevitable errors in it, and to be responsible for making its conclusions fit everything else believed. I have devoted much of my effort toward a solution of this problem. However, I believe Charniak erred on the right side; it is much better to make informed guesses than to become mired in searches through disjunctions.

II. How to Model the World

> ...The true idea of the human mind, is to consider it as a system of different perceptions or different existences, which are link'd together by the relation of cause and effect, and mutually produce, destroy, influence, and modify each other. Our impressions give rise to their correspondent ideas, and these ideas in turn produce other impressions. One thought chaces another, and draws after it a third, by which it is expelled in turn. In this respect, I cannot compare the soul more properly to any thing than to a republic or commonwealth, in which the several members are united by the reciprocal ties of government and subordination, and give rise to other persons, who propagate the same republic in the incessant changes of its parts.
>
> --David Hume, A Treatise of Human Nature

As we have seen, TOPLE's basic unit of knowledge is the assertion, a claim about a property or relation of one or more objects. For example, (EDIBLE BAN1) claims the object with name "BAN1" is edible. (MOVE JOHN (PLACE HAVANA)) asserts "JOHN is in the process of moving to HAVANA." (AGGLOMERATIVE EDIBLE) claims the property EDIBLE is true of any group of objects, each with that property.

TOPLE uses the facilities of the CONNIVER programming language (McDermott and Sussman, 1972) to embody its knowledge of the world. CONNIVER is for writing LISP-like programs which communicate through a pattern-accessed data base. Programs ADD list-structured "items" to this data base, thus making them present to others that find them by "fetching" all items that match a certain pattern. Since the truth of a belief may be

modelled by the presence of an item in the data base, programs which add items on the basis of the presence of other items may be thought of as doing deduction.

In particular, certain "pattern-directed" programs are called in conjunction with the CONNIVER primitives ADD and FETCH, which alter and test the data base. An "if-added method" has a pattern which matches an ADDed item. A pattern is a list structure with variables instead of constants in certain slots. The variables are marked with "!>". For example, the following method has pattern (IS !>X MAN):

```
(IF-ADDED (IS !>X MAN)
    (ADD (LIST 'IS X 'MORTAL))   ).
```

This method adds (IS SOCRATES MORTAL) when any other program adds (IS SOCRATES MAN). When the method is called, (IS !>X MAN) is matched against (IS SOCRATES MAN), with the result that X is bound to SOCRATES. So (LIST 'IS X 'MORTAL) evaluates to (IS SOCRATES MORTAL), which the method ADDs. (The symbol "'" is used, as in MIT LISP, to indicate that the following expression is a constant, not to be evaluated.) Thus the method can be thought of as drawing conclusions based on the belief "All men are mortal."

"If-needed methods" are called in conjunction with FETCH, and may be used to simulate or impose the presence of a belief specified by a pattern. For example, the following method is

another reflection of the belief that all men are mortal:

```
(IF-NEEDED (IS !>X MORTAL)
    (FOR-EACH-ITEM (IS !>X MAN)
        (NOTE)  ))
```

The function NOTE adds the current instance of the pattern (IS !>X MORTAL) to the list of items whose presence the method is simulating. A FOR-EACH-ITEM loop cycles through all FETCHed items, including those simulated by methods. Thus, if (IS SPIRO MORTAL) and (IS GEORGE MAN) are present, the loop

```
(FOR-EACH-ITEM (IS !>X MORTAL)
    (PRINT M))
```

prints

```
SPIRO
GEORGE,
```

the second name being produced by the if-needed method.

It is usual in CONNIVER programs to represent "atomic formulae" by items, and all other knowledge by programs and methods. In most cases, "other knowledge" includes deductive beliefs like "for all x, if x is a man then x is mortal." TOPLE also has need of more complex information regarding what data-base changes are plausible and what changes make certain other changes necessary, as well as information about what beliefs are entailed by its current beliefs. All of this is expressed in methods, mostly if-neededs (see Fig. 1), which are thus

responsible for the propagation of "the same republic in the incessant changes of its parts." TOPLE models its atomic beliefs, or assertions, as CONNIVER item data, the parts that "incessantly change."

An advantage of this representation is that it enables TOPLE to treat an assertion as an object with properties of its own. A routine can make assertions about assertions, or manipulate the item property lists CONNIVER provides. The second alternative is actually used to store the situation-independent "meta-knowledge" about an assertion. For example, all items have a property called FAITH which tells whether they are currently believable or not; normal beliefs have property LIKELY under this indicator. A more interesting type of meta-knowledge is that relating two or more assertions. This type will be dealt with in the next chapter.

Not all assertions about which the program has knowledge are assertions it believes. A subset of the LIKELY assertions are present in the CONNIVER sense-- findable by the FETCH pattern-directed search primitive. These are the items that are allowed to take part in deductions. Some facts are plausible but not present, because they are immediate conclusions from other facts, and would be confusing to have around as though they were independent. For example, in the monologue of Chapter I, the implication (AT BAN1 (PLACE TAB1 UNDER)) is as LIKELY as its two

impliers ([3] and [4]), but not present, to avoid having a record of two locations for BAN1. (Cf. Chapter IV.)

With devices like these, we can represent a static world structure, and indicate a rich variety of interconnections between its component data, an ability of great importance in a belief system of the kind to be discussed in the next section.

A limitation of the system as described so far is its inability to deal with time, with incompatible world states, and with processes as well as conditions. We cannot, for example, describe "a monkey's being at the table, then moving to the box and being there." We might solve this problem by introducing "world situations" as a new type of object, of the same logical status as monkeys and bananas, and make all static statements mention their situations. Then (AT SPIRO (PLACE TAB1) S0) and (AT SPIRO (PLACE BOX1) S1) would not be contradictory or confusing because they manifestly mention different situations S0 and S1. Typically (Green, 1969a; Hayes, 1971), such a scheme treats actions as functions (in the predicate-calculus, not procedural sense), which map situations into situations. Thus the moving monkey would be described by the assertions

```
(AT SPIRO (PLACE TAB1) S0)
(AT SPIRO (PLACE BOX1)
          (MOVE SPIRO (PLACE BOX1) S0)),
```

where (MOVE SPIRO (PLACE BOX1) S0) is a successor to S0.

For a variety of reasons, I have avoided this course. First,

those little situation symbols do not seem to fit into my notion of the world. A situation would seem to have completely different ontological status from boxes, monkeys, and even more abstract concepts; situations contain statements, not the other way around. Situation letters enable one to treat statements as immutably true, which is a convenience in first-order predicate calculus, but (I am convinced) a poor model of how people think. If the situation symbols could be suppressed, a problem-solver could treat the world as static some of the time, but dynamic when it wanted it to be. Notice also that the notation for successor states is long and clumsy, and forces us almost immediately to consider the abbreviation problem (Minsky, 1961) for chains of even a few states. (Hayes, 1971)

Finally, and most importantly, such a notation forces us to implement the semantics of situations by the same sort of mechanisms as for other sorts of objects-- anything to be known about a world-state must be proved. We are ultimately almost forced into the "frame problem" (McCarthy and Hayes, 1969; Hayes, 1971; Green, 1969a.), and the necessity of proving things about states from scratch, as though they were independent except for a few exceptions. In fact, the truth is the other way around; successive states are closely related, temporally, causally, and logically.

It would be nice for situations to have properties like

these:

-- They should be cheap to create, mention, and associate with data.

-- A situation's successor should be initially, and automatically, equivalent to it with respect to all assertions. Differences should be the exception, not the rule.

-- It should be possible to alter the state structure after creating it, enriching sequences of events better to fit one's beliefs.

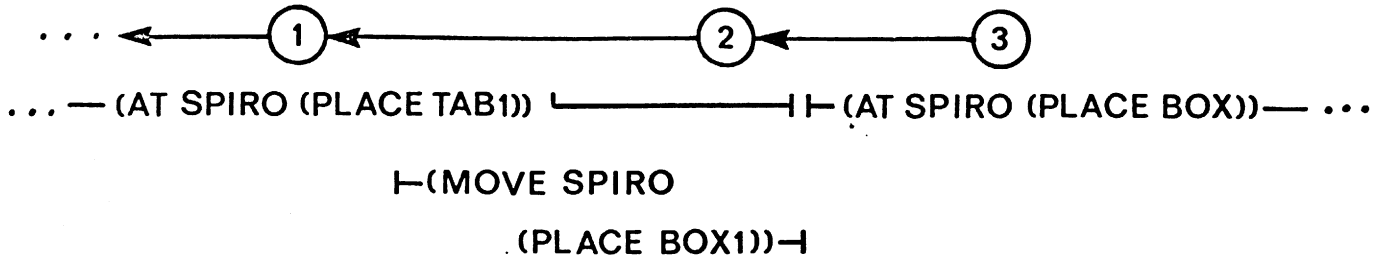
-- Actions must transform situations and conditions on one level, but be capable of being ongoing conditions on an expanded time scale. "Fidel went to Havana" and "Fidel was going to Havana" are event and condition, but reflect the same fact in the real world.

-- It should be possible to make hypothetical alterations to the entire world structure, retaining the freedom to discard them or make them final at low cost.

An important feature of CONNIVER provides a way of meeting these criteria. This is the multiple-data base feature. All of the pattern-oriented operations I described previously are performed with respect to a particular context, which may contain an arbitrary collection of items and methods. A context, when created, may be a "sub-context" of another context, which means

that it initially contains the same information. Later additions and removals of items and methods do not affect its superior. Thus the sub-context relation is a computationally efficient model of the successor relation for situations.

For example, if the sub-context relation is indicated by "<----", the following diagram models a sequence of situations (circles):



Time is from left to right. Below the situations are items representing beliefs about the situation, which are modeled as items present in the corresponding context. A line like "|—...—|" indicates the extent of an item. "|—" indicates when it is added; "—|", when it is removed.

Notice that this notation treats events the same as conditions; the programs reading it must realize the

difference. In the diagram, (MOVE SPIRO (PLACE BOX1)) is an event merely because it lasts for only one situation, and causes a change in Spiro's location, assertions about which are conditions. This second fact can only be represented as a belief about a belief, namely, something like "(MOVE SPIRO (PLACE BOX1)) in situation 2 caused SPIRO's location to change to (PLACE BOX1) in situation 3." This information is not necessary for all applications, and, in fact, TOPLE does not deal in such facts.

The distinction between event and condition is not perfect, and it doesn't have to be. A program may interpolate situations and events into the time span of another event; or elide enough situations from a situation sequence to foreshorten conditions into events. Thus, on one time scale, World War II is an event, on another definitely a condition. This difference is reflected not merely in terms of tempora persistence of an assertion, but in the types of implications and explanations it takes part in. "Because of World War II" could be a reason for why meat was rationed and why France lost her empire, in the first case as a condition, in the second as an event.

There are many problems with this scheme, but hopefully they reflect inherent difficulties with reasoning about process rather than artifacts of a clumsy formalism. That being as it may, they are serious enough so that no attempt has been made in the current program to exploit the event-condition ambivalence. Any

assertion is treated as one or the other. Of course, a more complicated context structure than a row of contexts will be necessary for dealing with a single fact as an event and a conditon. Real-world time seems to be organized into episodes and sub-episodes, with local conditions appearing as events on a more global scale. (See Chapter VII.)

There is another departure of a different kind from the simple linear structure of time. TOPLE models the future as a branching structure of situations, which reflects its ignorance about what is actually going to happen. For example, in the monologue of Chapter I, when TOPLE hears, "The experimenter attaches a bunch of bananas to the ceiling," it sets up two mutually exclusive futures:

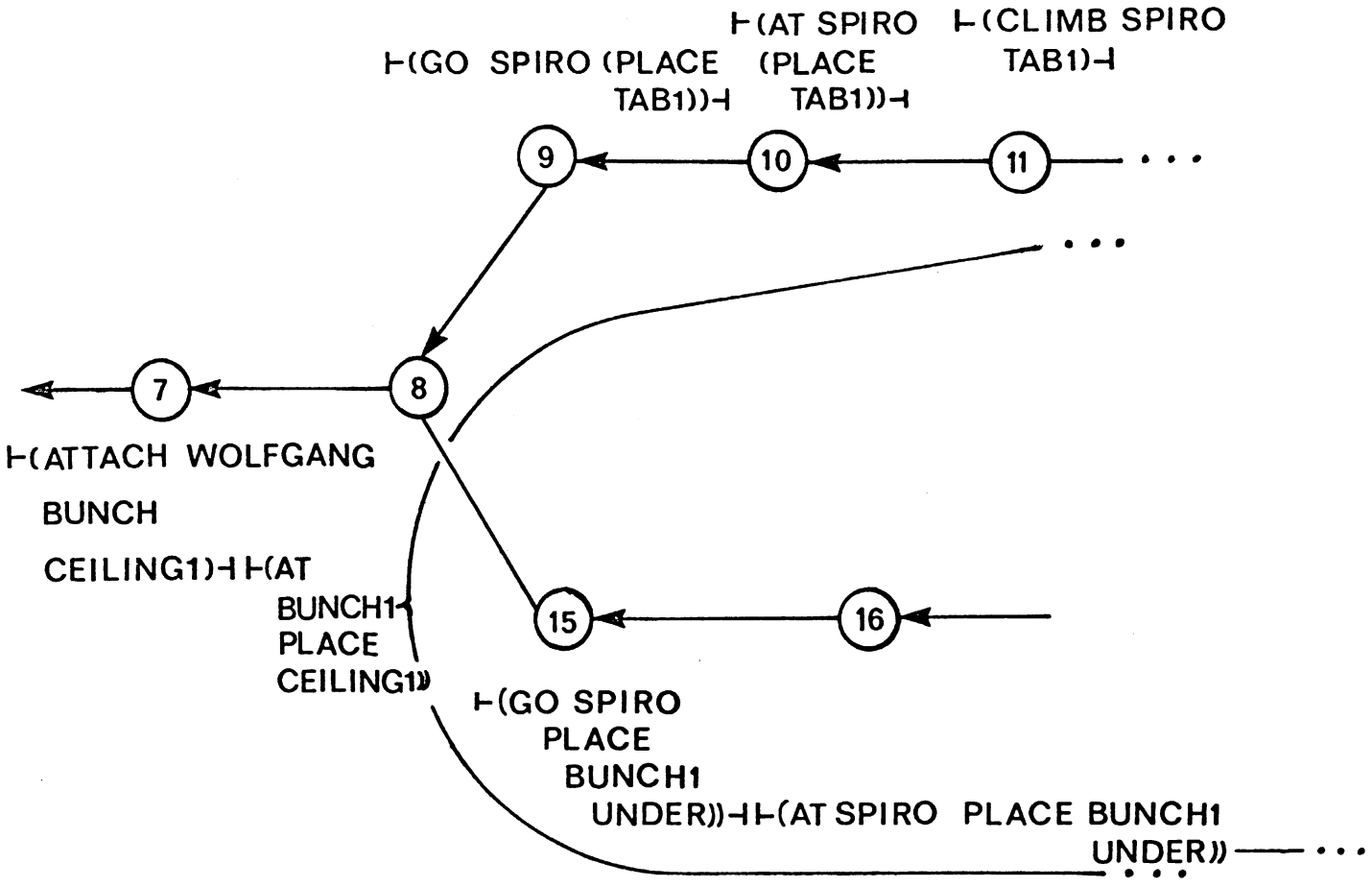


Figure 3.

(Cf. [25] - [40].) This sort of context structure is easy to construct in CONNIVER.

The straightforward use of contexts solves some simple problems for us, but leaves the deep ones untouched. The frame problem, which has been done away with in its trivial form, arises again in connection with deducing properties of a situation from situations which preceded it at various times before. Keeping a table in one place from event to event is a

bookkeeping problem which vanishes when the right model of time is used; deciding whether furniture or creatures are likely to be in the same place now as they were two minutes, two days, or two years ago (when the last information was received) is more difficult.

By using contexts, I have given the program's world a dynamic structure. Its overall "syntax" is now complete. But there are several elements missing:

-- I have only suggested what kinds of information are to be stored in such a model. So far it has been discussed in terms of symbols that have been given no "meaning." It remains to be explicated what the machine really "believes."

-- The type of information I have described is of a very particular sort. There is no room for general statements like "All men are mortal," or, "If you push anything hard enough, it will fall over." As suggested in my exposition of CONNIVER, such propositions are to be modelled by <u>programs</u>.

-- The model I have sketched takes world-time into account, but is so far static in another way: it makes no provision for change in the model itself, for learning, error correction, and ongoing experience. Such changes, too, will be accomplished by programs, in conjunction with an entirely different use of contexts. In fact, I shall show in the next chapter that some simplicity is gained by having them be the same procedures

responsible for the kinds of deduction mentioned in the last paragraph. In my program, all these functions have been subsumed by a set of routines for maintaining a "reasonable" picture of the world.

In the end, it is the interactions of programs like these that give TOPLE's symbols whatever meaning they have.

III. How to Change the World

> When we consider either the history of opinion, or the ordinary content of human life, to what is it to be ascribed that the one and the other are no worse than they are? Not certainly to the inherent force of the human understanding; for, on any matter not self-evident, there are ninety-nine persons totally incapable of judging of it for one who is capable; and the capacity of the hundredth person is only comparative....Why is it, then, that there is on the whole a preponderance among mankind of rational opinions and rational conduct? If there really is this preponderance--which there must be unless human affairs are, and have always been, in an almost desperate state--it is owing to a quality of the human mind, the source of everything respectable in man either as an intellectual or as a moral being, namely, that his errors are corrigible.
>
> --John Stuart Mill, On Liberty

What does it mean to believe an assertion? It can hardly mean only that it is written down in your data base (or head) in some notation, to be assented to whenever someone inquires about it. Instead of mere "disposition to assent," it seems to me that even for philosophical purposes the notion of belief requires two components:

-- Understanding: a belief is related to other beliefs in ways that reflect what they mean. Anyone who believes Fred is a bird must believe he can fly, unless he believes Fred is an ostrich, or is sick, etc. Even if the believer professes and practices complete ignorance on these issues, he must recognize their relevance. If told that Fred can fly, he should be pleased

by this agreement with his previous knowledge; if told he cannot, he should seek to know the circumstances surrounding Fred's condition.

-- Commitment: A belief does not merely serve to tie together other beliefs. In addition, believing something associates a cost with changing it. If a new proposition is understood to contradict an old one, then the cost of assimilating the new one must include the cost of backing up and undoing the consequences of its predecessor. To hold the belief that Fred is a bird is to associate such a cost with believing he has four legs. For in this case, either one must doubt that Fred is a bird at all, or doubt one's belief that all birds have two legs.

These two aspects of belief are difficult to separate; it is impossible to commit oneself to a proposition without understanding, e.g., what would contradict it. Both of these components are intimately tied up with a deductive ability.

However, a simple ability to deduce does not seem sufficient for understanding. In systems which admit of deduction (as opposed to mere representation, such as Schank et. al. (1970)), it is clear what the relations are between various beliefs: some can be proved from others. For this reason, Sandewall (1970) comes closest to tackling the problems of understanding and commitment of any of the many people (Rumelhart et. al., 1972;

Schank and Tesler, 1969; Quillian, 1969) who have thought about how to represent semantic knowledge. Sandewall (1970) and Green (1969b) both envision using the question-answering power of an information-retrieval system based on a uniform proof procedure to reject new axioms which contradict what it knows. Nonetheless, even an infinitely-powerful proof procedure can handle the problem of commitment in only the shallowest way. This is because first-order predicate calculus defines the answers to questions of consistency, but does not tell you which questions to ask, nor how to patch up a data base with an inconsistency in it.

A complete belief system (or knowledge system, after Abelson (1973)) contains a deducer among three components:

-- A deducer: Part of understanding is the ability to answer questions. In TOPLE, this component is essentially a group of CONNIVER if-needed methods, operating to retrieve data in ways similar to Winograd's (1971) PLANNER data base.

-- A believer: A more interesting component is that responsible for wedging new information into the data base. Since this component must have understanding of the same issues as the deducer, and also wishes to be pattern-directed, it, too, is a set of if-neededs, whose patterns match the items to be believed, and which alter the data base to reflect the new belief.

-- A doubter: If an if-needed runs into a contradictory (or to some degree implausible) set of beliefs, they must be reconciled in some way, or the old beliefs must be doubted. Doubting something is not simply removing it from the data base; this can happen, e.g., when some change occurs in the world being modelled. Instead, the system must recognize that every belief has a purpose in keeping the world-model consistent. If it is ripped out, the resulting hole must be filled by a new set of beliefs that plays the same role in a way more congenial to later findings.

Thus, these three components are intimately related. In fact, the deducer and believer are embodied in the same set of if-neededs, which can be called with different parameters. There are at least two good reasons for this:

1. In both cases, a reasonable first step toward considering a proposition is to check what you believe about it already. In the case of deduction, an estimate can then be made of the plausibility of the item to be deduced. In believing, the system may find itself ignorant on the subject, and be forced to assume something; or may discover implausibilities which must be cleared up by calling the doubter or considering domain-dependent alternatives.

2. The pattern of an if-needed method in both cases corresponds to an assertion that is close to the surface

linguistic structure of a question or statement. For example, the statement, "The monkey has a toothache," should be translated into (HAVE SPIRO G1), (IS G1 TOOTHACHE), and the HAVE-understander called. This if-needed method, however, should probably translate the concept further into (ACHE G2), (IS G2 TOOTH), (PART-OF SPIRO G2). In general, HAVE sentences really depend almost entirely on the haver and havee for their meaning, whether they are questions or assertions. Another example is the question "Where was the table?" or the statement, "There was the table." Clearly, the desired response from the range of possible answers (understood meanings) for this question (elliptical statement) depends on the situation, one's model of the speaker, etc. It would not seem to depend on whether information is being offered or requested between two creatures involved in the same situation. Hearing either sentence should cause a method with pattern (AT !>THING !>PLACE) to be called, which must solve the scope problem regardless of whether deduction or belief is desired.

Consequently, TOPLE maintains a set of all-purpose if-needed methods, managed by the function PLAUSIBLE?, which plays the same role in a belief system as GOAL in PLANNER (Hewitt, 1972) or TRUE in Sussman's (1972) HACKER, both deductive systems. PLAUSIBLE? takes a "goal type" argument, with value DEDUCE or BELIEVE+, which tells the methods it calls what they are to do.

(Fig. 1.)

When a method is doing pure deduction, it typically checks what it knows by searching item data structures. Then it returns a conclusion if the deduction is possible; NIL if the assertion is believed not to be true; or nothing if no conclusion can be reached. (It is possible for a CONNIVER if-needed method to return nothing, in which case a special argument to the calling PLAUSIBLE? is evaluated to give its value.)

When the same if-needed method attempts a belief, the typical sequence of events is as follows: if the assertion is already believed, it returns it; otherwise, it picks the best set of changes to the data base it can find, makes them, and adds and returns the new belief. It cannot return NIL or nothing--the belief must be added.)

(The reason for the "+" in "BELIEVE+" is that there is clearly room for goals of type BELIEVE- and just BELIEVE. The first would add and remove appropriate assertions in order to make a pattern false; the second would force a definite belief on a subject, even if in DEDUCE mode no judgment either way would be made. I have not implemented these goaltypes, but they would be very useful in certain situations.)

Whenever an if-needed method makes a data-base change, it must be aware of the possible future actions of the grim reaper called DOUBT. The best set of changes may not be best forever;

later considerations may cause the system to find another solution or abandon the attempt to believe the item requested. But DOUBT cannot know the range of alternative solutions without being told it by the method which made the changes.

Let us step back and examine the problem in the abstract. TOPLE is basically a very skeptical program. It wants to resist <u>any</u> change to the data base, but changes produced by its "sense organ," the teletype, are inevitable, so it forces itself to believe what it is told as cheaply as possible. Some changes to the data base, however, are more expensive than others, because TOPLE understands the world (at least to some extent), and knows that some interpretations of the situations that it is told about are unlikely. It resists having to believe in such interpretations at all if more plausible interpretations of what it hears are available; and, if it must accept them, it demands some kind of compensating belief. Put another way, a new belief, in combination with some old ones, often leads to a kind of "tension," an uneasiness about the world that must be resolved by the if-needed method that notices it.

I shall show later how these difficulties arise during the operation of the program, and how (using the function CHOOSE) it builds disjunctive goal trees to resolve them. For now, I will concentrate on a description of the belief structures this effort is aiming at.

Let me put this discussion in the context of an artificial example which will illustrate a lot of notational and organizational points. (The problem to be described does not, of course, arise in the monkey-and-bananas world.)

If TOPLE knew anything about birds, hearing (FLIGHTLESS FRED) after (IS FRED BIRD) would make it twitchy. This is not a contradiction, but causes it to look around for ways of clearing the problem up. Assuming the system knows penguins are antarctic birds, the relevant if-needed for penguinhood will record that being an antarctic bird is corroborating evidence that one is a penguin. If it knows (or can prove), in fact, that Fred lives in the Antarctic, that makes it easier to believe he is a penguin; if TOPLE is ignorant of Fred's habitat, it is slightly more expensive to believe in Fred's penguinity, but still can be done; with contradictory evidence, this entire effort to resolve the difficulty is in jeopardy.

I shall graph implausibilities and the assumptions necessary to reduce them in the following way. If a set A1,...,Ak of beliefs makes it desirable to find or assume some set of compensating beliefs Ak+1,...,An, we make a circle out of them, with arrows from the "tension-producers" to the "tension-reducers":

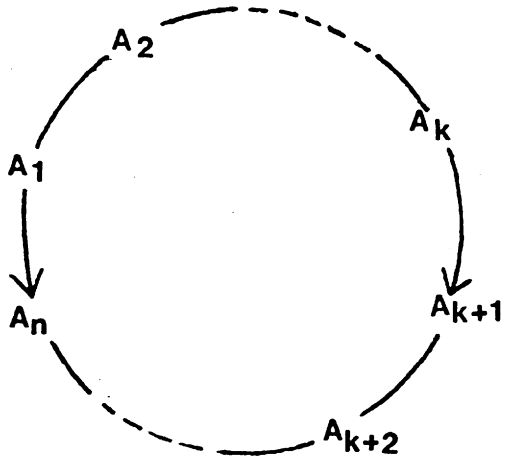


Figure 4.

Then the ring structure for our example is:

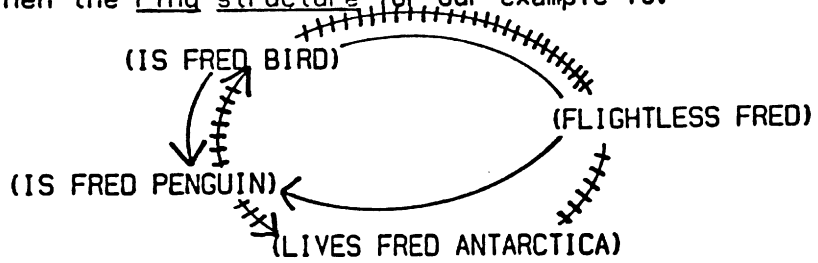
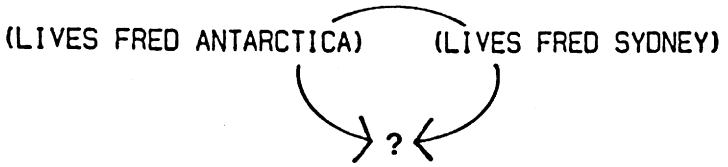


Figure 5.

There are two rings here, as indicated by the two types of line. The "———" ring is caused by the conjunction of (IS FRED BIRD) and (FLIGHTLESS FRED). (IS FRED PENGUIN) reduces the cost of believing these two, but cannot be believed without accepting "Fred is a flightless Antarctic bird," which causes the "+++++++" ring.

It is not enough merely to discover useful assumptions. Structures such as that of Figure 5 should be explicit enough so that DOUBT can use them. For example, what if the hypothesis that Fred is an ostrich is implausible for some reason, so that the system adopts the belief structure of Figure 5. Then say it

is told "Fred lives in Sydney." The following tension is set up:



The LIVES-method knows (by calling a geography expert) that Sydney cannot be in Antarctica by any stretch of its imagination. So it must doubt (LIVES FRED ANTARCTICA). This should lead it to look for another way of resolving (IS FRED PENGUIN) ?. Then it might assume Fred lives in a zoo in Sydney (for example), or doubt the PENGUIN assumption and try, e.g., (IS FRED OSTRICH) again. This time whatever implausibilities prevented this assumption may be compensated by the corroborating belief that (LIVES FRED SYDNEY). Our programs might now come up with:

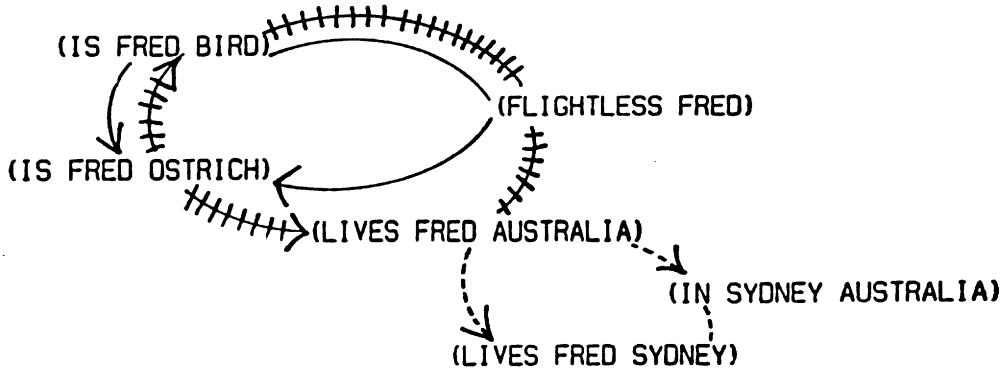


Figure 6.

which expresses three different rings.

These sorts of structures, called belief rings, are set up by if-neededs called by PLAUSIBLE?, and demolished, if necessary, by

DOUBT. DOUBT, however, is not an expert on everything; how can it know what are the allowable courses of actions to take to patch up a ring?

The answer is that each ring builder must associate with that ring a reconstructor form, a sort of continuation of the ring-building process in suspended animation. For DOUBT to question a ring, it must evaluate its reconstructor, which typically either revives the search for a tension-reducer or attempts to give up and doubt one or more of the beliefs that causes the tension.

With this model, DOUBT no longer has to understand anything about the world, but is reduced to a bookkeeper which finds all the rings the doubtee belongs to and fiddles with them in the appropriate way. If the assertion to be doubted is a primary ("tension-causing") member of a ring, doubting it makes the ring useless, and it is dismantled. If it is a secondary ("tension-reducing") member, the reconstructor of tht ring must be called to attempt to reduce the tension some other way. (Of course, a belief will probably be a member of several rings, in different capacities. Each of its roles is treated independently.)

Notice that all the information about the basis of a belief has been secreted away in its ring reconstructor form. Presumably some declarative information should also be saved for some assertions; for example, to be able to answer why-questions, it is probably desirable to tag assertions about a creature's

actions with an indication of his purposes. But the bookkeeping necessary to maintain and interrogate such properties (or assertions about assertions) is a responsibility of the if-needed methods and reconstructors associated with each domain. (In fact, the current TOPLE is not designed to answer such questions, so it stores very little information of this sort.)

Another point: the arrows from primaries to secondaries in a belief ring have nothing to do with implication strictly conceived. The direction is from hole to filler, from incompleteness to completion. This notion is closely allied to the "implicational molecule" idea of Abelson and Reich (1969), although less well-defined; it goes back before that to Gestalt ideas of good form and stable organization. (Koffka, 1963)

All the rings I show in this chapter involve only changes to the data base in which an assertion is added. There are other types which fit just as well into this framework. For example, as will be seen in Chapter V, the predicted situations generated by TOPLE are the secondary elements of rings whose primaries are the current conditions that give rise to the predictions. A less obvious example of a secondary change is the hiding of certain believed data. Whenever an already-believed thing can be subsumed under some new belief (for example, by transitivity of its relation), the new belief causes the old one to be hidden, for reasons of intelligibility of the data base. This is

represented by allowing "hidden datum" to be a legal type of secondary element. Demolishing a ring with a hidden secondary causes it to be brought back into the open. Other secondary effects that could be implemented would include attachment of new properties to data, for example the declarative information alluded to in a previous paragraph; one would like this to be directly linked to the beliefs that caused it to be attached, so that doubting them would make the links vanish.

With these (almost infinitely) powerful devices in hand, let us review how a mechanism of commitment to a belief is to be designed.

I have described what sorts of structures are the target of a plausibility investigation. The problem is for the system to enumerate the possible structures, and choose the least costly alternative as its chosen conception of the world.

The methods to be called by PLAUSIBLE? must be able to communicate the difficulties they have run into to the higher function concerned with picking the proper belief structure. A certain cost is associated with each obstacle; the cost must be estimated when obstacles are encountered, then discovered precisely if a decision is made to attempt to overcome them. These costs are of the sort mentioned previously: it takes effort to alter previous beliefs; it is assumed to be costly to make "unjustified" assumptions that may be painful to alter

later; it is very costly to doubt assertions made by TOPLE's human friend.

When the effort to reach a goal encounters a difficulty, it is usually crucial to be able to try an alternative goal at a higher level. However, since it might not succeed, it is important to keep all options open until one emerges as a definite winner.

This control function is accomplished by the function CHOOSE. This program attempts to choose between several goals--represented by CONNIVER forms--on the basis of which seems to be the least costly to achieve. Each of these forms attempts to construct an adequate picture of the world, yet they must not interfere with one another as they run "in parallel." Consequently, CHOOSE gives each form a separate hypothetical context to play with. Of course, the programs CHOOSE calls will often call CHOOSE (and PLAUSIBLE?) themselves as they attempt to massage the world model to fit what they hear. Hence, my program builds a goal-tree of processes, each trying to achieve an adequate view of the world. (Fig. 7.) At any given time, only one branch of this tree is active, the others either being too unpromising to have been run at all yet, or having encountered difficulties and given up for a while.

It is important to realize that the contexts in the tree of Fig. 7 have nothing to do with the contexts corresponding to

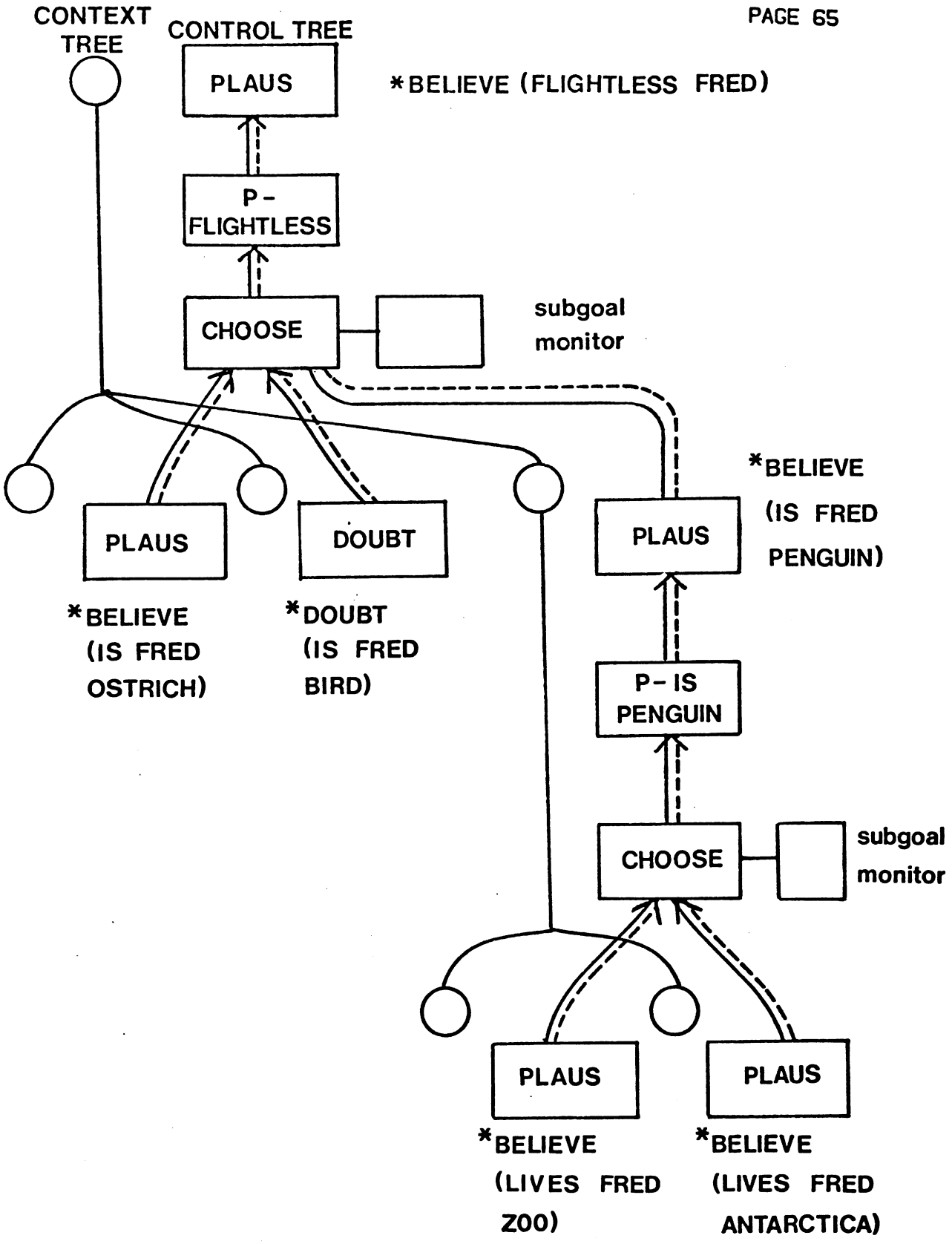
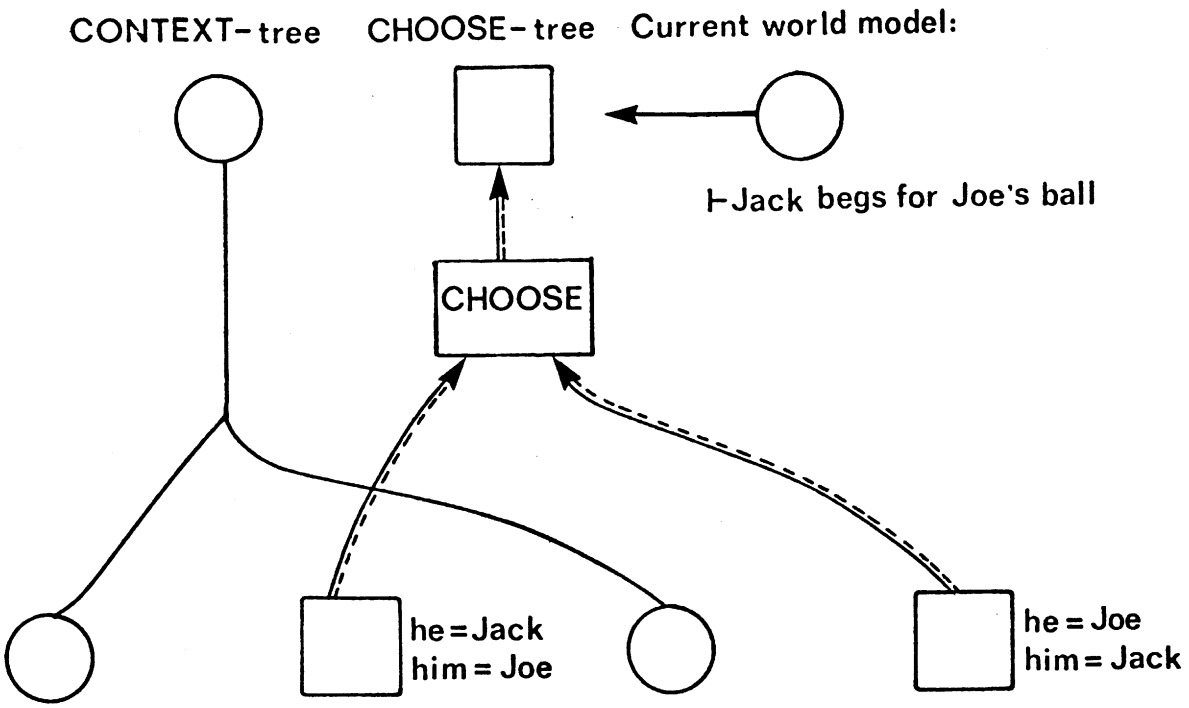
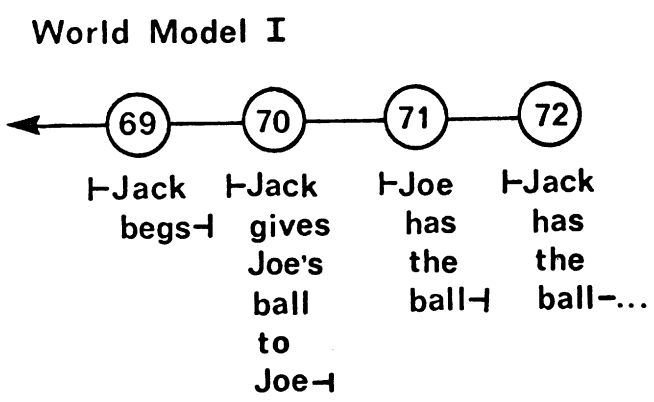

CONTEXT TREE
CONTROL TREE
PLAUS
*BELIEVE (FLIGHTLESS FRED)
P-FLIGHTLESS
CHOOSE
subgoal monitor
PLAUS
DOUBT
*BELIEVE (IS FRED OSTRICH)
*DOUBT (IS FRED BIRD)
PLAUS
*BELIEVE (IS FRED PENGUIN)
P-IS PENGUIN
CHOOSE
subgoal monitor
PLAUS
PLAUS
*BELIEVE (LIVES FRED ZOO)
*BELIEVE (LIVES FRED ANTARCTICA)

figure 7

situations. The former are called problem contexts, the latter, world contexts. A moment's reflection on the meaning of time in each should convince you of this. In the problem context tree (Fig. 7), the time is "now," and different contexts contain different hypotheses about the story; the world contexts each have a different "monkey time," which has nothing to do with "TOPLE time."

Since the hypotheses about the story depend on the problem-context tree, different hypotheses give rise to different world context trees. TOPLE understands only present-tense sentences, so most of its uncertainty is about the future, but there is no reason why it could not ponder different presents. For example, it if heard, "Jack begs for Joe's ball until he gives it to him. Now Jack has it," it could set up two interpretations in a CHOOSE-tree:


CONTEXT-tree
CHOOSE-tree
Current world model:
⊢Jack begs for Joe's ball
CHOOSE
he=Jack
him=Joe
he=Joe
him=Jack


World Model I
69
70
71
72
⊢Jack begs⊣
⊢Jack gives Joe's ball to Joe⊣
⊢Joe has the ball⊣
⊢Jack has the ball-...

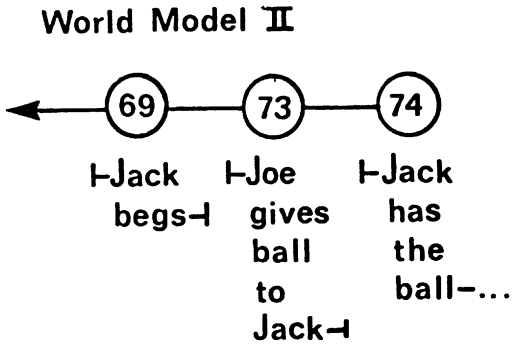

World Model II
69
73
74
⊢Jack begs⊣
⊢Joe gives ball to Jack⊣
⊢Jack has the ball-...

figure 8

(Hopefully, the version on the right is more believable to the system.) Of course, TOPLE does not actually understand giving; see Charniak (1972) for steps toward theories about such human activities.

Notice that with these two context mechanisms TOPLE has a choice as to how it will represent uncertainties about the future. It can maintain two different problem contexts, each with its model of the future, or have one model which branches, as in Fig. 3. The former makes it easier to compare features of the alternatives; the latter is more condensed and cheaper. TOPLE uses both, in different circumstances. It generates alternative futures in a goal tree, and saves those that survive in prediction trees.

When a program at the tip of a branch of a tree like that of Fig. 7 runs into a difficulty, it must estimate the cost of overcoming it so that CHOOSE may decide whether to let it continue, and if not, what to do next. If a process blocked in this way is resumed later, the estimated cost will be transformed into actual costs, such as those of believing various assumptions or altering the data base. In the current TOPLE, these costs are merely crude estimates of the number of "arbitrary assumptions" that a branch of the goal-tree involves. It might be thought that a goal-tree of this sort would become very bushy, mired in contemplation of a large set of mostly meaningless numbers that

do not differ very much. However, the troubles reported by various branches may involve symbolic information as well as numbers, and TOPLE and its subroutines take as much advantage of the symbols as possible.

Trouble reporting is accomplished by the function (CONTINUE trouble estimated-cost), which behaves like a no-op if it returns at all; however, it results in the trouble being communicated to the higher-level subgoal monitor of some CHOOSE, (cf. Fig. 7.) which may decide to run someone else. These messages may be arbitrary list structures, which can be decoded by message-handlers at the disjunctions of CHOOSE goal-trees. Currently, messages are almost all single atoms; the two the most common are IGNORANCE and CONTRADICTION, generated when a method does not know an assertion, or finds it to be false, when trying to believe it.

However, more complex messages are possible. For example, if-needed methods trying to believe an action are always on the lookout for difficulty CONTRADICTION in attempting to believe a prerequisite condition for their actions. If absolutely required, they will shoe-horn the belief in the prerequisite into the data base, but first they send a message (PREREQ condition) to whoever is above them, hoping this specific information will be of use. In fact, these messages are decodable at the top level by the routine CHECK-PREDICTIONS that chooses the current

situation sequence; it treats them as evidence that a person is leaving out certain actions of a predicted sequence, and it tries to move forward in the already-built predicted future to a place where enough assumed actions have taken place to satisfy the prerequisite. This comes in handy in accepting, "The monkey picks up the banana," when heretofore the monkey has been on a different side of the room from the banana, the assumption being that he went over to it first.

It would be nice if I could get rid of the numbers attached to trouble reports, and deal only in comparisons of symbolic difficulties, because these numbers do not really mean very much. They are used in a purely ad hoc manner to make the data base alterations come out the right way; there comes a point where TOPLE is choosing between resolving two CONTRADICTIONs, and has only the number of rings each is a secondary of to go on. (The use of these numbers reflects the notion that about one more arbitrary assumption will be made in the "second-best" way of believing a group of primaries, than was made in finding the best.)

At each level, when a program has decided what changes are needed to believe something, it bundles them into a ring as described; a ring is implemented as a list of the form (primaries secondaries reconstructor). The primaries and secondaries are (item-datum, situation) pairs defining the

relevant beliefs; each such datum points to the ring from its property list for that situation. The reconstructor is an ordinary CONNIVER form to be evaluated when DOUBT wants to destroy the ring. The ring is attached as a property to every item datum in it.

To summarize these points and make them concrete, consider the if-needed P-FLIGHTLESS which is part of the mechanism required to handle the example given before:

```
(IF-NEEDED P-FLIGHTLESS
      (FLIGHTLESS !>CREATURE)
   "AUX" ((C 1.0))
   (COND (%P(PLAUSIBLE? (LIST 'IS CREATURE 'BIRD) 'DEDUCE)
          (CHOOSE
                 ((OR %S(PLAUSIBLE?
                         (LIST 'IS CREATURE 'OSTRICH))
                      (FAIL))
                  (CSETQ C 0.0))
                 ((OR %S(PLAUSIBLE?
                         (LIST 'IS CREATURE 'PENGUIN))
                      (FAIL))
                  (CSETQ C 0.0))
                 ((CONTRADICTION 2.0)
                  (DOUBT (LIST 'IS CREATURE 'BIRD)))   ))
         (T (IGNORANCE 1.0))   )
   (ASSUME (LIST 'FLIGHTLESS CREATURE) c)
   (NOTE)
   %R(SUBST CREATURE
        'CREATURE
        '(PROG (CHOOSE ((DOUBT '(IS CREATURE BIRD)))
                       ((PLAUSIBLE? '(IS CREATURE BIRD)
                                    'BELIEVE+))   )
               (CHOOSE ((DOUBT '(FLIGHTLESS CREATURE)))
                       ((PLAUSIBLE? '(FLIGHTLESS CREATURE)
                                    'BELIEVE+))   )))   )
```

This method may be understood as follows (ignore the "%'s" for now). The function IGNORANCE is defined as

```
(CDEFUN IGNORANCE (COST)
   (COND ((EQ GOALTYPE 'DEDUCE) (ADIEU))
         (T (CONTINUE 'IGNORANCE (PLUS TOTCOST COST)))   )),
```

where TOTCOST = cost so far, and COST is an estimate of future expenses. That is, if creature is not a bird, P-FLIGHTLESS doesn't know whether it is flightless or not; if GOALTYPE=DEDUCE, it returns nothing with (ADIEU); if BELIEVE+, it reports difficulty IGNORANCE, and assumes FLIGHTLESSness if it proceeds from the CONTINUE.

The function CONTRADICTION is an analogous one for difficulty CONTRADICTION:

```
(CDEFUN CONTRADICTION (COST)
   (COND ((EQ GOALTYPE 'DEDUCE) (ADIEU NIL))
         (T (CONTINUE 'CONTRADICTION (PLUS TOTCOST COST)))   )).
```

In DEDUCE mode, if CREATURE is known to be a bird, P-FLIGHTLESS returns NIL (with (ADIEU NIL)), unless it can prove CREATURE is an ostrich or a penguin. If it proves it is not one, it calls FAIL, which means "report trouble of infinite cost," and causes that branch of the CHOOSE to be terminated.

In BELIEVE+ mode, the same CHOOSE behaves in a slightly different way. First, since PLAUSIBLE? never returns NIL, the FAIL's can't be executed; the CHOOSE merely picks the least

costly thing to do. The third branch causes TOPLE to report trouble CONTRADICTION, then go ahead and remove (LIST 'IS CREATURE 'BIRD) from the world model. (The CHOOSE of this program does not attend to the symbolic messages it receives, but passes them up to anyone above it with a more sophisticated message handler; if control returns to it, it picks the least costly branch to proceed.)

The variable C in P-FLIGHTLESS is the cost associated with the assumption it makes. This is 1.0 (measured in "arbs"--arbitrary units for arbitrary assumptions), unless it can be deduced, or reconciled with the world, when the cost becomes 0.0. ASSUME adds its argument to the current context, charges C units for it (by calling CONTINUE with the new cost), and associates this cost with the assumption.

%P, %S, and %R are macro-characters which unobtrusively represent the ring-building done by P-FLIGHTLESS. %P records its argument as a primary assumption; %S, as a secondary. If there are any secondaries, %Rform creates a ring out of the accumulated primaries and secondaries, and makes form its reconstructor. In this case, the reconstructor would call again the methods that originally put (IS FRED BIRD) and (FLIGHTLESS FRED) into the world model.

As a less artificial example of the operation of the system, consider the following sequence of events. TOPLE has been told

that there is a big table (TABLE), a slightly smaller box (BOX2), a little box (BOX1) (both boxes bottomless), a small rubber ball (BALL1), and a dinner plate (PLATE1), roughly the size of BOX1, in the arrangement of Figure 9.

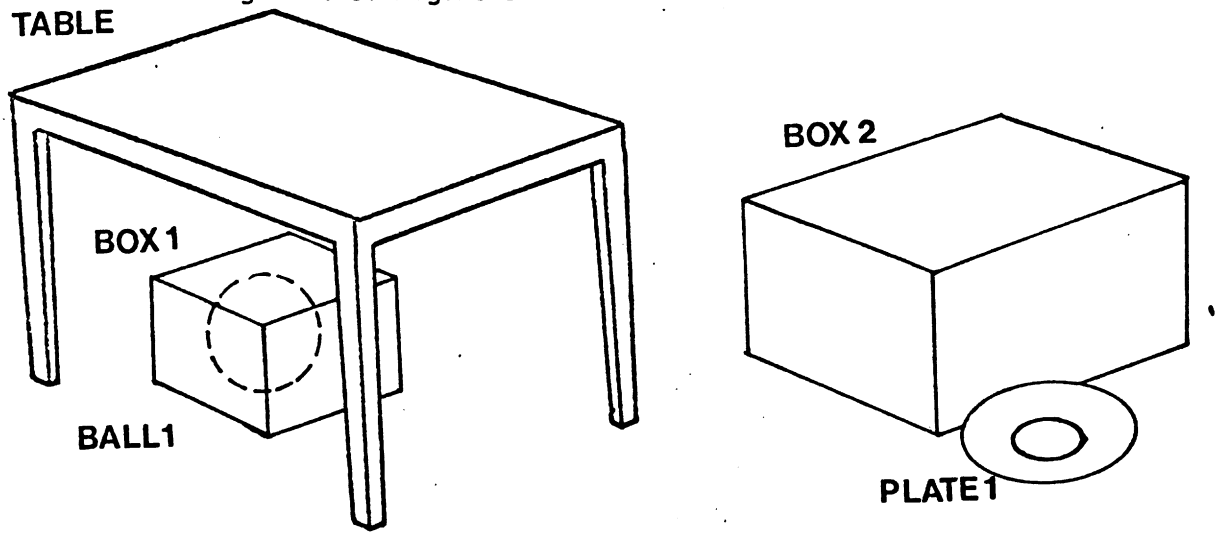


Figure 9.

The ball is under the little box, which is under the table; the plate is by the big box, which is somewhere in the vicinity of the table.

Now TOPLE is told, "The ball is on the plate," in the form

```
(AT (THE X (IS X BALL))
    (PLACE (THE Y (IS Y PLATE)) ON))
```

which it translates into (AT BALL1 (PLACE PLATE1 ON)).

Attempting to believe the given formula clearly sets up a tension, since it is not obviously compatible with TOPLE's current view of the world. What needs to be done is to be more precise about where BOX2 is in relation to TABLE. The nicest

thing to believe would be that BOX2 is under BOX1, so that BALL1 could nestle on PLATE1 in comfort; the CHOOSE called by method P-AT sets up a world-view branch to try to believe (PLAUSIBLE? '(AT BOX2 (PLACE BOX1 UNDER)) 'BELIEVE+), but runs afoul of its belief that BOX2 is bigger than BOX1. It should always be possible to doubt the truth of beliefs which cause trouble (Cf. Quine, 1963), in this case, that BOX2 is too big, but the costs on this branch are exploding, as more and more beliefs must be fiddled with for no good reason. So this branch of the goal tree is de-activated, and another one is set up, which tries to imagine BOX2 under the table only. This succeeds (at the cost of the one "unjustified" assumption that BOX2 is under the table), but the problem is only partially solved; the next thing to try is to assume that BOX1 is on the plate. Suppose this is also relatively cheap, leaving TOPLE visualizing Figure 10.

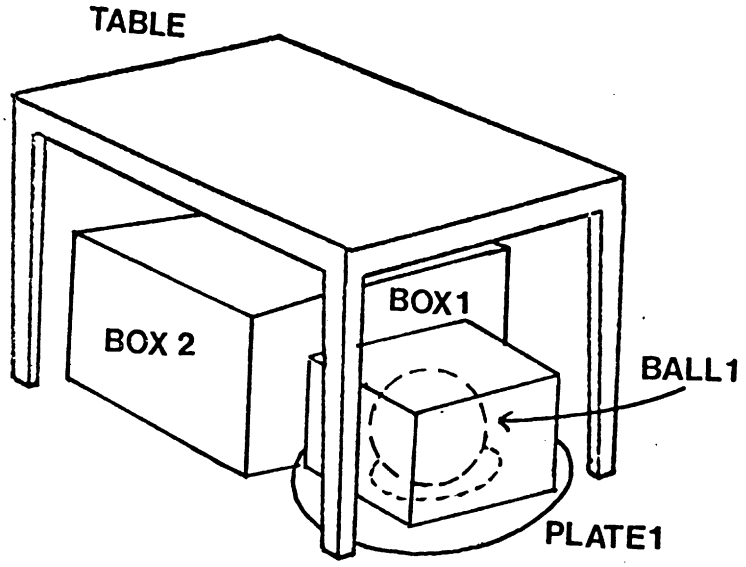


figure 10

Figure 10.

The if-needed P-AT, which generated this, now makes the following ring:

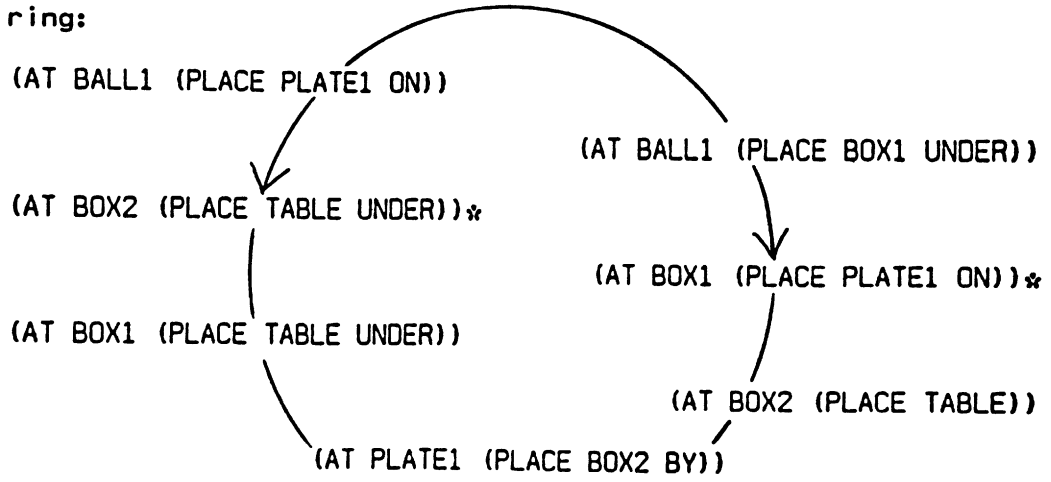


Figure 11.

with a cost of two assumptions (which are starred). (Many of the secondaries were already present.) It associates with it a reconstructor which will either DOUBT (AT BALL1 (PLACE PLATE1 ON)) or revive the attempt to find a new arrangement of objects.

In fact, suppose that a conflicting statement is discovered.

For example, imagine TOPLE is told that

```
(AT (THE W (IS W (SMALL BOX)))
    (PLACE (THE Z (IS Z FLOOR)) ON)).
```

The first THE-description is translated into (IS W BOX) and (ORDER SIZE (TYPICAL BOX) W <); i.e., W is a smaller-than-average box. This clearly refers to BOX1, so that TOPLE is attempting to believe (AT BOX1 (PLACE FLOOR ON)). Since this is contradictory to its current picture, the old ring will be smashed, and the reconstructor called to rearrange the objects. This time it tries putting PLATE1 under BOX1, to generate the visualization of Figure 12.

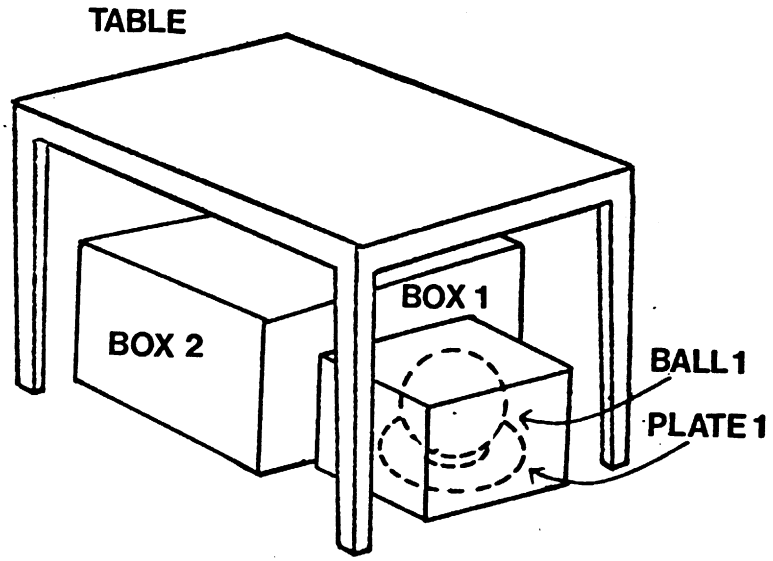


Figure 12.

and the ring

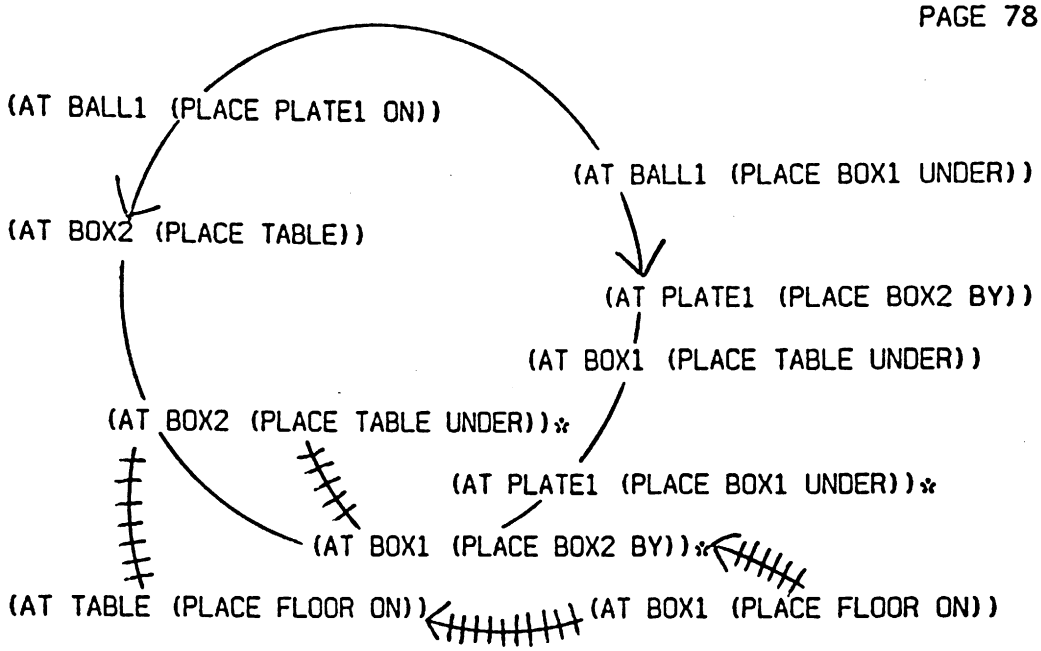


Figure 13.

(The "+++++" ring is shown for completeness; it is generated by a different process, the one that called (DOUBT '(AT BOX1 (PLACE PLATE1 ON))).) This solution to the conflict costs three (starred) assumptions, but is warranted by the tension from the new ring.

This scheme has worked out fairly well in the design of the rest of TOPLE, but there are problems with it. First, it is so general that by itself it obscures any regularities there are in types of possible reconstructors. It might be possible to find a declarative structure that would categorize the possible courses of action in case of doubt. Indeed, I believe this to be the case, and would like, in future versions of TOPLE, to narrow down exactly what a reconstructor is. However, it is one of the blessings of programming in a LISP-like language that you can let

EVAL be your "data decoder" before you are sure you know what sorts of possibilities might appear in a slot. While you use this provisional scheme, you are free to discover what regularities exist, and convert to a more appealing declarative system later.

Second, I am apprehensive as to exactly how much information and power must be put into reconstructors (or their declarative counterparts). So far, I have been able to assume that all knowledge is in the if-neededs for a particular domain. Reconstructors tend to be of the form, "Either doubt the primary of this ring or find some other way to believe it." Anyone wishing to doubt a secondary of such a ring usually has such a good idea of what the world should look like that he can add the assertions embodying his new world view to the data base, before calling DOUBT. The reconstructor then knows exactly what the doubter wants to believe, because it is right there in the data base. So far, I have managed to avoid more sophisticated PLAUSIBLE?-reconstructor interactions when they arose by picking notations that made this simple kind of communication possible, but I believe this problem must ultimately be faced and solved.

Finally, it is disappointing to me that a lot of the ring patterns which arise are determined purely pragmatically, on the basis of which assertions should cause reconsideration of which other assertions, rather than on any intrinsic deductive or

inductive relations between them. From a "tension-reducing" point of view, it is difficult to see why all but two assertions are secondaries in the main rings of Figs. 11 and 13. The answer is to be found in the action of DOUBT; I wanted doubting of either of a ring's two primaries to cause the ring just to dissolve; doubting any of the others should cause the two primaries to be reconsidered. This is okay (=it works), but makes the notion of "ring" too cloudy in my mind for comfort.

IV. Space

This chapter will treat in detail the area of competence I have given the most effort, that of thinking about the relative positions in space of the objects in TOPLE's world. Spatial concepts have been among the most elusive to formal analysis in the history of AI. Problem-solving programs (Hayes, 1971; Black 1964; Fikes and Nilsson, 1971) have usually used ad hoc formalisms for the solution of particular problems; vision programs have been able to indulge in precise number-handling, without regard to higher-order relations of support, containment, etc., separate from particular shapes and scenes. (An exception is Winston (1970), who uses semantic relations from particular scenes to generalize about visual concepts.)

Understanding real-world statements requires a different sort of notation and knowledge. One's task in understanding them is to visualize which relations are likely to be true, given what has been said. Examples of this are abundant in the dialogue of Chapter I. Another is as follows: if one is told something is under a bunch of bananas attached to the ceiling of a room, he usually assumes it is resting on the floor. If not, he must recognize an open question (in my system, a ring reconstructor) as to what does support it.

In understanding language, one must accept a wide, loose

range of locutions about location; one must attempt to understand how people think about space. On the other hand, certain very difficult problems can be finessed. While TOPLE must visualize, it does not have to visualize very precisely. For example, the "FINDSPACE" problem, that of finding room for an object in a location that may contain other objects, does not really arise; if your interlocutor tells you there are three loaves in a box, you may accept it; if you find yourself trying to picture an elephant in an oven on some branch of a CHOOSE-tree, forget it; and so forth.

So the problem is, how do people convey the essential properties of a spatial arrangement? First, we have an intuition that the world is divided into places (not necessarily disjointed). But a place is not an arbitrary point set. It is artificial to think of, say, a diagonal swath across a room from a lower corner to an upper one as a place. Places tend to be associated with objects. Some examples are obvious: inside a box is a place; between two boxes is a less well defined one. Some seem perceptually correct; every object defines one large place, which I denote (PLACE ob), which it is "AT": (AT ob (PLACE ob)). In addition, smaller object may be at the same place.

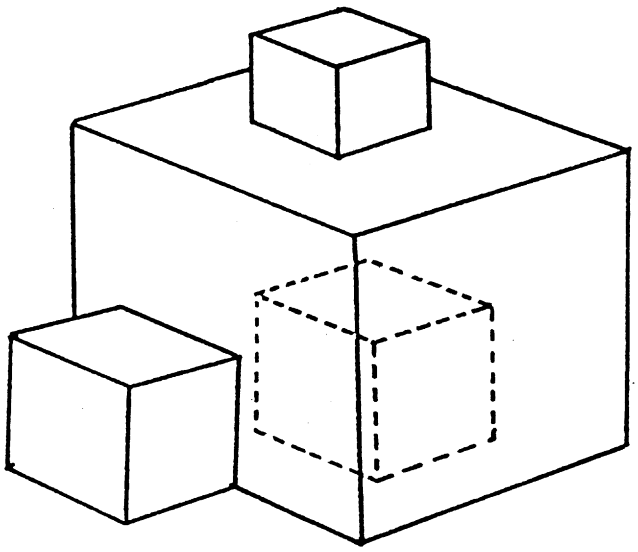


Figure 14.

BOX1 in Fig. 14 has caused a piece of space to separate out, become "visible," and (most important) arrange itself in ways that are useful to creatures with purposes. The space around this box has new properties it didn't have before. The place on top of BOX1 is special; it supports anything put on it, and stands out compared to other places above the box. I denote this as (PLACE BOX1 ON). I chose this notation, instead of breaking an object's place into its component spatial relations (as (AT BOX2 (PLACE BOX1 ON)) instead of (ON BOX2 BOX1)), partly for reasons of pattern-matching ease, and partly to reveal how it is a subplace of (PLACE BOX1); the modifier "ON" is sometimes relevant, sometimes not.

Similarly, I have the places (PLACE BOX1 IN) and (PLACE BOX1 BY), each with its special properties. (Fig. 15.)

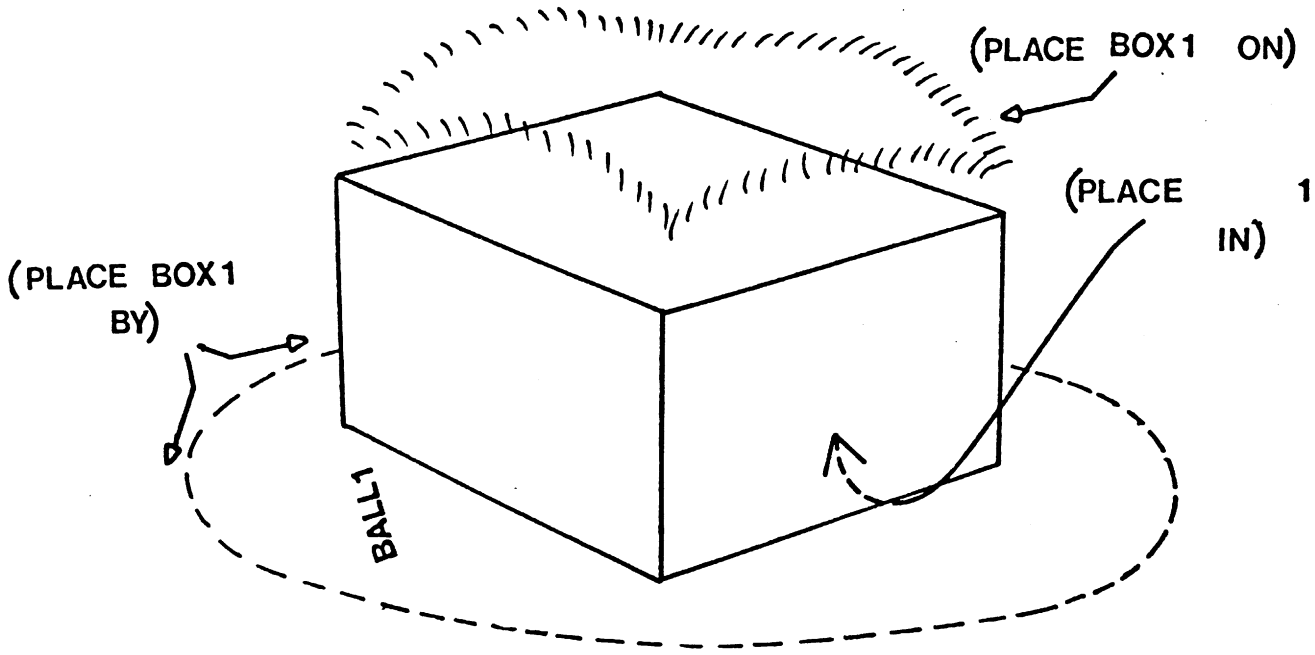
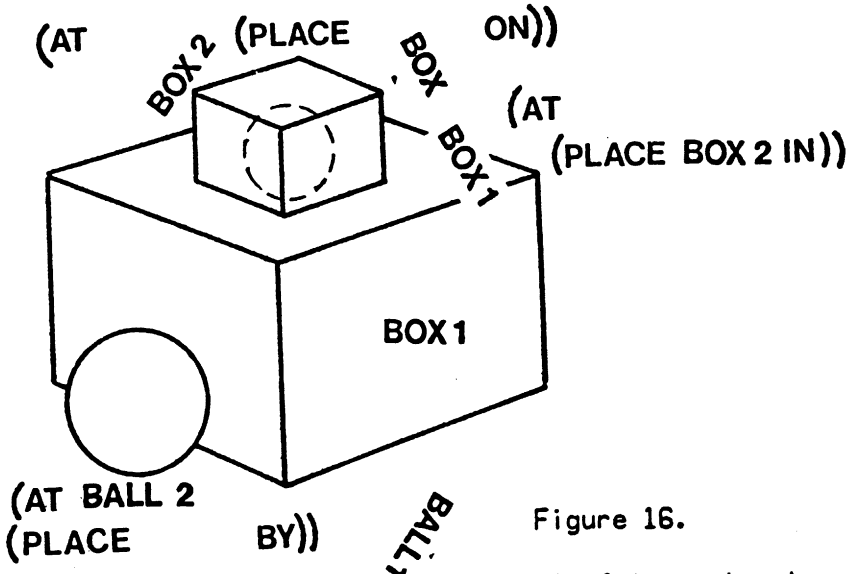


Figure 15.

Of course, there are new objects in all these places, each breaking up its piece of space.



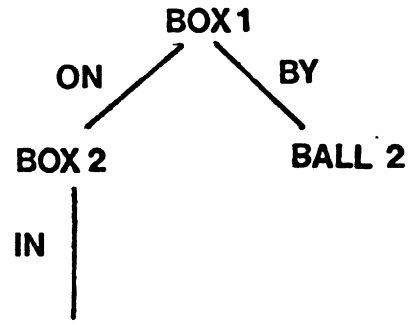


Figure 16.

This scheme enforces the sort of tree-structured hierarchy shown in Figure 16.

TOPLE now can bring to bear a good amount of knowledge in searching and manipulating these hierarchies. For example, it

assumes that IN-ness does not cancel ON-ness, so that BALL1 is ON BOX1 in Figure 16. If asked to believe two things are in the same box, it realizes that this fact and what it knows about their sizes constrain each other. If it is told a thing is in a box, it must believe it is not too tall. It is willing to allow a thing to be ON a smaller thing. And it

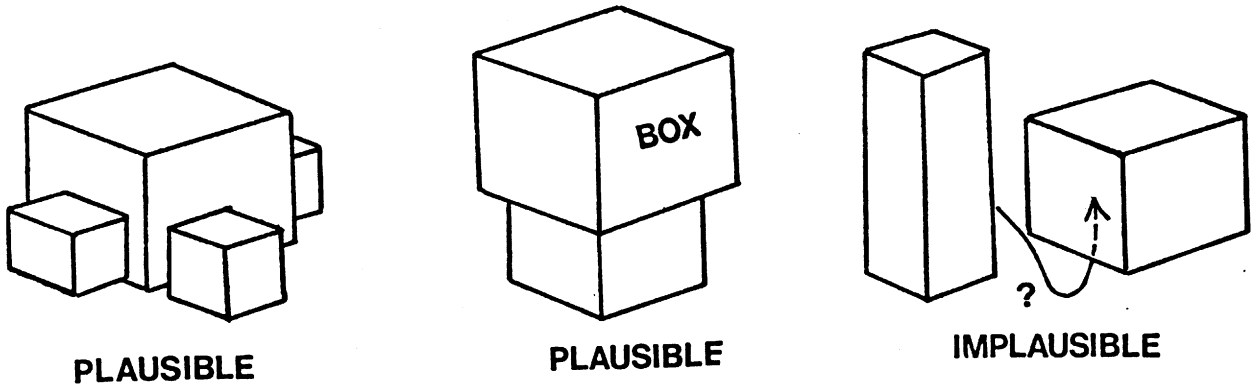


Figure 17.

allows an indefinite number of small objects to huddle BY a big one. (Figure 17.)

Of course, this scheme as outlined is too simple by far. There are several objections that can be made to it.

1. Half of you must be crying, "Space is not a tree! Look at Figure 18!"

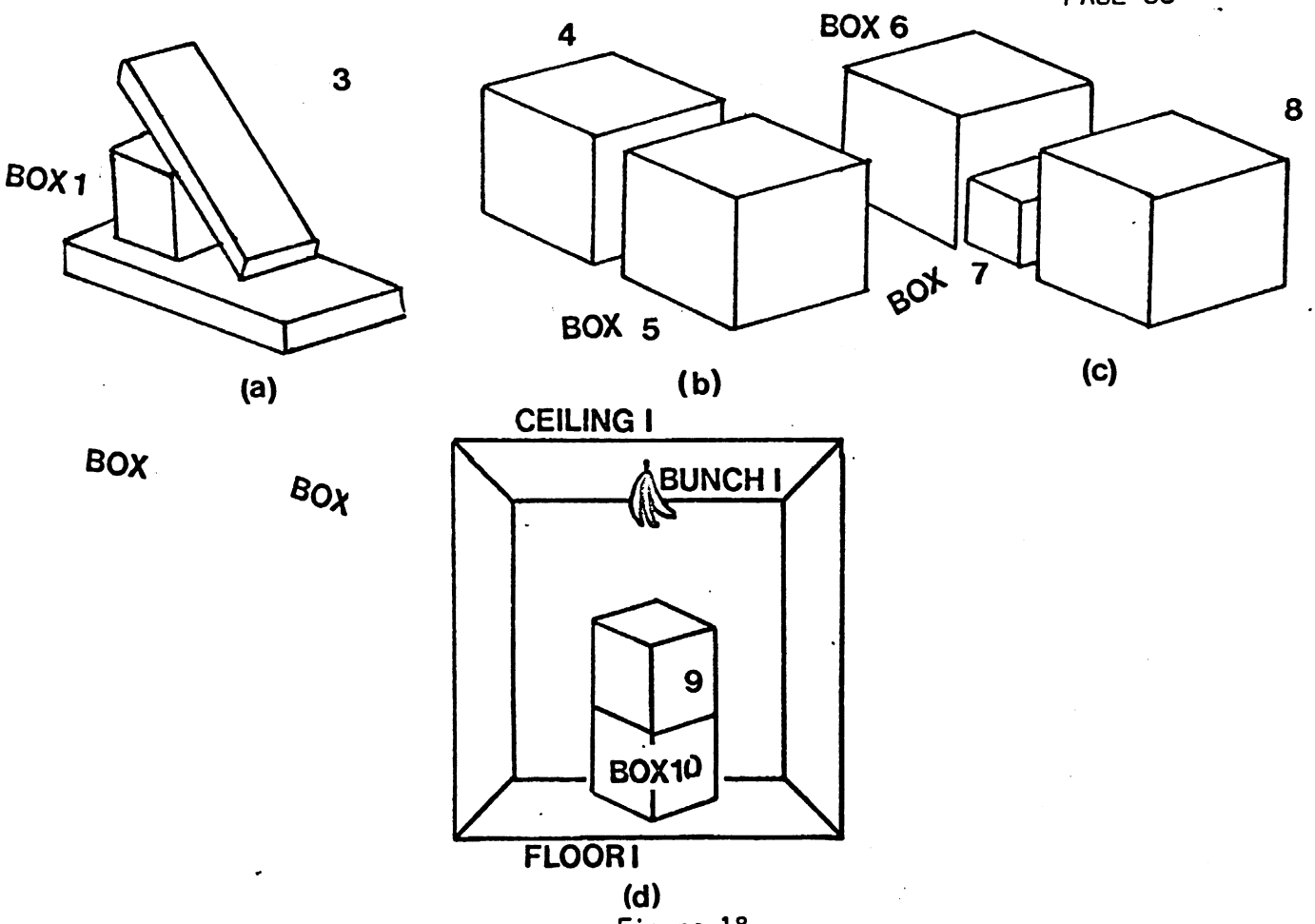


Figure 18.

True enough. BOX3 is in some funny relation to BOX2, as well as being ON BOX1. Some people would say that both (AT BOX4 (PLACE BOX5 BY)) and (AT BOX5 (PLACE BOX4 BY)) are true. BOX7 is between BOX's 6 and 8, a case of two objects creating a place. And it is unclear where BOX9 is, at (PLACE BUNCH1 UNDER), or at (PLACE BOX10 ON), or both, or are they the same place?

But, I claim, people think of space as tree-structured, after all. People ignore irrelevant spatial relations, concentrating on clustering by bigger and bigger objects. A person can tell

you the things ON his desk without usually being able to describe detailed interactions such as those of Fig. 18(a). A person thinks of the problem of getting from one place to another as getting from one place in a tree to another, often by passing through the root. (Fig. 19)

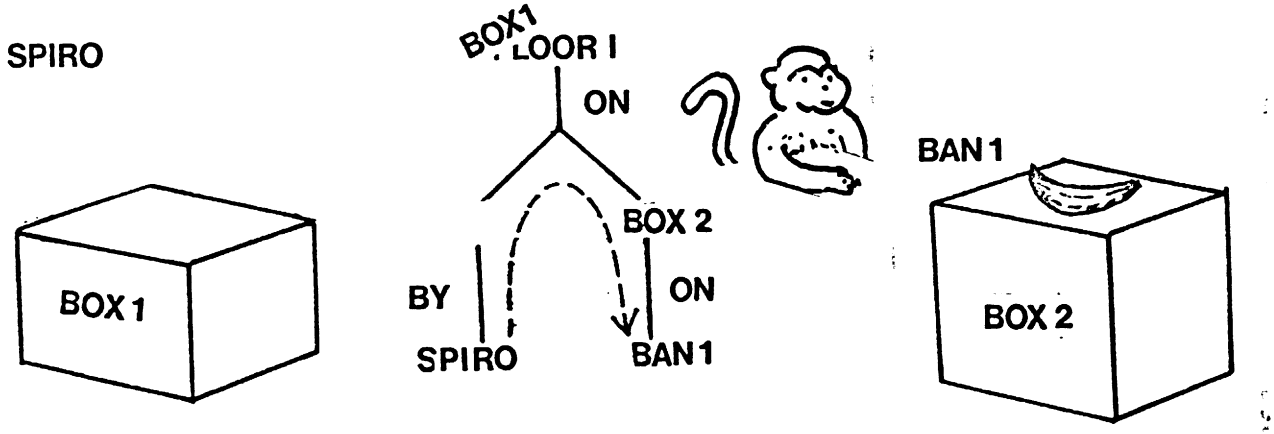


Figure 19.

People arrange cities in their heads as places hung off funny-shaped thoroughfares; good structuring leading to efficient traveling. (Some cities are hard to structure; people have lived in Boston for years before realizing they are going miles out of their way to ride the MBTA instead of just crossing a bridge.) If people had to think about where things were by searching out in several directions, it seems to me that they would bog down in a lot of time-consuming and irrelevant computation. So I have forced TOPLE to maintain a tree

structured world (modified slightly, as I shall show), as a way of forcing it to visualize things exactly. In this way, there is a place for everything, and everything is in its place.

And I have counter-objections to the objections regarding Fig. 18. Is it really as revealing as it could be to say BOX's 4 and 5 are AT each other? With both these assertions in the data base, there is an at-large symmetry, but it is obscured when each datum is fetched separately. It would seem better to note that BOX4 and BOX5 constitute a group, which divides space in its own way. Things can be on the group (as well as on each box separately). A 2-group also has a BETWEEN, just as each box in it has an IN. (Fig. 18(c).) Within a group, I suspect it is possible to describe the relative positions of its members with statements about their parts, as (AT FACE1 (PLACE BOX5)), (PART-OF BOX4 FACE1), and I envision a similar remedy for the inadequacy of my notation for Fig. 18(a). (E.g., (AT END1 (PLACE BOX2 ON)), (PART-OF BOX3 END1).)

However, I have not actually implemented any mechanism for thinking about parts and members of objects and groups. This is one of many places where I had to pull back rather than go too deeply into details in designing TOPLE.

Figure 18(d) deserves a separate section:

2. There are cases where grouping of this sort seems artificial. BOX9 is not BETWEEN BUNCH1 and BOX10 any more than

BOX10 is BETWEEN BOX9 and FLOOR1.

Let me switch to a simpler case. In Figure 20, I isolate a place on the floor, under the bananas.

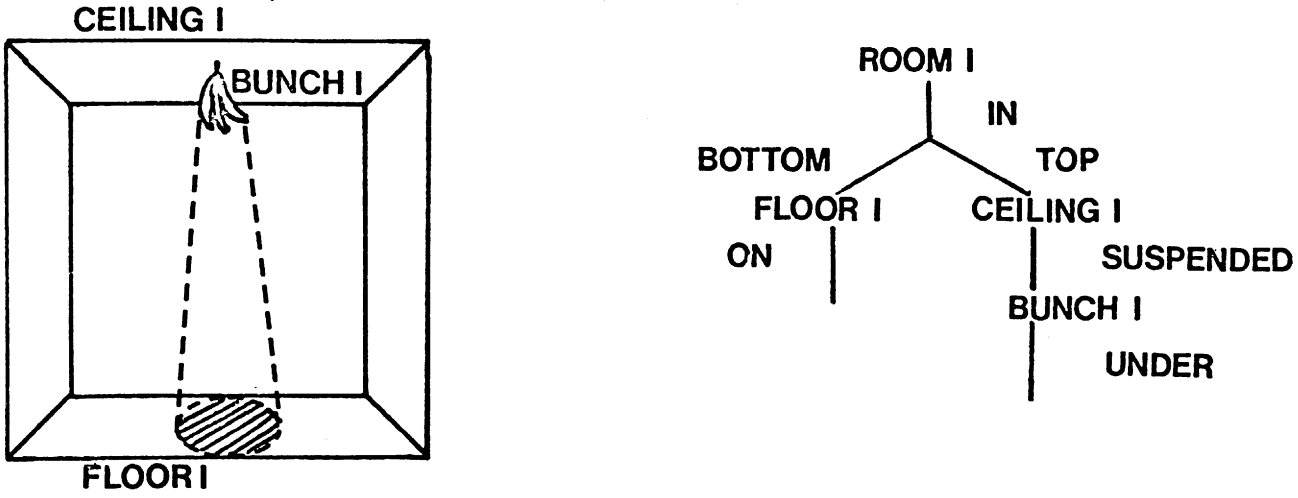


Figure 20.

How shall I talk about this place? It is a subplace of (PLACE FLOOR ON), but there is no object on the FLOOR to delineate it. In Figure 18(d), it is (PLACE BOX10), but before BOX10 is there, we have to be able to name it. (Since, e.g., the goal of getting BOX10 there may come up.)

The solution is to allow places to modify other places, if they satisfy certain abstract conditions. Thus, we say (AT BOX10 (PLACE FLOOR1 ON (PLACE BUNCH1 UNDER))), and have destroyed the tree-ness of our place hierarchies to the extent suggested in Figure 21.

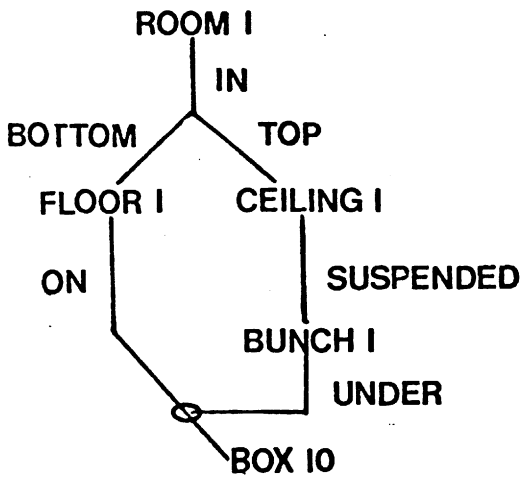


Figure 21.

Of course, not all places can modify a given place; here the condition is that the two place be on opposite branches of a tree, split by a relation (TOP-BOTTOM) that is in the same direction as the modifier (UNDER) of the intersecting place. (Other conditions are possible.)

This solution has worked quite nicely so far. In talking about objects on the floor, (PLACE BUNCH1 UNDER) works like any other modifier, but is treated as a place by the space-pondering routines when it is specifically mentioned by someone else.

Notice that the notation, while it expresses the seemingly symmetric idea of place intersection (cf. Figure 20) is asymmetric. Why not say (AT BOX10 (PLACE BUNCH1 UNDER (PLACE FLOOR1 ON)))? Principally because the support relation ON seems so much more important than UNDER; and, in fact, because the routines for thinking about Spiro's going from one place to

another need to use the trees constructed by the space-ponderer P-AT. My intention is to have the notation hide information that is probably useless.

3. The remaining objection is the same as for the other domains of competence the system has: how does it compensate for fallibility? When a new spatial relation conflicts with its old world model, how is it to be reconciled?

An example is that given at the end of Chapter III, in which TOPLE must make assumptions about the spatial relations of BOX1 and BOX2 in order to make BALL1 reside on PLATE1. Another would arise as follows: if TOPLE were told (AT BOX10 (PLACE FLOOR1 ON)), in Figure 20, then (AT BOX10 (PLACE BUNCH1 UNDER)). The correct solutions in these two cases are indicated by the tree-hierarchy transformations shown in Fig. 22.

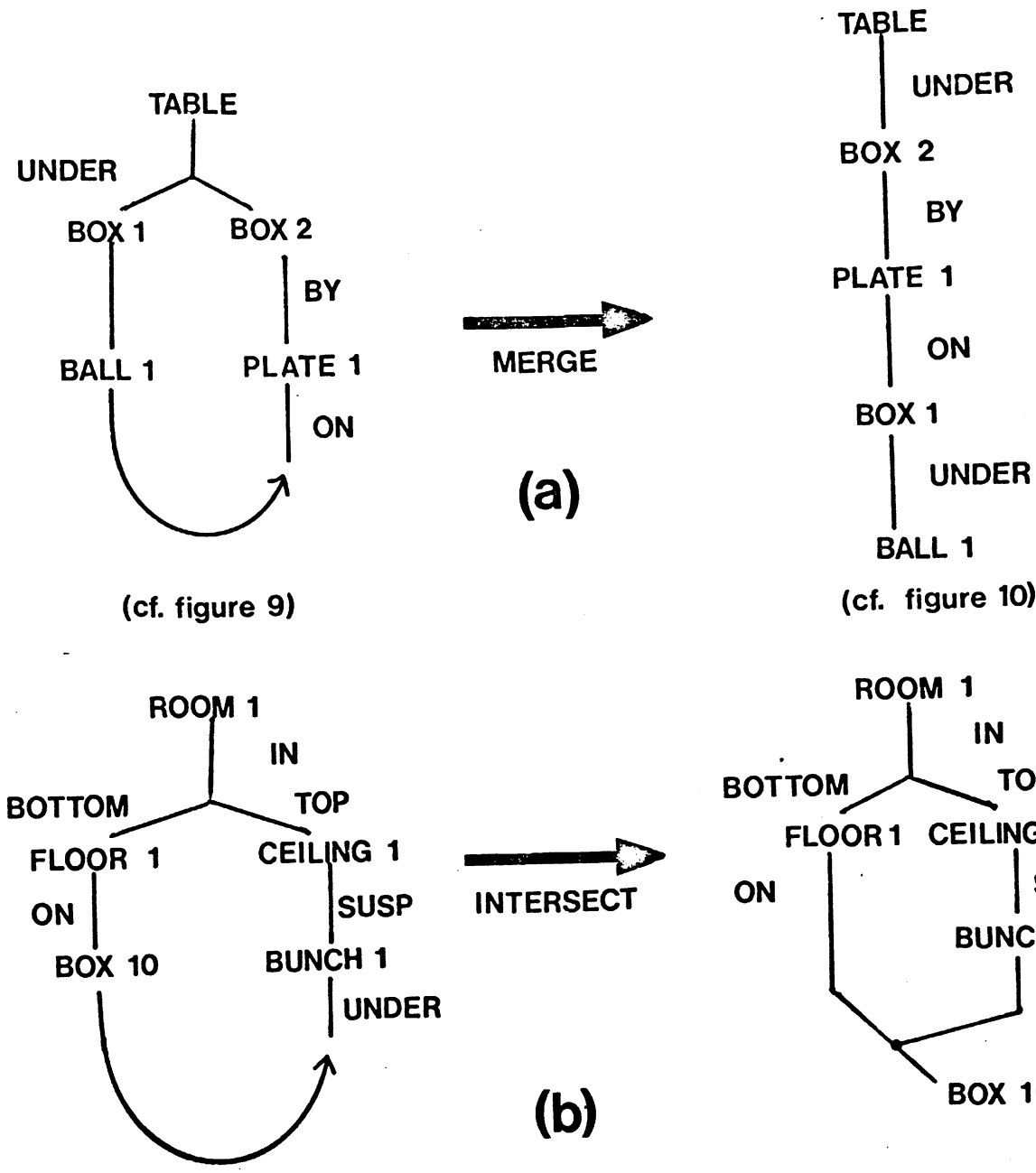


Figure 22.

These two transformations are called merging and intersecting place hierarchy branches. Which is appropriate is usually not difficult to determine.

With this in mind, I can describe in detail the operation of P-AT, the routine for deducing and believing AT-statements. It

is called by statements of the form (AT object place). The first thing it does is call (PLACEREL '(PLACE object) place). This function is a (relatively) high-speed checker of what is known. It replies to P-AT that one place is a subplace of the other, or that they are on two different branches of the place hierarchy. (It can also reply, in the case of a support relation to place, that (PLACE object) is indirectly supported by place. For example, if object is ON BOX1, which is ON the FLOOR, (PLACE object) is on the same tree branch as (PLACE FLOOR ON), but not a subplace of it.) If PLACEREL knows nothing about the relations of the two places, it tries guessing. (Currently it makes assumptions that objects with unknown supports are supported by the floor (FLOOR1); it should be more responsive to linguistic cues regarding the current setting and topic.)

If (PLACE object) is a subplace of place, P-AT creates a ring from the chain of AT-items between object and place; they are the secondaries, (AT object place) being the primary involved. The reconstructor is very simple: if you doubt any of the secondaries, reconsider (AT object place), i.e., choose again between believing it or doubting it. The reason for this is that P-AT is the most knowledgeable program in the system about space relations; so it should be given control if the item is doubted.

Notice what this implies about the behavior of the doubting routine (probably a later activation of P-AT). It must make its

changes to the world before calling (DOUBT '(AT object place)), so the reconstructor can find some new way to think about the world which is compatible with those changes. This is the simplest means of doubter-reconstructor communication imaginable, but has so far sufficed. (Cf. my reservations at the end of Chapter III.)

If P-AT is unsuccessful in deducing what it wants, it first checks to see if it is plausible that (FITS object place). This involves a simple check that the other objects at the place or the place itself do not crowd object.

If this is known or at least believable, P-AT sets up a CHOOSE among four alternatives: DOUBT that (PLACE object) is on a different branch of the place tree from place, merge the branches, look for an intersection, or imagine that object just moved. In the case of a merge, it constructs a ring of the sort I sketched in Figs. 11 and 13. An intersection ring has only the cross-links in the tree as its new secondaries. (Fig. 22(b).) In the case of imagining it GO-ing from its old place to place, it is necessary to call the routine P-GO (see Chapter V), which deals with how the place hierarchy changes with time.

The most important feature of the program P-AT is its goal of forcing a particular visualization of a scene from what it is told. It will not allow two facts about an object's location to float around independently in the data base, but forces them to

be compatible, even if this requires filling in details of the scene that might be wrong. The advantage it gains from this is that thinking about space is easier, at least until someone doubts one of these details. Then that detail can be fixed, in relative isolation from the rest of its beliefs. (Of course, doubting something with a connection to many belief rings is more difficult, but then such a belief is likely to be more plausible, too.) Thinking about space is not the only area in which I have tried to make TOPLE behave this way. In general, it picks a way to patch up superficially inconsistent beliefs and believes it as strongly as something it is told. This saves it from doing more deduction later, but requires something like the ring system I have developed in order to undo mistakes.

P-AT is only part of the space ponderer in TOPLE. There must also be routines for finding things at a place and finding where something is. There must also be an if-needed, P-SAMEPLACE, which decides whether two objects are in the same place on some scale (for purposes, e.g., of deciding whether Spiro has immediate physical control over something). The routines P-GO and A-GO decide whether GO-ing is believable and what changes to make to the place hierarchy when GO-ing occurs, respectively. (The "A" in "A-GO" stands for "if-Added.")

Finally, my solution to the problem of space is by no means complete, nor, in the parts it deals with, even final.

V. Time and Causality

I have now described the belief structure of TOPLE, and the devices it uses to express the relations between situations; it is time to turn to how it manipulates its situation-dependent beliefs so as to generate the most coherent structure.

So far, I have dealt with the details of the plausibility mechanism only in a static context. When TOPLE hears of various spatial relations, it must attempt to juggle them within a single world-situation to believe them. But when we tell it about actions, causes, and sequences of events, a new dimension of complexity is added. First, it must begin to extend the assumptions it makes into the past and future. Second, which situation is meant in a given piece of discourse becomes one of the variables to be picked by TOPLE; and this ellipsis is one that can never be fully eliminated by having our interlocutors be more precise, because natural language has no devices for specifying a situation absolutely. In this work, the problem has been alleviated by eliminating all tenses but the present. Thirdly, situations become individuals, with different properties in different contexts, but individuals of an ethereal kind. It becomes a problem when to refer to one situation with indeterminate properties; or to two or more situations, only one of which is "real." My partial solution to this deep problem is

determined by convenience rather than the consistent application of any fundamental principle.

To demonstrate these problems and my "solutions" clearly, I will work through an example given in Chapter I. In the situation of Figure 2, TOPLE is told, "The monkey goes over to the table and picks up the banana," or, after reference resolution, (AND (GO SPIRO (PLACE TAB1)) (PICKUP SPIRO BAN1)). Ignoring the details of the AND-understander, this involves a call to PLAUSIBLE? to check on the plausibility of (GO SPIRO (PLACE TAB1)). PLAUSIBLE? calls the if-needed method P-GO, a method with pattern (GO !>CREATURE !>PLACE).

As with other beliefs, the first thing P-GO does is check the likelihood that this action is already known to have happened or not to have happened. If a belief that Spiro went somewhere is in the data base, the relation between where he went and (PLACE TAB1) can be considered, leading to a conclusion of yes or no. If some other action is known to have been performed by Spiro, TOPLE assumes he didn't go anywhere, since it believes creatures capable of one action at a time.

All of this is similar to the world-checking phase of a static belief routine. In case of ignorance, however, P-GO must set about adding the assumption to the data base. The first thing it does, of course, is to report trouble IGNORANCE back to TOPLE, which considers whether it has guessed the wrong

situation, and thus whether predictions should be checked. But I leave a description of TOPLE's prediction-finding-and-flushing mechanism until I have described how predictions are generated.

Assuming TOPLE allows P-GO to proceed, it must now call routines in the space module to decide on the possibility of going from where CREATURE is to PLACE. There are several reasons this might be impossible: CREATURE might be locked in a cage (or, in general, have to get to the same "enclosure" as the place it is going); the PLACE might be too small to hold CREATURE (e.g., if the ball is in the box, (GO SPIRO (PLACE BALL1)) might really mean (GO SPIRO (PLACE BOX1)); the PLACE might not be in the same horizontal plane as CREATURE, necessitating some climbing or jumping as well as GO-ing.

These difficulties should give some idea of my conception of GO-ing, which I hope corresponds to a subclass of the full English meaning of the action. It is a primitive operation, corresponding to a creature's walking from one place to another, carrying with him everything he is holding or which is attached to him. (Path-finding is ignored as a problem in this setting.) However, P-GO recognizes that people (and other sections of the problem solver) may be careless, as we saw in Chapter I. Therefore, it may re-interpret statements about going, or fill in details about side actions and prerequisites. This "ambiguity tolerance" (Dreyfus, 1972) lets people and other sections of

TOPLE be relatively vague about going, knowing P-GO will figure out what they mean from what is possible. For example, P-PICKUP (see below) may have to believe a GO-statement to satisfy a proximity prerequisite. It can phrase it as (GO creature (PLACE thing)), knowing P-GO can correctly interpret this as going to the thing's container if necessary.

Assuming P-GO has guessed a meaning for its pattern, and has (possibly) made some space assumptions to satisfy the prerequisites for going, it is time to look for a reason for CREATURE to do what it did. This is typical of action routines. There are two components in believing the actions of animate creatures: believing they're possible, and understanding the motives for them.

In the case of going, the only motive TOPLE understands is that there is something at the destination that CREATURE wants to hold. Consequently, for each thing at that place, it attempts to believe (WANT creature (HOLD creature thing)). This calls the dull method P-WANT-HOLD, which knows almost nothing about monkeys except that they like to eat, and wanting to eat something is a plausible supergoal for wanting to hold it. (As a token concession to primatehood, it also grants monkeys an occasional desire to play with toys and unfamiliar objects.)

Notice that these WANT routines are in charge of believing conditions. P-WANT-EAT, for example, calls P-HUNGRY, which

checks what a CREATURE has eaten lately. However, when a WANT-method decides that its want is actually plausible, its work is only beginning. The addition of a statement of the form (WANT !>CREATURE !>CONDITION) to a situation causes TOPLE to simulate CREATURE with goal (ACHIEVE !>CONDITION).

It should be emphasized that this simulation of a "monkey" is extremely crude, for two reasons. First, the model of space and shape is just not good enough to handle detailed consideration of his activities; most complex motivations depend on non-simulable activities such as curiosity or boredom. Second, the simulation is meant to suggest what actually might happen, so if it is too detailed, most of the details will be useless. As a consequence, my "monkey" behaves a lot like a rigid pillar with a magnetic hand that reaches for, and sticks to, food, and can make it disappear.

Simulation is little different in most respects from the normal operation of the system. The variable ACTOR is bound to the person being simulated, and the goals tend to be of the form (ACHIEVE...) rather than DEDUCE or BELIEVE+, but all the methods that simulate a given creature are free to call PLAUSIBLE? at any time to discover things about the world. A typical ACHIEVE-method checks to see if its condition is already satisfied. If not, it decides what actions are necessary to achieve it, and executes (PERFORM action) for each one. It may do this via the

achievement of certain subgoals, which is done by recursive calls to ACHIEVE.

The creature simulator behaves like a robot problem solver (Fikes and Nilsson, 1971; Winograd, 1971; Fahlman, 1973), although it is fairly stupid, since studying such problem solving is not one of the major goals of this project. In fact, where a typical problem solver usually constructs a plan for achieving some goal, TOPLE is interested only in predicting the behavior of some creature with that goal. Thus, where other systems have their PERFORM primitive add an instruction to a plan, TOPLE turns (PERFORM '(action . objects)) into (PLAUSIBLE? '(action actor . objects) 'BELIEVE+). For example, when ACTOR=SPIRO, (PERFORM '(GO (PLACE BAN1))) calls (PLAUSIBLE? '(GO SPIRO (PLACE BAN1)) 'BELIEVE+).

Now we have gone full circle. (Fig. 23.) P-GO was called by PLAUSIBLE?; P-GO calls P-WANT-HOLD, which calls P-WANT-EAT, which assumes (WANT SPIRO (EAT SPIRO BAN1)), which simulates Spiro with goal (EAT SPIRO BAN1). The routine ACH-EAT tries to ACHIEVE (HOLD BAN1), which causes it to PERFORM (GO (PLACE BAN1)) (to be followed by (PICKUP BAN1)).

But PERFORM calls PLAUSIBLE?, which calls P-GO. How do we avoid an infinite recursion? By having P-GO assert the action (GO SPIRO (PLACE TAB1)) before calling P-WANT-HOLD. This assertion causes A-GO to create a new successor situation with

SPIRO AT the new place. Now, when the call to P-WANT-HOLD results indirectly in a new call to P-GO, it does something entirely different: it merely verifies the GO assertion. There is no infinite recursion.

Now ACH-HOLD PERFORMs (PICKUP BAN1). This call to PLAUSIBLE? to BELIEVE (PICKUP SPIRO BAN1) is different. There is nothing to verify; P-PICKUP must act similarly to P-GO: verify that its pattern is possible, add it to the data base, and then explain it.

A difficult design decision here is what it means to add such a thing to the data base. Even though this is a call to PLAUSIBLE? with goaltype BELIEVE+, the intent is clearly not to assert something about the situation of the present (i.e., immediately following (GO SPIRO (PLACE BAN1))), but about the near future. This would seem to justify the creation of a predicted world-situation. However, this new state is not different in logical content from its predecessor. An alternative might be to make certain context-dependent changes to the current situation instead of creating a new one. Then verifying a prediction would mean accepting a context (cf. Fig. 8), rather than following a situation branch (Fig. 3). This would be analogous to the way I work reference resolution (Chapter VI), where a context is generated for each possible referent, and each is tested in the environment of the

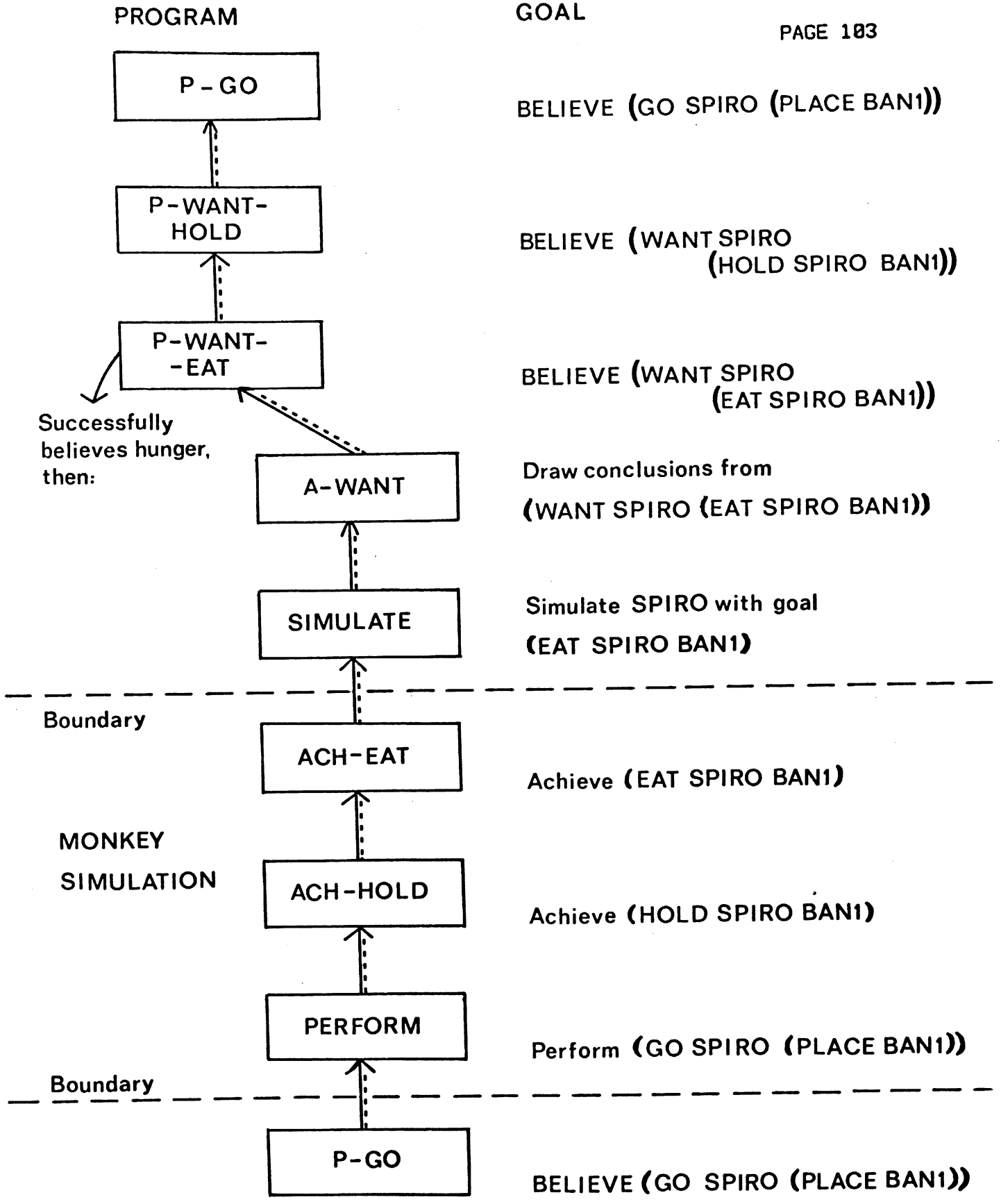

PROGRAM
GOAL
P-GO
BELIEVE (GO SPIRO (PLACE BAN1))
P-WANT-HOLD
BELIEVE (WANT SPIRO (HOLD SPIRO BAN1))
P-WANT--EAT
BELIEVE (WANT SPIRO (EAT SPIRO BAN1))
Successfully believes hunger, then:
A-WANT
Draw conclusions from (WANT SPIRO (EAT SPIRO BAN1))
SIMULATE
Simulate SPIRO with goal (EAT SPIRO BAN1)
Boundary
ACH-EAT
Achieve (EAT SPIRO BAN1)
MONKEY SIMULATION
ACH-HOLD
Achieve (HOLD SPIRO BAN1)
PERFORM
Perform (GO SPIRO (PLACE BAN1))
Boundary
P-GO
BELIEVE (GO SPIRO (PLACE BAN1))

figure 23

surrounding phrases. In fact, this is the way uncertainties about the future are handled within a single SL paragraph; in the dialogue of Chapter I, the uncertainity between [5] - [13] and [14] - [20] is one of context. But from one paragraph to the next, uncertainties can be packaged only as assumptions attached to alternate possible futures; the uncertainty between [25] - [30] and [31] - [40] is of this sort. It is clear why the two cases differ: uncertainty about the present is too expensive to indulge in for too long, but must be tolerated (for a while) right after it is encountered. However, doing things this way requires creation of a new situation solely to get the future it represents out of the way. In this and in other cases, the boundary between one situation and the next is often one of convenience.

At any rate, someone has to decide when to forecast a situation rather than add a datum to the current situation. These decisions are made by "hidden" modules of SIMULATE and TOPLE which are sensitive to the troubles of the belief routines beneath them, and automatically search for or create futures in which these beliefs are true. In this way, a picture of one or several futures is built up.

Although I have not described the process in detail, these futures are checked at several places whenever some if-needed reports difficulty in matching some pattern to a particular

situation. At the top level, when some belief cannot be matched to the current situation or a predicted one, it is crunched in forcibly by the process I have now described a couple of times (for P-AT and P-GO). In a simulation, however, such a difficulty causes generation of a prediction.

Hence, when one tells TOPLE about Spiro going to some place, it adds any necessary prerequisite information, then assumes his GO-ing and being at the new place. The call to P-WANT-HOLD causes consideration of a set of motives, leading to a simulation, including a re-checking of going and a prediction of picking up and eating.

Leaving aside the question how justified these predictions are, notice how nicely this mechanism implements a sort of logical "completion" phenomenon. No matter at which point entrance is made into this set of routines, it attempts to impose much the same structure on the world. If TOPLE were told the monkey was hungry, it would predict the whole sequence; if told he wanted to pick up the banana, it would fill in the hunger part, then make the prediction; if told, "he picks up the banana," it would put in the prerequisite (GO SPIRO (PLACE BAN1)), then proceed; finally, if it is told this last item directly, its consideration of motives leads it to envision the whole sequence.

However, there are some pieces of knowledge that cannot be

gotten to from just any direction. P-GO believes that the only motive for going anywhere is to get something. But ACH-AT will PERFORM GO-ing to fetch, say, a box to climb on. This box is not wanted to play with or to eat, so P-GO will be confused if it is told this element of the episode before being told the monkey's motives. However, it is perfectly capable of understanding such an action in the context of a large, predicted plan. This is a reasonable restriction; it would be absurd, when hearing of a creature GO-ing, to imagine all the indirect goals he might have. Behavior which is only indirectly related to a goal is often hard to understand; stories (especially children's stories) often have a character go through a series of actions that are unintelligible until his overall motives are revealed. It is more important that a process-understander be able to relate a creature's goals to his actions once he has heard them than to be able to deduce everything the creature wants from what it does.

In this way, I have implemented one aspect of the "scenario" idea (Abelson, 1973; Winograd, in conversation), which threatens to become as catchy as "procedural embedding of knowledge." The theory is that particular events are to be understood as belonging to wholes of identifiable types, which should in turn give rise to expectations of new particular events. It should be clear what parts of TOPLE correspond to the parts of this mechanism.

My program is deficient in two respects with respect to this theory: (a) the scenario is not represented very explicitly, but inheres in one or more programs which understand things; (b) the notion of a "worry list," or set of open, unanswered questions is not implemented at all. The best TOPLE does in the last department is to force answers to some questions, but leave a ring reconstructor associated with the answers which asks the question again if the provisional answer conflicts with later revelations. An example is its attitude toward physical objects; as soon as it begins considering where one is (in PLACEREL), it pictures it as being supported by the floor. (It does not know about birds and other "unsupported" objects.) This is only an assumption, so it is willing to do away with it, but only if the question, "What does support it?" can be answered. In this way, a kind of "scenario" for physical objects' spatial relations is implemented, with the question of support on the "worry list." But this is not really a satisfying implementation, because the list of important questions is not associated well with the success or failure of an overall scenario, and because the list cannot include unanswered questions.

One aspect of prediction I have not dealt with. Not all predictions come true. But not all not-immediately-verified predictions are false. How is it to be determined that a

prediction is inaccurate?

I do not believe there is a single, universal answer to this question, although some general principles present themselves. If a prediction has a time limit--e.g., it will happen immediately or not at all--the passage of time can be a signal. More generally, if a prediction is based on one or more assumptions, and if what happens subjects those assumptions to doubt, it makes the predictions doubtful, too.

Specifically, TOPLE possesses the ability to make mutually exclusive predictions, based on conflicting assumptions. This is a nice case, because if one possible future occurs, its alternatives are painlessly disconfirmed. In the sample behavior of Chapter I, an example of this occurs: when Wolfgang attaches the bananas to the ceiling, the position of the table determines what Spiro will do. Since the table is not, in fact, under the bananas, he does not climb it, but goes over to try to jump for them. This results in a decisive disconfirmation of the alternative, as well as a discovery as to where the table is relative to the bananas.

How these mutually exclusive possibilities are generated deserves some comment. When to split a future-generation into two pieces is clear: whenever a deduction made during the simulation reveals difficulty IGNORANCE. In this case, TOPLE saves a tag to the problem spot at the top level of the

simulation, but goes on, assuming the assertion in question is false. When the simulation is done, it goes back over the tag it has generated, and starts it up again, with a different context, this time assuming it true. Each run creates a different future. When the simulation is done, it finalizes the two opposing contexts in such a way that any assumptions made in one of them about the past are saved in a "recipe" for entering the possible future it represents.

My concern over these techniques is a reflection of my feeling that programmed monitoring of other programs is an important approach to intelligence. In this case, one set of programs is simulating a monkey; the monitor is using its ignorance to generate alternate possible futures. There is no reason why it could not use a similar technique to simulate the program on a test case, to study or debug the program rather than the world model. In the case of my monkey program, TOPLE did not write, and does not understand, the pieces of itself it watches; if it had written them itself, from English sentences about monkeys in general, its monitoring could be more informed and important.

After all of this, there still remains the problem of a prediction that has not been explicitly confirmed in the story, nor passed by on the way to a clear alternative. I have no general solution to this, but adopt the following course: I

associate with each predicted situation a reconstructor (Chapter III). Its primary element is, in the case of a simulation prediction, the WANT-assertion that gave rise to it. The ring structure for all the assumptions made during a simulation reflects the subgoal structure that existed when those assumptions were made: individual actions depend on WANTs which are the record of the ACHIEVEment of subgoals of higher-level WANTs.

If TOPLE finds it impossible to verify a prediction in believing something new, it typically has two alternatives: move the prediction so it follows the new belief, thus postponing confirming it; or doubt it. These are explored under the control of the reconstructor, a program which is given the difficulty (e.g., IGNORANCE or CONTRADICTION) that stopped the attempt to verify that predicted state of the world. The typical reconstructor for the ring which was built by achieving a (simulated) subgoal of some creature reruns that piece of the simulation, on the assumption that the creature found a different way to accomplish that one subgoal, but that his overall plan remained the same. Then it tries to salvage as much as possible of the rest of the ring structure created by the simulation, by BELIEVing each of its pieces in the future following the new prediction.

As a less obvious example, consider what happens with

jumping. The routine for achieving AT knows that it can achieve getting to places slightly higher than the actor by jumping. The if-added A-JUMP, which records the effect of this action, makes a prediction that the jumper will fall back to the surface he started from. It associates with the predicted situation a reconstructor which flushes the prediction if there is a real contradiction--if, for example, the creature grabbed something solid. Otherwise, it assumes the situation has come to pass anyway, even though it hasn't been explicitly mentioned. (No one is likely to add, "He falls to the floor," to, "He jumps up and grabs/misses the bananas.")

I am not too happy with this technique as it stands, but I think it can be refined. One problem with it is that it is not a solution to the problem of how to add information to the past. TOPLE avoids this problem with the real past, by suppressing tense, but the present is unavoidably the past of the future. With respect to predictions, one always has the recourse of re-predicting to see how new information has changed the future, but this is hardly a useful technique for fiddling with the past; re-processing everything since then is unfeasible. But something must be done. If it had concluded a ball was in a box, how shall new knowledge that something else was there affect TOPLE? Perhaps it is less likely now that the ball will fit. Or imagine that TOPLE has believed (rightly) that the monkey moved BOX1

across the room at one point for some reason. Now it discovers an anvil was in the box, an anvil light enough to extract from the box, but too heavy to carry with it. It must add that the monkey took the anvil out before it lifted the box.

I am hopeful that these problems can be solved by extensive use of demons (Charniak, 1972) to record the assumptions TOPLE makes in believing something. This sort of demon would be a pattern-directed procedure that would be executed whenever someone added an assertion that upset such an assumption. For example, when P-FITS is satisfied that it knows of nothing else in a box to get in the way of a ball, it could generate a demon to notice the insertion of anything else into the box in that situation. If this happens, the demon checks to see if the ball still fits--i.e., it triggers the recontructor of the ring the way any other doubter of it would. This demon itself would become part of the ring.

There are problems with this scheme. The most prominent is writing demons with general enough patterns. For the anvil example, the pattern (AT !>THING (PLACE BOX1 IN)) will not do; if BOX2 is in BOX1, (AT ANVIL (PLACE BOX2 IN)) should be detected also. Evidently, the if-added method requires improvement if it is to implement demons.

I have not studied demons extensively in this work, partly because of this difficulty, and partly because I was more

interested in creating a framework (situations, rings, and other structures) they could operate in than attempting to encode much knowledge in them. My treatment of belief about the future is different from Charniak's (1972) work mainly in my attempt to specify declaratively what is to be true about the future, so that PLAUSIBLE?, rather than a simple pattern-matcher, can be used to verify a given alternative. Nevertheless, his work has shown the advantages to be gained by using demons cleverly, and I am confident that many of his results can be used in a TOPLE-like framework.

VI. Other Knowledge

Size and Shape

The current version of TOPLE has a limited ability to think about the size and shape of the objects in its world. It uses this information in deciding whether a group of objects will fit in a place, in thinking about climbing to reach an object, and in calculating other distances.

A full analysis of size and shape, especially the latter, would be very difficult, but an essential part of a complete system able to reason about space. As suggested in Chapter IV, one way to describe the relative orientation of objects is to describe the locations of their parts relative to each other. To do this would require a language for describing groups and multi-part objects. (Presumably this would be in terms of standard shapes with standard parts adjacent or attached to each other.) A full solution would take into account that the size and shape of an object may change with time. (For example, in one sense, all creatures change shape whenever they move.)

My analysis is a great deal simpler. I assume every object is constant in shape and size. Each one is to be treated as having a SIZE, HEIGHT, and WEIGHT. The meaning of the latter two is fairly clear. SIZE is roughly proportional to cross-sectional area, but of two objects with about the same area, the shorter

has the smaller SIZE.

There are no absolute measurements of these properties, but a partial ordering of objects by each of them. Assertions about the order of property "p" are stored as (ORDER p obj1 obj2 modifier), where "modifier" gives the details of the relation between obj1 and obj2. For example, (ORDER HEIGHT BOX1 SPIRO >=) means "Spiro is slightly taller than BOX1." These modifiers may be as specific as a numerical factor ("obj2 is n times as p as obj1") or as general as ">*", meaning, "from slightly bigger to enormous by comparison."

When an order relation is added to the world model, the if-needed P-ORDER of TOPLE makes sure it is consistent with everything else, then deduces anything new it can from the new relation. Because of modifiers in ORDER assertions, it is often possible to draw conclusions from two assertions with the same upper (or lower) bound. For example, from "BOX1 is slightly larger than BOX2," and "ROOM1 is enormous compared to BOX2," we may deduce, "ROOM1 is enormous compared to BOX1." These deductions are done by looking up the relevant modifiers in tables which tell how they are to be composed, complemented, combined with numerical modifiers, etc. The modifiers currently implemented are >> (much greater), > (two to five times greater), >= (slightly greater), = (approximately equal), 1.0 (exactly equal), the other numbers, and >* (greater to some degree), and

the corresponding "<"-modifiers. P-ORDER maintains order assertions as a partial ordering, in which all redundant information is hidden; for example, if there are two assertions with the same object as lower bound, it means that the upper bounds cannot be related.

In the partial ordering of objects, there is occasionally a node of the form (TYPICAL kind), as (TYPICAL TABLE). For example, (ORDER SIZE (TYPICAL BALL) (TYPICAL BOX) >) is believed by the system. This information is used by P-ORDER whenever it has no size information about an individual, but knows its type.

These beliefs are not as useful as they could be. There is no way currently to record a range of typical sizes for a class of objects. (E.g., "Boxes can be arbitrarily small, are never bigger than rooms, and are usually slightly smaller than a table.") Also, TOPLE knows no way to doubt information about TYPICAL objects. This is because, like all information about more than one event or condition, it must be modified and debugged rather than just confirmed or discarded. (See Chapter VII.)

ORDER assertions are used by the space routines in various ways. P-FITS allows any number of very small objects into a container, but only a few bigger ones, and only one object if it is just slightly smaller. ACH-AT uses height information to decide if a creature can jump for an object. P-PICKUP allows a

creature to pick up only objects lighter than himself.

This system is the beginning of a complete mechanism for understanding size information, but is obviously inadequate. It lacks ability to express a lot of comparisons. There is no way to compare heights with anything but heights; comparison with widths might be useful in describing a cube. There is no way to express information about functions of size quantities. For example, "the box, stacked on the table, reached to the ceiling," would seem to imply, "the sum of the heights of the box and table is just less than the height of the room."

One problem with generalizations like these is that they involve unlimited statements about equality and inequality, which traditionally present a lot of difficulty to theorem provers. (Robinson and Wos, 1969; Nevins, 1972). If this domain can be handled at all, it will be by reducing problems in it to the point where calculation will be required instead of deduction (cf. Bledsoe et. al., 1971). Then formula manipulation techniques could be applied.

On the other hand, it may be that for language comprehension one should do no better at thinking about inequalities than people, and it may be that people use only the crudest rules for summing sizes and comparing heights and widths. It shouldn't be too hard to add to TOPLE knowledge like, "the sum of a box's height and a slightly larger box's height is slightly larger than

2 times its height."

Reference Resolution

TOPLE stores objects internally as atomic CONNIVER objects, present in the contexts of the situations in which they exist. But references are not normally to such objects in conversation. ("Spiro" is an occasional exception.) Instead, people are allowed to refer to objects with pronouns or descriptive noun phrases.

TOPLE allows objects to be referred to with pronouns or noun phrases beginning with THE or A, albeit only in simple ways. All such references occurring inside a predicate formula must be turned into definite objects before PLAUSIBLE? is applied to it. However, there is in normal discourse often more than one candidate for a given referring expression. Some of these can be weeded out by modifiers occurring in the referring expression. (For example, "ripe" in, "Spiro ate the ripe banana and threw the green one away.") However, there are always under-described objects, especially those referred to with pronouns. ("Jack threw the ball to Bill. He threw it back to him." (Charniak, 1972)) Here judgment must be suspended as to which meaning is meant until a wider context is taken into account.

TOPLE's solution to the reference problem is elegant, as far

as it goes. It sets up at each level of an SL paragraph and formula a CONNIVER generator function. Each of these generates either an object (if it belongs to a referring formula like (THE...)) or an interpretation (for a predicate formula like (AT...)). Each generator also returns a separate context, as returned by PLAUSIBLE?, which is used to sprout the alternatives returned by later generators. Thus, if a predicate formula has two references (e.g., (AT (THE X (IS X BALL)) (PLACE (THE Y (IS Y BOX)) ON)) for, "the ball is on the box"), and each reference has two potential objects it might refer to, there are four branches of the goal tree that might be generated before (AT ball box) is handed to PLAUSIBLE?. Clearly, other references and ambiguities increase the possible size of this tree multiplicatively.

However, TOPLE does not start out blindly generating all possibilities. It tries to take the first thing generated by each generator and stick with it. Only when an implausibility is encountered does it suspend operations on its established branches and make new ones. (There are still inefficiencies in this procedure that may be avoidable; in the example given TOPLE currently regenerates all possible boxes each time it picks another ball. It may be possible at least to save the boxes and try them first with the second ball.)

The generator of interpretations for predicate formulae is PLAUSIBLE? and its if-needed methods. Two other if-neededs, R-

THE and R-A, generate noun-phrase referents. R-THE calls PLAUSIBLE? in BELIEVE+ mode on each clause of (THE var - clauses-), in an attempt to find methods that can generate objects with the characteristics described in that clause. (If one runs into trouble CONTRADICTION, it fails completely. IGNORANCE, however, can be overcome. For example, in "One banana [BAN1] was green. Spiro ate the ripe banana [BAN2]," the assumption (RIPE BAN2) will be added to the data base; if BAN1 is considered, it will be rejected.)

R-A assumes its object is brand-new. It gives it a new name, and BELIEVEs all of its properties. (Obvious inadequacies in this scheme will be discussed below.)

Pronouns are handled by R-HE, R-SHE, R-IT, and R-THEM. (SHE X) is treated almost like (THE X (IS X ANIMATE) (SEX X FEMALE)); (IT X), like (THE X (IS X INANIMATE)); etc. However, the order of generation of objects is not random, as in the case of THE. Instead, the reference-resolution equipment keeps a list of referred-to objects in reverse chronological order. It generates them in this order for us by the pronoun generators. This is a very crude implementation of the heuristic advice that the plausibility of noun-phrase candidates is less closely tied to recency of reference than candidates for pronouns; in TOPLE, recency is ignored for noun phrases. Over the short time suggested by Chapter I's sample dialogue, this works. However, a

more sophisticated system should take recency into account.

There is one more frill in this system. R-THE and R-A have (IS var kind) as their first clause. In the case of kind=CORNER, the method P-IS-CORNER knows that CORNER is not really a kind of object, but a kind of place. Therefore, it checks that it was called in the context of generating referents of a THE-expression, and that this expression occurs in the context (PLACE...IN). If it was not, the method fails with trouble MEANINGLESS. Otherwise, it short-circuits the generation of (PLACE...IN)'s to generation of places of the form

```
(PLACE (THE X (IS X FLOOR))
       ON
       (PLACE (THE Y (IS Y WALL)) BY)
       (PLACE (THE Z (IS Z (OTHER WALL))) BY)).
```

Many other tricks of this kind could be imagined. (For example, "the clutch" refers to nothing in "He was a good player in the clutch.") However, only the CORNER device has been implemented, since it was needed for the sample dialogue, and since a systematic study of such pseudo-references might take years.

In fact, my treatment of reference handles only the most obvious corner of this vast problem. What is remarkable about what I have implemented is that it came virtually free with the plausible inference mechanism.

Even if no wider uses of reference were to be handled, my

program could be much more clever if it were more intelligent about generating possible referents of an expression. Methods for doing this have been explored by Winograd (1971) and Charniak (1972).

TOPLE is more sophisticated about choosing a referent from a set of possibilities than Winograd's SHRDLU. Charniak's "restriction method" for evaluating referents is quite different from TOPLE; it is less likely to get bogged down in a bushy tree than TOPLE, but I suspect it is too jumpy about rejecting alternatives that do not immediately conform to restrictions encountered in Micro-PLANNER theorem patterns. On the other hand, Charniak's method is able to generate candidates on the basis of previous predictions of what sentences are likely, as opposed to generating objects and testing them. Thus, in

> "Whenever something is put in the oven, the indicator comes on. John put a loaf of bread in the oven and it turned on,"

it is foolish to try "oven," then "loaf of bread," and finally "indicator" as the referent of "it." The first sentence should give enough information so that the first clause of the second could cause a prediction that the indicator would come on. The other alternatives should not even be looked at. There is no reason why an ability of this sort could not be incorporated into TOPLE; it would require having R-IT return "!>" (which matches

any object). It would also require placing of monitors which would not allow any if-neededs called by PLAUSIBLE? to make up an object if there were no useful predictions; in that case, recently referred-to objects would just have to be tried in order. (In other words, in this case, difficulty IGNORANCE would be insuperable.)

No programs I have seen can yet cope with the fact that many references are not references to individual objects, or, in many cases, to anything at all. For example, "A monkey likes a banana now and then," does not refer to a particular banana or monkey. In, "Fred is a monkey," "a monkey" does not refer; it predicates of Fred. In, "The steam engine is here to stay," "the steam engine" may refer to a particular machine, but the more conventional reading is "steam engines will continue to be used," or some such. Gilbert Ryle (1949) has pointed out that "the British Constitution" refers in only the most devious way; it is really a sort of proper noun with institutional uses. "The American Constitution" would seem to be easier to handle, but, here too, if the piece of paper referred to by the phrase were to disappear, it would still have a use.

Other problems with reference cannot be solved until more progress is made in representing partial ignorance about a situation. TOPLE assumes (A var...) refers to a brand-new object, which is absurd. (For example, "a wall," "a block in the

box," etc.) Sometimes this assumption causes no trouble, but in others (counting walls, for instance), possible equalities may have to be taken into account. This problem would get worse if TOPLE had to understand accounts of dialogues, since different characters may have different knowledge. (E.g., Fred: "I lost my dwarf"; a little later, Joe: "I saw a dwarf in the forest yesterday.") Hopefully linguistic and common-sense clues can be used to avoid as full a treatment of equality in solving this problem as might be required for proving mathematical theorems.

Syncategorematic Adjectives

Predicate calculus notation biases you towards treating nouns and adjective as predicates (or terms of predicates, as (MALE X) or (SEX X MALE)). In fact, it is not always possible to separate a noun phrase into separate formulae, one for the common noun header, and one for the modifying adjectives. For example, a former judge is not a judge; and whom would he be former than?" A little elephant is not little; it is not clear that words like "big" and "little" even have a meaning when not applied to a particular noun.

Such adjectives are called "syncategorematic" (Quine, 1960). A method to understand such words must see the whole phrase they

occur in. Thus, besides the method P-IS which understands assertions like (IS ob kind) for atomic kinds, there must be a method for each kind modified by a syncategorematic adjective. Currently, there are only three such adjectives TOPLE understands: BIG, LITTLE, and OTHER. The two size adjectives (others could easily be added) are understood by a method with a pattern which matches (IS object (BIG kind)) or (IS object (LITTLE kind)). This method understands this to mean (IS object kind) and (ORDER SIZE (TYPICAL kind) object >☆) (or (...<☆)).

OTHER is understood by a method with pattern (IS object (OTHER kind)); it sets up a generator for (IS object kind), but rejects the generated objects which have been mentioned previously in this paragraph. This could be honed down a good deal, but the principle is clear.

Other syncategorematic adjectives can be handled by similar ad hoc methods. For example, "former" could be understood by a method which altered the tense to past of an expression it appears in; this is clearly beyond the abilities of TOPLE, since it cannot handle tense at all.

A "wholist" might make a case that all adjectives are syncategorematic; after all, an adjective may be applied to almost any noun with some metaphorical meaning, and that meaning must depend on the noun. (A green idea might be one tinged with envy, whereas a green government would be an inexperienced one.)

However, somewhere this interdependence of word meanings on each other must end, or become negligible, or there would be no place for a machine or a human to start in understanding a sentence or discourse. In some sense, "syncategorematicity" is a useful, systematic dimension of ambiguity, which is worth attending to only when it can profitably be distinguished from other sorts. For example, literally "green" means "colored green" or "unripe." Although which meaning is intended depends to some extent on what noun it applies to (a "green fruit" is more likely to be unripe than a "green vegetable"), the noun does not determine its meaning as exclusively as for syncategorematic adjectives. Thus, in the case of such an adjective, one might as well generate the possible meanings and pick the one that is easiest to believe, rather than decide the meaning on the basis of its noun.

Some adjectives are ambiguous between a syncategorematic and categorematic meaning. "Poor" is an example: a poor violinist may be untalented or impoverished. Someone might argue that "poor" is just as syncategorematic as "untalented," because a poor corporation is likely to be richer than a rich man, but it seems to me that the linkage between adjective and noun in these cases is rather loose. For day-to-day purposes poor men, poor countries, and poor corporations are on similar scales of power and influence, and therefore "so-and-so is poor" and "so-and-so is a violinist" ought to be stored independently in a data base.

VII. Assessment

> ...You can build a machine to draw demonstrative conclusions for you, but I think you can never build a machine that will draw plausible inferences.
> --Georg Polya, Induction and Analogy in Mathematics

In fact, there is no reason why machines cannot do plausible reasoning (although creative reasoning, plausible or rigorous, is as difficult for machines as for anyone else). But the possibility of error in reasoning means that a machine must record its reasons for believing what it does, and must be prepared to find alternatives to uncomfortable beliefs.

This is exactly TOPLE's ability, over a limited, but expandable domain. There is tremendous room for expansion of its abilities, in almost every direction, as I shall discuss, but TOPLE demonstrates the following points already:

1. Semantic comprehension, the aspect of linguistic understanding most difficult to systematize, is within reach of computer world models. It has always been clear that such comprehension involves more than conversion of natural language to some superficially more rigorous notation; the assertions so generated must be related in intelligent ways to what is already known. To some extent, TOPLE can do this. It knows that "going to the bananas" translates into a different internal representation in different situations; it knows that some

interpretations of sentences about monkeys are implausible because of what it knows about their abilities and wants; etc. There are many things it does not realize, and consequently does not fully comprehend sentences about, but sentences on topics it knows about, it "understands."

When I started this work, it seemed clear to me that a deductive model of the world (such as that of Green (1969b), Black (1964), or Winograd (1971)) should be able to serve as a model of belief, if it could be run in a kind of "monitored mode" that would cause a supervisor to step in and force a conclusion when a deductive program (a PLANNER theorem) failed for a non-fatal reason.

The resulting program is not quite like that. The belief system has been woven into the deductive system in a much more intimate way. Each method is allowed to bring all the expertise it can to bear on problems they encounter. But the monitor lives on, as a class of control programs that allow crosstalk between problem programs that wish to compare views of the world.

This system provides a clever module for a linguistic understander to communicate with. Assuming it is operating in the context of actually understanding coherent discourse, it is the final arbiter of what meaning is intended by an utterance. All syntactic and pragmatic "clues" to meaning are subordinate to the question of what makes sense uttered by a given speaker to a

given hearer in a given situation.

TOPLE is deficient in two major respects: at the linguistic end, it is almost blind to syntactic clues that could shorten its search for an interpretation; at the intellectual end, it is quite stupid about things it pretends to understand, such as space, and about things it knows nothing about, such as the knowledge of other creatures' motives and minds, especially the methods they use to communicate.

But with what it knows, TOPLE is able to throw away almost all linguistic information and still come up with plausible interpretations of what it is told.

2. As Charniak (1972) has pointed out, it is completely impracticable, in a system for understanding stories, to postpone all deduction until someone asks a question. There are too many loose possibilities and multiply-branching cases for deductions to be feasible, unless plausible assumptions are forced as the story progresses. In many cases, some true conditions and events in the story are never explicitly mentioned, but should be added once and for all when certain linguistic or logical circumstances arise.

An important case of forced assumptions is TOPLE's beliefs about the future. These <u>predictions</u> are important for several reasons: they may be the only way that certain events (e.g., the fall of an object) will get recorded; they provide a way of

cheapening deductions in the future by packaging them ahead of time; they allow certain conditions, notably the motivations of another creature, to impose a coherent structure on events before they occur.

The method TOPLE uses for performing this last task is worth repeating: for systems sufficiently like itself to simulate, it requires no special interpreter to apply its knowledge of them; it lets the CONNIVER evaluator do it. This allows for a unity of notation in describing its own and other creatures' views of the world. On the other hand, it embodies an admission that, for a system like TOPLE, understanding other creatures really well is as tricky as understanding itself.

3. The control structure of a system doing plausible reasoning must be more complicated than that of a deductive system, for which simple backtracking might work. In rigorous deduction, contradictions are final, and can cause failures; in plausible reasoning, inconsistencies are merely difficulties to be overcome. Choosing which difficulty is to be worked on is too difficult to decide using only local features of the problem, which are all that can be examined when there is only one process left un-FAILed at a particular time. Instead, there must be communication between a program that started an investigation and the subgoal-achievers it creates, and between these subgoal-achievers themselves.

However, even a rigorous deductive program would seem to require the ability to stop and see where it is and what it is doing. An expert deduction program should treat the deductive rules as a problem, not a solution. It must be able to draw conclusions from them, and notice contradictions, but that is only a small part of what it must do. It must know which sub-theorems are unlikely to be provable (cf. Gelernter, 1959); it must know what kinds of proofs are useful in what kinds of situations; how to plan by breaking a proof into sub-proofs and then filling in the details. This information about formal rules must be flexibly integrated with the information in them; they cannot be useful (no matter how cleverly the order in which they are applied is chosen), if they are applied more or less blindly until the answer pops up, as so many recent theorem provers do. (Robinson, 1965; Green, 1969a)

The CONNIVER programming language (McDermott and Sussman, 1972) has been a valuable tool for expressing knowledge of both deduction and problem-solving. Its tree-structured control and context data are perfect for building "informed goal trees." (Sussman, 1972; Fahlman, 1973)

Directions for Further Research

One goal of this master's thesis research was to gain the high ground for an assault on the problem of telling computers things in English. But, like everyone else, I find myself mired in a morass I only saw dimly before. TOPLE is too narrow in every respect. Unfortunately, broadening it in only one of those respects would be of little use, and broadening it in all of them is an enormous task. Unfortunately, also, it seems that the essence of intelligence is to be broad.

Nonetheless, the fact that the problem can be factored is encouraging, as is the fact people are working on each factor.

First, TOPLE could use an enormous amount of new knowledge about its world. It is still naive about space and partial orderings, as can be seen from Chapters IV and VI. It contains only a crude simulation of Spiro, so crude that it makes terrible, stupid, impossible predictions. (Literally, TOPLE could not simulate its way out of a paper bag.) TOPLE misses so much in the real world that it is, to put it mildly, not a very subtle listener. It could know more about distance, size, shape, what it means to be in something, what it means to hold something, what monkeys really want, what they feel, etc., etc.

Specifically, TOPLE could know an awful lot more about language than it does now. Its only purely linguistic knowledge

is in some of the details of pronoun understanding. Phenomena like sentence topic and presupposition are not dealt with at all. Some I am quite confident about; presupposition, for example, would seem to involve running TOPLE in DEDUCE mode, as is done in THE-phrases.

Lexical ambiguity is an area I anticipated more work on than I have actually done. A lot of the structure of TOPLE is derived from the paradigm of testing the alternative meanings of words in parallel. What I have found, however, is that even when there is no ambiguity, the same type of analysis is required. A simple statement of location, with no apparent ambiguities, may mean different things in different situations, and the possible meanings must be explored and compared. I believe that analysis of lexical ambiguities beyond the little I have implemented would be relatively simple to add.

Other logical nuances of language are more perplexing. What does "but" mean, for example? In some cases, it means little more than "and," which it is logically equivalent to, as in, "Father gave a balloon to Billy, but not to Susy." Here, whether the conjunction is "and" or "but," the same problems of understanding remain; why did Father favor his male child in this case? (Previous or subsequent events may give an explanation, or it may be cultural.)

In other sentences, "but" expresses a good deal more. From

the following example,

> "The monkey was hungry. He ate his bananas. There was another banana in the box, but he couldn't see it,"

most people would conclude that the monkey was still hungry. Hence, "but" means "the second clause of this sentence invalidates a conclusion (here, a prediction) that would ordinarly follow from the first," namely, that the monkey will eat the hidden banana. The same sort of meaning appears in, "He wanted to give Susy a balloon, but he couldn't find her." However, from the sentence about the monkey, you cannot deduce the prediction to be invalidated without the extra assumption that the monkey was still hungry; thus, "but" signals that a deduction is to be made that is invalidated by the following clause, even if new beliefs must be assumed to do it! At this point, I have no idea how the set of possible falsehood-generating beliefs is to be chosen.

TOPLE does not now store in any fashion what is being talked about; people can usually summarize in a single word or phrase what they are talking about. How such information is recorded, how it is changed, how it is deduced, and how it is to be used are more or less mysteries to me. If TOPLE knew "where" it was talking about, for example, it could make better guesses about the locations of new objects that are mentioned. Currently, it always assumes they are on the floor.

TOPLE is innocent of syntax. This I would remedy by hooking it up to a SHRDLU-like parser (Winograd, 1971) and using "semantic structures" instead of formulae in my semantic language. The parser could use purely syntactic clues (or simple semantic ones, like "selection restrictions"), to suggest best guesses at interpretations, and try them out using PLAUSIBLE?

Opening up the syntax in this way demands a much more flexible internal representation than is currently allowed. Otherwise, the "English" the program understood would be little more than disguised predicate calculus. Consequently, the parser must be allowed to dump a lot of free-form modifiers on a formula without disturbing the methods which understand it. Most adverbial phrases which modify clauses, for example, will not fit into a simple slot structure for formulae with the verb of that clause. The predicate GO presumably takes a destination argument normally, but does it take a standard "instrument" also, as, "in a car," or "on a boat"? How about, "with a girl"? How about "as fast as you can"? These must be attached to the representation of GO in some loose way and they must result in useful data base changes when they are understood. Many authors (Rumelhart, et. al., 1972; Schank and Tesler, 1969; Quillian, 1969) have given thought to how information could be crammed into some sort of network representation. Such representations seem easy to make up; undoubtedly a version can be properly parenthesized and read

into a CONNIVER. But using it seems more difficult.

For example, how are modifiers to be attached to the correct piece of a sentence? The classic example of this is, "I saw the man on the hill with a telescope." There is no way to translate this into the current TOPLE's SL without obliterating ambiguities. One way around this limitation is to have a parser hand the pieces of sentences like this to the believer one at a time; the believer would try to disambiguate "the man," by considering recently referred-to men, and if this failed, it would ask for more qualifying information, and receive "on the hill." The problem with this is that sometimes the correct way to qualify a reference further is to postpone picking a referent until a wider context is generated; i.e., give the understander "I saw" when it asks for more of the sentence. An alternative is to pass a less organized formula to the if-neededs that understand things; the example might produce something like (MODIFIED (SAW (I) (THE X (IS X MAN))) (ON (THE Y (IS Y HILL))) (WITH (A Z (IS Z TELESCOPE)))). Then the if-neededs for the various tasks of believing and resolving references could look for appropriate modifiers; for example, P-SEE would look for (WITH optical-instrument) and add whatever internal data it implied to the data base. Clearly, R-THE would get first crack at any modifiers to aid in disambiguating what its pattern refers to; also, somebody has to realize that SAW can be a verb of

physical action that could take place ON a hill.

Another area I have not touched is tense and modality. The use of past tense opens the problem of references to past situations, which are done in a natural language by the use of conjunctions like BEFORE and WHILE that relate events and conditions. Once TOPLE is made responsible for adding statements to the past, it must make sure that changes it makes do not contradict its beliefs about situations since those. This problem was dealt with in Chapter V. It can probably be solved by the intelligent use of demons of one sort or another.

Future tense seems easier. The word "will" is essentially a device for allowing a speaker to make predictions in the data base of the hearer; the latter is responsible for explaining any forecasted events that need explaining, but otherwise can be as free as in believing his own predictions. But how are the nuances in the word "may" to be captured? Possibly there are grammatical phenomena, such as word pairs of the form "may...if" or "may...unless" that can be exploited; perhaps the remaining forms are just "will's" with less certainty; I don't know. At least "may do" is better than "may have done," which involves a hypothetical addition to the past. Other modals, like "must" and "can" seem to be resolvable into concepts of obligation, ability, knowledge, social institutions, etc., as well as predictions.

Before tense can really be understood by TOPLE, the way it

structures time must be re-thought. Currently it admits of only one short episode, with one past and several hypothetical futures. Clearly, it cannot expect to string out its set of situations indefinitely. Not all mentioned situations are to be definitely orderable. Not all episodes occur on the same time scale; the same fact is a short event in one scenario and a long condition in another. Time must be broken into short sequences, whose major results become the only visible elements on a higher level where sequences are made up of more complex events. For example, "He committed a murder," is an event with a single consequence at a high level--the victim is dead. Lower down, the details of how he did it are visible and important; the important outcome occurs somewhere in the middle of the story. It, too, is broken into finer episodes, such as, "He drove across town," which have as much fine structure as is needed. Hopefully, each layer can be kept simple and short by throwing away possibilities when going up the hierarchy.

Many questions remain to be answered. Is this structure a tree, or is there more than one way to group events? How much information can be carried from one episode to the next? For example, in a story you hear how Fred moved the piano across the room; at the end, Fred and the piano are in a new place. How do you formalize the fact that Fred's new location is not an "important result" of his moving heavy furniture, but that the

furniture's location is? How is the fact to be usable that all events can be totally ordered, and that two events may share the same time? When, in particular, can two event sequences be merged or re-structured?

Before TOPLE can really understand linguistic knowledge, it must have a model of language, meaning language use. It must understand its interlocutor's motives, and how what he says serves them. It should be able to use language for its own ends. In other words, it should know how to converse. Nowhere is it written in TOPLE now that, "If you don't know something you wish to know, and there is a friend close by who probably does, ask him. His reply will probably be an answer or an explanation of why he can't answer." Thus TOPLE does not converse intelligently, nor can it understand accounts of conversation.

Language as an institution will be hard to understand until TOPLE is better at modelling the minds of other creatures. This is done on an extremely ad hoc basis currently. There is no way to record that a creature doesn't know some known fact, or that he does know an unknown one. The latter is especially difficult. To know that there is an object with property P is to know there is an A such that (P A). (This is just the Skolem-function form of (EXISTS X (P X)).) But then you appear to know what object that is, namely, A.

For example, on January 13, 1973, the Boston Globe ran a

headline that may be paraphrased, "Super Bowl winners play second-best team tomorrow." I defy any existing language-understanding program to interpret that as, "Super Bowl to decide best team tomorrow," which it actually meant. In this case, the "other creatures'" minds we are modelling are our own, in the future, on January 14. In some sense, the phrase (WINNERS SUPERBOWL) refers in the future, but doesn't refer properly now; it behaves as a constant in the future, as a variable now. If Spiro is in the room with a table, its location is a constant to him, but a variable to us. On the other hand, if Spiro has a box that rattles, he only knows some unknown thing is inside, but TOPLE may have been told what; here the roles are reversed.

Notice that it is not possible to write my "simple fact" about how to get the answer to a question until TOPLE can express the difference between what it knows and what its friend knows. Nor is it possible to express the concepts "look for" or "can" (counting "knows how" as a subcase of the latter).

Finally, when communication in natural language is allowed, there is no easy way to avoid the problem of being told "quantificational" statements, statements that would most likely have quantifiers if expressed in predicate calculus. The worst case here is being told facts which should alter programs already carefully built into the system (like, "monkeys can climb ropes," which alters the AT-achiever and other modules).

Not all "general" statements are so bad. (ARE TABLE PHYSOB) has no variables in it. (ORDER SIZE (TYPICAL BALL) (TYPICAL TABLE) >) expresses a plausible deduction in a single assertion, when interpreted by a clever method. (I call these statements "general," or "quantificational," because they are most intuitively written with quantifiers in a first-order theory, and because doubting one of them might involve qualifying it (see below). The first might be expressed as (FORALL X (IMPLIES (IS X TABLE) (IS X PHYSOB))) Of course, (ARE TABLE PHYSOB) would be a legal statement in such a theory as is; it would <u>not</u> be legal if "is a table" were expressed as (TABLE ...) rather than (IS ...TABLE). The second formula could be written, using the modal operator "NORMALLY," (McCarthy and Hayes, 1969), as:

```
(FORALL X
   (FORALL Y
      (NORMALLY (IMPLIES (AND (IS X BALL) (IS Y TABLE))
                         (ORDER SIZE X Y >)))))).)
```

Since both of these items are variable-free, one could imagine methods for deducing and believing them. A TYPICAL-size item, for example, could be checked by comparing instances of the classes at issue and averaging, or something similar.

Most of TOPLE's general information is buried in places where it is not accessible to the believing or doubting machinery in any way. For example, the information about numerical modifiers (">>," "<*," etc.), and how to compose them ("x>>y and y>=z,

therefore, x>>z") or conclude new assertions from inequalities with the same upper bound ("x>>y and x>=z, therefore z>y"), is contained in tables stored as variables, not as item data. TOPLE knows exactly when it should use this information, looks it up, puts it to use, and forgets it. It does not have to go through a tedious demon-call every time it asserts a fact that requires a deduction; why should it, when the program P-ORDER that does the assertion is an expert on numerical orderings? In particular, this program, which knows all the alternative ways of cramming seemingly contradictory orderings into the data base, cannot and does not consider doubting "> is transitive" as one of those alternatives (nor of doubting itself, another piece of quantificational information).

I consider this way of writing if-needed methods a strong feature of TOPLE; it does not waste time with searches or deduction when calculation is what is required; it does not consider everything doubtable. I quite consciously wrote it so the knowledge programs are experts on the world, and TOPLE is an expert on how to call those programs. However, if it is true that deduction and routine-doubting are too expensive to try in all but exceptional cases, then a solution to the problem of being able to be told new methods and general beliefs is all the more urgent. When a new belief is received, it will <u>not</u> fit smoothly into the system at first (see below); it will be called

too often as a separate routine, instead of being integrated into the proper places in already-existing routines; it will be full of bugs, or be too general or too specific, and will have to be mistrusted, perhaps all the time until it has proved itself.

A false assertion about a single event or condition is a bug; a false generalization has bugs, but is usually worth keeping in some modified form. The reaction to a disconfirmation of even a simple generalized belief is different from that to a "singular" assertion, as we may call assertions about a single fact. If someone has been describing a particular farm, and says, "Melons generally run larger than sows," he probably means, "on this farm" as an obvious qualification. Whether this is the intended meaning depends heavily on linguistic cues in context; but in any case an investigation should make the proper qualification clear.

In the case of an assertion about typical sizes, we can in principle test its truth by taking averages, and looking for qualifications that make the average come out right. When we turn to generalized knowledge expressed as a program or (worse yet) a piece of program, with no clear form or destination, the hope vanishes that even in principle such a simple system as averaging will be of use. Instead, the program-assimilating program will just have to understand what it is asked to believe, and understand its interactions with what it already knows.

This is an incredibly difficult problem: how to inform a

problem-solver of a new method and have it be able to use it intelligently. Always it does either too much with it or too little. Systems like QA3 (Green, 1969b), QA4 (Derksen et. al., 1972), PLANNER (Hewitt, 1972), and CONNIVER (Sussman and McDermott, 1972) pretend they have solved everything by associating a pattern with every piece of knowledge and mentioning it by pattern instead of by name. Then a new theorem is absorbed smoothly by being picked up by exactly those programs that have need of it, as shown by their use of matching patterns.

Of course, it doesn't work that way. Theorem-provers typically bog down unless given axioms in tightly optimized bundles. (Cf. the solution to the "tower of Hanoi" problem in Green (1969a).) PLANNER-type systems suffer from the opposite defect: nobody calls your shiny new theorem or method until you find all the places that should be calling it and patch them. If there do happen to be callers in existence already, your new theorem is probably competing with whomever they used to call, and the two should be combined. Calls to the new program probably interact in funny ways with old problem-solving equipment. You end up putting machinery into your new and old programs so they can communicate better, or you redesign the whole system so it breaks the problem up more naturally, or you suffer the inefficiency of mis-matched subparts. Work on mechanising debugging processes of the complexity required is

only beginning. (I am indebted to Gerry Sussman (personal communications) for many of these insights.)

The problem is that there is a lot of knowledge about how programs should work which is not contained in modules that apply it when those programs are run, but must be put to use when those programs are written. There is no guarantee even at that time that it will fit in one place or another, because it might be expressed in the very structure of the routine being constructed. To a great extent, the sophistication with which pieces of TOPLE-like systems talk to each other is wasted when the topic is trivial; instead, they should rewrite themselves in ways that make such sophistication unnecessary.

Now add to these problems those of ambiguity and gross under-specification, and you have the problem of telling computers interesting things in English.

There is only one solution I can see to the problem of writing expert routines: the design of expert routine writers. It took me a while to translate my piecemeal conscious intuitions about space into programs; no doubt the super-TOPLE of the future will have to be about as good a programmer as I before it can be told about space. Or perhaps space is something you cannot be told about; an expert space thinker or learner must be wired in. Right now no one knows what things can or cannot be communicated. No one knows what <u>is</u> communicated when one person tells another

about a program he wants. How does he partially specify a formal process? How does he specify which things he doesn't care about? How does a programmer do heuristic reasoning about programs--something quite different from proving programs correct? If we do not solve these problems, telling computers things will always be as painful as it is today.

Bibliography

Abelson, R.P. (1973) The Structure of Belief Systems, to appear in Computer Simulation of Thought and Languag (Colby, K. and Schank R.C., eds.), W.H. Freeman & Co..

Abelson, R.P. and Carroll, J.D. (1965), Computer simulation of individual belief systems, Am. Behav. Sci. 8, p. 24.

Abelson, R.P. and Reich, C.M. (1969), Implicational molecules: a method for extracting meaning from input sentences, Proc. Inter. Joint Conf. Art. Intel. 1, (Walker, D.E. and Norton, L.M., eds.), p. 319.

Black, F. (1964) A Deductive Question-Answering System, Ph.D. dissertation, Harvard. (Also in Minsky, 1968, p. 354.)

Bledsoe, W.W., Boyer, R.S. , and Henneman, W. H. (1971) Computer Proofs of Limit Theorems, Proc. Inter. Joint Conf. Art. Intel. 2, p. 586.

Charniak, E. (1972) Toward a Model of Children's Story Comprehension, M.I.T. A.I. Laboratory Technical Report 266.

Colby, K.M. and Smith, D.C. (1969), Dialogues between humans and an artificial belief system, Proc. Inter. Joint Conf. Art. Intel. 1, (Walker, D.E. and Norton, L.M., eds.), p. 319.

Coles, L.S. (1967) Syntax Directed Interpretation of Natural Language, Pittsburgh: Carnegie Institute of Technology, Ph.D. Thesis.

Derksen, J.A., Rulifson, J.F. and Waldinger, R.J. (1972) The QA4 Language Applied to Robot Planning, Proc. FJCC 41, p. 1181.

Dreyfus, H.L. (1972) What Computers Can't Do, New York: Harper & Row.

Fahlman, S. (1973) A Planning System for Robot Construction Tasks, M.I.T. unpublished S.M. thesis.

Fikes, R.E. and Nilsson, N.J. (1971) STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Proc. Inter. Joint Conf. Art. Intel. 2, p. 608.

Gelernter, H. (1959) Realization of a geometry-theorem proving machine, Proc. Inter. Conf. on Information Processing, Paris: UNESCO House, p. 273. (Also in E.A. Feigenbaum and J. Feldman, Computers and Thought, New York: McGraw-Hill Book Company, p. 134.)

Green, C.C. (1969a) Application of Theorem Proving to Problem Solving, Proc. Inter. Joint Conf. Art. Intel. 1, (Walker, D.E. and Norton, L.M., eds.), p. 219.

Green, C.C. (1969b) Theorem-proving by resolution as a basis for question-answering systems, Machine Intelligence 4 (Meltzer, B. and Michie, D. eds.) New York: American Elsevier Publishing Company, Inc., p. 183.

Hayes, P.J. (1971) Robot Logic Applied to Problem Solving, Machine Intelligence 6.

Hewitt, C. (1972), Description and Theoretical Analysis (Using Schemata) of Planner: A Language for Proving Theorems and Manipulating Models in a Robot, M.I.T. A.I. Laboratory, Technical Report 258.

Koffka, K. (1963) Principles of Gestalt Psychology, New York: Harcourt, Brace & World.

McCarthy, J. and Hayes, P.J. (1969) Some philosophical problems from the standpoint of artificial intelligence, Machine Intelligence 4 (Meltzer, B. and Michie, D. eds.) New York: American Elsevier Publishing Company, Inc., p. 463.

McDermott, D.V. and Sussman, G.J. (1972) The CONNIVER Reference Manual, M.I.T. A.I. Laboratory Memo. No. 259.

Meyer, G.S. (1972) Infants in Children Stories - Toward a Model for Natural Language Comprehension, M.I.T. A.I. Laboratory Memo. No. 265.

Minsky, M. (1961) Descriptive Languages and Problem Solving, Proc. Western Joint Computer Conf., p. 215. (Also in Minsky, 1968, p. 419.)

Minsky, M. (ed.) (1968) Semantic Information Processing, Cambridge: MIT Press.

Nevins, A.J. (1972) A Human Oriented Logic for Automatic Theorem Proving, M.I.T. A.I. Laboratory Memo. No. 268.

Newell, A. (1962) Some problems of basic organization in problem-solving programs, in Self-Organizing Systems--1962 (M. Yovitts, G.T. Jacobi, and G.D. Goldstein, eds.), New York: Spartan.

Polya, G. (1954), Patterns of Plausible Inference, Princeton: Princeton University Press.

Quillian, M.R. (1969) The Teachable Language Comprehender: A Simulation Program and Theory of Language, Comm. ACM 12, p. 459.

Quine. W.V.O. (1960) Word and Object, Cambridge: MIT Press.

Quine, W.V.O. (1963) Two Dogmas of Empiricism, From a Logical Point of View, New York: Harper & Row, p. 20.

Raphael, B. (1964), SIR: A Computer Program for Semantic Information Retrieval (Ph.D. dissertation, M.I.T.)

Robinson, J.A. (1965) A machine-oriented logic based on the resolution principle, J. ACM 12, p. 23.

Robinson, J.A. and Wos, L. (1969) Paramodulation and theorem-proving in first-order theories with equality, Machine Intelligence 4 (Meltzer, B. and Michie, D. eds.) New York: American Elsevier Publishing Company, Inc., p. 135.

Rumelhart, D.E., Lindsay, P.H., and Norman, D.A. (1972) A Process Model for Long-Term Memory, Organization and Memory (E. Tulving and W. Donaldson, eds.), New York: Academic Press.

Ryle, G. (1949) The Concept of Mind, New York: Barnes and Noble.

Sandewall, E. (1970) Formal Methods in the design of Question-Answering Systems, Uppsala University Dept. of Computer Sciences, Report NR 28.

Schank, R.C. and Tesler, L.G. (1969) A Conceptual Parser for Natural Language, Proc. Inter. Joint Conf. Art. Intel. 1, (Walker, D.E. and Norton, L.M., eds.), p.569

Schank, R.C., Tesler, L.G. and Weber, S. (1970) Spinoza II: Conceptual Case-Based Natural Language Analysis, Stanford A.I. Project Memo. AIM-109.

Sussman, G.J. (1972) Teaching of Procedures--Progress Report, M.I.T. A.I. Laboratory Memo. No. 270.

Sussman, G.J. and McDermott, D.V. (1972) From PLANNER to CONNIVER--A Genetic Approach, Proc. FJCC 41, p. 1171.

Sussman, G.J. and Winograd, T. (1970), Micro-PLANNER Reference Manual, M.I.T. A.I. Laboratory Memo No. 203.

Waltz, D.W. (1972) Generating Semantic Descriptions from Drawings of Scenes with Shadows, M.I.T. A.I. Laboratory Technical Report 271.

Winograd, T. (1971) Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, M.I.T. Project MAC Technical Report 84.

Winston, P.H. (1970) Learning Structural Descriptions from Examples, M.I.T. Project MAC Technical Report 76.

CS-TR Scanning Project
Document Control Form

Date : 4/30/96

Report # AI-TR-291

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 150 (156-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other: Copy

Check each if included with document:
☒ DOD Form ☐ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number):_____

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-150) UNNUMBERED TITLE PAGE, UNNUMBERED ABSTRACT, UNNUMBERED TABLE OF CONTENTS,	5-150
(151-156) SCANCONTROL, COVER, DOD, TRGT'S (3)	

Scanning Agent Signoff:
Date Received: 4/30/96 Date Scanned: 5/7/96 Date Returned: 5/9/96

Scanning Agent Signature: Michael N. Cook

REPORT DOCUMENTATION PAGE

Form Approved
OBM No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1974	

4. TITLE AND SUBTITLE

Assimilation of New Information by a Natural Language Understanding System

5. FUNDING NUMBERS

6. AUTHOR(S)

Drew V. McDermott

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Massachusetts Institute of Technology
Artificial Intelligence Laboratory
545 Technology Square
Cambridge, Massachusetts 02139

8. PERFORMING ORGANIZATION REPORT NUMBER

AITR 291

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Information Systems
Arlington, Virginia 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AD-780194

11. SUPPLEMENTARY NOTES

None

12a. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

14. SUBJECT TERMS

15. NUMBER OF PAGES
160

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**