Technical Report 266

Toward A Model Of Children's Story Comprehension

Eugene Charniak

MIT Artificial Intelligence Laboratory

BIBLIOGRAPHIC DATA SHEET	1. Report No. AI-TR-266	2.	3. Recipient's Accession No.

4. Title and Subtitle

Toward a Model of Children's Story Comprehension

5. Report Date: December 1972

6.

7. Author(s): Charniak, Eugene

8. Performing Organization Rept. No. AI TR-266

9. Performing Organization Name and Address
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square,
Cambridge, Mass. 02139

10. Project/Task/Work Unit No.

11. Contract/Grant No. N00014-70-A-0362-0003

12. Sponsoring Organization Name and Address
Office of Naval Research
Department of the Navy
Information Systems Program
Arlington, Va. 22217

13. Type of Report & Period Covered: Technical Report

14.

15. Supplementary Notes

16. Abstracts

How does a person answer questions about children's stories? For example, consider "Janet wanted Janck's paints. She looked at the picture he was painting and said, 'Those paints make your picture look funny'". The question to ask is "Why did Janet say that?".

We propose a model which answers such questions by relating the story to background real world knowledge. The model tries to generate and answer inportant questions about the story as it goes along. Within this model we examine two questions about the story as it goes along. Within this model we examine two problems, how to organize this real world knowledge, and how it enters into more traditional linguistic questions such as deciding noun phrase reference.

17. Key Words and Document Analysis. 17a. Descriptors

Computational Linguistics
Computer Problem Solving
Disambiguation
Language Comprehension
Machine Understanding
Pragmatics
Pronoun Reference
Question Answering
Semantics

17b. Identifiers/Open-Ended Terms

Artificial Intelligence
Computer Aided Cognition
Natural Language Understanding

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Unlimited Distribution Write A.I. Publications	UNCLASSIFIED	304
	20. Security Class (This Page) UNCLASSIFIED	22. Price

PROOF COPY

TOWARD A MODEL OF CHILDREN'S STORY COMPREHENSION

Eugene Charniak

December 1972

Artificial Intelligence Laboratory

Massachusetts Institute of Technology

Cambridge Massachusetts 02139

Second Printing January, 1974

TOWARD A MODEL OF CHILDREN'S STORY COMPREHENSION*

ABSTRACT

How does a person answer questions about children's stories? For example, consider "Janet wanted Jack's paints. She looked at the picture he was painting and said, 'Those paints make your picture look funny'". The question to ask is "Why did Janet say that?".

We propose a model which answers such questions by relating the story to background real world knowledge. The model tries to generate and answer important questions about the story as it goes along. Part of the information connected with a "concept" is the set of facts which might be relevant to stories which include the concept. When the concept occurs in the story these facts are "made available" in the sense that they can then be used to make deductions. In general all the necessary information to make a deduction may not be around at the time the fact is made available. Hence the facts are allowed to wait around "looking" for the necessary information. For this reason these facts are called "demons". This model also sheds light on some problems of reference and disambiguation (such as interpreting "funny" as "bad" in the above example). The demons (serving as "context") can assign a particular meaning to a word, or a particular referent to a noun phrase.

A major problem is formalizing our real world knowledge to fit into the comprehension model and we explore in detail one small topic (piggy banks). Note that it is the researcher, not the model, who discovers and organizes the basic facts. That is, the model does not learn.

An earlier version of the model described in the thesis was computer implemented and handled two story fragments (about 100 sentences). The problems involved in going from natural language to internal representation were not considered, so the program does not accept English, but an input language similar to the internal representation is used. Naturally this is only a first attempt at a model for children's stories and many suggestions for further work are included.

---

*This is a revised version of a dissertation submitted to the Department of Electrical Engineering on August 25, 1972 in partial fulfillment of the requirement for the degree of Doctor of Philosophy.

ACKNOWLEDGMENTS

My thanks to Professor Marvin Minsky (thesis supervisor), Professor Joel Moses and Seymour Papert (thesis committee), Jeff Hill, Gerry Sussman, and Terry Winograd (listened and commented from the beginning) and Carl Hewitt, Mitch Marcus, and Andee Ruben (read and commented on later versions of the thesis).

The work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0003.

The views and conclusions contained in this document are those of the author's and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

TABLE OF CONTENTS

1 Introduction

1.1 What This Is All About

Let us consider the problem of constructing an abstract model of story comprehension. To determine what the model, or program, has "understood" about what it has read, we will ask it questions. (Note 1) (All "notes" are at the end of the thesis.) So a typical story might start:

> Fred was going to the store. Today was Jack's birthday and Fred was going to get a present.

Some typical questions would be:

> Why is Fred going to the store?
> Who is the present for?
> Why is Fred buying a present?

There are two points which we should note about such questions. First, they are not answered explicitly in the text. That is, the story did not say "Fred was going to the store <u>because</u> he ..." The story does not even contain a full implicit answer; one cannot logically deduce an answer from the statements in the story without using general knowledge about the world such as:

> Objects "got" at stores are usually "bought".
> Presents are often bought at stores.
> If a person is having a birthday, he is likely to get presents.

Our use of questions that go beyond the story as a test of understanding the story raises a methodological problem. <u>How</u> far beyond the story may the questions go? It would obviously

be absurd to allow any question, however remotely connected. However, we have no way to draw a line formally and so will have to rely on an intuitive sense of what constitutes an "ordinary question" "directly related" to the story.

The average reader of our "story" has no difficulty in answering a question like "Who is the present for?". In fact, he often has to go back to the story before he believes that the story did not give the answer explicitly. It would seem that one applies "every day knowledge" to the story as one goes along. But how does one do it? How do we incorporate common sense knowledge into the process of understanding natural language? This thesis is an attempt to elucidate this question.

As the story is "read" our model will be making decisions on the basis of common sense knowledge. (I will also use the term "real world knowledge". In all cases I mean the kind of facts which are shared by most people in a given culture, like middle class USA.) An example of a particular kind of decision that is involved in "understanding" is reference decisions. That is, what previously mentioned object does a noun phrase refer to? Consider:

> Janet and Penny went to the store to get presents for Jack. Janet said "I will get Jack a top" "Don't get Jack a top" said Penny. "He has a top. He will make you take it back. "

The trouble here is the "it" in the last sentence. If we were naively to decide that it referred to the last inanimate

object mentioned, we would conclude that the referent was the top Jack currently has. But, this is not the way the story is understood. This example will be covered in detail later. Suffice it to say that it is our knowledge of "presents", and returning presents, which is coming into play.

To explicate the role of world knowledge in understanding, we will describe a model which answers questions about children's stories. Here are some of the issues, and an outline of the strategy I will adopt.

1) I have decided, as a first approximation, to divide the problem into two parts. The first half concerns taking the natural language and translating it into an "internal representation". This internal representation is a form which is convenient for making deductions. "Internal translation", as we shall call the first part, will be restricted to processes which could be performed by a person seeing the line completely out of context. The second part of our two part division will be called "Deep Semantic Processing (DSP).

2) Rather than consider the entire problem of natural language comprehension, only DSP will be considered in this thesis.

3) The model will try to "fill in the blanks" of the story on a line by line basis. That is, as it goes along, it will try to make connections between events in the story (usually causal connections) and fill in missing facts which seem important (such as who was going to receive the present).

4) The model will build up a data base which contains the facts given by the story along with any deductions made in accordance with (3) above). The model will try to keep this data base consistent and non-redundant.

5) A fact is only "made available" for making deductions when there is reason to believe that it is relevant to the story. For example, going back to our first story, the fact "people get presents from others on their birthdays" would be associated with the concept "birthday", and would hence be introduced to the data base by the line "Today was Jack's birthday". Typically the model will immediately try to use the newly introduced fact. However, in our "birthday" example it is not until the next line, "Fred was going to get a present" that we find anybody obtaining a present, and so it is then that our fact can be used and allow us to conclude that the present is for Jack. So in many cases facts introduced earlier in the story

are used to interpret lines which occur later in the story. Such facts are called "demons".

6) The model will attempt to handle many kinds of statements about intentions, desires, and responsibilities which frequently come up in children's stories. In particular, they will cause portions of our general knowledge to become relevant to the story. Hence this information can affect later lines, as in (5).

7) The problem of reference will be covered in some detail. I will try to show how the information which is used to "fill in the blanks" (see (3) above) can be directly used to help with reference.

Needless to say, what follows is not a complete answer to the problem of incorporating world knowledge into the understanding process.

1.2 An Example

To provide a better idea of what is involved in understanding children's stories, this section is devoted to a detailed example. This example was handled by a computer program written by the author. It was given the story and the questions, and it responded with the indicated answers. The computer version is not, however, identical to the model presented in the thesis.

What the Program Does

The program's input (and output) is not English, but "pre processed" English, in the sense that it has already been put into a form which closely resembles the internal format. This however leaves open the question of what information is explicit in the input, and what the program deduces for itself. We adopt the policy that the only information which can be included in the input is information which a normal reader can get solely out of an individual sentence without recourse to context. For example, since any noun phrase might be referring to an object which was introduced by an earlier line of the story, such reference decisions are not resolved in the input language. The actual input is given in Appendix A.

This story is taken directly from "Up and Away", a first grade reader published by Houghton Mifflin (Mc Kee et al. 66).

Several sentences have been deleted for brevity. Many of the examples throughout this thesis were inspired by stories in the first two books of this series, though this example is the only one I have quoted directly.

The Example

Jack and Janet were in the house. Jack was holding a box of pencils and a box of paints.

> Everything indented at this level is commentary for the reader. The first two lines of the story are not in the original version which has a picture instead.

"Janet, see the paints and pencils that Daddy got for us," said Jack.

Janet went to look at them.

"Are the paints for me?" she asked.

"No, the paints are mine," said Jack. "The pencils are for you, Janet."

Question: Did Daddy get the paints for Janet?

Answer: No, Daddy got the paints for Jack.

> It should be noted that the original sentence about getting the paints and pencils for Jack and Janet was ambiguous. It might have meant that they were for the children to share. If we had asked this question 3 lines ago the program would have responded "I don't know".

Janet said to herself, "I want the paints."

Jack began to paint a picture of a red airplane. Janet went to look at it.

"Those paints make your airplane look funny," she said.

"You could make a good picture of a red airplane with these pencils."

Question: Why did Janet say that the paints were bad?

Answer: She wants the paints.

> This question involves several points. The program interprets "funny" as "bad" in this context (this is not done for it in the input format). Janet really said that the picture was bad, and it is necessary to transfer this to the paints. We must know that if you want something another person has, you may make nasty comments about it in order to get it. Also, if we had asked, "Is the picture funny?", the response would have been (in essence) "No, she said so but she had an ulterior motive." At this point it should be realized that the program needs a lot of information about wanting, trading, giving, owning, and strategies for getting something from another person. While this example shows that the program has some knowledge of these topics, it knows virtually nothing about everything else.

(Janet continues)" I will let you have the pencils. I will take the paints."

Question: What does Janet want to do?

Answer: Trade the pencils for the paints.

Jack looked at the pencils.

"No, thank you, Janet," he said. "I want to paint more pictures."

Janet said to herself, "I do not want these pencils."

Soon Janet came back with something for Jack.

"Jack, here is a toy cat to put your money into," she said. "I will give you this cat and all my pencils for your paints."

Question: Why did Janet get the toy cat?

Answer: To trade with Jack.

The program realizes that "something" (in "Soon Janet came back with something for Jack.") and toy cat are the same object. (Again, this is not done in the input language.) It has to know that returning to a place where you just were with a given object constitutes getting the object. We might note that we could remove the "for your paints" from "I will give you this cat and all my pencils for your paints," and still understand Janet as meaning "trade". Of course, "give" does not always mean trade; only in certain contexts. This and the "funny" example point out the fact that inference is also necessary for deciding "meanings".

Question: Why did Janet offer to give Jack the cat and the pencils?

Answer: She wanted the paints, and Jack would not trade for the pencils.

"I have no money," said Jack. "What do I want with a toy cat? I want to paint more pictures."

Question: Will Jack trade?

Answer: No.

Question: Why won't Jack trade?

Answer: He doesn't want the cat and he wants the paints.

Actually "What do I want with an X" is treated as an idiom, meaning "I don't want an X" and is input this way to the program.

"I'll have to get some money for Jack," Janet said to herself.

Soon she came back with it. "Jack, here is some money to put into the toy cat," she said. "Now you will have some money, a toy cat, and all these pencils."

The phrases "some money" and "a toy cat" out of context may or may not refer to the cat and money

Janet is holding. The program decides that they do since it is anticipating a trade offer, and this can only be construed as a trade if the objects in question are Janet's.

Jack laughed and said, "Take your things and go away.

Question: Will Jack trade?

Answer: No.

The program interprets the phrase "go away" as implying that the person who said it does not want to do any current or suggested activity.

I want to paint pictures."

Question: Why did Janet kick Jack?

Answer: I was not aware that Janet kicked Jack.

Question: Where is Bill?

Answer: I don't know, but it seems like a silly question since no Bill is mentioned.

Question: When did Janet get the cat?

Answer: Before she offered to trade the pencils and the cat for the paints.

"When" questions are answered by mentioning an event which happened at roughly the same time. The major problem is making sure that the event used is important enough in the story so that the reader will be certain to remember it. In a longer story this becomes very important, though even here it makes a difference. The response could have been "Just after Jack looked at the paints" which occurred a few lines before Janet goes to get the cat. However, this response seems much less natural than the one given. Currently the test for importance is checking that the line has causal links with other parts of the story.

Question: Does Jack know that Janet wants the paints?

Answer: Yes, he was there when she offered to trade which

implied that she wanted the paints.

> When Janet said "I want the paints" she said it to herself. While the program can make use of this information, Jack cannot. So instead the system notes that "wanting the paints" is the reason for Janet offering to trade. It then assumes that Jack could figure this out also.

1.3 Some Comments

In this section I would like to clarify a few points which, in my discussion of my work, I have found to be potential sources of misunderstanding.

Learning

Many people in artificial intelligence seem to believe that a program, in order to be considered "smart", should be able to do its tasks without knowing very much. "Learning" is typically held out as a way to accomplish this task.

When this bias is combined with the fact that children's readers are intended to help children "learn" to read, many people assume that a program to read stories must be a "learning program"

In this regard I would like to emphasize that the chief concern motivating the model discussed here is relating a large body of knowledge to a particular story. (To get some idea of how much I believe is needed the reader might take a glance at section 5.1 where I try to list what ought to be known about piggy banks.) This knowledge is "put into" the program by hand. How to acquire such knowledge is not the main concern here.

The Relation of the Program to the Thesis

The program which did the example in 1.2 handled one other story fragment, also about trading, which was roughly twice as long as the example given. At that point, rather than extending the program to more stories I considered what should be done if I were to start all over again. The model presented in in the following pages is the result of this speculation. In the presentation of the "thesis model" I make no attempt to specify the exact points of correspondence with the computer program though for the most part the two are quite similar. Hence the existence of the computer program does not "guarantee" that the "thesis model" is completely specified, or at least could be extended to become completely specified. The reader will have to decide for himself how precise my model is. This is as it should be.

There seems to be a myth that a model is shown to "work" by making it into a working computer program. But most advanced programs in Artificial Intelligence handle only a few kinds of selected test examples. No one is interested in a program which works for three cases. What makes such a program seem important is the belief that it could do more, or perhaps, if the program were extended in some easily imaginable way, it could handle more examples. But deciding that this is the case means the reader must understand the ideas presented, work out a few test cases of his own, and

mentally simulate these cases himself. But such a check out procedure can usually also serve to convince the reader of the validity of a non programmed theory.

This issue should not be confused with a separate, but related belief. This is that programming makes for good mental hygiene. It seems to me that hygiene is a personal issue, and if a person wishes to keep himself "honest" by programming his ideas, that is up to him.

As I said, the model presented in this thesis is the result of rethinking the computer model. There were many reasons for not actually continuing with the program. I had the usual problem of poor design decisions which needed correcting but which would require rebuilding virtually the entire program. The major reason, however, is that making a working program involves solving a large number of time-consuming but theoretically uninteresting problems. Even more seriously, it does not allow the possibility of solving one sub-problem at a time. Very often one can have very good reasons for believing that some process must exist, which has a certain well-defined effect, even though one has no idea of how to construct such a process. One should not be restrained by this from studying other processes which depend on this unknown one.

A Note on the Examples Used

With the exception of the example in 1.2, the examples used in the thesis are not "real stories" but specially contrived examples to illustrate the point at hand. Most real stories have many different things going on in a given sentence so it is hard to determine the influence of the issue being discussed. To figure out all the factors in the story would mean continually straying from a logical development of the ideas. Since these examples are designed to be as simple as possible, they tend to be dry and somewhat unnatural.

By sticking to such examples I am excluding the more complex developments which can take place only in longer stories. However, the shorter examples can be considered fragments of larger stories, so the machinery we develop will be applicable to more realistic stories.

1.4 A Note to the Reader

The reader who only wants to get a taste of this thesis is advised to read chapters 1, 2, the first three sections of chapter 3, and chapter 10. If you are interested in reading more, I recommend the rest of chapter 3, all of 4, and the first four sections of chapter 5. The rest of 3 is optional in the sense that later chapters do not depend on it. Chapter 4, however, is must reading before going on to later chapters. (On the other hand, since chapter 4 is primarily concerned with notation you might try proceediing without it and if you encounter some unintelligible notation you can just look it up in 4.) Once chapter 4 is out of the way, the later chapters (5 6 7 8 9) can be read in most any order, though the given order is slightly preferable.

The program which did the example in 1.2 was written in LISP and Micro Planner. While later chapters assume a slight knowledge of LISP, it should be possible to read this thesis without knowing Micro Planner. (Probably the easiest way to become familiar with LISP is with (Weissman 67).) For those who do know Micro Planner I should point out that in my attempt to simplify Micro Planner syntax, I have removed the TH from the front of function names. Hence THCOND comes out COND, which is the name of the similar LISP function. Througout this thesis it should be assumed that in such cases I am referring to the Micro Planner function.

Finally as an aid to the reader, Appendix B is a list of abbreviations and special terms. It gives the unabbreviated forms and one or more pointers to portions of the text which explain the terms' significance.

2 An Overview

Section 2.1 discusses what tasks have to be done in going from natural language to our proposed internal representation. Section 2.2 gives examples in which "understanding" is necessary to accomplish some of the tasks mentioned in 2.1. We use such examples to explain the advantages of a system which understands as it goes along, over one that does deduction only when asked a question. Sections 2.3 and 2.4 introduce the four main components of inference or deep semantic processing (DSP) as I shall call it. They are:

Demons - Facts which are introduced by "concepts" occurring in the story are called "demons" since in many cases they must wait for further information. In such cases we can think of them "looking" for the appropriate fact. So "not being willing to trade" might put in a demon looking for a better offer.

Base routines - These constitute what we know about a "concept" independent of "context" So, for example, if A gives B to C then C now "has" B. This is not dependent on what happened earlier in the story.

Bookkeeping - This does chores like keeping the data base relatively consistent and non-redundant. So, should a person in the story change location, we must update the old location statement.

Fact finders - These are utility routines for doing standard deductions which aren't worth asserting separately. A typical fact finder might say, "If you want to know if person P knows fact F, just see if when F occurred, (or was said by some character in the story) P was around."

2.1 From Natural Language to Internal Representation

Finding the Relations

Several sentences can have (pretty much) the same meaning, as in:

(2.1) A woman who is pretty painted the room.
(2.2) A pretty woman painted the room.
(2.3) The room was painted by a pretty woman.
(2.4) The room was painted by a woman who is pretty.

We are told that a certain woman painted a certain room, and furthermore that this woman is pretty. One trouble with natural language is that its syntax is quite complicated. So in both (2.1) and (2.2) we are speaking of a "pretty woman", yet in (2.1) "pretty" occurs after "woman" in the sentence, while the reverse is true in (2.2). We would like an internal representation to provide a simpler syntax. So we might express (2.1) - (2.4) as:

(2.5) (PAINT WOMAN ROOM)
(2.6) (PRETTY WOMAN)

Now one might object that (2.1) - (2.4) do not have the same meaning, hence representing them all the same way is a mistake. So while (2.1) and (2.2) are about a certain woman, (2.3) and (2.4) are about a room. The sentences do not have the same focus. To put this another way, in chosing passive or active forms, the story teller has made a decision about what is important in the story. However, he made a more important decision when he chose to tell us that the woman

painted the room, as opposed to, say, "blew it up". In this thesis we will be concerned with what it means to "paint" or "blow up". While the shifts in meaning caused by the active passive distinction, and the positioning of modifiers are interesting, they are second order effect.

Objects such as (2.5) will be called "assertions." An assertion will be assigned a standard word order. (In (2.5) and (2.6) the first item in the assertion is called the "predicate".) So in (2.5) the fact that WOMAN comes before ROOM will be taken as meaning that it was the WOMAN who did the painting. We could have decided on an order like:

(2.7) (ROOM PAINT WOMAN)

but once we have decided on the order inside the assertion it cannot be changed. Of course, we haven't accomplished a great deal by putting PAINT first, or having all the words in capital letters. We still won't know that (2.6) implies that the woman is not ugly, but at least there is no longer any doubt about what "pretty" refers to, or the fact that it was the woman who painted the room, and not the room who painted the woman. Establishing the relations in a sentence is probably the most important part of the translation from natural language to internal representation.

I should mention that I have expressed the relations in prefix notation simply because it is the most natural for me. There are arguments about the desirability of prefix vs. infix

notation. These arguments deal primarily with the ease of expressing certain grammatical rules with the two notations. While these arguments are clearly beyond the scope of this thesis, the reader might want to consult (McCawley 1970).

The internal representation of a line from the story will enter the data base. In the course of processing the line we will deduce new facts about the story, and this will cause other assertions to enter the data base. To avoid confusion, when I speak of the internal representation of a line I do not include any assertions which may be deduced from the original assertions.

Separating Word Senses

Now consider the sentences:

(2.8) Jack caught a cold.
(2.9) Jack came down with a cold.
(2.10) Jack got a cold.

Most people would agree that these have the same significance. If we were to represent (2.8) - (2.10) as:

(2.11) (CATCH JACK COLD)
(2.12) (COME-DOWN-WITH JACK COLD)
(2.13) (GET JACK COLD)

every time we had to answer the question "Is Jack ill?" we would have to look for each possible format. Clearly it would be better, since we agree that they all have the same significance, to translate all the sentences into a single format. Since we can disambiguate these examples without

further context, we will not be concerned with such cases; the words will be disambiguated in the input. A system which did handle these cases would have to know facts about possible meanings of "catch" "get" and others.

But if we assume that (2.8) - (2.10) will be translated into, say,

(2.14) (BECOME-SICK-WITH JACK COLD)

we will have to separate meanings of words. That is, we would not want

(2.15) Jack got a ball.

to become

(2.16) (BECOME-SICK-WITH JACK BALL)

Then our second translation activity is to separate meanings of words so that a given fact will "tend" to be represented in a single form in the internal representation. The "tend" is a qualification due to imprecision in what constitutes "a given fact". While it is clear that (2.8) - (2.10) are all the same fact, what about:

(2.17) Jack caught the ball.
(2.18) Jack got the ball by placing some part of his anatomy in the flight path of the ball and after the ball hit that part of his anatomy he did not let it go.

While (2.17) and (2.18) are in some sense giving the same information, the differences in expression are quite large. In general, only when the differences are "small" as in (2.8) - (2.10) can we assume that the internal representation will

be identical.

Determining Reference

When we say that "Jack has a top" we mean that he has a particular top, which can be distinguished from, say, the top which Mary has. Our internal representation must use different symbols for the two objects, so as not to confuse them, as in (HAVE JACK1 TOP1). We put a number after the "top" to distinguish it from other tops. (To be consistent we have also put numbers after JACK since there could possibly be more than one person in a story with a given name.) Henceforth an "object" is an indexed element in an assertion. However, when we talk, we don't mention TOP2, we say "Mary's top" and rather than TOP1 we might say "The one Jack got". So it is necessary to translate between the English descriptions and our indexed objects. This, simply, is the problem of reference, and it is another problem syntax and semantics must deal with.

Syntax and Semantics

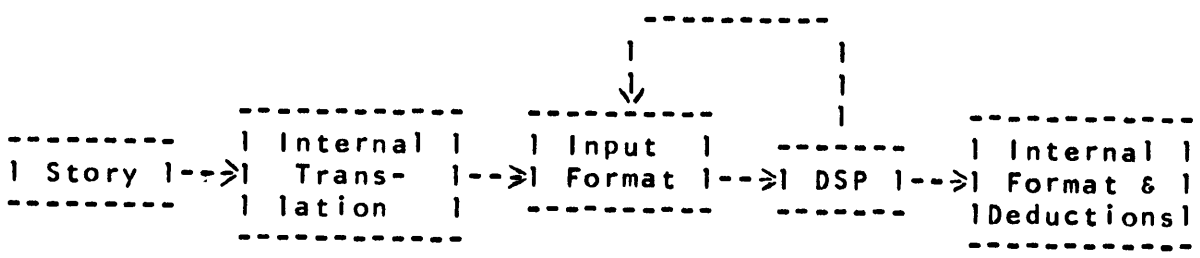
I have introduced the transformation into internal representation by looking at the properties we want of our internal format. I could equally well have started by looking at what aspects of language current syntatic and semantic theories account for. Syntax with its divisions of words into

categories like nouns and verbs, and its "rules of syntax", can account for much of thestructure of language. To use my terminology, syntax helpsdetermine the relations in a sentence. While syntax is far from being completely understood, there exist several competent syntactic parsing programs (Dewar et al. 69), (Winograd 71), and (Woods 69).

Semantics also aids in determining the sentence structure besides helping to separate word senses. "Semantics" means different things to different people, from the study of how to use language more effectively, to deriving the significance of diplomatic communique, to everything about language not included in syntax. When I use the term, I simply mean what is called semantics in much current linguistic work. See (Chomsky 65), (Fodor and Katz 64), and (Lakoff 71). "Semantics" in this tradition is limited to "linguistic information" as opposed to "non-linguistic information", the latter roughly corresponding to what I call "real world knowledge". This level of analysis has been incorperated into several comprehension programs to date, (Alexander 71), (Simmons et al. 68), and (Winograd 71).

I will define "internal translation" as these two processes (syntax and semantics) and henceforth drop the words syntax and semantics. The important fact about internal translation is how little information it uses since its usefulness is directly related to this limitation. One wants

to get some toehold on the sentence to start the analysis. It seems to be easiest to construct models where the initial analysis uses little information. Hence it is the paucity of information used in internal translation which makes it useful as a way of getting "into" the sentence. On the other hand, as we will see in the next section, this lack of information will also insure that internal translation will not be able to complete the transformation of the sentence into the internal representation. We will call the result of internal translation the "input format" since, it is the input to DSP. The relation between input representation and internal representation is expressed in the diagram.



The arrow from DSP back to input format is to indicate that the input will be steadily modified in the process of applying DSP.

2.2 Internal Translation and Its Problems

In order to do a complete job of translating into internal representation, we need information which we have not made available to internal translation. For example, consider the problem of establishing the relations in the following sentence:

(2.19) I saw the man on the hill with the telescope.

This is three ways ambiguous, depending on whether "with the telescope" modifies "see", "hill" or "man". There is nothing in the sentence which tells us one way or the other, although there seems to be a natural order of preference, with "see" the most likely, and "man" the least likely to be modified by "with the telescope". This would suggest that we might try the possible sentence structures in order of likelihood, but no matter how we do it, we will need to interrogate our data base to see who or what is known to have a telescope. This would be part of the information about the particular story, hence not part of our internal translation information.

Just separating the senses of a word like "give" can be quite complex. If Bill gave his knife to Jack, does it mean that Jack now owns the knife, or merely that Jack is now holding it? Consider:

(2.20) Jack, Bill and Jack's dog Tip were outside. Bill said, "I like Tip. Will you trade him to me, Jack? I will give you my pogo stick." "I don't know" said Jack. "I will have to ask my father." "Show him the pogo stick" said Bill. "He will like it." So Bill gave Jack the pogo stick and Jack went home.

The first time Bill mentions "give" he clearly means "trade" and a change of ownership. However, the second time, we understand that Bill is only giving Jack the pogo stick temporarily, and hence the "give" means "hold", and not "own".

Determining reference is just as complex. We saw one example in the introduction. We give a slightly modified version here.

(2.21) Today was Jack's birthday. Penny and Janet went to the store. They were going to get presents. Janet decided to get a top. "Don't do that" said Penny. "Jack has a top. He will make you take it back."

Would he make Janet take back the new top or his (old) one? It's obvious to us, but why? In the next chapter I will try to demonstrate that much of thestory must come into play in determining the correct referent for the "it" in the last sentence. It seems that there are no particular limitations on the information which can come into play in determining reference. Let us now focus on one particular implication of this assumption.

Making Deductions as One Goes Along

One popular conception of English comprehension has the program applying internal translation to each input sentence,

but not applying what I am calling deep semantic processing until a question is asked that requires the program to connect the events in the story to make a coherent whole. This model seems to be assumed by work using predicate calculus theorem provers as a "back end" for natural language question answering systems (Black 68), and (Green 69). (This is also observed by Simmons in (Simmons 70).)

Such a "delayed deduction" model has many problems. Suppose in doing a deduction we needed a particular line L which has a pronoun like "it" in it. If it is indeed the case that deductions from anywhere earlier in the story might effect our interpretation of "it" we would have to go back and "understand" all the story previous to L. (By "understand" I just mean make the kind of deductions which I intend to do on the fly.) If this is the case, then there would certainly be no point in waiting to make deductions at all.

In the case of a pronoun we at least know that we need prior deductions in order to complete our understanding of the sentence. But what about a sentence like:

(2.22) Jack got on the elephant.

There is nothing in this sentence to indicate that it does not mean "what it says". But it could be preceded by:

(2.23) Jack, Fred, and Ann were playing circus. Fred said "That rock over there is an elephant".

So it would seem that any sentence in the story could have its significance altered in a major way by earlier lines in the

story. Naturally, this is problematic for a delayed deduction scheme.

Another Comment on the Input

In chapter 1 I said that the input to DSP was pre-processed, with the limitation that it could not contain information which could not be derived from the sentence totally out of context. Actually, given the development so far, a more natural limitation is what can be done by internal translation. To see the difference, consider a sentence like "Jack slid into second base". Out of context we would guess that Jack is playing baseball and was involved in a close play at second. Nevertheless, such deductions would not be part of internal translation, since the deductions seem to require information which goes beyond the bounds of internal translation.

Since internal translation works on individual sentences without context, it is clear that the internal translation restriction is a stronger one than the "out of context" restriction. While I have actually adopted the "context" restriction, since it is more precisely statable, in the thesis I have tried to limit myself to the "internal translation" restriction since it seems to define the most natural breaking point for the problem.

2.3 Demons and Base Routines

Consider a fact like:

(2.24) "If it is raining" and

"If person P is outside" == "P will get wet"

We have an intuitive belief that (2.24) is a fact about "rain", rather than, say, a fact about "outside". Many things happen outside and getting wet is a very small part of them. On the other hand only a limited number of things happen when it rains.

We will embody this belief in our system by associating (2.24) with "rain" so that only when "rain" comes up in the story will we even consider using rule (2.24). We will say that rain is the "topic concept" of (2.24). (By "concept" I mean a symbol without an index (TOP1 has an index, GO does not) other than tense, type and negation markers (see 4.1) which appears in assertions, or relations between such symbols.) To put this another way, when a concept is brought up in a story, the facts associated with it are "made available" for later use. (Wewill also say that the facts are "put in" or "asserted".) So, if "circus", say, has never come up, the program will not be able to make deductions using those facts associated only with "circus".

Note however that we are not saying that "rain" has to be mentioned explicitly in the story before we can use (2.24).

It is only necessary that there be a "rain" assertion put into the data base. Other parts of the story may provide facts which cause the program to assert that it is raining. For example:

(2.25) One afternoon Jack was outside playing ball with Bill. Bill looked up and noticed that the sky was getting dark. "I think we should stop" said Bill. "We will get wet if we keep playing."

Here, the sky's getting dark in the afternoon suggest that it is going to rain. If this is put into the data base it will be sufficient to bring in facts associated with "rain". (Actually, to account for (2.25) we would need to modify (2.24) slightly, since (2.24) requires that it <u>is</u> raining, and in (2.25) we only suggest that it <u>will</u> rain. However, I am not suggesting that (2.24) is a real fact the program must know. It is simply used for illustration.)

Also note that at topic concept need not be a single "key word". A given set of facts may not become available to the system until a complex set of relations appear in the data base. How this can be implemented will be seen later.

Looking Forward, Looking Back

Only considering facts after we have seen the "topic" concept allows that we might see the topic concept before we have all the information needed to make use of the fact. This would be no problem in a delayed deduction scheme because a rule is only used when the user askes a question. If the

facts which enable the rule to work are missing, it simply means that the rule will not be used. But, when making deductions "on the fly", if the necessary information comes in after the rule has been introduced we want to make the deduction when the information comes in. So we might have:

(2.26) Jack was outside. It was raining.
(2.27) It was raining. Jack was outside.

In (2.26) there is no problem. When we introduce "rain" we have sufficient information to use (2.24) and deduce that Jack is going to get wet. But in (2.27), we only learn that Jack is outside after we have mentioned rain. If we want to use (2.24) we will need some way to have our fact "look forward" in the story. To do this we will break a fact up into two parts, a pattern and a body (a program). We will execute the body of the fact only when an assertion is added to the data base which matches the pattern. So with (2.24) the pattern would be "someone outside". Hence in (2.27) we introduce (2.24) when we see "rain". At that time no assertion matches the pattern. But the next line will create a matching assertion, so the body of the fact will then be executed. Hence we will say that a fact is "looking forward" when the assertion which matches its pattern comes after the assertion which made the fact relevant to the story. When the assertion which matches the pattern comes before we will say that the fact is "looking backwards".

We can see how important looking forward is with a few

examples.

(2.28) In example (2.21) we interpreted the line "Jack has a top" as meaning that he did not want another. The common sense knowledge is the fact that in many cases having an X means that one will not want another X. This piece of information would probably be filed under "things to consider when about to get something for somebody else". Naturally it was an earlier line which mentioned that Janet was thinking of getting Jack a top.

(2.29) "Jack was having a birthday party. Mother baked a cake." The second line is interpreted as meaning "for the party" on the basis of information about birthday parties brought in by the first line.

(2.30) "Bill offered to trade his pocket knife for Jack's dog Tip. Jack said 'I will ask Janet. Tip is her dog too." The last line is interpreted as the reason Jack will ask Janet because of information about the relation between trading and ownership.

(2.31) "Janet wanted to get some money. She found her piggy bank and started to shake it. She didn't hear anything." The last line means that there was nothing in the piggybank on the basis of facts about piggybanks.

In each of these cases it is an earlier line which contains the information which is used to assign the interpretation. So in the first example there is nothing inherent in the line "Jack has a top" which means "don't get him another". Suppose there were. Changing the example to "Jack has a ball", something in the line would have to key a check for the following situations as well:

(2.32) "Bill and Dick wanted to play baseball. When Jack came by Bill said "There is Jack. He has a ball."

(2.33) Tom asked his Father if he would buy him a ball. "Jack has a ball" said Tom.

(2.34) Bill's ball of string was stuck in the tree. He asked Jane how he could get it out. Jane said "You should hit it with something. Here comes Jack. He has a ball."

Those familiar with Planner might notice that our "facts" look quite similar to Planner antecedent theorems, with the exception that our facts can "look back" as well as "look forward". Antecedent theorms are only designed to look forward. Another difference is that while antecedent theorems are automatically executed when a relevant assertion is entered into the data base, demons will be called some time after the assertion has been created. I initially formulated facts as antecedent theorems because I was so impressed with the need to "look forward". However, rather than call the facts antecedent theorems, we will call them "demons" since it is a shorter name. "Demon" is also the older term first coming into computer literature with (Selfridge 59) and (Minsky 63).

Possible Futures

We require a concept to be introduced by the story before facts associated with it become available for making further deduction. One aspect of this decision is the significance it gives to statements about future events. In general, statements in future tenses are problematic. While it is usually clear what a present tense statement means, what is one to make of some statement that "Jack must buy a new suit"?

It does not mean that we can be certain that at some point in the future we will see Jack buying a new suit. It certainly does not mean that Jack is currently at the clothing store. So what is one to do with such a statement? In the system I propose, the answer is fairly clear. Such statements ("possible futures") introduce facts that can be used to interpret some of the things which may happen in the story. Once we see "must buy a new suit" we will be looking (forward) for the person's going to a clothing store and, when and if he does so, we will know why. Note that this analysis applies equally well to "can buy a new suit", "will buy...", "should buy...," etc. This is not to say that all these phrases mean the same thing. Rather, the concept of possible future is designed to capture what they have in common. Their differences will have to be accounted for by other means.

In some sense the name "possible future" is of a misnomer, since it does not really describe a "future" at all, but at best some aspect of the future. Perhaps a better name might be "possible features of the future". In this respect it corresponds roughly to what Mc Carthy calls "fluents", though the two concepts are used somewhat differently in their respective schemes (Mc Carthy and Hayes 1968).

Specification and Removal of Demons

It should be emphasized again that the model does not

"learn" the information contained in the demons. On the other hand, the demons are not specific to the story in the sense that they mention Jack, or "the red ball". Rather, they talk about "a person X" who at one point in the story could be Jack, at another, Bill. In chapter 4 we will briefly discuss how a demon will get the correct assignments for its variables.

We want demons to be active only while they are relevant to the story. A story may start by talking about getting a present for Jack, but ultimately revolve around the games played at his party. We will need some way to remove the "present getting" demons when they have outlived their usefulness. (an irrelevant but active demon not only wastes time and space, but can cause us to misinterpret a new line.) This will be discussed later in 8.3.

Base Routines

So far we have said that demons are introduced to the story when the proper concept has been mentioned. But this implies that there is something attached to the concept name telling us what demons should be put in.

If we look at a particular example, say (2.30), it is Bill's offer to trade the pocket knife for Tip, which sets up the context for the rest of the fragment. So we must have some information, which this line somehow accesses, telling us

to activate certain demons. We will assume that this information is in the form of a program. Such routines, which are available to set up demons, will be called "base routines" and will be designated by -BASE at the end of their name, as in TRADE-BASE which is activated when an assertion with the symbol TRADE is placed in the data base..

These base routines will be responsible for more than setting up demons. Suppose we are told that Jack had a ball, and Bill a top. Then Jack traded his ball to Bill for the top. One question we might ask is "Who now has the top?" Naturally since questions of "who has what" are important in understanding stories we will want to keep tabs on such information. In this particular case, it must again be the "trade" statement which tells us to switch possession of the objects. To be more formal, we would say "trade" entails the fact that the objects have switched owners. (Though note that I am not using the formal logical definition of "entails") Since every time a trade occurs we will want to exchange objects, it must be the casethat whenever we see "trade" we execute TRADE-BASE. Of course, the program can't be too simple minded, since it must also handle "I will trade..." and perhaps even "Will you trade ...?"

A good test as to whether a given fact should be part of a base routine or a demon is whether we need several lines to set it up or whether we can illustrate the fact by presenting

a single line. (Naturally several lines could be made into one by putting "and's" between them, but this is dodging the point. I am only suggesting an intuitive test.) So we saw that "Jack has a ball" was not enough by itself to tell us that Jack does not want another ball. Hence this relation is in a demon, not in a base routine for, say, "have". On the other hand, often a single line can tell us quite a bit as in "Jack slid into second base". This indicates that the base routine for "second base" interacts with "slide" to tell us that we are talking about a baseball game.

There is some question as to whether demons and base routines are really distinct entities. When a demon is looking forward it is waiting until an assertion is added to the data base which matches its pattern. This is exactly what a base routine does, but while the demon will only be asserted when we have encountered the proper concept, base routines are always available.

Of course, we are only looking at children's stories. Will we want the same set of base routines for reading both fiction and chemistry, or will base routines also be shuffled in and out, like demons, but only on a broader scale? Rather than being added and removed within the confines of a single story, they might be added and removed depending on the type of literature we are reading. While I tend to think of base routines and demons as distinct, in practice the view adopted

does not seem to make too much difference in the course of the thesis.

2.4 Bookkeeping and Fact Finders

Updating and Bookkeeping

Up to this point we have introduced two parts of DSP, demons and base routines. In this section we will introduce the remaining two parts

Again let us consider the situation when Jack had a ball, Bill a top, and they traded. When we say that Bill now has the ball, it implies that Jack no longer does. That is to say, we must somehow remove the fact that Jack has the ball from the data base. Actually we don't want to remove it, since we may be asked "Who had the ball before Bill did?". In this particular case we might be able to construct the answer from the fact that there was a trade, though we would also have to be on the lookout for "give" and even Jack losing the ball and Bill finding it. While this would be difficult, in other cases such an approach would be seemingly impossible. For example, suppose Janet wanted a doll, and she did several things to get it, but then decided that she didn't want a doll, but a paint set. If we were to erase the fact "Janet wants a doll" we would no longer be able to answer questions like "Why did Janet go to the toy store?" since we would have erased the reason. Instead, we want to mark the assertions in some way to indicate that they have been updated. So, going back to Bill, Jack and thetrade, we might have:

(2.35) (N1 HAVE JACK1BALL1) TROUBLE:(NEG-UPDATE N4)
(N2 HAVE BILL1 TOP1) TROUBLE:(NEG-UPDATE N5)
(N3 TRADE BILL1 JACK1 TOP1 BALL1)
(N4 HAVE BILL1 BALL1)
(N5 HAVE JACK1 TOP1)

The N1 etc. are names given to the individual assertions so we can refer to them. The TROUBLE marker is on the property list of the assertion. It basically says that the assertion cannot be taken as currently true. NEG-UPDATE indicates that the reason is that the assertion has been negated at a later date. There are other ways an assertion can get a TROUBLE property. For example, in our story in chapter 1, when Janet said "Those paints make your airplane look funny", we gave the assertion a TROUBLE:(ULTERIOR-MOTIVE JANET1) property. (The distinction between an assertion and its property list is a function of the workings of Micro Planner in the same way the distinction between an atom and its property list is a function of the workings of LISP.)

Throughout this thesis we will assume that there is a separate section, pretty much independent of the rest of DSP, which is responsible for doing such updating. We will call this section bookkeeping. However, there are some problems with this conception of things, and in 7.3 we will consider (and reject) an alternative (having the base routines do updating). Besides updating, bookkeeping will also be responsible for noting if the new assertion is a duplicate of an old assertion (children are constantly repeating things in

children's stories). If the new line is a duplicate it will not be re-entered into the data base.

Fact Finders

But even deciding that one statement updates another requires special knowledge. Suppose we have:

(2.36) Jack was in the house. Sometime later he was at the store.

If we ask "Is Jack in thehouse?", we want to answer "No, he is at the store." But how is bookkeeping going to figure this out? There is a simple rule which says that ((state) A B) updates ((state) A C) where C is not the same as B. So (AT JACK FARM) would update (AT JACK NEW-YORK). But in (2.36) we can't simply look for Jack AT (someplace which is not the store), since he is IN the house. To make things even worse, we could have:

(2.37) Jack was in the house. Sometime later he was in the kitchen.

To solve this problem we will add a theorem which knows about location. Such a theorem, we will call it AT-NOT-FF, might go:

(2.38) To establish that PERSON is not at location LOC

Find out where PERSON is, call it X
If X = LOC , then theorem is false so return "No".
If X is part of LOC then return "No".

This handles, given appropriate information, cases like (2.37).

If LOC is part of X, then try to find a different X
Else return "Yes"

In (2.36) the bookkeeper would try to prove that Jack is not at the store, and it would succeed by using AT-NOT-FF and the statement that Jack is in the house. Bookkeeper would then mark the earlier statement as updated. Theorems like (2.38), called fact finders, will be indicated by -FF at the end of their names, such as AT-NOT-FF.

Like demons, fact finders have a pattern and a body. A particular fact finder is called when something else (either a demon, base routine or bookkeeping) wants to establish a goal which matches the pattern of the fact finder. This is different from demons which are called when we encounter a given fact. In Micro Planner fact finders are Consequent theorems, while demons, as we have already mentioned, are Antecedent theorems.

We introduced fact finders via bookkeeping. However, fact finders are needed elsewhere, and in fact, are a more secure part of the model than is bookkeeping. To take one example, we could have:

(2.39) Jack was in the house. Janet was at the park.

We would then ask the program "Is Jack at the park?" and AT-NOT-FF would supply the answer. To take another case, typically when a person offers a trade, we will want to make sure he owns the object being traded. One good rule of thumb is if you don't have an explicit "own" assertion, then the person who introduced the object into the story is the owner. This rule came into play in the main example in chapter 1. Naturally, this rule is also a fact finder.

The basic idea behind fact finders is that they are used to establish facts which are comparatively unimportant, so that we do not want to assert them and hence have them in the data base. So in (2.36) we do not want to assert "Jack is not in the house" as well as "Jack is at the store". In the same way we will have a fact finder which is able to derive the fact "(person) knowns (fact)" by asking such questions as "was the (person) there when (fact) was mentioned or took place?". Again, since this information is easily derivable, and not all that important, so there would seem to be no reason to include it explicitly in the data base.

Fact finders, then, act like axioms in traditional theorem provers in that they are used for making chains of deductions. Hence there is the traditional problem of infinite or exponential search. I am assuming that the extra machinery which a Planner based system gives one will be sufficient to handle this problem. (Since this is one of the

features of Planner, any discussion of the language will describe how it helps eliminate infinite or exponential search problems.) Whether this assumption is justified is a topic for future research.

In the current scheme of things fact finders are always around in the same way that base routines are. They are not activated and deactivated according to the context. Such an ability would not be inconsistent with the rest of the model, but it has not been needed so far. Presumably this is because such context dependent facts have been important enough to include in the data base so they can be handled by demons.

2.5 Facts and Search

To answer questions one must have (1) the relevant facts, (2) the ability to access them, and (3) the ability to do deductions. In this section I will make a few comments on the first and second problems. Since I have, in some sense, been "thinking" in Planner, and since Planner itself determines how deductions are made, I have given little thought to methods of making deductions.

Infinite Stories, Finite Facts

Just as a theory of syntax must account for an infinite number of sentences with a finite number of "rules of grammar", a theory of story comprehension must account for an infinite number of stories with finite"every day knowledge". If we are going to accomplish this task, the facts we include in the system must really capture the facts that people use, and must not be special cases. (Though technically we could still need only a finite number of facts if we used a finite number of special case facts for each "real" fact.)

Capturing the facts which people use is no easy process. Let us look at a fact used in last chapter's example. We wanted to be able to understand why Janet said "Those paints will make your airplane look funny," (from chapter 1). The rule we suggested there was as follows:

(2.40) If a person P wants an object X, then look for P saying that X is "bad". If P does this it is because P wants X.

We can easily see that this rule does not completely capture what people know. Consider:

(2.41) Janet had decided that she would bake a cake that afternoon. WhenJack came by she said, "Jack, I am going to make a cake this afternoon, would you like to help?" "I am going to the beach this afternoon" said Jack. "It will be cold" said Janet.

While our intuition says that the same thing is happening in both the example of the last chapter and (2.41), (2.40) will not handle (2.41). So, while superficially reasonable, (2.40) is only a special case of the rule which people seem to know.

In other cases we can easily fall into false generalizations. It is not unreasonable to believe that the phrase "take (object back" means "take object to a place were the object was at some prior time". But, this is falsified by:

(2.42) Bill and Frank were out camping and looking for indian relics. Bill found a large bowl. After cleaning it off some he said, "I really need some of the tools we left at the campsight. I will take it back now and give it a careful cleaning."

Search Planner Style

The method of finding relevant facts which was used in the forementioned program, and which was in the back of my mind while writing this thesis is the method which is most natural in Planner. (Also Micro Planner. Though the

differences between the two are many, at the level of generality found in this section the two can be considered the same.)

Demons, fact finders and assertions are stored in separate data bases, according to a filing system based on their patterns. (We have already mentioned the patterns of demons and fact finders. The pattern of an assertion is just the assertion "proper", as opposed to the property list attached to the assertion.) So, for example, when we encounter a given concept, an associated demon is put into the appropriate data base on the basis of its pattern. When we want to find whether there are any demons looking for a given assertion, we use the same filing scheme to look in the demon data base.

Subtopics and Search

There are naturally many other search schemes. Some of these are realatively straight forward modifications of the system already outlined in the sense that the major assumptions of this chapter are left unchanged. So, for example, rather than asserting all the demons associated with a given topic, perhaps one could cluster the demons into larger groupings. So, the facts about "present" (gift) might be clustered about subtopics like "buying", "surprise", "choosing", etc. These subtopics themselves become demons of

a sort, but since there are many facts, and only a few subtopics, we would save a lot of time by asserting only the "subtopic demons". Naturally, when new assertions enter the story they will look to see if there are any subtopic demons which might be applied to them. If there are, then there will be pointers from the subtopic demon to the "real" demons, and the rest of the deduction procedure will be the same.

There are two important questions to ask about this scheme. First, will there be enough demons connected with any given topic to make this scheme worthwhile? And second, will the demons fall easily into subtopics? Before we can answer these questions we need a large body of established facts. Since we do not have such a body, I have not pursued this idea any further.

Significant Search Differences

The subtopic search scheme just presented leaves all the basic assumptions of this chapter unchanged, so our ability to do deductions remains unchanged. More important search issues actually call these assumptions into question.

For example, in section 2.3 we suggested that no demon would be asserted until its topic concept had appeared in an assertion. Can we stick to this assumption? Naturally, we can keep this assumption by asserting a topic for the sole purpose of putting in the necessary demons. Of course, if we

were forced to do this we would instead admit that the "topic" restriction on demon asserting should be relaxed. This will be discussed in section 5.7.

In the last section we added fact finders to our model. They, along with actual assertions, are our only way to answer a question, whether the question is asked by the user, or generated internally in the course of making some deduction. We might ask whether these two are sufficient. In section 5.4 we will suggest that the answer is no. In particular we will argue that demons must also be allowed to "answer questions." (Keep in mind that demons, as we have formulated them do not "answer questions" but simply respond to the presence of assertions which match their patterns. They may produce assertions which answer later questions, but demons, so far, are never called upon to answer a specific question.)

To the extent that our suggestions in chapter 5 are correct, we will have demonstrated that the model presented in this chapter is incorrect. However, the changes required in chapter 5 can be viewed as extensions of the model presented here, so it seems best to present the simplified model first.

Of course, these are only a few of the assumptions we have made in the course of this chapter. For a brief discussion of some of the others the reader might consult section 10.2

3 Reference and Deduction

In this chapter we will see how complicated deductions can come into reference decisions. I am singling out this topic for several reasons. First, in section 7.1 we will discuss ordering the parts of the comprehension model presented in chapter 2. The arguments put forth there will depend primarily on the model of reference we outline in this chapter. Secondly, I find the way reference fits into my comprehension model very "natural". This "naturalness", it seems to me, constitutes a powerful argument for the merit of the all-over model. Finally, although we have been discussing "understanding" a story, "understanding" is a poorly understood term. Reference, however, is clearly defined. Yet, most of the process which we would call "understanding" seem to be necessary to decide reference. Hence, reference can be viewed as a paradigm of "understanding".

We saw in 2.2, particularly in example (2.21), that fairly complicated deductions can be necessary in choosing the correct referent. For convenience we repeat (2.21) here as (3.1).

(3.1) Today was Jack's birthday. Penny and Janet went to the store. They were going to get presents. Janet decided to get a top. "Don't do that" said Penny. "Jack has a top. He will make you take it back."

(3.2) Penny wanted to go to Bill's party. Mother had to tell her that she had not been invited.

(3.3) When Penny heard about the costume ball she started thinking about what Mother could wear. Mother had to tell her that she had not been invited.

In (3.1) the problem is choosing the correct referent for the "it" in the last line. In (3.2) and (3.3) the second sentence remains the same, but due to the difference in first lines we understand the last "she" differently in each case.

In an example such as (3.1) we will be making deductions in order to answer questions like:

(3.4) Why did Janet and Penny go to the store?
(3.5) Who are the presents for?
(3.6) Why would Jack make Janet take the top back to the store?

In this section we will suggest a way to determine references while making such deductions. That is, the routines we introduce to handle such questions will also aid in reference decisions. (I will limit discussion to definite noun phrases. For further discussion of reference see chapter 6.)

Note that we would need some means of answering (3.4) - (3.6) even if the story (and questions) were given to the machine with all the referents already determined (i.e. the input would explicitly mention TOP1 etc.) That such routines will also help in deciding reference is the "naturalness" mentioned above.

3.1 Kinds of Information Used in Reference Decisions

We decided that deductions about the story allowed us to pick the right referents in (3.1) - (3.3). However we did not argue the point, but rather relied on the reader's intuitive understanding of the factors which can influence the choice of referents. Let us take a closer look at exactly what some of these factors are.

Descriptive Information

When we say "Jack", or "the ball" we know that the object in question is named Jack, or is a ball. We could further specify the object by adding adjectives, as in "the red ball". A slightly less trivial example, "Jack's house", specifies some house which satisfies a possessive relation with respect to Jack. In the case of "house" it probably means the house in which Jack and the rest of his family live. If we needed even further specification we could add subordinate clauses, as in "the red ball which was thrown on the roof yesterday".

The information in these cases is "descriptive" in the sense that the noun phrase describes the referent it is aiming at. In most cases it is easy to use descriptive information. When we look for the referent for "the ball" we can exclude anything which is not labeled "ball". Other cases can get more complex however. We can't just look for a house which Jack owns when we see "Jack's house". We need information

about houses to tell us that often a possessive applied to "house" means that the person's family lives there. Nevertheless, "Jack's house" gives us a great deal of information about the house under consideration, and it is this information which I call "descriptive".

Most natural language programs make use of descriptive information. Probably the most complete and satisfactory account is found in (Winograd 72). Naturally, in examples (3.1) - (3.3) descriptive information played a role (we didn't even consider the possibility that "it" might refer to Janet), but by itself, descriptive information was not sufficient, since in (3.2) and (3.3) the "she" could refer equally well to either Mother or Penny, and in (3.1) "it" or "the top" could refer to either top.

Recency Information

One simple kind of information is "recency" information. If I am talking to you and mention "the fig" I am referring to the one I mentioned two sentences back, not the one I last mentioned in our conversation two weeks ago. Such information plays an even more crucial role in determining pronoun reference, since in most cases the major clue to the identity of the referent is the fact that it most likely has been mentioned in the last two or three sentences.

Differences in "recency" do not have to be very large in

order to be significant. Consider:

(3.7) Bill threw Jack a green ball.
Jack was holding a red ball.
Jack threw it to Dick.

(3.8) Jack was holding a red ball.
Bill threw him a green ball.
Jack threw it to Dick.

Since we wish to emphasize context, we will choose examples where recency information by itself is insufficient to choose the referent, such as (3.1) - (3.3).

Syntactic Information

Syntax also has a role in reference. Consider the sentence "Bill washed him." We know that "him" does not refer to Bill, since if it did we would have to say "Bill washed himself." There are several ways to convince ourselves that this is fact. First we needed no context to make our decision. Also, we could replace "wash" with a nonsense word like "setan" and we would still know that in "Bill setaned him" "him" is not the same as "Bill". Furthermore, if we knew that "him" did refer to "Bill" we would say that the sentence was ungrammatical, in the same sense that "I washed me" is ungrammatical. The meaning of "I washed me" is perfectly clear, so its rejection must be on the basis of syntax.

Such grammatical rules are only capable of ruling out possible referents. So while in "Bill washed him" we know that "him" does not refer to "Bill" we do not know to whom it

does refer. I might point out that for more complicated sentences the rules which govern pronoun behavior are poorly understood. We will come back to this point later in section 6.2.

Selectional Restrictions

Verbs often have "selectional restrictions" on what kinds of things can serve as their subject or object. For example, in "He told Bill about the riot," "he" can refer to Jack or "the boy in the yellow shirt", but not "my dog Rover". In the same way "it" in "She kicked it" cannot refer to "the race", though in "She won it" it could.

To summarize, we have seen the following kinds of knowledge being used to decide reference:

1) Descriptive
2) Recency
3) Syntactic rules
4) Selectional restrictions

But these four are not capable of deciding the referent for (3.1) - (3.3) so we need to add on:

5) Other Contextual Information

Exactly what is subsumed under this heading is not clear. It will certainly include the effect of multiple step deductions, which is the topic of this chapter. (Note 2)

3.2 Deciding Reference Requires Complex Deductions

In many cases we need to understand a lot of the story to get the references right. But understanding is not a monolithic thing. One can understand more, or understand less. Then how much 'understanding' do we need? What do we need to understand? In particular, are the facts which need to be understood to solve reference problems the same as those we will find in our attempt to "fill in the slots" of the story? Again let us consider example (3.1) repeated here as (3.9):

(3.9)
1) Today was Jack's birthday.
2) Penny and Janet went to the store.
3) They were going to get presents.
4) Janet decided to get a top.
5) "Don't do that" said Penny.
6) "Jack has a top.
7) He will make you take it back."

A good guess at the rule which deduces that "it" refers to the "present top" might be:

(3.10) Since Jack may not want a top he may have it returned to the store

(There are other candidates for the "rule" operational in (3.9). We will discuss one other in section 3.4). If our model were "expecting" such a situation to take place, this expectation would lead to assigning "it" to the present Janet is thinking of getting. So we need to understand the story well enough to apply (3.10). But (3.10) assumes that we know that Jack may not want a top. How do we know that? The story

never said so. (That is, no single previous sentence, when taken out of context, would imply that Jack may not want a top.) Presumably we know that in the context of buying an X as a present for someone, if he has an X then he may not want another X. But how do we know that Janet intends to buy a top for Jack? Again the story never said so. We can carry this chain of argument back till we get to the first line of the story. When we are done we will need roughly the following deductions (listed by the line number when the deductions should be made):

(3.11) Line 3: Realize the present is intended for Jack.

Line 4: The top would be Janet's present for Jack.

Line 6: That Jack already has a top suggests that he might not want another. That he might not want another suggests that Janet should not get him one

Line 7: The reason he might make Janet return the top is that he doesn't want another one.

Note that every one of these facts will be needed to answer questions like (3.4) - (3.6). If our model is going to answer such questions "on the fly" then the deductions like those in (3.11) must be made and left in the data base for future use.

Or look at the "party" example (3.2). In that case we might ask "Do you think Janet was disappointed?". In order to answer that question we would have to combine Janet's wanting to go to the party with Mother's comment that Janet had not been invited. Intuitively at least, it is an interaction like

that which leads to our understanding of "she" as Janet in (3.2).

As for "how much understanding" we need to determine the referent, in the "top" example most of the crucial points of the story as listed in (3.11) are needed in helping to determine the referent. Naturally in longer stories we would not expect every line to come into play. Even in the "top" story we could delete the second line and still get the reference correct. However I see no way to delimit which information in a story can be used to determine a particular referent.

3.3 Using Common Sense Knowledge in Reference Problems

In this section we will sketch a method for using multiple step deductions based on common sense knowledge in the reference decision. Let us suppose that when we initially encounter a noun phrase we apply definitional, syntactic and selectional information to get a quick list of which objects in the story might be the referent. (For pronouns we will only consider "recently mentioned" objects, in which case we will be using recency information also.) We will call the objects found "possible referents" (the list of them is called the PRL), and we will call the process Finding Possible Referents (FPR). We are interested in the case where there is more than one possible referent.

I have been assuming that the English sentence gets translated into some internal represenation. So we might have:

(3.12) Jack's top is on the table.

(3.13) (ON TOP2 TABLE1)

(Only the major assertion of (3.12) is represented in (3.13).)

Presumably in (3.12) we know that TOP2 belongs to Jack. If we further assume that we know of only one top which belongs to Jack, by the time we are done with definitional information we are able to reduce the number of possible referents to one. We then take the one remaining item, TOP2, and place it in the proper position in (3.13). We need to go

through a similar analysis for "the table".

A logical extension to the case where we have more than one possible referent would be to put in a "variable" in the proper position in the assertion. We will indicate a variable with the prefix ? so ?X is the variable X. (As necessary, we will introduce special syntax, usually based on Micro-Planner (Sussman, et. al. 72).) In the case where Jack had more than one top, example (3.12) would become:

(3.14) (ON ?X TABLE1)

Now ?X cannot be any possible object; we want to limit it to the tops which belong to Jack. We will say that ?X is "restricted" to the possible referents already determined by FPR. Restrictions on variables are checked before the variable is assigned a value, to insure that the value is consistent with the restrictions.

Once we have placed the restricted variable in the assertion we will go on and do our various deductions as if we knew the objects being talked about. That is, we finish the processing of the sentence with the variable "replacing" the undecided noun phrase (NP). There is one obvious limitation, however. We cannot allow ourselves to ask directly "What is ?X ?", since we don't know the answer to that question. Now this seems like a very serious limitation, and it might appear that we have tried to make the problem go away by legislating it out of existence. But the key phrase is "ask directly".

As we shall now see, there are ways to " ask indirectly" about ?X.

We saw earlier that, even ignoring the problem of reference, in order to answer the question "Why would Janet have to return the top?" we would need a rule of deduction something like (3.10). That is, something which says, if a person doesn't want a present it might be returned to the store where it was bought. Rephrasing this slightly we get:

(3.15) If we see that a person P might not like a present X, then look for X being returned to the store where it was bought. If we see this happening, or even being suggested, assert that the reason why is that P does not like X.

The phrase "look for (something happening)" in (3.15) suggests that it is a demon.

What might it mean for (3.15) to see the activity it is looking for? In particular, (3.15) is looking for X being returned to the store. Instead, in (3.9), we get the line "Jack will make you take it back". Now, the internal representation for "returning an object" and "taking an object back" will presumably be the same since they mean the same thing. However, even assuming this much, there will still be a major difference between what we are looking for and what we find. We are looking for the present (i.e., the top Janet is thinking of getting) being returned. We are told that "it" will be returned where we are not sure what "it" is, except that it is restricted to the two tops. The logical thing to

do would be to let the demon choose TOP2 (the potential present) since that is the one which it is expecting. This is exactly what we do, and by choosing TOP2 we get the correct referent for the "it". Now in some sense our demon did ask "What is ?X ?". It was told that the variable did not have a value as of yet but that either TOP1 or TOP2 would be acceptable. (Also note that by the same process we get "it" being returned to the store, though the original sentence did not state where the object was to be returned.)

To summarize briefly, when we encounter a noun phrase we immediately apply FPR, which uses definitional, syntactic, selectional, and sometimes recency information. If this is not sufficient to determine the referent uniquely we put a variable in the place where the internal symbol for the object would go if we only knew what it was. The variable will be restricted to the list of possible referents. If the variable is ever matched against one of the possibilities, it will be assigned that value. (If more than one possibility matches we can just further restrict the possibility list.) This happens, for example, when the current line matches a demon. In the process we will permanently assign the variable to its new value, and hence assign the noun phrase its referent. If the variable is never assigned, we will finally fall back on the "last mentioned rule" (LMR) to pick the referent. (Later, in 5.1 we will see that base routines can also assign

referents. Still later, in 6.5, we will argue that in special cases even bookkeeping comes into play.)

3.4 Other Ways to Use Context in Reference Determination

In the last section we outlined, in a general way, a method of using context to help decide referents. Referring to our method of 3.3 as the "restriction method" we will now consider, and reject, a few alternatives to the restriction method. Naturally, the few methods discussed do not exhaust the possibilities. Nor do the alternatives rejected here correspond to any published proposals for applying contextual information to reference, since, as far as I know, I am the first to present such a scheme. Rather, these alternatives are attempts to formalize partially schemes which have been suggested during discussions of my work with others.

A Backup Scheme for Choosing Referents

At a glance the "last mentioned rule" is a pretty good rule. If we apply our early referent analysis to get a list of possible referents, and then pick the one most recently mentioned, probably ninety percent of the time we will get the correct referent. Perhaps rather than go through the bother of creating a dummy variable, and hoping that eventually it gets assigned to an object, we could assume that the last mentioned object is the correct one unless it somehow gets us into trouble. (Note 3)

This scheme can be characterized as a "backup" method since it assumes one possibility until that is proved

incorrect. Since all backup schemes need a "plausible move generator" to decide in which order the possibilities will be tried, our proposal is characterized by a "recency" method for picking "plausible moves". Other "move generators" will be discussed later.

There are several reasons why a backup approach might be the wrong way to do things. One obvious reason is that the number of possibilities necessary to try would be so large as to be prohibitive. However, since we are seldom left with more that three or four possible referents after early referent analysis, this reason does not apply. A really serious problem in using backup is finding a way to decide when one possibility has failed and we should try the next.

The difficulty will arise because in the backup method, when we decide that a particular referent is "bad", we must do so without looking at the other possible referents, and in particular without comparing the chosen referent with the other possibilities. So, for example, we would not be able to do something like "pick the referent which 'fits' best" since that requires comparison.

To get some idea of how a backup scheme might work, let us again consider our "top" story (3.1). Let us suppose that our system had a rule which went:

(3.16) If an object X is returned to the store then it must just have been purchased.

Presumably when we try Jack's top as the referent of "it" we would reject this decision on the basis of (3.16). Now it seems to me that there are serious difficulties with this analysis. Our particular story did not specify that "it" would be returned to the store. Furthermore, while we do not know that Jack's top had just been purchased, it is not clear that that is sufficient reason for believing that it wasn't. While I believe that these are valid objections, we can better illustrate our basic point, that it is necessary to compare referents to decide on the correct one, with another example.

(3.17) Mother made some cookies and left one out on a plate. She put the plate on the kitchen table, and went into the living room. "I am sure Janet will like it" thought Mother.

On the basis of recency, the list of possiblities, for the last "it", in order of preference, would be:

(i) the living room
(ii) the kitchen table
(iii) the plate
(iv) (the) one cookie

The problem is deciding that "it" does not refer to (i), (ii), or (iii). Since the backup method requires that we "get into trouble" in order to eliminate the preferred (i.e. "last mentioned") referent, the problem boils down to determining what in (i), (ii) or (iii) will cause trouble.

There are many possible facts which could lead us to the decision that the "it" refers to the cookies. For example:

(a) Children are more likely to "like" cookies than the other objects listed

(b) People are more likely to be concerned about whether others like things which they have had a hand in making
(c) Since Janet is presumably aware of the other things in the list, the cookies are the only things about which Janet must form a new opinion.

But none of these is absolute in the sense that if disobeyed the story would be nonsense. So rule (c) which is, in my opinion, the most likely explanation, would not prevent Mother from saying "I am sure Janet will like this dish." We could easily imagine mother going on to say "I bought it just for her." Rather, each of these rules simply says, in effect, "If I had the choice, I would pick ..." But this means that the rules must be able to compare possible referents, and we have already seen that backup schemes do not allow for such comparison.

We have seen other cases earlier in which one referent simply fits better, (3.2) and (3.3). There is no reason why Mother could not be the person "not invited" in example (3.2). Nor, of course, does it suffice to say that "Mother's not being invited doesn't make any sense since we were talking about Penny going." What matters is not that Mother's not being invited would make no sense, it is that Penny's not being invited would make more sense. Indeed, if we had been told, "Penny was told that Mother had not been invited," we would have understood as best we could. Perhaps we would have waited for a line like "Penny knew that Mother would never let her go alone." We might try to handle (3.2) and (3.3) by

postulating a rule which rejects "mother" as the referent since the sentence with her as referent does not "advance" the story. However, this is quite dangerous since in many cases ((3.7), (3.8)) neither referent advances the story, and such a rule would then lead to rejecting all the referents.

So far we have formulated our backup scheme with the preferred referent being the last mentioned referent. There are other possibilities. One interesting one is to have the program define an object, or set of objects, as "topic" or "important objects". These objects would be tried first by our backup scheme. Such a proposal would seem to help in (3.17) where we could argue that the story was "about" the cookie Mother left for Janet. This referent would then be tried first and it would go through without any problems. There is, of course, the problem of deciding that the story is about the cookie, but even ignoring this, our latest proposal has seemingly fatal problems.

(3.18) Mother was baking a cake. It would be done in an hour, at which point Mother would put the cake on a plate to cool. Meanwhile, she went into the living room and sat down in the easy chair. "I am sure Janet will like it" thought Mother.

(3.19) Mother was baking a cake. It would be done in an hour, at which point Mother would put the cake on a plate to cool. Meanwhile, she went into the living room and sat down in the easy chair. It collapsed.

While in (3.18) the "it" is understood as "the cake", in (3.19) "it" is "the easy chair". If the story is "about" the cake (as would be necessary to explain (3.18)) then we should

understand "it" as being the cake in (3.19) also.

Or again, consider a modification of (3.1):

(3.20) Today was Jack's birthday. Penny and Janet went to the store. They were going to get presents. Janet decided to get a top. "Don't do that" said Penny. "Jack has a top. It is green."

If in accounting for (3.1) we had "the present top" as the topic of the story, our topic rule should give us "the present top" as the "it" in (3.20). In fact, we understand the last "it" as referring to Jack's top.

So while "topic" seems to help "backup" methods, it does not seem to help enough. This does not mean that the concept "topic" is useless in general. Indeed a more complete theory will surely include it. It just is not sufficient to enable "backup" methods to work.

A Breadth First Method

The idea behind the restriction method was that since a demon represented a potentially relevant fact, it should be allowed to assign a referent to an undecided noun phrase. However, it is quite possible to have several demons looking for similar patterns, in which case only one of them (the most recently added) will get to select the referent, since by the time the others get to the assertion the NP will no longer be undecided.

Our breadth first method removes the asymmetry between the first demon to apply and all the rest. In this method we

go through DSP once for each possible referent, keeping the facts learned in each case separate. That is, we process the sentence several times, each time assuming a different referent for the undecided noun phrase. (We can think of the different referents being tried in parallel.) The referent which allows the most causal links between the current sentence and previous facts is the one selected and the facts generated by the other referents are to be discarded. The idea here is simple. We have been suggesting that the referent we want is the one which "fits" best. That might be the referent which allowsthe most links to what has already happened.

Such a scheme would seemingly require much more computation than the restriction method, especially when we consider what happens when more than one referent is left undecided by early referent analysis. So, unless we had strong evidence that it were necessary we would not want to implement it. I have no stories which indicate that this method is needed.

The reader may then wonder then why I even brought the topic up. The reason is that there are several potential problems associated with the restriction method which do not apply to the breadth first method.

First, suppose we are executing a demon or base routine and we come to some conditional where the alternatives are

equally likely. For example, part of TRICK-BASE might ask "is the animate object which is going to do the 'trick' a person or animal?" (It helps to know in order to decide what constitutes a trick.) If we do not know who the animate object is, we may have no way to answer. On the other hand, in the breadth first method we will "know" who the animate object is, even though later we will throw out all the possibilities but one. While this is a problem for the restriction method, in chapter 7 we will discuss a possible solution which involves only a minor change in the model.

The second potential problem deals with the restriction method's asymmetry between first applied demons and later demons. One can easily imagine that at certain points in a story some demons will be more "important" than others. Perhaps they will explain more of the events of the story. At the same time, there may be other, "minor", demons also asserted. Further, let us assume that some of the minor demons would not assign the same referent to a particular NP as the major one would. In the breadth first method the minor demons would be overruled by the major one. In the restriction method however, we could have the stiuation where a "minor" demon picks up an assertion and assigns a referent so that the major one would not even apply. I have not constructed an example of this sort since by and large when analyzing a story I have only considered the most important

demons which could apply. Only when we are more advanced, and have also included "minor" demons, might we encounter this situation. Hence, while my personal opinion is that the breadth first method will not prove to be necessary, it is still a possibility.

3.5 Demons and Details

Section 3.3 gave the basic ideas about getting context into reference determination. In this section we want to elaborate the idea of demons. Demons, as used in this paper stem from the programming language Planner, designed by Carl Hewitt (Hewitt 69).

Let us construct a demon to account for the following example:

(3.21) Janet wanted a nickel. Her piggy bank was in her room, so she went there and got it. "I will get that nickel" she said. She shook the piggy bank very hard. Finally it came out.

Again we note that we will need a demon to answer the questions "Why did the money fall out?" and "Did Janet get the money?". (What follows is a very simplified example. For more detail see chapter 5.)

Now the incoming sentence "It came out" might look like:

(3.22) (GO ?IT) ----> (OUT-OF ?IT ?UNMENTIONED)

Actually, all assertions have "names", so in (3.22) we will point from the "go" assertion to the "out of" assertion by mentioning the latter's "name". However, it is easier to go through the example if we ignore the names. ?IT is a variable restricted to the two possible referents NICKEL1 and PIGGY-BANK1. The ?UNMENTIONED is an unrestricted variable which represents the answer to the question "came out of what?". The demon which picks up (3.22) is outlined on the

next page.

Now (3.22) will activate this demon since it matches the pattern of PB-OUT-OF. Compare:

```
(OUT-OF ?IT ?UNMENTIONED)
(OUT-OF ?M ?PB)
```

During this match ?IT and ?M are linked up (that is, anything which happens to ?M will also happen to ?IT), as are ?UNMENTIONED and ?PB. When ?IT and ?M are linked, the variable ?M also becomes restricted to the possible referents for "it", NICKEL1 and PIGGY-BANK1. When we check (in the second box) that ?M is some form of money, we will be looking for objects which are labeled MONEY. In the data base we will have the assertion:

```
(IS NICKEL1 MONEY)
```

which is to match the pattern:

```
(IS ?M MONEY)
```

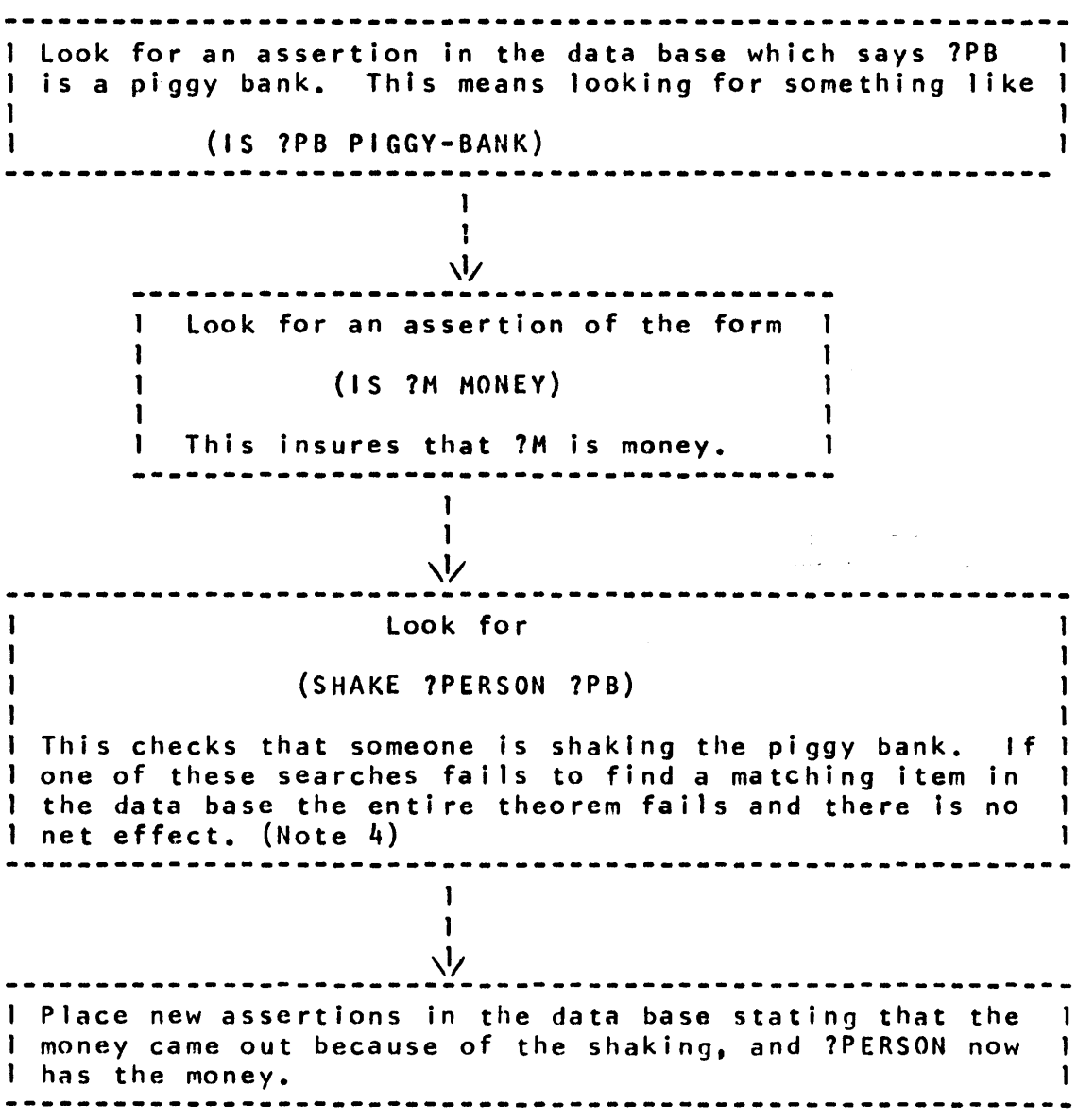
This will cause ?M to match NICKEL1. This value will be checked against the restriction on ?M which came from ?IT via the linking mentioned earlier. Since NICKEL1 is an acceptable referent, the match will go through. When the theorem has been executed ?IT will remain assigned to NICKEL1 even though the variable ?M will no longer "exist". In this way we decide that "it" in example (3.21) must refer to the money.

PB-OUT-OF

This is a demon called PB-OUT-OF.
The local variables are (M PB PERSON)

(OUT-OF ?M ?PB)

This is the pattern of the theorem. Since ?M and ?PB have no value at this point the pattern will match any assertion about one object coming out of another.



4 A Few Details

In previous chapters we have presented basic ideas, but suppressed details about the representation of facts, etc. In this chapter we will specify some details needed in the rest of the thesis. For brevity I have eliminated justifications for most decisions. Often alternative methods seemed quite reasonable, and some will be discussed in chapters 8 and 9.

4.1 The Internal Representation

We represent a statement like

(4.1) Bill got the ball before he went to the park

by several assertions:

(4.2)
```
(N1 BEFORE N2 N3)
(N2 GET BILL1 BALL3)
(N3 GO BILL1 PARK1)
```

"N1", "N2" etc are "names" (assertion numbers) for cross reference of assertions. In a given story each name will be associated with only one assertion, but I will not attempt to keep the names unique in the thesis.

We need to represent the tense of a statement, which we will assume to be either past (PAST), present (PR), or future (FUT). Naturally there are more tenses than these three, but they will do for the stories we will be working with. The type of sentence may be imperative (IMPERATIVE), question (QU), or statement (ST). We will put the necessary markers right after the predicate in the assertion, so we will have:

(4.3) Jack is at the park.
```
(N4 AT PR ST JACK1 PARK1)
```

(4.4) Did Jack go to the park?
```
(N5 GO PAST QU JACK1 PARK1)
```

(4.5) Will Jack go to the park?
```
(N6 GO FUT QU JACK1 PARK1)
```

(We will discuss the representation of imperatives on the next page.)

Except for chapter 9, we will only be concerned with statements, as opposed to imperatives or questions. Also, most of the statements will be in present tense. Hence, in the interest of readability, we will suppress all PR and ST markers. So (4.3) will look like:

(4.6) (N7 AT JACK1 PARK1)

We mentioned in 2.3 that future tense statements were to be considered "possible futures" in that associated with the concepts introduced by the possible future would be facts which might interpret later parts of the story. We also mentioned that certain other types of statements should have the same effect. It doesn't make very much difference, from the standpoint of what we are looking for, whether we say:

(4.7) Janet said, "I { can / should / will / must / want to } bake a cake."

This suggests that we represent statements like these in such a way that they have a common core which expresses the "bake a cakeness" of the statement. So we will have

(N8 WANT JANET1 N9)
(N9 BAKE FUT JANET1 CAKE1)

Now, suppose Jack and Janet are outside, and they decide to pretend they are eating supper. If Janet says

(4.8) "Jack, get a plate"

we would want to interpret Jack's going into the house as part of getting the plate. Hence (4.8) and other imperatives act

like possible futures. For this reason we will represent (4.8) as:

```
(N10 IMPERATIVE N11)
(N11 GET FUT JACK1 PLATE1)
```

Negation will be indicated by a NOT in the assertion. So we get:

(4.9) Jack was not in the house

```
(N12 IN NOT JACK1 HOUSE1)
```

Besides singular specific objects like HOUSE1, we need plural objects like "the pencils" which constitute a set of unknown number, and "non-specific" objects such as "a ball", as in "Jack wanted a ball", which may refer not to any particular ball, but to any one of the set of all balls. Finally we will allow definite sets where we know all the members such as "The ball and the top" in:

(4.10) "Jack was holding the ball and the top."

```
(N12 HOLD JACK1 DGROUP1)
(N13 MEM TOP1 DGROUP1)
(N14 MEM BALL1 DGROUP1)
```

We will call such groups DGROUPS for Definite GROUPS. (Note 5)

So far we have only discussed the gross structure of the internal representation. There are many fine points which will have to be decided. For example, how would we represent

(4.11) Jack put the money in the piggy bank.

Some possibilities are:

(N15 PUT-IN JACK1 MONEY1 PB1)

(N16 PUT JACK1 N17)
(N17 IN MONEY1 PB1)

(N18 PUT IN JACK1 MONEY1 PB1)

There are many decisions of this kind which need to be made. I will try to explain my choice whenever a new representation is introduced. I do this because I believe that representation is best discussed in the context of how it is going to be used, and my chief concern usually is with what makes deduction easiest. There have, however, been several more abstract studies of possible internal representations for English, see (Sanderval 72), (Shank 69) and (Simmons 70a).

4.2 What Demons Look Like

At the very end of section 3.5 we used a simplified diagram to described a demon which connected money coming out of a piggy bank with the person who shook the PB now having the money. In this section we give a more detailed notation.

The basic form of a demon is:

```
(DEMON  (demon's name)
        (list of variables)
        (pattern the demon is looking for)
        (e1)
        (e2)
        .       Program to be run if the
        .       proper assertion is found
        .
        (en))
```

So our PB-OUT-OF demons would have the outline

```
(DEMON  PB-OUT-OF
        ((variables))
        (?N OUT-OF ?M ?PB)
        .
        .
        .
        )
```

To fill in the rest of the demon we will need two primitives, GOAL and ASSERT.

The Primitive GOAL

GOAL, as used in this thesis corresponds to THGOAL in Micro Planner. (Those who know Micro Planner, however, should still note the names given to GOAL specifications, as mentioned in the second half of this sub-section, since I do not use standard Micro Planner notation.) GOAL is primarily

an information-obtaining primitive. So for example:

(GOAL (IN JACK HOUSE))

asks if the assertion (IN JACK HOUSE) is in the data base. If the answer is yes, the next line of the demon will be executed; if no, then, roughly speaking, the demon will "fail" and the system will act as if the demon had never been run. If we just want to know if Jack is in anything we would ask:

(GOAL (IN JACK ?))

Anything will match a "?" so any assertion which has IN in the first position, JACK in the second, and anything at all in the third will be retrieved. (If several matches are possible, the first one found will be used, and the others "saved". The purpose of "saving" the rest will be described later.) If there still is nothing which satisfies the pattern the GOAL will fail as in the first case. A more useful form uses a variable instead of a "?", as in:

(GOAL (IN JACK ?X))

Variables are allowed to be "unassigned" in which case ?X would match anything, and in the process get assigned to the object it matched. Hence if the last example matched (IN JACK BAR) ?X from that point on would have the value BAR. However, if ?X already had the value STORE our pattern would only match (IN JACK STORE).

It is possible to add further specifications to the GOAL command. We will only be interested in two particular

specifications. We can add restrictions (or filters) to what kind of assertions we want to match our pattern. Usually these filters will refer to the property list of the assertions. For example, we will often use filters that say we are only interested in assertions which are currently true. That is, we don't want any updated or otherwise untrue assertions. Since the TROUBLE tag is placed on the property list of assertions which are no longer true, our filter will simply look for this tag. We will indicate this filter by a $TRUE following the pattern in the GOAL statement. So we would have:

(GOAL (IN ?P HOUSE) $TRUE)

Another modification of the GOAL statement is to tell it to use fact finders (consequent theorems) in trying to establish the GOAL. While in general it is possible to tell GOAL which or what kind of fact finders should be used, we will always request all fact finders whose pattern matches the GOAL pattern. We indicate this by including a $DEDUCE in the goal statement. So if we wanted to establish that Jack is currently in the house, we might ask:

(GOAL (? IN JACK HOUSE) $TRUE $DEDUCE)

In this example the first "?" says that we don't care what assertion number the assertion has. Note that a GOAL statement may have side effects. For example, a fact finder called by the GOAL may put new assertions in the data base.

The Primitive ASSERT

The other needed primitive is ASSERT which is somewhat like THASSERT in Micro Planner. Like THASSERT, ASSERT adds a new assertion to the data base. Unlike THASSERT, it also puts the assertion on the TO-BE-DONE list, which is a list of assertions which still have to be processed by DSP. The exact role of the TO-BE-DONE list will be given in the next section.

The other difference between ASSERT and THASSERT is that since in most cases when we assert a new assertion we will not have chosen a particular "name" for it (like N38), we will let ASSERT make up a name. So if we have:

```
(ASSERT (? IN JACK HOUSE))
```

ASSERT will interpret the initial "?" as meaning that it should give a name to the assertion.

Failure and Backtracking

At this point we can fill in the demons which we initially started with. It will look like

```
(DEMON PB-OUT-OF
        (NOLD PB PERSON M N)
        (?N OUT-OF ?M ?PB)
        (GOAL (? IS ?PB PIGGY-BANK))
        (GOAL (? IS ?M MONEY)$DEDUCE)
        (GOAL (?NOLD SHAKE ?PERSON ?PB)$TRUE)
        (ASSERT (? HAVE ?PERSON ?M))
        (ASSERT (? RESULT ?N  ?NOLD)))
```

Earlier we said that should a GOAL fail, the entire demon fails. A little closer look reveals that this is not exactly

the behavior we want. Suppose we have two piggy banks, PB1 and PB2, and it is PB2 which is the one which Jack is shaking. Furthermore, suppose that the first PB to be found by:

(GOAL (? IS ?PB PIGGY-BANK))

is PB1. Our third GOAL:

(GOAL (? NOLD SHAKE ?PERSON ?PB))

will then fail. Rather than having the entire demon fail, we will simply go back to the first GOAL and try the other possibility, PB2. That is to say, when a failure occurs, the program is backed up to the last choice point (in the sense that the program chose between PB1 and PB2) and then trys again. In Planner or Micro Planner this backup will go into previously called sub programs and sometimes into the program which called the program with the failure. Such capabilities are needed for the deductions in children's stories, in particular when we use a chain of fact finders to establish a GOAL. However, in the thesis we will not discuss any cases where it is necessary to take this into account.

Specification of Demons

Let us take a closer look at one of the lines in PB-OUT-OF:

(GOAL (?NOLD SHAKE ?PERSON ?PB))

In the demon this line has the effect of checking that the piggy bank which the money is coming out of is the one which

was shaken. It also binds the variable ?PERSON to the person who shook the PB so that in a later line we will know who it is that now has the money. However, this demon will not be put in until we see the "shake PB" assertion. Hence at the time the demon is put in we know which PB we are interested in, and who the person is who will get the money in case it comes out. So at the time the demon is put in we will pass on this information directly in the demon. The way we will do it is by "specifying" the variables in the demon. So just before we put the demon in we will change the demon so that ?PERSON will be bound to JACK1 (assuming he is the one who is shaking the PB), etc. The net result will be:

```
(DEMON PB-OUT-OF1
        ((NOLD 'N59) (PB 'PB1) (PERSON 'JACK1) M N)
        (?N OUT-OF ?M ?PB)
        (GOAL (? IS ?M MONEY))
        (ASSERT (? HAVE ?PERSON ?M))
        (ASSERT (? RESULT ?N  ?NOLD)))
```

The variable declaration list now says that ?PB is bound to PB1, ?PERSON to JACK1 and NOLD to N59, which we will assume is the assertion number of the "shake" assertion. (In all demons the variable ?NOLD will be bound to the assertion number of the assertion which caused the demon to be asserted.) Note that by specifying these variables the demon can dispense with two of the GOAL statements which were in the original formulation, namely the one which would have checked that ?PB is a piggy bank, and the one which would have looked for the "shake" assertion.

Also note the change in the name of the assertion. It now has an index after the name. The reason is that it is quite possible that we could need the same demon for two different situations, where the people and objects are different. Hence when we specify the demon we are really creating a new instance of the demon, which is given a unique indexed name.

There are two primitives which put in demons, D-ASSERT (Demon-ASSERT) and LOOK-BACK. They both have the same syntax.

```
(D-ASSERT (PB-OUT-OF 'N59 'PB1 'JACK1))
```

In general we can have more than one demon asserted by a single D-ASSERT statement. For each demon we need to specify the name, and a list of variable specifications. D-ASSERT just asserts the demon and binds its variables. LOOK-BACK does this plus applies the demon to the information in the data base (see section 2.4).

4.3 Summary of Flow of Control

In chapter 2 we introduced four parts of DSP. We did not explain in what order we apply these parts when we get a new assertion. Actually we only need order three of the four sections, since fact finders are called when one of the other three sections wants to prove a given fact, not whenever a new assertion comes in.

1) Apply internal translation to the sentence to get it in assertion format. This will include FPR as mentioned in chapter 3.

2) Place all assertions on the TO-BE-DONE list and in data base.

3) Take each entry on TO-BE DONE in turn and apply first the demons, and then base routines.

4) At any point in (3), if a new assertion is generated, put it at the end of TO-BE-DONE.

5) When (3) and (4) are finished, begin TO-BE-DONE again, and apply bookkeeping to each assertion.

5 Piggybanks and the Problem of Formalizing Knowledge

As we have seen, it is necessary for us to take our everyday knowledge, and put it in a form which can be used by the machine. In this chapter we will study one topic in great detail. The subject is piggy banks.

I will be trying to do two things at once in the chapter. The primary goal is to articulate some parts of a complex body of information on a given topic. At the same time, I will try to give some idea of the kinds of problems one encounters while formalizing a body of knowledge. The second emphasis is the development of several ideas based on examples drawn from piggybanks.

5.1 Getting Our Feet Wet

We start out just writing down various things we know about piggy banks (PB's henceforth).

> PB's come in all sizes and shapes, though a preferred shape is that of the pig. Generally the size will range from larger than a doorknob, to smaller than a bread box. Generally money is kept in PBs, so when a child needs money he will often look for his PB. Usually, to get money out you need to be holding the bank, and shake it (up and down). Generally holding it upside down makes things easier. There are less known techniques, like using a knife to help get the money out. If, when shaken, there is no sound from inside, it usually means that there is no money in the bank. If there is a sound it means that something is in there, presumably money. You shake it until the money comes out. We assume that after the money comes out it is held by the person shaking, unless we are told differently. If not enough comes out you keep shaking until you either have enough money, or no more sound is made by the shaking (i.e., the bank is empty). In general the heaver the PB the more money in it. Some piggy banks have lids which can be easily removed to get the money out. Sometimes it is necessary to smash the PB to get the money out. To put money in, you need to have the money and the bank. The money is put into the slot in the bank, at which point you no longer are directly holding the money. Money is stored in PBs for safe keeping. Often the money is kept there during the process of saving in order to buy something one wants. PBs are considered toys, and hence can be owned by children. This ownership extends to the money inside. So, for example, it is considered bad form to use money in another child's PB. Also, a PB can be played with in the same way as, say, toy soldiers, i.e., pushed around while pretending it is alive and doing something.

Though the above is not exactly a short list, it is not all that long either, and surprisingly enough it contains almost everything I can think of about piggy banks. There are some

facts which pertain to PBs which are not included in the above list because they are really facts about much broader classifications. For example, we could say that one can throw a PB but this applies to small objects in general, and would most likely appear in a discussion of "throw".

PB as Container

On the other hand, some of the facts listed above could be generalized somewhat from PBs. The trick of shaking a PB to find out if anything is in it applies to other containers also. In treating such cases I have ignored the more general aspects of the problem and instead acted as if the facts were local to PB. My belief is that the best way to proceed in these cases is by treating individual containers, say, separately and only trying to combine the facts after we have had experience with several specific examples.

Of course, we cannot use the information in the form it has on the previous page It is not at all clear how our system would use a fact like "Money is stored in PBs for safe keeping." The basic problem then is to take such information and get it into a more usable form. We will do this by presenting very short story fragments along with a question, which together illustrate the fact being examined. We will then propose a way to answer the question. As we shall see, often we will have to go back and revise our solutions when

later examples are considered.

What I am proposing then is to study special cases, particular stories, particular containers. This leaves many questions unanswered. Which are the best containers to study? How can we be sure we have all the facts about topic X? To answer such questions one would need a theory of the structure of knowledge, and neither I nor anyone else has one. My belief is that the only way to obtain such a theory is to plow ahead, as we shall do in this chapter.

Allowable Assertions

Before we can get started on this program we should say a word about what kinds of assertions we will allow to be used in describing our facts about piggybanks. Naturally we will need some standard ones like:

IN	also other location assertions
HAVE	A has B
RESULT	A is the result of B
SHAKE	A is shaking the object B

However, note that SHAKE is a composite activity, consisting of many smaller motions. We might have the policy of trying to replace composite activities with their subactions, though in the case of shake there would be little point since its subactions have no significance by themselves. On the other hand, do we want to allow a composite activity like:

(5.1) (N6 GET-FROM FUT JANET1 PIGGY-BANK1 MONEY1)

This says that there is a possible future with Janet getting

money from the PB. I will allow (5.1) since it and other composite activities embody my belief that a story (or real life) is not just a linear string of actions. Rather those actions are structured in many ways. One particular way is that actions are clumped together into larger activities. So, getting the PB, shaking it, listening to see if there is anything in it, grabbing the money as it falls out, are all part of trying to get money out of the PB.

Furthermore, not having composite activities would complicate the process of answering questions about composite activities such as "Why did Janet get money from her PB?". We are not interested in why Janet moved the PB upwards, then downwards, or why she turned the PB upside down. We want to know why she did all of these things. At best we would be forced to have a separate RESULT assertion linking each activity to Janet's desire to get money from the PB. At worst we might have trouble screening out "noise" such as "the reason Janet shook the PB was that she could not find a knife to use to get the money out".

5.2 LOOK-BACK and Generalizations on Demons

LOOk-BACK

Suppose we were given:

(5.2) Janet needed money. She got her piggy bank.

Even at this point, were we to be asked what Janet is going to do with the PB we would still say, "get money from it." This suggests that when we enter our PB base routine we want to check if the actor needs money, and if so, assert that the intention here is to get money from the PB.

However, we could have the story go like:

(5.3) Janet got her PB. "I want a nickel" she said.

In (5.3) we don't find out about the need for money until after the statement concerning the PB. This implies that PB-BASE will have to put in a demon looking for "want money" if it can't establish it already. The only alternative would be for "want money" to look for "getting a PB". This seems to be less reasonable, since there are many ways of getting money, but only a very limited number of reasons a person gets a PB. Besides, unless we use a demon we must place the information under both PB and "want money". With the demon it need only appear once.

This demon would look something like this:

```
(DEMON PB-NEED-MONEY
   (NOLD PERSON PB N1 MONEY N)
```

NOLD is specified to the assertion number of the "get PB" assertion, which is the assertion which caused this demon to be asserted. PERSON is specified to the person who got it. PB is specified to the piggy bank.

```
   (?N HAVE FUT ?PERSON ?MONEY)
```

The demon is looking for a possible future with the person "having" money.

```
   (GOAL (? IS ?MONEY MONEY))
```

Make sure that what the person needs is money.

```
   (ASSERT (?N1 GET-FROM FUT ?PERSON ?PB ?MONEY))
   (ASSERT (?  SUB-ACT ?N1 ?N))
```

Says that the "get-from" assertion is a sub action of wanting to get money.

```
   (ASSERT (? SUB-ACT ?NOLD ?N1)))
```

And getting the PB is a subaction of getting money from it.

To account for (5.3) we suggested that PB look for the person wanting to get money, and then assert statements about how getting the PB fitted into his goal. This is exactly what this demon does, except that the demon waits for the "want" statement to be made, whereas the PB routine will look back to see if we already know the fact. We can collapse these two activities into one by using LOOK-BACK (see sections 2.3 and 4.2). LOOK-BACK not only puts the demon in, but also looks back in the data base to seeif it already contains an assertion which matches the demon's pattern. If there is a

match in the data base, then this demon is applied to the assertion.

A Second Demon

So far we have written one demon which captures a small part of the relationship between piggy banks and money. Let us go on to another.

(5.4) Janet shook her piggy bank. Finally she got a nickel.

We could ask several questions

(5.5) How did Janet get the nickel?
Where was the nickel before Janet had it?
Is the nickel in the piggy bank now?

The last question is not trivial, even if we assume that we have some way to place the nickel in the PB before Janet got it. The problem is that usually we say that a person has (or even "is holding") an object if he is holding something which contains the object. So for example, if Jack is holding a box and in the box is a kitten, were we asked "Is Jack holding a kitten?" we would say "Yes." So when we are told that Janet got the nickel, it is by no means obvious that this implies that the nickel is not in the PB. The conclusion must be that we know that when dealing with piggy banks, "getting" the object means getting it out of the piggy bank.

The following demon is proposed to handle such problems. It will be put into circulation when we first encounter "get PB". (Note that it assumes, as we have all along, that "get"

entails "have".)

```
(DEMON PB-HAVE-MONEY
   (NOLD PB PERSON N1 M N)
```

NOLD, PB, PERSON are specified as before.

```
   (?N HAVE ?PERSON ?M)
```

Look for the person currently having money.

```
   (GOAL (? IS ?M MONEY))
   (IF-NEED (?N1 GET-FROM FUT ?PERSON ?PB ?M))
   (IF-NEED (? IN PAST ?M ?PB)))
```

IF-NEED does an ASSERT provided the item is not already in the DB. So we are asserting (if we don't already know) that the person got the money from the PB.

```
   (IF-NEED (? IN NOT ?M ?PB))
```

Now assert that it no longer is in the PB.

```
   (ASSERT (? T-RESULT ?N ?N1))
```

Getting the money is a Trivial-RESULT of getting the money from the PB.

```
   (ASSERT (? SUB-ACT ?NOLD ?N1)))
```

Getting the PB is a sub action of getting money from the PB.

A Digression on T-RESULT

We have introduced a new predicate here: T-RESULT (Trivial-RESULT). When we say that A is a T-RESULT of B we mean that "by definition" B leads to A. To put this another way, there are some actions which are partially "defined" by their results, such as "trade", "find", "drop", "kill" "eat" "paint something a given color", etc. So, "the ball is red"

is a T-RESULT of "painting the ball red". T-RESULT differs from RESULT in one important way. If X is a T-RESULT of Y then X is not a T-RESULT of Z unless Z = Y. This means that if Jack got a certain object by finding it in the woods, he could not have gotten it from Mother. If we painted the ball red, it could not have become red for any other reason. Note that for RESULT this is not the case, i.e., a given action can be the result of several other actions. For example, in our opening story, when Janet offers Jack both the toy cat and her pencils for the paints she does it because she wants the paints, and because Jack won't trade for just the pencils.

Other possible evidence for the distinction between T-RESULT and RESULT comes from the distinction between "how" and "why" questions in English. Consider:

(5.6) How / *Why did Jack get the ball?

Bill gave it to him.

(5.7) How / *Why did Jack dispose of the grape?

He ate it.

In each of these cases the * before the "why" indicates that the answer given would not be acceptable if the question started with "why" rather than "how". My theory is that this is related to the distinction between T-RESULT and RESULT. However, "how" and "why" usage in English is sufficiently

complex that more work will have to be done before we can be sure. (Note 6)

Two Demons Into One

Returning to our demons, there seems to be some duplication of information in the two demons we have written. One recognizes why a person would get his PB, and the other recognizes the results of his action. We might write this as:

Need money -- Try to get money from PB -- Have money

We might note that such a paradigm occurs elsewhere also. So we have:

Want cookie --> Get from cookie jar --> Have cookie
Need new suit --> Go to tailors --> Have suit
Want the chairr red --> Paint the chair --> Chair is red
Want a cake --> Bake a cake --> Have a cake

What seems to be emerging is the fact that our two demons represent only one fact, "PBs are useful for getting money." This one fact is then used in two different ways, which are related by the idea of "goal oriented behavior". But our representation of this information as two separate demons would seem to indicate that there are two separate facts. Now we could combine the two demons, but this by itself will not necessarily capture the generalization we have in front of us. After all, we could just make one demon which consists of one very large COND, which acts like one demon if the tense is FUT, and the other demon if the tense is PR. What we really want is some "meta demon" format. Then into this format we

could stick "piggy banks are useful for getting money", or "painting an object is useful for making it a given color". Exactly how this is to be done is not clear, since it depends on our analysis of "paints", "baking", etc. However as a start towards such a theory we might try to consolidate our two demons in a natural fashon.

```
(DEMON PB-FOR-MONEY
   (NOLD PERSON PB MONEY N N1 TN N2)

        As before, this demon is put in by "get PB".

   (?N HAVE ?TN ?PERSON ?MONEY)

        Look for the person having had, having, or to
        be having, money (that is, tense is variable).

   (GOAL (? IS ?MONEY MONEY)$TRUE $DEDUCE)
   (IF-NEED (?N1 GET-FROM FUT ?PERSON ?PB
                          ($R ? (IS-OBJ '(MONEY)))))

        IS-OBJ and $R will be defined below.

   (COND ((EQUAL ?TN 'FUT)(ASSERT (? SUB-ACT ?N1 ?N)))

        EQUAL here is different from LISP EQUAL in
        that if the first argument is unassigned, it will
        be set to the second argument.  Here we are
        saying that if the person will be getting money,
        then "get from" is a sub-action of this desire.

         ((ASSERT (?N2 T-RESULT ?N ?N1)))))

        Otherwise getting the money is a direct
        result of "get from".
```

Note that this demon does not account for the fact that if the person now has the money it was previously in the PB. We could put the fact into PB-FOR-MONEY, though at the cost of separating the FUT and PR "versions" even further. (That is, we would make the one demon look more like two different

demons put into the same shell.) Instead we should probably give the question a lot more thought. For example we should note that roughly the same information seems to be involved in:

(5.8) Janet needed some money. She went to get her PB.

Question: Is there money in the PB?
Answer: Janet seems to think so.

IS-OBJ and $R

The expression ($R (variable) (predicate)) says that (variable) is to be restricted so that it only matches objects which satisfy (predicate). The restriction of variables mentioned in chapter 3, is done by using the same symbol.

As for the new primitive in our demon, IS-OBJ, it does the following:

When the pattern is used as a goal IS-OBJ restricts the variable (in this case the variable is simply a "?") to objects which are of the type given by the argument (in this case MONEY). That is, IS-OBJ will only allow itself to match objects about which we can prove

(5.9) (? IS (object) MONEY)

When the pattern is used to make an assertion IS-OBJ says to create a new NP defined by its argument. This NP will fill the spot in the assertion corresponding to that taken by IS-OBJ in the pattern. Note that it is possible for the NP "money" to refer to previously mentioned objects. Should

there be more than one the NP will be represented by a restricted variable.

We will see other uses for IS-OBJ in later sections.

5.3 Jumping to Conclusions

Why we Jump to Conclusions

Note that PB-HAVE-MONEY works in situations like:

(5.10) Jack and Janet were outside. The ice cream man came along and Janet needed money for some ice cream. She went to the house and got her PB. Soon she came back with a dime.

This is all well and good, since PB-HAVE-MONEY gives the correct interpretation of (5.10). The trouble is that (5.10) could go on to say:

(5.11) "Look, Jack, see the dime I found on the sidewalk!" said Janet.

This shows that our program jumped to a (wrong) conclusion when it asserted that Janet got the money from her PB.

To take another example, suppose we have:

(5.12) Janet needed some money. She got her piggy bank and shook it. Finally some money came out of the PB.

If we now asked, "Does Janet have the money?" the best answer would be yes, and could be answered with a demon like:

```
(DEMON PB-OUT-OF
   (NOLD PERSON PB X N N1)
```

PERSON, and PB are specified as one would expect. NOLD is pointing to a "shake" assertion since we are assuming that this demon is put in only when we have seen "shake".

```
   (?N OUT-OF ?X ?PB)
```

Look for the money being "out of" the PB.

```
   (ASSERT (? RESULT ?N ?NOLD))
```

Assert that it got this way because of the shaking. Also assert that the person now has the money.

```
   (ASSERT (?N1 HAVE ?PERSON ?X)))
```

However, (5.12) could go on to say:

(5.13) "The money rolled under the bed."

Again, the problem we must solve is how to answer questions when in actuality we don't have enough information, though we have enough to give a plausible answer.

How to Jump to Conclusions

We will handle this problem by allowing an ASSUMPTION marker on assertions, indicating that the assertion may be false. (Actually, when we get a story like "Janet went to the store. She wanted to get a ball," we are assuming that "getting the ball" was the reason for "going to the store". While this might be false, if it were we would (justly) accuse the story teller of trying to mislead us. An assertion will only receive an ASSUMPTION tag if the fact could be false

without our feeling misled.) So, when we see that the money came out of the PB we will assert that Janet now has it, but mark the assertion "ASSUMPTION" so that if the ensuing lines somehow contradict the assumption we will know that it can be safely dumped. We will check for contradictions in bookkeeping when we look for facts which must be updated by our latest fact. If the old fact is a recent assumption the program should not update it, it should erase it. (Note 7)

The ASSUMPTION tags will be placed on the assertions by a function in the demon which deduced the assumed assertion. In PB-OUT-OF we would add one more line of code, (ASSUMPTION ?N1). Later, when (5.13) followed (5.12), bookkeeping would automatically look to see if "money under the bed" updates any previous information. In this case it would update the fact that Janet has (in the sense of "immediately controls") the money. But before we marked the "have" statement as updated we would note that it is just an assumption, and that it was assumed quite recently (say within the last 5 lines or so). So rather than updating the fact, we could erase it. The idea is that we want the program to jump to conclusions, with the proviso that it can retract them later. (Note 8)

Un-assuming an Assumption

We also need to be able to "un-assume" an assertion. That is, we need to have the ability of removing the

ASSUMPTION tag but leaving the assertion in tact so that it can no longer be erased by contradictory evidence. For example, suppose we have assumed that Janet got the money when it fell out of the PB. Now suppose the next sentence is:

(5.14) Janet gave the money to Betty for her jump rope.

Presumably "give" will specify that Betty now has the money. But this contradicts the fact that Janet has the money, as assumed by our demon. On the other hand, suppose we knew that Betty was around when the money came out. If the next sentence were

(5.15) Betty got the money

we would interpret this as contradicting the assumption that Janet got it. The problem is resolved by the fact that "give" must also check to see if Janet has the object which she is giving away, and if we don't have the fact available, assert it. So, when we try to establish that Janet has the money, we will stumble across our assumption that she got it when it fell out of the PB. We then "un-assume" that fact, since it is necessary for the consistency of the story.

Note that there is some question about exactly what is capable of "un-assuming" an assertion. We will assume that only consistency checks have this ability. Hopefully when more research is done on assumptions in stories we will get a more definite answer.

When Do We Jump to a Conclusion?

The mechanism detailed here for removing incorrect assumptions is probably adequate for our immediate purposes but there remains a problem, deciding exactly when we have an uncertain situation. At this point we must actually write into the code of a particular demon that its results are uncertain. However, a given demon can produce very reliable facts in one situation, but much less reliable information in another. We have actually seen this in our examples, but not being able to go into everything at once, I chose to ignore it until now. Consider:

(5.16) Jack got his PB. He shook it and shook it. Finally he got some money.

as opposed to:

(5.17) Jack went home and got his PB. He had a nickel when he got back to where Bill was waiting.

In both cases, our PB-GET-MONEY demon should operate. In the (5.16) its result (that Jack got the money from the PB) can be completely relied upon; in (5.17), it can't. At first glance the difference seems to be that the "get" statements differ in tense. (Because the narrator always talks in past tense, anything he says in simple past should be interpreted as present tense with respect to the story. Hence in (5.16) we have a present tense statement, whereas (5.17) is a past tense statement.) We might reasonably argue that in general, present tense statements lead to more reliable conclusions

than past tense statements. This seems often to be true, but the reason goes deeper. Consider:

(5.18) Jack and Bill were outside. The ice cream man came along, but Jack had no money. Jack went home and soon he was back. "Where were you?" asked Bill. "I went home and got my PB" said Jack. "I shook and shook it. I got some money and came back here."

This seems perfectly clear. We should have no doubt that he got the money from the PB, even though his statement is past tense. So what seems to matter is how reliably we can pin down the time when the money was "got". In (5.16) and (5.18) we can know the time fairly exactly. It was right after he shook the PB, which also implies it was while he was holding the PB. This means that there was not enough time for anything else to happen. In (5.17) we cannot be so sure. All we know is that he got the PB and some indefinite time later (after he returned to the place he had been before he went home) he had some money. This leaves, say, five minutes in doubt.

Assuming this analysis is reasonably correct, several questions arise. 1) Who is doing this analysis? The demon in question, or some specific routine designed for doing it? I would suspect the latter since the process would seem to be a complicated one, but then, how do we transfer this information to the demon in question? (Or does the demon in question call the program?) 2) What is the algorithm involved?

A clear problem for future research is to collect examples of "jumping to conclusion" problems, and working out suitable procedures to explain them.

5.4 Demon-Demon Interaction

In section 2.5 we discussed some of the assumptions we had made earlier in the chapter. One particular assumption was that explicit questions (i.e., GOALs) could only be answered by assertions and fact-finders. In this section we will argue, in effect, that this assumption is incorrect. We will see that in some instances the information needed to respond to a GOAL is bound up in a demon. Rather than duplicate the information and include it in a fact-finder also, it would be more economical to somehow use the demon to answer the goal.

Evidence of the Phenomenon

Getting presents for someone often requires money. So if we saw

(5.19) Janet was going to get a present for Jack. She needed some money.

or

(5.20) Janet needed some money. She was going to get a present for Jack.

we would assert that the reason she needs money is to get a present for Jack. Since the "need money" statement can occur on either side of the "get present" statement, it is clear that we want "get present" to apply LOOK-BACK to a demon which says "If the 'getter' needs money, it is because of 'getting present'." But now consider the following fragment:

(5.21) Janet was going to get a present for Jack. She went to get her PB.

Once again we want to assert that Janet gets her PB because she wants money. But this time there is no previous assertion which says that she needs money. Of course, what is at work here is the fact that often presents are bought, in which case the person will need money to do the buying. But, originally we represented this fact as a demon, and so far we have no way for two demons to interact with one another. That is, somehow we want our new "need money" demon to make use of the previous "need money" demon which was activated by "present". So roughly speaking, when LOOK-BACK does a GOAL on the pattern of PB-FOR-MONEY it will find not a fact-finder, but the PRESENT-NEED-MONEY demon. This mechanism (we will call it demon-demon interaction) is a clear extension of our basic model as described in chapter 2.

"Implementing" Demon-Demon Interaction

We will extend LOOK-BACK so that not only does it try to find an appropriate fact earlier in the story, but it also "pseudo-asserts" the demon pattern. Asserting a fact involves, in general, two distinct activities. The most obvious is putting it into the data base, the other is placing the assertion on the TO-BE-DONE list. When we pseudo-assert something, we only look for applicable demons. Now, suppose we specify in LOOK-BACK that if any pseudo-asserted pattern

activates some demon which succeeds (that is, we get through the entire demon) then the pattern will actually be put into the data base. To make this clearer let us take the example at hand. The first line of (5.19) ("Janet was going to get a present for Jack,") puts in a demon which is looking for Janet's needing money (PRESENT-NEED-MONEY). Its pattern is:

(?N HAVE ?TN ?PERSON ?MONEY)

We now come along to the second line ("She went to get her PB.") and PB-BASE is going to assert the demon PB-FOR-MONEY by using LOOK-BACK. PB-FOR-MONEY's pattern is identical to the one above (the pattern of PRESENT-NEED-MONEY), and in both cases ?PERSON is assigned to Janet. If we now pseudo-assert the pattern of PB-FOR-MONEY, it will naturally match the pattern of PRESENT-NEED-MONEY, which will then be executed. LOOK-BACK, noting that the pattern successfully activated a previous demon, (PRESENT-NEED-MONEY) puts into the data base the pattern which it originally just pseudo-asserted. (This will be the pattern of PB-FOR-MONEY.) Naturally, this new assertion causes PB-FOR-MONEY to be activated. The net result is that we now have the fact that Janet needs money in the data base, plus the facts generated by the two demons (that she needs money to get the present, and she got her piggy bank in order to get money).

In our explanation we have skipped over one point. Notice that we end up with an assertion in the data base which

mentions "money". But "money" was never mentioned in the story. Both demons (PB-FOR-MONEY and PRESENT-NEED-MONEY) will have a check to insure that the object needed is money. These checks will naturally fail, since we have not said that the object needed is anything at all. The solution to this problem makes use of IS-OBJ. Rather than specifying that the object must be money by using

```
(GOAL (? IS ?MONEY MONEY)$TRUE $DEDUCE)
```

as in PB-FOR-MONEY, we can specify the fact in the demon's pattern with:

```
(?N HAVE ?TN ?PERSON
          ($R ?MONEY (IS-OBJ '(MONEY))))
```

In this way we will both specify that the object must be money when the pattern is used as a goal, or as a demon pattern, and when it is asserted we will "create" the needed "money" symbol. (Note 9) We might then rewrite PB-FOR-MONEY as:

```
(DEMON PB-FOR-MONEY2
   (NOLD PERSON PB N N1 N2 TN MONEY)

        This demon is put in by a "get PB" assertion.

   (?N HAVE ?TN ?PERSON ($R ?MONEY (IS-OBJ '(MONEY))))

        Still is looking for the person having had,
        having, or to be having, money, only now the
        pattern itself requires that ?MONEY be money.

   (IF-NEED (?N1 GET-FROM FUT
                 ?PERSON ?PB
                 ($R ? (IS-OBJ '(MONEY)))))
   (IF-NEED (? SUB-ACT ?NOLD ?N1))

        Assert, if needed, that getting the PB is a
        sub action of getting money from the PB.

   (COND ((EQUAL ?TN 'FUT)(ASSERT (? SUB-ACT ?N1 ?N)))
         ((ASSERT (?N2 T-RESULT ?N ?N1))(ASSUME ?N1 ?N2))))
```

A Restriction on Demon-Demon Interaction

We should add that demon-demon interaction is probably more complex than we have indicated so far. Consider:

(5.22) Janet was going to get a present for Jack. "I will get something for him at the store" she thought.

It would not be unlikely that both "present" and "store" would put in a "need money" demon. However, in (5.22) we cannot assume that Janet really needs money. For all we know she has as much as she needs in her purse. If demon-demon interaction were as simple as we have made it out to be, the two instances of the "need money" demon would join up to produce a "need money" assertion. So it is not sufficient for two demons to be looking for the same pattern. Looking at example (5.21) we note that one of the demons gave a reason why Janet might need

money, and the second suggested that needing money was the cause of a certain action. So we have:

Will get present --> Need money --> Will get PB

To put this in everyday terms, in (5.21) we have both a motive for needing money (getting present), and a result of needing the money (go and get PB). In (5.22) we have two motives. The natural suggestion is that demon-demon interaction be restricted to cases where we have both motive and result.

How do we recognize when we have both motive and result? As it stands now one demon looks pretty much like any other. We might just try to label all demons as "motive" or "result" with respect to their pattern. In fact, we already have seen some justification for this. PB-FOR-MONEY was differentiated somewhat in section 5.2 where it was called a "goal-oriented behavior" type demon. If we look back, we see that all the examples given were of the "result" variety. However, it would be premature to formalize such concepts at this point. We simply don't know enough. But it seems that this is the path we must eventually follow.

5.5 Putting Money into a Piggy Bank

The primary concern of this section is to give another example of demon-demon interaction. At the same time, this example is one of my best illustrations of fact determination. We commented in section 2.5 that it is necessary to decide exactly what facts a person uses to make a given deduction. In this section a deduction which at first seems to stem from a single fact about piggy banks will be shown to result from at least two separate facts.

A Piggy Bank Problem

One fact we know about PB's is that they are good places to keep money. This fact seems to come into play in:

(5.23) Penny said to Janet, "Don't take your money with you to the park. (You will lose it.) Go and get your PB!"

(5.24) After Janet helped Ms. Jones with her groceries Ms. Jones gave her a dime. Jack came along and said "Come with me to the park, Janet." "OK" said Janet. "But first I am going home to find my PB. I do not want to take the money to the park."

(5.25) Janet put some money on the sink. Mother said, "If you leave the money there it may fall in the drain. Let's find your PB."

In each case the natural question is, "Why should Janet get her PB?" Now we might try to construct a "piggy bank" demon which responds to some common element in (5.23) - (5.25) and then make the necessary assertions. A close look at the examples even gives a start at what such a common element

might be, say "a particular location for the money is negatively evaluated." The trouble with such a solution would be that it would not account for:

(5.26) Janet said "I am going to put my money away. I will get my PB."

(5.27) Janet helped Ms. Jones with her groceries. Ms. Jones gave Janet a dime. Jack came along and said "Janet, let's go to the park." "OK," said Janet. "But I want to put my money in a safe place. I am going to get my PB."

Now there is nothing saying that our demon needs to account for (5.26) and (5.27). However, it seems quite obvious that we are using the same information in all the examples above. The only difference is that in (5.23) - (5.25) we are expressing the need for a "safe place" by making negative comments about another location. Again, if this is a single fact we would like a single demon to express it. The trouble is finding what (5.23) - (5.25) have in common.

A Non-Piggy Bank Problem

In the course of looking at examples like (5.23) - (5.25) I noted examples like:

(5.28) Penny said to Janet, "Don't take your money with you to the park. Put it on the shelf."

(5.29) After Janet helped Ms. Jones with her groceries Ms. Jones gave her a dime. Jack came along and said "Come with me to the park, Janet." "OK" said Janet. "But first I am going to put my money in the house. I do not want to take the money to the park."

(5.30) Janet put some money on the sink. Mother said, "If you leave the money there it may fall in the drain." Janet put the money in a drawer.

(5.31) Janet said "I am going to put my money away. I will put it in my toy box."

(5.32) Janet helped Ms. Jones with her groceries. Ms. Jones gave Janet a dime. Jack came along and said "Janet, let's go to the park." "OK," said Janet. "But I want to put my money in a safe place. Then Janet went into the house and put the money in her room.

These examples exactly mirror (5.23) - (5.27), except that (5.28) - (5.32) don't mention PB's. Naturally, in these examples the question to ask is "Why did Janet put the money in the drawer?", or "Why will Janet put the money in the house", etc.

Such examples tend to indicate that the problem facing us is wider that just PB's. We will name this wider problem the "put away" problem. However it is not the case that our problem with PB's can be completely reduced to the "put away" problem. So while in (5.28) on, we mention that Janet has or actually intends to "put" the money some place, in the PB examples all we needed to say was that Janet was going to get the PB. To put this another way, our knowledge of PB's allowed us to interpret "get PB" as meaning that Janet was going to put money into it. However our knowledge of houses or shelves does not allow us to make similar deductions in (5.28) - (5.32).

The Put-Away Demon

Ignoring piggy banks for the moment, what would a solution to (5.28) - (5.32) look like? We will have some demon, called the PUT-AWAY demon, which is activated by lines like:

(5.33) Don't leave the money by the sink.
(5.34) I do not want to take my money to the park.
(5.35) I will put my money away.

These lines will put in a demon looking for "put away". Ultimately we will want a theory of why people put things away (i.e., what lines put in the "put away" demon), but it is not necessary to know this in order to continue our study of piggy banks. Our "put away" demon would have a rough outline like:

```
(DEMON PUT-AWAY
   (NOLD OBJ PERSON LC TN PL N)
   (?N ($R ?LC LOC) ?TN ?OBJ ?PL)

        Look for assertion which says that OBJ
        is at (will be at) some location

   (Code goes in here which will check to make sure that
    ?PL is an "appropriate" place to "put away" ?OBJ.)

   (ASSERT (? RESULT ?N ?NOLD)))
```

(Note 10) The check needed to make sure that the place is "appropriate" is complicated. Finding good rules is a problem for further research. But we will need such a demon, and the demonstration of its usefulness is independent of our problems with PBs.

The Piggy Bank Demon

What we will now see is that if we assume the PUT-AWAY demon, all the examples in (5.23) - (5.27) fall out easily, plus a few others which we haven't even looked at yet. But first we need to consider a new PB demon entitled PB-MONEY-IN. It is parallel to PB-FOR-MONEY, but while the latter was for recognizing that money was going to be taken out of the PB, PB-MONEY-IN is for recognizing that money is going to be put in.

```
(DEMON PB-MONEY-IN
   (NOLD PB PERSON N1 N M TN)
   (?N IN FUT ($R ?M (IS-OBJ '(MONEY))) ?PB)

        We are assuming that ?NOLD is bound to
        the "get PB" assertion.  We are looking for
        money to be in the PB.

   (ASSERT (?N1 PUT-IN FUT ?PERSON ?PB ?M))

        The intention is to put the money in the PB.

   (ASSERT (? SUB-ACT ?NOLD ?N1))

        And getting the PB is a sub-act of that intention.

   (COND ((EQUAL ?TN 'FUT)
          (ASSERT (? RESULT ?N1 ?N)))
         ((ASSERT (? T-RESULT ?N ?N1)))))
```

This demon will account for examples like:

(5.36) Ms. Jones gave Janet a dime. Janet went to get her PB. "I want the money to be in my PB" she thought.

(5.37) Janet got her PB and dropped some money in.

(5.38) After Ms Jones gave Janet a dime, Jack came by and asked Janet if she wanted to go to the park. "OK", said Janet. "I will go home first and get my PB." Soon Janet came back and said "My money is in the PB, let's go!"

Demon-Demon Interaction

Now, if we assume demon-demon interaction as discussed in section 5.4, PB-MONEY-IN plus PUT-AWAY will interact to solve all the examples from (5.23) to (5.27). Let us see how this will happen.

First note that the restrictions we placed on demon-demon interactions are met here. That is, in 5.4 we said that we needed both a motive and a result before we could "combine" demons in this manner. In the case at hand, PUT-AWAY is a motive for having the money in the PB, and "get PB" is a result of intending to put money in the PB. So when we hand PB-MONEY-IN to LOOK-BACK, it will pseudo-assert the pattern, which essentially says "look for some money in this particular piggy bank". This assertion will activate PUT-AWAY which is looking for "this particular portion of money being at some location". PUT-AWAY naturally goes on to check that the location is "appropriate", but this test should be easy for PB's to pass.

Finally, note that this solution extends to the following case:

(5.39) Janet got a dime from Ms. Jones. She said "I am saving my money to buy a bicycle. I am going home to get my PB."

Here we know that Janet is going to put the money in the PB because of the "save" statement. However, we immediately note that we have cases like:

(5.40) Janet got a dime from Ms. Jones. Janet told her "I am saving my money to buy a bicycle. I am going home to put the money away. (I am going home to put the money in my drawer.)

Naturally, (5.40) indicates that "save" must activate PUT-AWAY. If this is the case, then (5.39) is accounted for in exactly the same manner as all the initial examples. While the reader may not be surprised at this result, I am, since initially I thought that the relationship of "save" with piggy banks would need a separate PB demon.

5.6 Sound and Choosing Semantic Representation

Another possible story fragment:

(5.41) Janet went and got her PB. She shook it. She heard the PB rattle.

We might now ask if there is anything in the PB, and expect to get the answer "yes". Or, if Janet went on to say, "Good, there is something in there," we might ask how she knew. Or again, if she hears nothing, we can draw parallel but opposite conclusions. It would be easy enough to write a demon which basically says that sound implies something in the PB and no sound implies the PB is empty. One immediate problem in formulating such a demon however, is the many ways we can talk about an object or action making a sound. For example:

(5.42) The PB (coins) made a sound (noise).
(5.43) The PB (coins) rattled (jingled).
(5.44) Janet heard a coin (the PB).
(5.45) She heard a coin (the PB) rattle (make a noise).
(5.46) She heard a noise (rattle).
(5.47) There was a noise (rattle).
(5.48) A sound came out of the PB.
(5.49) A sound came from the PB.
(5.50) Janet heard a sound from the PB.
(5.51) She heard the coins hit the inside of the PB.

Ultimately, this profusion of ways to say much the same thing is a problem for the semantic portion of the program, and as such really should not be covered here. The problem is that we don't know what our demon should look for until we have created some order out of the chaos. Certainly we can't afford to have a separate demon looking for each individual situation. We must figure out what all these forms have in

common, so we can force all of them into a few standard forms, and just have demons looking for them.

Fortunately it seems to be possible. Let's start with (5.42). It says that a particular object "made a sound". Of course, the "made" seems to be somewhat superfluous, since, as we see in (5.43) we can just say "rattled" "jingled" or if the coins were very heavy, "banged". What we might do to standardize (5.42) and (5.43) would be to say

(N9 SOUND (coins, PB, etc.))

which would be translated into English as "The PB 'sounded'". Note that N9 does not specify if the sound was a "rattle" etc. There are several ways to do this, but it doesn't seem to be necessary to go into it now. Then (5.47) would become:

(N10 SOUND UN-MENTIONED3)

Carrying on the same way, we translate the sentences of the form "Janet heard (an object, a sound, an object make a sound)." (i.e., (5.44) , (5.45), and (5.46)) as:

(N11 HEAR JANET N12)
(N12 SOUND PB1)
or
(N12 SOUND COIN1)
or
(N12 SOUND UN-MENTIONED4)

Note that what we have done so far is to translate every line into a SOUND assertion plus, in some cases, a HEAR statement. If we wanted to be even more uniform we might say that the statements which don't mention that someone heard the sound should be translated into a "hear" statement anyway, since

someone must have heard it. But there doesn't seem to be much reason for this. Our PB demons for example will just be looking for the SOUND assertion, and will ignore the HEAR statement anyway. Furthermore there is a very good rule for sound and sight, which says that if a sound (or a visual presence) is at a certain location at a certain time, anybody there will have heard (seen) it unless we have specific reason to believe otherwise.

The last four sentences (5.48) - (5.51) are still not accounted for. Example (5.50) is easy once we have (5.49); we just add a "hear" assertion. After looking at some examples we find we can account for (5.48) and (5.49) with the following rule:

(5.52) "A sound came out of (from) X" gets translated as follows

a) if X is capable of making a sound by itself
(N13 SOUND X)

b) If X is a container (in the loose sense that it is usually considered to have an inside and an outside
(N14 IN UN-MENTIONED4 X)
(N15 SOUND UNMENTIONED4)

c) If X is part of a very large container (house size)
(N16 OTHER-SIDE-OF UNMENTIONED5 X)
(N17 SOUND UNMENTIONED5)

Examples illustrating part (a):

(5.53) A sound came out of (from) the loudspeaker.
(5.54) A sound came out of (from) the radiator.

Examples illustrating part (b):

(5.55) A sound came out of (from) the PB.
(5.56) A sound came out of (from) the building.
(5.57) A sound came out of (from) the wall.

Examples illustrating part (c):

(5.58) A sound came out of (from) the window.
(5.59) A sound came from the roof.

Though the fact that we cannot say "came out of the roof" indicates that there is some distinction between the two phrases which we have not yet captured.

There is some question as to exactly where this translation process will be carried out in these cases. For example, for "out of" we might have internal translation produce a more literal translation, saying that an NP named "sound" has moved from inside the PB to outside. Then GO-BASE would look and see that what "went" wasn't a physical object, and it would translate the rest of the way to our format as in N13 - N17. However, it would probably be better if this were done immediately, or else we will have a slight bit of trouble with our "come out of PB implies get" demon which would see this object coming out of the PB and assert that Janet got it, completely oblivious to the fact that it is only a sound. Of course, this could be rectified by the demon checking to make sure that the object is physical, but then we would have to make at least two checks on its "physicalness", whereas if we checked immediately in internal translation, and translated accordingly, we could have the whole thing over and done with.

At this point we have only (5.51) to deal with. Although

this particular example has an awkward phrasing, sentences like (5.51) are a common form of statements about sound. For example "Janet heard Jack come into the house." "Janet heard the leaves fall to the ground." "Janet heard Bill talking to Dick." are all statements which say that Janet heard some action taking place. However, such statements do not fit our scheme of having "hear" always refer to a "sound" assertion, since our "sound" assertions take an object as their argument, not an assertion. There are several possible ways out. We could allow "sound" to take an assertion as argument and get:

```
(N19 HEAR JANET1 N20)
(N20 SOUND N21)
(N21 HIT COINS PIGGY-BANK-SIDE)
```

The trouble is that our "sound" demons (for PB and any other action which depends on sound) will have to make a special check to see if the "sound" assertion is referring to an object or an assertion. We would like to avoid this if at all possible.

A second possibility is to do a more radical translation and get:

```
(N22 HEAR JANET1 N24)
(N23 HIT COINS1 PIGGY-BANK-SIDES)
(N24 SOUND COINS1)
(N25 T-RESULT N24 N23)
```

But while it would work for this situation it would produce unhappy results for "Janet shook her PB and dropped it. She heard the PB hit the floor." This would create a "PB sounded" assertion which our demon would pick up. To prevent this from

being interpreted as something being in the PB we would have to add a check in the demon to note what caused the sound.

Note that we get exactly the same problem if we try to convert all our "sound" statements so that they take an assertion as their argument. We might represent (5.43) as:

(N26 SOUND N27)
(N27 MOVE PB1)

instead of "the PB 'sounded'", but we still have the problem that we saw before. If we just say we heard the PB we want our demon to work, but if we have "hear the PB hit the ground" or "slap against her hand" or whatever, our demon should remain inoperative.

This all seems to be suggesting that when we say "heard the PB" we are saying that the PB is emitting sound in some way, and it is this "emitting" which our demon responds to. But when we say "hear (some action)" any conclusions we draw from this fact should be based on the action itself; the PB-SOUND demon should not come into play. Further evidence for this view is provided by:

(5.60) "When Janet picked up the PB a few coins hit the side of the PB."

While (5.60) does not mention "hear" or "sound" at all, we still understand that the coins are in the PB. So whatever tells us in (5.51) that the coins are in the PB is independent of sound. Then since we don't need, or even want "hear (action)" to use the "sound" predicate we can translate (5.51)

as:

```
          (N28 HEAR JANET1 N29)
          (N29 HIT COINS1 PIGGY-BANK-SIDE1)
```

It is only now, after this lengthy analysis of "sound" that we can go on to write our PB demon.

```
(DEMON PB-SOUND
   (NOLD PERSON PB X N N1)

        PB and PERSON are specified as before.
        Since the demon is only for present tense
        statements we will assume that NOLD is
        a present tense "have PB" assertion.

   (?N SOUND ?X)
   (COND ((EQUAL ?X ?PB)

        As in "the PB made a sound" or "There was a sound"
        in which case we assign PB as the "sounding" object

          (IF-NEED (?N1 IN ($R ? (IS-OBJ '(SOMETHING))))))

        If we are simply told that the PB "made a sound"
        we just want to assert that something is in the PB.

          ((AND (GOAL (? SIZE ?X SMALL) $TRUE $DEDUCE)
                (GOAL (? HARD ?X)$TRUE $DEDUCE))
           (IF-NEED (?N1 IN ?X ?PB)))

        If we are told "(obj) made a sound" where (obj) is
        hard and small, then assert that (obj) is in the PB.

          ((GOAL (?N1 IN ?X ?PB)$TRUE $DEDUCE)))

        If none of the above, and we can't deduce that ?X
        is in the PB then we have picked up a stray fact
        like "Janet heard Jack" and we should fail.  (If a
        COND has no successful branch then we fail.)

   (ASSERT (? RESULT ?N ?N1))

        Assert that the sound is a result of something
        being in the PB and the shaking.

   (ASSERT (? RESULT ?N ?NOLD)))
```

The corresponding demon for no sound being made:

```
(DEMON PB-NO-SOUND
   (NOLD PERSON PB N X N1)

        NOLD is the "shake" assertion.

   (?N SOUND NOT ?X)
   (COND ((EQUAL ?X ?PB)

        As in "the PB didn't make a sound" or "There wasn't
        a sound".  We assert that nothing is in the PB.

          (IF-NEED (?N1 IN NOT
                        ($R ? (IS-OBJ '(SOMETHING)())))))
         ((AND (GOAL (? SIZE ?X SMALL) $TRUE $DEDUCE)
          (GOAL (? HARD ?X)$TRUE $DEDUCE))

        If the thing which didn't make a sound is hard and
        small then it is not in the PB.  Otherwise the no
        sound assertion has nothing to do with PBs.

          (IF-NEED (?N1 IN NOT ?X ?PB))))
   (ASSERT (? RESULT ?N ?N1)))
```

One trouble with these two demons is that they really represent the same fact, "If a PB is shaken, there is something in the PB if and only if the PB makes a sound". In order to combine the two we will most likely need a new form for negation. Rather that having the NOT in the assertion proper, we could put it on the property list. This way "there was a sound" and "there wasn't a sound" could match the same pattern, only the latter would have a tag on its property list. However, I am not sure of all the implications of this change in notation, so I have only adopted the new negation notation for this one demon.

```
(DEMON PB-SOUND-BOTH
   (NOLD PERSON PB X N N1 TAG)

        ?TAG will be NEG if the assertion is negated, POS
        otherwise.  ?NOLD is still the "shake" assertion.

   (?N SOUND ?X)
   (COND ((EQUAL ?X ?PB)
          (IF-NEED (?N1 IN
                        ($R ? (IS-OBJ '(SOMETHING)())) ?X)
                   (PROP (LIST 'POLE ?TAG))))

        If the PB did (not) make a sound then something is
        (not) in the PB.  We put the appropriate negation
        tag on the IN assertion's property list.

         ((AND (GOAL (? SIZE ?X SMALL) $TRUE $DEDUCE)
          (GOAL (? HARD ?X)$TRUE $DEDUCE))

        If ?X is hard and small then it is (not) in the PB.

          (IF-NEED (?N1 IN ?X ?PB) (PROP (LIST 'POLE ?TAG))))
         ((AND (EQUAL ?TAG 'POS)
               (GOAL (?N1 IN ?X ?PB)$TRUE $DEDUCE))))
   (COND ((EQUAL ?TAG 'POS)(ASSERT (? RESULT ?N ?NOLD)))
         (T))
   (ASSERT (? RESULT ?N ?N1)))
```

We will need another routine to pick up the "not in PB" assertion generated by PB-NO-SOUND and translate that into a statement that Janet will no longer try to get money from the PB. This should be a separate demon since we also need this to account for:

(5.61) Janet needed some money. She got her PB. There was nothing in it.

Presumably we could now ask "Will Janet get money from the PB?"

```
(DEMON PB-EMPTY
   (PB PERSON TN OB OBJ OB)

        This demon is put in by a "get from PB" assertion.
        ?PB and ?PERSON are bound as usual.  OB is bound
        to the object which is to be removed from the PB.

   (? IN ?TN NOT ?OBJ ?PB)
   (SUBSUME ?OB ?OBJ)
   (ASSERT (? GET-FROM FUT NOT ?PERSON ?OB ?PB)))
```

The "subsume" function just checks to make sure that whatever was said not to be in the PB can be construed as referring to the thing the person wanted to get out of the PB. So "nothing" would subsume pretty much everything, while "money" would subsume "the nickel", but not "a paperclip". While it is clear that we need some such check, at the moment SUBSUME seems somewhat ad hoc, coming out of nowhere as it does. However, we do need something. For example, we certainly don't want this demon applying if Janet is trying to get a paperclip out of the piggybank, but comments as she is trying to do so that there is no money in it.

There is an uneasiness about introducing a function like this without any idea if it is really needed in general, or if it is needed just for this particular demon. While I suspect that the former is the case, it would be comforting, when introducing a new function like this, to be able to find many situations where the same function is needed, as we did with IS-OBJ.

5.7 Accessing the Information

At this point we have constructed several demons for piggy banks, PB-FOR-MONEY, PB-SOUND-BOTH, PB-OUT-OF, PB-EMPTY, and PB-MONEY-IN. We suggested that these would be put in by the piggy bank base routine. On the other hand, we might want to argue that PB-BASE does not activate PB-SOUND-BOTH, or PB-OUT-OF. The argument will revolve around the acceptability of story fragments like:

(5.62) Janet got her piggy bank. It made no sound.
She could hear the coins rattle.
Some money came out.

In each case the story is clearly acceptable if we insert "She shook the PB" between the two lines. We will need a PB-SHAKE demon anyway to account for the connection between "shaking PB" and "getting money from PB". So, if we feel that a given fragment is unacceptable without "shake" then PB-SHAKE could put in the corresponding demon. If not, then PB-BASE should put it in. I personally have no strong feelings one way or the other. I find the stories definitely "odd" but understandable.

Significant Sub-Actions

Throughout this chapter we have assumed that the line which started us off on piggy banks was "(person) got piggy

bank." However, we could equally well have " (person) get (money) from PB" as our initial "piggy bank" line.

(5.63) Janet needed some money. "I will get some money from my PB," she said.

(5.64) Jack and Janet were outside when the ice cream truck came along. "I am going to get some money from my PB," said Janet. Soon Janet came back with some money.

In these cases we would like to use PB-FOR-MONEY to connect the "get from" assertion to the "have money" assertion. While we might use other means, since we already have this demon available, we might as well use it. The only trouble is that if we look at PB-FOR-MONEY we see that it assumes that "get piggy bank" is the assertion which is pointed to by ?NOLD. If we started out with "get from" we would get such silly statements as "getting money from the PB is a sub-action of getting money from the PB". Nor is this an isolated example. A quick look at PB-MONEY-IN reveals that exactly the same problem occurrs there.

What we are seeing here is not a simple bug in our code. The problem is actually quite widespread; let us look at it from a slightly different angle.

The two lines which might set up a "piggy bank" context are quite different. The line "get money from PB" is a description of a complex activity. On the other hand "get PB" is only a part of the complex activity described by "get from", and it can be part of other activities, like "put

into". But, when we combine the "get PB" line with the fact that the person needs money, we are able to guess that the person intends to perform the complex "get from" activity. So what we have done is to take two facts which are part of the "get from" complex, put them together, and deduce that we are witnessing the "get from" activity. In a sense we are deriving more encompassing contexts from specific facts. So a fact like "get piggy bank" can be viewed in two ways. First, it relates a small activity to a known larger activity. Secondly, if we were not aware of this larger activity, our fact gives evidence that this larger activity is indeed taking place.

Other examples of this dual usage:

(5.65) Today was Jack's birthday. Mother got a cake.
(5.66) Jack went to get his hammer. His play house was broken.

In both cases we deduce a larger activity (birthday party and fixing toy house) from more particular facts (getting a cake, and getting a hammer) which are part of the larger activities.

We might call such particular actions "<u>significant sub-actions</u>", since not all sub-actions lend themselves to dual usage. For example:

(5.67) Jack got some string and went outside.
(5.68) Jack got his kite and went outside.

Here, (5.68), but not (5.67), can be interpreted as "going to fly a kite", even though both "get string" and "get kite" are sub-actions of "fly kite". Ultimately we will want a theory

of the exact circumstances under which such things happen. For the moment we will just note that adapting the PB-FOR-MONEY (or PB-MONEY-IN) demon so that it can serve a dual purpose will also enable it to handle both "get PB" and "get money from PB" being bound to ?NOLD, which was the problem we started out with.

Significant Sub-Actions and Search

The concept of signficant sub-actions suggest that it might be necessary to modify the rule that a demon can only be made available when its "topic" concept is introduced. The reasoning goes as follows: Topic A says look for topic B (birthday says look for "party"). But C, a significant sub-action of B is sufficient by itself (given A) to indicate B (get a cake + birthday ==> party). Suppose the demon relating C to B has B as its "topic concept" (get cake has "party" as its topic). Then whenever we set up a demon looking for B (party) we also want to also activate those significant sub-actions of B which can serve as evidence for B. Hence we would be activating demons whose "topic" had never appeared in the data base.

However, there is a crucial assumption in the above, namely that C has B as its topic concept. We might argue, for example, that "get cake" is not a sub-action of "party" but of "birthday" itself. As evidence one way or the other we might

use the following example.

(5.69) Today was the Fourth of July. Mother got a cake.

If the average reader understands (5.69) as implying that Mother got the cake for some Fourth of July celebration, then the connection is definitely between "party" and "cake" rather than "birthday" and "cake". While I find (5.69) odd, my informants have no problem in interpreting it correctly.

But even if we accept that the connection is between "party" and "cake" we are still have not shown that we need to change our topic rule, for if it is "cake" which is looking for "party" then we only need demon-demon interaction. In (5.65) both "birthday" and "cake" would be looking for party, and we have both a "motive" (the fact that it is someone's birthday) and a "result" (getting a cake). On the other hand, if "party" looks for "cake" then we need to change our topic rule to account for (5.65) since "birthday" looks for "party" which in turn looks for "cake". So the evidence is not conclusive in either direction.

Looking at (5.66) we see that again the crucial assumption is whether "get hammer" has the topic concept "fix". If so, then we need a new "topic" rule. If, on the other hand, "get hammer" itself puts in a demon looking for "fix" then we again can make do with demon-demon interaction. That is, both "get hammer" and "broken" will be looking for "fix" and we will have the necessary "motive" and "result".

When Do We Execute PB-BASE?

All the demons we have written assume that the PB has some purpose being in the story. So, if we were describing a room and mentioned that a PB was on the shelf, none of our demons would be called for. This suggests that perhaps something like

(N31 HAVE (tense) (person) (PB))

should be required before we will activate PB-BASE.

There is still a loose link in all this. Suppose that we have a "have PB" assertion coming into the data base. At what point in the processing do we execute PB-BASE? When we initially defined base routines, we said that they were executed whenever they were mentioned in the internal representation. Since all our initial examples dealt with base routines connected with predicates, we have tacitly assumed that DSP took a particular assertion, looked at its predicate and got the appropriate base routine. We cannot use this simple method with base routines connected with objects, however. (We will distinguish the two kinds of base routines as "predicate base routines" and "object base routines".) So if Janet gets her PB, what appears in the assertion is not "piggy-bank" but PB1, or some indexed object. Furthermore, a given object can appear in more than one assertion. Do we call the base routine more than once?

Another problem is that in the case of pronouns, we can't call the object base routine until we know the pronoun's referent (except in the odd case that we can't decide between two objects of the same type). However, we can't postpone calling the object base routines until after we know all the referents since sometimes object base routines themselves can be useful in deciding other referents. For example:

(5.70) Janet was going to buy a present. Today was Mary's birthday. She got her PB and shook it.

In this case the "she" seems to be determined by the previously mentioned demon demon interaction between "Janet needing money" and "she gets PB". Note that the later demon is put in by PB-BASE. Hence if the call to the object base routines took place after all the referents had been decided, the "she" would already have been incorrectly assigned to Mary before we got into PB-BASE.

Instead, we will call object base routines just before the assertion goes into bookkeeping, and if necessary, just after bookkeeping also. Also note that in many cases it is not really necessary to know the correct referent, since "the piggy bank" should call the PB base no matter which PB is being referred to.

5.8 Further Topics

Tense and Time

So far we have concerned ourselves primarily with present tense statements. Our demons have reflected this fact. For example, PB-SHAKE specifies that we are to look for a person shaking a PB. Since the pattern specifically requests a present tense statement (by our convention of chapter 4 we left out the PR marker), this demon will not work for other tenses. But we also need to handle future tense:

(5.71) Janet and Jack were outside. Janet needed some money. She said "I am going home. I will get my PB. I will shake and shake it."

While this may sound slightly stilted, it is nevertheless clear, and we are capable of answering the question "Why will Janet shake her PB?" In exactly the same way we need to handle past tense:

(5.72) Jack and Janet were outside and saw the ice cream man. Janet ran home. Soon she was back. Where were you? asked Jack. "I went home and got my PB. I shook it and got some money."

As a first attempt to handle (5.71) and (5.72) we could replace the (deleted) PR in the pattern of PB-SHAKE with a variable ?TENSE. In this way our demon would also be executed in the last two examples.

However, allowing different tenses creates problems. For example, we could have:

(5.73) Jack asked Janet if she wanted to go to the movies. Janet said "I need some money. I will get my PB." "But you got a quarter earlier" said Jack.

We do not want this last past tense statement to activate the PB-GET-MONEY demon which would then assert that the money came from the PB which Janet has not yet gotten. This seems to imply that the "get-money" demon will have to check explicitly that the money was received after the person decided to get money from the PB.

We can solve the problem in an ad hoc way by including in PB-FOR-MONEY the check:

```
(COND ((EQUAL ?TN 'PAST)(GOAL (? BEFORE ?N1 ?N)))
      (T))
```

though even this has some difficulties. But the problem is a very general one. Sub-activities of a super-activity will be partially ordered in time with respect to each other. Such a basic fact should ultimately be expressed in the very notation we use to describe facts. How such a scheme would work is another open problem.

Necessity and Surprise

Consider:

(5.74) Janet got her PB and shook it. She was going to put some money in it. There was no sound. She put the money in and shook it again.

Two questions to ask are, "Will there be a sound this time?" and "Why did Janet shake the PB the first time?" Now, our current scheme will not handle the first question. If we had

said "Janet heard a sound," then our demon would be able to say why. But we have no way at present to predict what will happen.

What we need is the ability to use the same fact both to account for known facts (in my model this means acting like a demon) and answering explicit questions about what is likely to happen whenever a person asks (this means acting like a fact finder). Since both my system and Planner make a firm distinction between the two, both seem to be inadequate in this respect. This is another situation which indicates the need for allowing demons to respond to GOAL requests.

One very different solution might be to use a predicate calculus theorem prover. In such a system we might have the rule:

(5.75) "shake PB" ==> "sound"<==> "money in PB"

Presumably we could use this rule to deduce the fact that there would be a sound. However, this would still leave a problem with:

(5.76) Janet got her PB and shook it. She was going to put some money in it. There was no sound. She put the money in and shook it again. There was still no sound. She was surprised.

The question now is, "Why was Janet surprised?". While we might be able to patch such a predicate calculus scheme to account for this example, it really appears that our "expectation" approach is more realistic. However, if we are going to account for both examples with demons, then most

likely we will need some concept of a "must occur" demon. Probably, when shake puts in the demon it can be made to look to see if there is anything in the PB. If so then PB-SOUND is in a "must occur" situation and should be so marked. Presumably, when we have made PB-SOUND and PB-NO-SOUND into a unified fact, knowing that "sound" must occur will mean that "no sound" should be surprising.

Note that "must occur" is not applicable to all demons. So getting a PB does not necessarily imply that the person will be geting money from it. "Must occur" only seems applicable in those situations which seemingly depend only on the laws of physics, so to speak. Again we seem to need a classification of demons.

Other Problems

The second question about (5.74) was "Why did Janet shake the PB the first time?" We do not want to get the answer "She was trying to get money from the PB." This is problematic because initially when we see the "shake" assertion that is exactly what we assume. Once we recognize that this conclusion is contradicted by the next line, by "She was going to put some money in it," then our "assumption" mechanism is sufficient to remove the offending assertion from the data base, and the worst that could happen is that the model would respond "I don't know" which is not all that unreasonable.

But it is not that easy to have the model recognize that this is a contradiction. So after the first line in (5.74) we would have internally something like:

(5.77)
(N33 GET JANET1 PB1)
(N34 SHAKE JANET1 PB1)
(N35 GET-FROM FUT JANET1 MONEY1 PB1)
(N36 SUB-ACT N33 N35)
(N37 SUB-ACT N34 N35)

When we get the second line in we will have added:

(5.78)
(N38 PUT-IN FUT JANET1 MONEY1 PB1)
(N39 SUB-ACT N33 N38)

The only real clue we have to go on is the fact that N33, the "get" assertion, is now a sub-action of two different main activities, "put in" and "take out". But in general a given activity is allowed to be a sub-action of two different activities. So we have:

(5.79) Janet and Jack were outside. Jack said "Janet, I was looking at your PB and it had a crack in it." "I will go to see," said Janet. "I need some money anyway." Janet went to get the PB.

In this case "get PB" is a sub-act of both "seeing the crack" and "getting money from the PB". What we need to know in this case is that taking money from a PB and putting it in are mutually incompatible. Again the necessary information is there, the incompatible (and "contradictory") end results of the two activites. Again the problem is making use of it.

If we are told that Janet had taken money from her PB this implies many things,most of which are really irrelevant given that the action is over with, such as the fact that she

shook the PB. Suppose we were asked if she did so? Do we say "I don't know"? Do we assert "shake" when we are told that a person got money from the PB? Do we use some other technique to get a yes answer? The same problem applies on any complex activity, such as baking a cake. It doesn't seems as if we would want to list out in the data base every action which goes into the activity. So what do we do?

If Janet went home to get her PB and comes back and says that there was no money in it, it implies that she at least tried. How do we know this? There seems to be an interaction between "know" and PBs.

Ownership and Possession

When we listed the facts we knew about PBs we mentioned that PBs are owned by a person, and this extends to the contents of the PB under normal circumstances. Now we might ask about the significance of ownership. That is, what difference does it make who owns a particular object? This, as we shall see is a very complicated question, and probably not really part of piggy banks, since it applies to all objects which are owned by a particular person. We can also see that it will probably present very different problems of analysis than we have seen for piggy banks. After all, we only looked at two kinds of "piggy bank" behavior, putting money in and taking it out. Both are goal oriented, and hence

have a certain strucure as we have indicated. Finding the significance of ownership first requires isolating situations in which ownership makes a difference, and then trying to specify what can happen in such situations so we can have demons on the lookout.

> If you don't own a particular object and you break it, some possible responses are: an apology to the owner, buying a new one, offering to do so, pretending you didn't do it. You may not sell something which you don't own, unless you have permission. If you are using someone else's toy, and the owner wants to use it, you should surrender the toy to the owner. If you want to use someone else's toy, you should ask permission, though if you are not taking it from where it was put by the owner (or at least from where you found it) this rule is not imperative. (That is, you won't be considered "bad" if you disobey it.) If you do take it away from its former location, and the owner comes looking for it, he has the right to be angry at you. Finally, if you are part owner of an object you still are restricted from selling without permission, and the consequences are roughly the same for breaking it (your guilt in this case is roughly that of a person who had permission to use the object.) On the other hand, it is not usually necessary to get permission to use the object from the other owner and in cases where both owners wish to use the object, a complicated negotiation process is likely to ensue.

These are just some situations where ownership is important. If we want to start considering what can happen in such cases the amount of information gets much larger. Let's just consider what might happen if person A takes OBJ which belongs to B. Person A then sells it.

> Person B might never notice it was missing. He might note that it was not there, but think that he himself mislayed it. Noting it is gone, B goes around asking who has it, or he might go around

accusing everyone of taking it. He may immediately decide that A must have done it and go after him, or go to tell Mother that someone (perhaps A) took OBJ. If B eventually finds A what happens? A might deny having done anything, hence becoming a liar, and hence a bad person. A could admit doing it, but maybe there had been a legitimate mixup, say A and B were brother and sister, and A thought the ball was his, when it was really B's. A could say "I am sorry" and give B the money (or the object he got in trade)or offer to try to get the object back. A might admit doing it, but say "tough luck". In such a case B is allowed to retaliate by doing something nasty to A, though Mother might try to talk him out of it.

6 Problems of Reference

We saw earlier that our internal representation must have symbols to represent the objects being talked about, and hence it became necessary to go from an English description to the correct internal symbol. Then in chapter 3 we briefly outlined how various contextual features would be included in the inputs to the reference decision. In this section we will accomplish two goals. The first is to take a closer look at what we previously called "finding possible referents". In particular we will look at definitional information in section 6.1 and syntactic information in 6.2. I have nothing to add on recency or selectional information.

The rest of the chapter will be concerned with completely new topics. First we will consider what happens when an NP does not refer to a previously mentioned object at all, but rather introduces a new object into the story. Then in 6.4 and 6.5 we will consider special cases of the reference problem and observe how they place special demands on DSP.

6.1 What's in The Words

That a noun phrase can systematically be converted into a description of its referent is not a new idea. Many comprehension programs use the fact (Craig et al. 66), (Quillian 69), (Woods 68). The most complete use of such information is in Winograd's program. This section is based on his work.

In this and the next several sections we will only be considering definite NP's, like "the ball," "it," "my kite," but not like "a ball," or "one of the kites." Such NPs are more likely than not going to refer to particular objects in the world. As we shall see later, deciding what does or does not refer to particular objects is not easy, but there are clear cut cases, and we shall concentrate on them for the moment.

Given the NP "the red ball" the program will collect each word's DESCRIBES property. For example, the entry under BALL would be:

```
(6.1)    (HEAD 'BALL)
```

Note that the entries are programs to be evaluated. In this case the HEAD program would produce:

```
(6.2)    ((GOAL (? IS ?B1 BALL)$TRUE $DEDUCE)
          (NOT (PLURAL ?B1)))
```

This simply says that the object is currently a ball and is singular. The symbol ?B1 is a variable assigned to this NP.

Collecting the DESCRIBES properties from all the words in "the red ball" we would get:

```
(6.3)    ((GOAL (? IS ?B1 BALL) $TRUE $DEDUCE)
          (NOT (PLURAL ?B1))
          (GOAL (? COLOR ?B1 RED)$TRUE $DEDUCE)))
```

This would be handed off to the program in charge of deciding what internal objects are possible referents. It would create:

```
(6.4)    (FIND ALL
           ?B1                              says to find B1's
           (B1)                             variables to be bound
           (GOAL (? IS ?B1 BALL) $TRUE $DEDUCE)
           (NOT (PLURAL ?B1))
           (GOAL (? COLOR ?B1 RED)$TRUE $DEDUCE))
```

This program, (6.4), would then be run, and would produce a list of all the red balls the program knows about (called the Possible Referent List (PRL)). If the PRL has only one member, we can assume that it is the correct referent.

Let's consider a more complicated example, "Jack's kittens." Since my programs don't do any syntactic analysis (or very little) this NP would come in as ((JACK) 'S PLM KITTEN) where 'S indicates a possesive relation and PLM stands for PLural Marker. (In the first example the HEAD routine knew the NP was singular by the absence of a PLM.) The routine for 'S calls the top level reference routine recursively on (JACK) and in the manner described earlier we produce and execute:

```
(6.5)    (FIND ALL ?B2 (B2)
            (GOAL (? NAME ?B2 JACK)$TRUE $DEDUCE))
```

This should find the symbol used for Jack. Then this symbol is inserted in (? POSSESSIVE *** ***) in the first *** location and it is made into a goal statement. 'S also will insert the fact that the NP is determined, so the final PLANNER expression will be:

```
(6.6)    (FIND ALL ?B3 (B3)
            (GOAL (? IS ?B3 KITTEN)$TRUE $DEDUCE)
            (PLURAL ?B3)
            (GOAL (? POSSESSIVE JACK1 ?B3)$TRUE $DEDUCE))
```

Subordinate clauses are handled in pretty much the same way. For example "the boy who gave Jack the ball" would come in as:

```
(6.7)    (THE BOY SUBCL (? GIVE PAST *** (JACK) (THE BALL)))
```

SUBCL will process the clause which involves determining the referents for (JACK) and (THE BALL). If these can both be determined uniquely, the result would be:

```
(6.8)    (FIND ALL ?B3 (B3)
           (GOAL (? IS ?B3 PERSON)$TRUE $DEDUCE)
           (GOAL (? SEX ?B3 MALE)$TRUE $DEDUCE)
           (GOAL (? AGE ?B3 YOUNG)$TRUE $DEDUCE)
           (NOT (PLURAL ?B3))
           (GOAL (? COLOR ?B3 RED)$TRUE $DEDUCE)
           (GOAL (? GIVE PAST ?B3 JACK1 BALL1)$TRUE
$DEDUCE))
```

Of course, we may not be able to determine, say, "the ball". As we mentioned in chapter 3, when this happens we put in a variable which is restricted to the members of the PRL for the NP. The PLANNER expression would then be:

```
(6.9)    (FIND ALL ?B3
           (B3 ($R B4 (MEMQ '(BALL1 BALL3))))
           ...
           (GOAL (? GIVE PAST ?B3 JACK1 ?B4)$TRUE $DEDUCE))
```

It may be that our description is not sufficient to determine the referent, in which case the next type of information to be used is "recency". If one possible referent was used recently (say in the last five sentences) and all the others were last used considerably prior to that (say at least ten earlier) we can be reasonably sure that the most recent one is the intended referent.

6.2 Syntax Information for Reference

The "she" in

(6.10) She hit the girl next to Mary.

cannot refer to "Mary". This phenomenon is syntactic in nature, in that it does not seem to depend on context or meaning. The questions we would like answered are 1) what are the rules which govern this phenomenon, and 2) how can we incorporate these rules into our procedures for determining reference?

The Lees and Klima Rules

The answer to the first question is still not known, despite the labors of many linguists. It turns out that the behavior of pronouns is quite difficult to explain when complex sentence structures are considered. However, all the cases which come up in children's stories can be accounted for with two rules postulated in (Lees and Klima 63).

1) If a particular object is referred to twice within the same "simple" sentence, the second occurrence must be made a reflexive pronoun.
2) If a particular object is referred to in both the "matrix" sentence and an embedded sentence, the reference in the embedded sentence must be pronominalized. (The matrix sentence in (6.10) is "She hit the girl.", the embedded sentence is "The girl was next to Mary.")

Hence in (6.1) if "she" did refer to Mary, by rule 2 we would have had to make it a pronoun. These two rules are expressed in a generative format. That is, they say what to do when we are generating the sentences. We now want to use them in our interpretive context. As a start, we can translate the Lees and Klima rules into the following interpretive pronoun rules (IPRs).

IPR1) If a pronoun is reflexive, it must refer to previous NPs in the same simple sentence (call it SS1).

IPR2) Else, remove from the PRL
  a) all NPs in SS1
  b) All non-pronoun NPs in sentences which are embedded in SS1

IPR1 covers "He washed himself". IPR2a covers "He gave him the ball" saying that "he" and "him" are different. IPR2b covers "He hit the girl next to him" and allows "he" and "him" to be the same, but would not allow "He hit the girl next to Jack" to have "he" and "Jack" the same.

Using the Rules

What we must decide next is when to use the IPR's. When we discussed "definitional knowledge" this question never came up, we just assumed that we used definitional information first. With our IPRs we have to be more careful. Suppose we assume that we apply the IPRs as soon as we see the pronoun. What would we do with a sentence like "He hit Jack"? We want to remove Jack from consideration as a possible referent for

"he", but when we first see "he" we haven't yet seen "Jack", much less figured out to whom this name is referring. What we need then is a converse to IPR2. It would read:

IPR2-CONV) If we know the referent for an NP in SS1:

a) Unless the NP is a reflexive pronoun, remove the referent from the PRLs of all other NPs in SS1

b) If the NP is not a pronoun, also remove the referent from the PRLs of all NPs in SS1 or sentences which embed SS1.

In this way, when we find that "Jack" refers to JACK1 (in "He hit Jack"), we removed JACK1 from the PRL of "he".

We have assumed that the rules for pronouns should be applied when the pronoun is encountered. Suppose that instead we wait and first apply definitional information to all the NPs in the sentence. We could then go through all the NPs and just apply IPR2-CONV to the ones whose referent had been determined. This scheme has the advantage that we can eliminate IPR1 and 2.

The program uses the method first proposed (applying IPR (1 and 2) when the pronoun is first encountered) because that is the method I thought of first. While this new scheme seems to offer a slight simplification, there are potential problems with it. However, the issue does not seem to be very important at this stage of development, so we will skip it.

Other Work on Syntax and Co-referentiality

The Lees and Klima Rule is only sufficient for the simplest sentence constructions. Consider the following sentences, in each case interpreting "he" and "Jack" as coreferential:

(6.11) Jack went to the store before he had breakfast.
(6.12) Before he had breakfast Jack went to the store.
(6.13) *He went to the store before Jack had breakfast.
(6.14) Before Jack had breakfast he went to the store.

Now a quick look back at our Less and Klima formulation will show that it is not sufficient to account for (6.11) - (6.14). In particular it does not predict that (6.13) will be bad. Rules which account for both our earlier examples and (6.11) - (6.14) were found independently by Ross (Ross 69) and Langacker (Langacker 69). The basic idea is that a pronoun is not allowed to appear earlier in the sentence than its referent if it "commands" the referent. ("Command" is a term invented by Langacker.) Basically A commands B if the clause which contains B is a portion of the clause which contains A. In (6.13) the "he" is in the main clause which contains the subclause "before Jack had breakfast". So in this case "he" both commands and precedes "Jack".

But things get even more complex. So in (Lakoff 68), Lakoff comes up with examples like:

(6.15) *In John's apartment, he smokes pot.
(6.16) In his apartment, John smokes pot.

Again, the * on (6.15) is for the interpretation with "he" and "John" coreferential. In this case our "command" rule does not predict that (6.15) is incorrect. (Note 11)

So far we have only looked at "he and "she". If we consider pronouns like "each other" or "both," things get even more complicated as shown in (Dougherty 69).

6.3 Introducing New Objects

There are times when an NP does not refer to a previously mentioned NP, namely when the NP introduces a new object into the story. Naturally this happens often. So in

(6.17) Jack and Janet went to the store.

we must introduce three new objects, Jack, Janet and the store. There is some question about how to tell when an NP is introducing a new object. As a first approximation we will assume that if, after applying definitional information there are no possible referents, then the NP is new. We shall see, in sections 6.4 and 6.5 that this is not necessarily the case, but we can worry about the exceptions then.

Once we have decided that the NP is new we want to

a) create a new symbol for it, and

b) assert all the properties of the new symbol (e.g. that it is "a top" and "blue" etc.).

Generating a new symbol is easy; part (b) is slightly harder. Here the idea is to take the PLANNER program constructed as in 6.1 and extract from it a list of appropriate assertions with the new object substituted for the variable. So if we took (6.4) which was the expression to locate "the red ball" we would want to get:

(6.18)
```
(? IS BALL2 BALL)
(? COLOR BALL2 RED)
```

The other two facts,

(6.19) (NOT (PLURAL BALL2))
(SPECIFIC BALL2)

should be interpreted immediately and the facts would be placed on the property list of BALL2.

These assertions would then be added to the front of the list of assertions to be handed over to DSP, and would be processed like any other assertions. In particular, they would be put into the data base, and the base routines would be allowed to entail other assertions where appropriate. So "Jack" would become:

(6.20) (? NAME JACK1 JACK)

but NAME-BASE would entail:

(6.21) (? IS JACK1 PERSON)
(? SEX JACK1 MALE)
(? AGE JACK1 YOUNG)

6.4 Over-Specified Noun Phrases

So far we have been concerned with the application of knowledge to NPs where the definitional information was not sufficient to determine the correct referent, i.e., when the PRL had more than one member. There is another possibility, however, when we are done with the definitional information, there might be no possible referents. This can occur for two reasons. Possibly a new object is being introduced into the story, or it might be that the noun phrase refers to a previously mentioned object, but contains new information about it. We will be concerned in this section with the latter situation.

We have seen an over-specified NP (henceforth referred to as OSNPs) already in the first spoken line in the story in the introduction, "Janet, see the paints and pencils which Daddy got for us!" The previous line said that Jack was holding paints and pencils, but we did not specify where they came from. So a system as we outlined in section 6.1, finding that it knew of no pencils which Daddy got for Jack and Janet, would promptly decide that the NP described some object that it hadn't heard about yet. Of course, this is not the conclusion we want to reach, so something has to be done.

To my knowledge OSNP's have not been studied to any great degree in the literature on reference. In fact, I am only aware of one study (Vendler 67) which even considers the

possibility at all, and he seems to indicate that an OSNP can never refer back to an earlier mentioned object, though he is not completely clear on the point.

Some Easy Solutions That Don't Work

One way out might be to say that subordinate clauses actually act to give us new information about the object, rather than to specify it further, and even in the cases where it was used as specification, we could probably do without it, depending instead on DSP to find the correct objects.

The trouble with such a scheme is that not only do we lose the often considerable selective power of subordinate clauses, but the problems that motivated this scheme still persist, as indicated by the following story fragment.

(6.22) Jack and Bill went outside to fly a kite. After some trouble they managed to get it to fly. Suddenly the string broke and Jack said, "My new kite is flying away."

In "my new kite" the over-specification comes not from a subordinate clauses in "the paints and pencils which Daddy got for us", but rather from the adjectives "my" and "new".

The next simplest strategy is that in the cases where we have over-specified the NP, we can go back and find the object which satisfies the largest number of the attributes mentioned in the NP. This would work for both the examples we have given, but would produce nonsense if we changed the last line of (6.22) to read:

(6.23) "That is OK" said Jack. "My new kite is at home."

Here over-specification is used to indicate that the object mentioned is not the one we were previously talking about. Note that if the story had gone

(6.24) "That's OK" said Jack. "The kite is at home."

the reader would be quite confused. In much the same way, the first story line in chapter 1 could have been

(6.25) "Janet, the paints and pencils which Daddy got for us will be here tomorrow."

and we would have understood the referent quite differently.

Do We Look for Positive or Negative Information?

What seems to be happening is that there is something in the sentence which either hints that the OSNP is the one we are familiar with, or hints that it isn't. So when we are told that the kite is flying away, we are not surprised since we know that the string broke. In (6.23) we would suppose that if the kite is in the house it cannot be the one which we know to be outside. The question then becomes, do we assume that the object is the one we know about unless we have reason to believe that it isn't, or do we assume that it is a new object unless we have reason to suspect that it is the old one? (We shall call these the conservative theory and the radical theory respectively.) The evidence is not particularly strong in either direction. Suppose the first spoken line of the chapter 1 story had been:

(6.26) "I like the paints and pencils which Daddy got for us." said Jack.

"See" as an imperative often implies that the object is there to be seen, so it offered reasons to believe that the paints and pencils were there. "Like," on the other hand, has no such implication. The trouble is that since we said that the paints and pencils are for Jack and Janet, we have reason to suspect that he has them, though it is not completely clear. However, let us change (6.26) to read:

(6.27) "I like the paints and pencils which Bill got from the store"

I still read it as meaning the ones he is holding though the sentence is almost a non sequitur in the story context. However, the evidence seems to be leaning toward the conservative theory. But, as we shall see later, a complete theory, even one based on the conservative theory , will also have to accomodate positive information. That is, information which says "the sentence is better if the referent is X".

The Double Negation Technique

Having decided on this, let us make a first attempt to specify exactly how we will decide that a given OSNP either may or may not be a previously mentioned NP. One thing we can be sure of; if the NP says it's a ball, we can restrict consideration to balls. So we can create a tentative PRL consisting of all objects which have the head "ball". (Note

12) In particular, if there aren't any previously mentioned "balls" we may assume that the NP is new and treat it as specified in 6.3.

Now if our NP were "the red ball" we certainly would not accept green balls as possible referents. This suggests that we want to go through the other modifiers and "loosen" their hold, but not to the extent that we allow in objects which contradict the modifier. So we might replace

(6.28) (GOAL (? COLOR ?B5 RED) $TRUE $DEDUCE)

with

(6.29) (NOT(GOAL (? COLOR NOT ?B5 RED) $TRUE $DEDUCE))

The latter statement says that we should not accept obects which can be shown not to be red. Of course, for this to work, our program will need to know that if an object is some color other than red, it is not red, but it would have needed to know this anyway. For future reference, we will call this the "double negation" technique.

As our examples have shown, the place where we run into trouble might not be in the NP proper, as in example (6.23), where we knew that it was a different kite because of the stated location. This implies that we have to apply our double negation technique to the main assertion(s) which contain the NP. In this way for (6.23) we would not consider any kites definitely known not to be in the house, e.g., the kite Jack and Bill were flying. To restate, applying our

double negation technique to the statement that the kite is in the house gives us:

(6.30) (NOT (GOAL (? IN NOT ?B7 HOUSE1) $TRUE $DEDUCE))

However, we would be able to establish that the kite being flown is not in the house, so it would be eliminated from consideration. Presumably in the case that all the possible referents are eliminated, the program decides that a new object is being referred to and creates one as discussed in the last section.

Why DSP Must Come Into Play

Some serious problems remain with this analysis. Consider the following story fragment:

(6.31) Janet was in her room playing with her paints. When she finished, she put them in the drawer and left the house. When she came back, Jack was in the living room. When she walked in Jack said "Janet, see the paints which Daddy got for us!"

Presumably the fact that we noted earlier, that for something to be seen it has to be there, applies here to eliminate Janet's paints from consideration, although we might argue that "see1" implies in some cases that the object is not known to the listener. (In fact, I will argue later that this fact must be the crucial one.) But in either case, if we take a close look at exactly how this is supposed to take place we find things amiss. We said earlier that we would have to apply double negation to the main assertions the OSNP appears

in. In this case that would be the imperative "see". The question we must ask ourselves, if we are going to negate this, is what does it mean to negate an imperative? Or for that matter, what does it mean to derive an imperative, negated or not? The best answer seems to be that we don't want to negate the imperative, we want to see if any of the conclusions which we derive from the imperative are negated. So "see" as an imperative usually implies the object can be seen or is new to the listener. If we can show that the relevant assertion is not true, we will have shown that our OSNP is inconsistent with the imperative. This same point is shown by the following example (Note 13):

(6.32) Jack was outside. He was holding a red ball. Janet came up to him and said, "Why is the beach ball in the middle of the lake?"

We have assumed that entailment is one of the functions of the base routines. If deciding OSNP's depend on entailment, perhaps we want to do something like we did with under-specified NPs (USNP - NPs where we can find more than one possible referent) - hand OSNP's off to DSP in the form of a restricted variable and let DSP do the work.

There is other evidence that this is what we must do. Consider:

(6.33) Jack was outside playing with a ball. Bill came by. "What was that smoke I saw" asked Bill. "My father burned my yellow ball" said Jack. "Why did he do that?" asked Bill.

We do not understand "my yellow ball" as the one Jack is playing with, presumably because if it had been burned it would be useless as a ball. But this, of course, is an entailment of "burn". Unless we allowed DSP to act on OSNPs we would have gotten the last example wrong. (Note 14)

The third example of needing DSP to help with OSNPs is fairly complicated. I will first show that in example (6.31) it is the "newness" criteria which is doing the selection. Then I will argue that the "newness" rule is properly a part of DSP. First let's show why we can't depend on the fact that the paints which Janet was using are not in the living room. Consider:

(6.34) When Jack got up one morning he looked out his window and saw a cat on the fence of the yard. Jack got dressed and went downstairs to have breakfast. After breakfast he put on his coat and went outside. The yellow cat was on the stairs.

Most people understand "the yellow cat" as referring to the one mentioned earlier. But, if we take our double negation theory seriously, then the fact that it is on the stairs, "contradicts" the fact that it is on the fence. The answer is that really there is no contradiction here, the cat had plenty of time to get from the fence to the stairs. But this does indicate that we cannot use the double negation technique unmodified. We will have to change it so that rather than looking for "updates" (which is what it is currently looking for) it will look for real "contradictions" in some sense.

Now, in (6.31) there is no reason for assuming that the paints Jack mentions could not be the same as Janet's paints, purely on the basis of location, since there was plenty of time for the paints to be brought into the living room from Janet's room. So the real influence must be the "newness criterion" (assuming that there aren't others I haven't thought of.)

What I want to show next is that deciding when a "seel" implies that the object is new to the listener is not trivial, and requires a degree of semantic processing which could only happen in DSP. To see this, consider:

(6.35) When Jack walked outside, Janet and Tip were there. Janet said "Look Jack! See Tip!"

(6.36) Janet had taken the large box from the basement and painted it bright colors. When Jack came into the house she said "Look Jack! See the box!"

(6.37) Jack went into the house. Janet was in the living room with a large box. She said "Look Jack! See the box!"

In (6.35) and (6.36) we do not assume that Jack is not familiar with the object. In (6.35) we simply know that Jack already knows of Tip's existence, whereas in (6.36) the combination of the fact that the box was previously in the basement, and that Janet has another reason for calling Jack's attention to it, lead to not asserting that it is new to Jack. In (6.37) however, since we have neither of these reasons, we assume that it is new to Jack. Such analysis seems to me clearly part of DSP, so we have satisfied the second half of the argument

We mentioned earlier in passing that we would need to include positive information in the process also. Consider:

(6.38) Jack went out to fly a kite. After a while he managed to get it up. Bill came by and watched. Suddenly Jack shouted "My new kite is flying away."

Presumably we will have a demon looking for such a situation, since it is one of the problems involved in kite flying, and one thing this demon should do is assert that Jack no longer has control over the kite. Given our current model, since this contradicts the fact that Jack has control over the kite, we would conclude that it must not be the same kite. The irony of the situation is the fact that the demon which asserted that he no longer had control would only work provided that the kite referred to was the same kite. The solution fortunately is easy. We simply allow demons the same selection power over OSNPs which we give them over USNPs. In fact, we can make the two look almost identical to DSP. When we realize that we might be dealing with an OSNP we make up its PRL and restrict the variable exactly the same way in both cases, except, with an OSNP we add one extra possible referent, a new symbol representing the case that in reality the NP is not meant to refer to a previous one at all, but really introduces a new object.

How to Handle OSNPs

Let us now summmarize how we shall treat OSNPs.

1) We recognize the situation when the "find all" created in descriptive information finds no suitable objects.

2) Create a PRL of all NPs which match the head noun. Then take each modifier in turn and remove all NPs from the PRL which conflict with that modifier. Eventually, one modifier will cause all the NPs to be deleted from the PRL. Call all modifiers prior to this one "tested" modifiers. This last modifier, plus all remaining ones are the "untested" modifiers.

3) Re-instate the PRL prior to the point when all NP's were deleted.

4) Make up a new symbol for the possible new object, and add it to the PRL.

5) All the tested modifiers should be asserted about the new symbol, so that DSP will know the appropriate facts about it. However, these assertions will not be processed by DSP themselves unless we decide that the object is a new one, in which case they will be processed as in the "new object" situation.

6) All untested modifiers plus any other assertion in which the OSNP appears will have the double negation test applied to it, and will then be handed off to DSP for normal processing

7) If in DSP we generate any statement assertion which contains an undecided OSNP we will apply the modified

double negation test to it immediately.

8) If the NP is decided to be an old one, the assertions generated in step 5 will be erased, otherwise they will be processed by DSP as mentioned in step 5.

Note in particular that really, except for step 7, DSP need take no account of whether a particular NP is over or under-specified.

6.5 Determining Indefinite Reference

The Existence of Non-Specific Objects

In chapter 3 we noted the distinction between the definite NP "the ball" as in:

Jack was holding the ball.

and the indefinite NP "a ball" as in:

Jack wanted a ball.

There is a real distinction between definite and indefinite NP's. Consider:

(6.39) Jack wanted a kitten. Bill had a kitten and Jack offered to trade his ball for the kitten. Bill wanted to keep his kitten, so Jack went to look for George who also had a kitten. George was willing to trade so Jack got his kitten.

In this example the "a kitten" in the first line cannot refer to any particular object, not even a particular object which is not yet known to us. Suppose it did refer to a particular object. Then when Jack offered to trade with Bill, in order to answer the question "Why does Jack want to trade" we would have to assume that the kitten Bill has is the particular one Jack wants. Of course, then, when Jack offers to trade with George we are in a bind since then we need to assume that it is really George's kitten that Jack wanted. Naturally, the answer is that Jack does not want any particular kitten. Rather, he wants one of a class of objects called "kitten". It so happens that both Bill and George have objects which are

members of that class.

The Relation Between (Non)Specifics and (In)Definites

We will call the class object a "non-specific" object. The alternative is a "specific" object. Specific and non-specific are "semantic" in the sense that they refer to the meaning of the referent. Definite and indefinite are "syntactic" in that they refer to the form of the noun phrase. In this section we will assume that objects can be marked +,- specific. Nevertheless, there is evidence that this is not exactly true (Karttunen 71), (Jackendoff 71). However, this simplification should have no effect on the basic points in this section.

The distinction between specific versus non-specific objects is often mirrored by definite vs. indefinite. This, however, is not always the case. In fact we can find exceptions in both directions as illustrated by:

(6.40) Janet said "I want a ball. The ball must have a blue stripe."

(6.41) Janet wants to marry a man who plays the violin for the Boston Symphony Orchestra (BSO).

In (6.40) the "the ball" refers not to any particular ball, but rather to the "a ball" mentioned in the first line. We could have replaced "the ball" in (6.40) with "my ball" or "this ball" and we would have still understood it the same way. (In passing we should point out that the existence of examples like (6.40) indicates that non-specific objects

should also be represented by unique internal symbols.) In (6.41) on the other hand we understand the "a man who plays violin for the BSO" as a particular person, while in

(6.42) Janet wants to marry a man who plays the violin

the "a man who plays the violin" could be either a class reference, or a particular person.

So the "indefinite reference problem" really becomes two problems. First, when is an NP non-specific? Second, finding the object to which the NP refers. Prior to this point we have only considered examples in which all the noun phrases were definite. Now we want to expand our scope to handle indefinites.

When Does an NP Refer to a Non-Specific Object? The Setting Constraint

An indefinite NP can only be a class referent in certain settings. So in

(6.43) Janet picked up a ball.

we recognize the "a ball" as being a particular ball. Those interested in an exact formulation of what "settings" do and do not allow non-specific interpretations might consult (Jackendoff 71). For children's story purposes we should recognize that some settings which allow class objects are modals, negation, questions, imperatives, and certain verbs which indicate that the indicated activity has not occurred, e.g., like, want, intend. These are illustrated respectively

by:

(6.44) Jack might get a ball.
(6.45) Jack does not have a ball.
(6.46) Does Jack have a ball?
(6.47) "Jack, get a ball!"
(6.48) Jack wants a ball.

On the other hand, straightforward past and present tense statements generally do not allow class interpretation as in "Jack had (has) a ball". We will refer to settings which allow non-specificity as non-specific settings. Those which don't will be call specific settings.

Semantic Considerations in Determining Specificity

We have already noted that when an indefinite NP does not refer to a prior object it may still be either specific or non-specific as indicated by:

(6.49) Janet wants to marry a man who loves music.
(6.50) Janet wants to marry a man who plays the violin.
(6.51) Janet wants to marry a man who plays the violin for the BSO.

Most people consider (6.49) non-specific, (6.50) ambiguous, and (6.51) specific.

These examples are misleading in that they seem to indicate that the number of people who satisfy the description is the relevent factor. While this is probably a factor in some cases, it does not work across the board, as in:

(6.52) I want an early Rembrant etching.
(6.53) Janet wants to marry a Beatle.

While I don't know how many early Rembrant etchings there are,

for all I know it could be less than the number of men playing violin for the BSO. Certainly the four (ex) Beatles number fewer than the violin players in the BSO.

This is an interesting problem, but it is not one which is crucial to children's stories. In our simple stories we can (and will) assume that any new indefinite NP in the proper setting will be non-specific.

Determining Reference - Definite NPs

We saw earlier that only non-specific settings allowed an indefinite NP to be interpreted as non-specific. The other way to look at this is that specific settings force the indefinite NP to be interpreted as specific. The same is true for definite NPs. Consider:

(6.54) Janet wanted a ball. She would give it to Jack as a present.

(6.55) *Janet wanted a ball. She gave it to Jack as a present.

The difference is that "would" creates a non-specific setting so the "it" can be the non-specific "a ball" introduced by the first line. On the other hand, the past tense "gave" creates a specific setting so the "it" must be a specific object. Since the only object around is the non-specific "a ball", the line sounds quite strange.

Actually, what makes (6.55) strange is not especially dependent on the "ball" in the first line being non-specific. Consider:

(6.56) *Janet wanted the red ball in Macy's window. She gave it to Jack.

In both (6.55) and (6.56) the problem is that one simply can't give something away which one dosen't have. But note that one can "get" an object which one does not have, hence we find examples like the one in our story of section 1.2.

(6.57) Janet said "I must get some money for Jack." Soon she came back with it.

Here the "it" is in a specific setting and is accordingly interpreted as specific. However, the "it" is also interpreted as being "money" which is "for Jack". To use conventional terminology, we would say that the "it" shared "identity of sense" with the "some money", but not "identity of reference".

Given the above analysis, we can now extend the reference procedure for definite NP's to cover the possibility of non-specific referents.

1) In early referent analysis, do not bother to distinguish between specific and non-specific referents. Put both on the possible referent list.

2) Allow normal selection mechanisms in base routines and demons to operate. So, for example, the consistency check in GIVE-BASE will prefer objects which the giver has, hence explaining (6.55) and (6.56).

3) Just after applying the "last mentioned" rule, see if we have a non-specific referent in a specific setting.

If we do, create a new specific object which will replace the non-specific referent in all new assertions.

4) At the same time, take all facts known about the old non-specific referent, and assert as facts about the new specific referent. This will account for the fact that "it" in (6.57) is both "money" and "for Jack".

How Do Indefinite NP's Refer?

Before we get started on extending our reference procedure to handle indefinite NPs we should first decide in what sense indefinite NP's refer. (Actually, we will only consider indefinite NPs with heads, so we will not consider NPs like "some" or "one". Some of the problems connected with "one" will be discussed in 6.6.) If we are presented with a line like "Jack bought a ball" in isolation, we create a new specific object BALL1, say. So indefinite NPs can refer to newly created objects, but can they refer to previously mentioned objects? If they do, it is with far greater difficulty than definite NPs. Consider:

(6.58) Janet was playing with her top. Jill came along and they were both playing with it for a while. "That was fun," said Jill. "I want to get a top."

(6.59) Mother and Jack were in the living room. Since the ceiling light was off, when the sun went down it became dark there. Jack said "I will turn on a light."

(6.60) Janet was in the house playing with a ball. Some time later when mother came home, Jack was outside playing with a ball.

(6.61) Jack wanted a green ball. Bill wanted a green ball also.

In (6.58) and (6.59) we have indefinites ("a top" and "a light" respectively) with possible referents which are definite ("Janet's top" and the living room light). In both cases however we do not assume that the definite is the object referred to. In (6.60) we have a syntactic indefinite and semantic definite with a possible referent which is also syntactically indefinite and semantically definite. Again we don't assume that the second "a ball" refers back to the first. In (6.61) we have two semantic indefinites, both being "a green ball". It would appear that assigning the same internal symbol to both NP's would get us into trouble if our example (6.61) went on to say:

(6.62) Jack wanted his to have a red stripe. Bill's should have pink dots.

However, there are cases where it is reasonable to assume that an indefinite is referring to a prior NP.

(6.63) Janet wanted to get a dog. When Ms. Jones gave her a dime, Janet said, "I am saving my money. I want to get a dog."

(6.64) Jack traded a yellow ball for Bill's top. When Jack got home, Mother asked him where he got the top. "I gave Bill a ball for it" said Jack.

(6.65) Jack and Bill were outside flying a kite. A strong wind came by and the string broke. Janet and Alice were outside the house. Janet looked up and said, "Look Alice, there is a kite flying away."

In (6.63) there seems to be no reason for not using the same symbol for both occurences of Janet's wanting to get a dog. In fact, not to use the same symbol would seem to imply that there are potential differences between the dog she wanted in the first line and the one in the last.

On the other hand, there are potential reasons for assigning a new symbol to the "a ball" in (6.64). For example, we could ask, "Does Mother know that the ball which Jack traded was yellow?" If we created a new "a ball" then we would have associated with it only those properties which Jack tells Mother (plus some indication that the two balls are really the same.) On the other hand, we could equally well solve this problem by keeping the same internal symbol, only including the modifiers in the data base description of the sentence. Since we will see in the next section that we will need the modifiers anyway, we will adopt the latter solution.

Bringing DSP into the Process

Both (6.63) and (6.64) can be handled by our standard procedure of looking for data base redundancies during bookkeeping. (That redundancy checking should be done by bookkeeping is so far just an assumption. In chapter 7 I will try to indicate why this should be the case.) Example (6.65) on the other hand seems to depend on our expectation that once the string breaks the kite will fly away. Hence it is a demon

which would assign the "a kite" in Janet's sentence to the one Jack and Bill were flying.

Furthermore, the base routines also come into play. Suppose the first line of (6.63) were changed from "Janet wanted to get a dog" to "Janet wanted a dog", the latter being translated in the data base to something like "Janet wanted to have a dog." The responsibility for bridging the gap between "get" and "have" falls on the base routines. (That is, "get" entails a "have" assertion, and it is the latter which matches up with the new first line of the story.) The conclusion is that, just like definite NPs, indefinite NPs should be left open (with a variable holding the place) so that DSP can work on the assertion.

The "Immediately Aware Of" Rule

As we initially pointed out, indefinites do not refer back to previously mentioned NPs with the "ease" of definite NPs (as illustrated by examples (6.58) - (6.61)). While it might be possible to account for those examples by simply not using the "last mentioned rule" on indefinite NPs, other examples seem to show that we need something like an "inverse last mentioned rule". Consider:

(6.66) Jack offered to trade his kite for Janet's top. Janet said "That is a nice kite, and I do want a kite."

In this example, a demon would presumably pick up the "I do

want a kite" and indicate that the line sheds some light on the possible outcome of the trade offer. But the trade offer, and hence the demon, is concerned with the particular "kite" which Jack has. Hence if the "a kite" were a variable, it would be assigned to the kite which Jack has. But we understand the "a kite" as a non-specific, applying to kites in general, not simply Jack's.

The solution to this problem comes from considering why, if a person is referring to an object which was previously mentioned, would he use an indefinite, rather than a definite NP? The intuitive answer is that the speaker is not sure that if he used the definite form the listener would be able to identify the referent correctly, or that the object in question, while known to the reader, is new to the speaker. We saw an example of the latter in (6.64). In (6.64) while Jack knew which ball he had traded, he did not expect Mother to be aware of it.

Actually, the speaker can know that the listener is aware of the object and still use the indefinite, as in:

(6.67) Jack came home with a top. Mother asked him where he got it. "I got it from Bill in exchange for a ball. It was the beach ball which you said I should throw out."

Suppose we had already encountered both the conversation between Jack and Mother where she told him to throw the ball out, and the trade between Jack and Bill. We would still be able to have a conversation like (6.67) in spite of the fact

that we know that Jack knows that Mother had previous aquaintance with the traded ball. So we really want to know if the listener is "immediately aware" of the object in question.

To account for these cases, we will add a rule which says that if a specific NP has occurred in a conversation, it cannot be the referent of any indefinite NP later in the conversation. This can be checked simply by looking for the last time the object was mentioned in conversation (or was physically present) and making sure that (1) it was in the current scene (no "gaps" in the events separating then and now), and (2) both speaker and listener were present at the time. (Note 15)

Given that (6.67) seems to indicate that the "immediately-aware-of" rule has precedence over demons, the application of the rule should be during early referent analysis.

Some Possible Difficulties With "Immediately-Aware-Of"

As formulated there are several problems, or potential problems, with our immediately-aware-of rule. For example, when we get to more complicated stories, we could have a situation where the story jumped back and forth between two conversations, both mentioning a particular object X. In conversation B we note that the last time X was mentioned

neither participant in B was present (because it was mentioned in conversation A). Needless to say however, this would not be relevant to whether the participants in B were immediately aware of X. So we will ultimately need to refine our definition of conversation.

There are also problems with examples like the following, which is taken from our opening story (section 1.2).

(6.68) "Now you will have some money, a toy cat, and all these pencils."

This was said by Janet near the end of the story in her last attempt to get a trade. At this point Jack was clearly immediately aware of both the money, and the piggy bank (the toy cat), yet Janet used indefinites. The question we must decide then is whether it is correct to assume that "a piggy bank" in (6.68) should have identity of reference with the piggy bank in Janet's hand. The alternative is that we interpret the "a piggy bank" as non-specific, and then realize that the PB in her hand is one possible object which she could be referencing. In concrete terms, if we assume indentity of reference, then the internal representation of her sentence has her speaking of the particular piggy bank. If we assume that "a piggy bank" is non-specific, then the internal representation of the line talks of a non-specific piggy bank, and we must derive new assertions which state that she is "really" offering the PB in her hand. (We ultimately need to make the connection in order to answer a question like "Why

did Janet get that PB?")

Since I know of no good evidence to decide between the two possibilities, I will assume that the "a piggy bank" is non-specific, since that allows keeping the "immediately aware of" rule intact.

One other potential problem with the "immediately aware of" rule deals with how it applies to the narrator. Presumably, the narrator (and reader) are "aware of" everthing which happens in the story. (We are talking of the "intangible" narrator who says things like "Jack and Janet were outside the house".) So one would expect that the narrator cannot use an indefinite to refer to a non-specific previously mentioned object.

In some cases, like (6.60) there is some doubt whether the newly mentioned non-specific indefinite is the same as a previously mentioned object. In these cases we normally note the doubt, but assign the indefinite to a new object since it seems clear that the narrator must at least intend us not to be sure whether the object is the same or not. There are other cases however which are much more difficult. Consider:

(6.69) Jack and Bill were flying a kite. A strong wind came by and the string broke. Janet and Alice were outside Janet's house. When Janet looked up she saw a kite.

My personal interpretation is that we are to assume that the "a kite" in the last line is the same kite which Jack and Bill were flying. However, if this is the case, it would break the

"immediately aware of" rule since both narrator and reader are aware of that kite. What seems to be happening is that the narrator is "empathizing" with Janet's "unawareness" of the kite, and hence uses the indefinite form. How we should handle this problem is another open question.

Summary of Indefinite NP Reference

1) Do early referent analysis without regard to the specificity of the possible referents.

2) Remove all specific NPs which fail the "immedately aware of" test. Create a new object as described in section 6.3. If the setting is specific, then the object should be specific, if non-specific, then non-specific.

3) Go through DSP as with definite NPs except the "last mentioned" rule should not apply.

4) At some point after we have checked for redundancy, if the indefinite NP has still not been assigned to a referent, assign it to the new object created in (2).

6.6 Further Topics

There are many topics which have not even been mentioned. For example while we understand the constraints that "it" places on possible referents, "this" and "that" are not so clear. Broadly speaking, "this" means "associated with the speaker" and often refers to an association based on distance, but it does not have to, as in "I don't like this situation."

Another ignored problem is sentential reference, as in "Don't do that". To give some idea of how complex this can be consider:

(6.70) A goat wandered into the yard where Jack was painting a chair. The goat kicked the paint bucket and got it all over himself. When mother came out she saw the goat and asked, "Jack, did you do that?"

There is no one line in the story which is the referent of "that" in the last sentence. It seems to refer to something like "cause the goat to be covered with paint".

However, for the rest of this section we will only look at a few problems which are closely related to issues which came up in this chapter.

"One" and "Some"

Looking at "one", ("some" has similar problems,) let us try to extend the procedure for handling syntactic indefinites in the most straightforward manner possible. We will create several new objects, each corresponding to the "description"

of a possible referent.

(6.71) The boys were going to play ball at the park. Jack brought a bat. Bill brought a ball. Sam brought one also.

In (6.71) initially the "one" may be either "a bat" or "a ball" and we will create two new objects, one for each possibility. Eventually we will decide on "ball" because it was most recently mentioned. (This seems to indicate that while LMR does not apply on syntactic indefinites for indentity of reference, it does come into play for identity of sense. This would be a simple change to our theory.) But there is a deeper problem here.

What constitutes the "description" of the previous object? A good rule seems to be "anything mentioned in the previous NP except for any information carried by a word which also acts like a determiner. So we see:

(6.72) Jack brought a soft yellow ball and Bill brought one also.

(6.73) Jack brought his soft yellow ball and Bill brought one also.

(6.74) Frank said to James, "Jack found that soft yellow ball, and Bill found one also."

In all three we understand "one" as meaning "a soft yellow ball". This means that in (6.73) we ignored the fact that the first ball belonged to Jack, and did not interpret the "one" as meaning that the one Bill brought also belonged to Jack. In (6.74) we interpret the one Jack found as being right there with James and Frank. The one Bill found is not interpreted

that way. Such examples are the justification for the section of the rule which says to ignore any information carried in a word which also acts as a determiner. However, if we keep looking at such examples we quickly find that things get very complicated. All sorts of ambiguities arise.

For example:

(6.75) Jack brought a sword which had belonged to his great-great-grandfather. Bill brought one also.

I find the last sentence at least three ways ambiguous. The "one" which Bill brought might have been originally owned by Bill's great-great-grandfather, perhaps by Jack's great-great-grandfather, or it could simply be a sword with no implications whatever about who owned it first. One can clearly get these readings by preceding (6.75) with the following:

(6.76) The fencing committee was worried that not enough people would bring swords to enable the match to take place. However,...

(6.77) The topic of the historical society meeting was early American weapons.

(6.78) The topic of the historical society was Jack's great-great-grandfather who was a captain during the Civil War.

How we will handle such examples is another topic for future work.

As mentioned earlier, we will need information about what the last NP specified about its referent. This is important as seen in:

(6.79) Jack owned a yellow ball. He brought the ball to the park. Bill brought one also.

In this case the "one" is still a ball, but it is not necessarily yellow. In our model we have made no provision for storing information such as what an NP specified about its referent.

The Structure of Conversations

When we said that pronouns referred to some object mentioned recently in the story, we arbitrarily set a limit of five lines on how far back we would look. While in most cases this limit is more than adequate it is not too difficult to make up an exception. Consider:

(6.80) Janet, Bill, and Bill's sister Helen were outside. Janet said, "I can't keep this kitten. Would you like to have it, Helen?" "Yes," said Helen. Bill said, "I don't know. Remember how Mother objected to that robin. She would not let us keep it." "But Mother said that it is not good to keep a robin indoors," said Helen. "It is not fair to the robin". "Look," said Janet, "Do you want it or not?"

In (6.80) the last sentence to mention the kitten was seven lines prior to the last line. Furthermore we could extend Bill and Helen's conversation quite a bit and still would not have any trouble interpreting the "it" from the last line. What is operational here is the fact that we have a "sub-conversation" going on, and Janet's comment brings us up out of it. In such a case the length of the subconversation

is not important. This, of course, is not a theory, just an indication that one is needed. We must specify the new pronoun rule, and explain how we will recognize sub-conversations.

7 The Rough Organization

In chapter 4 we outlined the order of execution of the various parts of DSP. The order we gave was

Demons
Base Routines
Bookkeeping

In section 7.1 we will try to justify this ordering. While this organization has worked fairly well so far, we will see examples in 7.2 where a base routine cannot complete its work because it does not know the referent for a noun phrase. Such examples have caused us to divide each base routine into two parts. The new part will be ordered after bookkeeping. This would give us:

Demons
Base Part 1
Bookkeeping
Base Part 2

The primary difference between the two parts of base routines will be that the first section will not be able to depend on knowing the referent of a particular NP and the specific meaning of a potentially ambiguous predicate. (This is necessary since, for example, many referents will still be undecided at this point.) By the time control get around to the second section these things will be known. Part one will assert new assertions (like TRADE asserting two HAVE assertions) and activate any demons which are put in by LOOK-BACK. Base routines part 2 activates all other demons

and does deductions which depend on knowing the NP referents.

In section 7.3 we will take a closer look at bookkeeping. The basic issue we will confront is whether the base routines for verbs like "trade" or "eat", which cause a change in the world, should be responsible for their own updating. This issue comes up because we can show that "trade", for example, must look to see if the person doing the trade really has the object. Hence if the base routine has gone this far, why not go just a bit further and update the assertion right then and there? We will not adopt this idea, however, since it seems to lead to real problems in handling reference.

7.1 Ordering the Sections

In this section we will try to show that it matters in what order things are done. Our arguments will be based on the model of referent determination presented in section 3.3.

Before one can speak of "ordering" sections of a process, one must believe that the process does indeed break up into those sections. In chapter 2 in particular, and throughout the thesis in general, we have tried to justify the "reality" of our four part division of DSP. Note however that saying that the model does something which corresponds to demon application is not to say that there is necessarily any place in its flow chart to which we can point and say "That is where it looks for applicable demons." Instead we might have twenty different kinds of demons, each checked at a different point in the process. Now in working on a model of the sort we have here, one always starts with simple assumptions, and only adds complexities when they are clearly justified. In the present case there is only one kind of demon, and it does make sense to talk of the "time" when the process looks for applicable demons.

However, this makes the whole argument seem to rest on the correctness of the assumption that DSP will never get more complicated. Since models in this domain certainly will increase in complexity, we would seem to be left on shaky footing. We shall argue, however, that the major points are

largely independent of the complexity of DSP.

First let us recall the scheme for definite noun phrase referent determination which we gave in chapter 3. The idea was that if early referent analysis was unable to narrow the choice down to one, we would substitute a variable in the assertion and allow portions of DSP (demons in particular) to assign the referent. However, if a referent were not assigned, we would finally just pick the most recently mentioned of the possible referents (that is, we would use the last mentioned rule, LMR).

This model was elaborated in chapter 6 in order to account for indefinite and overly specified noun phrases. Nevertheless our ideas about definite noun phrases have remained intact, and it is this scheme we will be using in our argument. Accordingly, throughout this section whenever we refer to "referents" we mean referents of definite noun phrases.

Given this reminder, we can now outline the points we want to make.

(7.1) There is a time in the process by which we must know all the referents, or at least be able to say that given the information we have so far in the story it is not possible to decide.

(7.2) There are certain processes (demons in particular) which know enough to be allowed to make "suggestions" about referents. Hence these processes must come before the point mentioned in (7.1).

(7.3) There are other process (bookkeeping in particular) which need to know the referents of NP's but which cannot be allowed to make suggestions as in (7.2). Such processes must come after the point mentioned in (7.1).

When expressed in this fashion, the ideas of this section become independent of our assumptions about the complexity of our DSP model. So if at any point we wish to add another section to the model we only need to see if it fits the description in (7.2) or (7.3). If it fits both we are in trouble and would be forced to adopt a more complex theory of definite NP reference, or a more complex organization of DSP. (That is, we might want DSP to be organized in such a way that we could "skip around" more in the processing of the sentence, depending on what information we had available. So if we didn't know a particular referent we could go on to other portions of DSP which, while not requiring knowing that referent, might shed some light on its identity. Such an organization scheme has been called "heterarchical" (Winston 72).)

Demons Before Bookkeeping

To argue that demons should be applied before bookkeeping we will show that in general definite NPs must be decided before we can do bookkeeping. That is, we won't know which previous "book on table" assertion to update if we don't know which book fell to the floor. On the other hand there are

cases where we might not know all the referents, but could still update. So if Jack has BALL1 and we are then told that "she" got BALL1, we can update the "Jack has" assertion without knowing the referent for "she". However, starting again with the least complicated model, we will assume that since bookkeeping in general cannot be done until after the referents are known, we can then order bookkeeping so that it will only apply after all definite referents are known.

Since in our current model all definite noun phrases are finally decided by the "last mentioned" rule, we will refer to the LMR as the point before which we cannot be sure of referents, and after which we can. There is however, nothing in the argument which depends on this point being the LMR. Consider the story fragment:

(7.4) Jack was outside. He wanted to play ball, but didn't have one. Sometime later Janet met Mary in the park. She told Mary that Jack really wanted to play ball, but couldn't because he didn't have a ball to play with. "We will make Jack happy" said Mary. "Bill has a ball. He is at home."

Assuming (incorrectly) that bookkeeping comes before we apply LMR, the "he" in the last sentence will still be undecided when we get to bookkeeping, since there is really no evidence indicating if it should be Bill or Jack, except that Bill was mentioned more recently than Jack. But bookkeeping must know who "he" refers to since if it is Jack it needs to update his location, whereas if it is Bill, no such action is needed. In this particular case, if bookkeeping did nothing due to its

lack of knowledge everything would be OK, but (7.4) could end with:

(7.5) "Bill has a ball" said Mary. "We will make Jack happy. He is at home."

Since again we have no way to be sure that "he" is Jack as opposed to Bill, the "he" will still be unassigned when it gets to bookkeeping, if bookkeeping occurs before applying LMR. But if this is the case then we will not be able to update our data base to reflect the fact that Jack is no longer outside. Nor is it fair to object that we really don't know that Jack is in the house, arguing that Mary's information might be old. As we saw in 5.3, (jumping to conclusions about money coming out of piggy banks), there is very little in this world that we can really be sure of, but somehow we go on doing the best we can. Any person reading (7.5) would probably conclude that Jack is indeed in the house unless given reason to believe otherwise. So we need to update.

Since we already established in chapter 3 that demons must be tried before we invoke LMR, we can conclude that demons apply before bookkeeping.

Base Routines Before Bookkeeping

Next we want to show that the base routines come before bookkeeping. The idea here is to show that we can't apply LMR until after the base routines have applied. (Though note that

if we consider base routines a special form of demons as suggested in (2.3) this would fall out automatically.) Consider the following fragment which is a repeat of an example in chapter 3.

(7.6) Janet wanted a nickel which was in her piggy bank. She went to her room for her piggy bank, and she shook it very hard. Finally she got it.

Once again, LMR would guess wrong, saying that "it" referred to the piggy bank, rather than the nickel. But now, note that the last line of (7.6) could have equally well been "Finally she had it" where all we have done is substitute "had" for "got". So the question arises, what is the demon going to look for, "have", or "get"? We could use two identical demons, only differing in that one is keyed to "have" and the other to "get". Naturally this is an unsatisfactory solution. The way around it would be to note that "get" will entail the fact that the "getter" now "has" the object. Or for that matter we might want to say that if we are told that a person suddenly "has" an object it means that he "got" it somehow. But no matter which way we solve the problem, the information that "have" implies get, or vice-versa, will be in the base routines for these predicates. So let's assume that the demon looks for "have money" and when we see "get money" the base routine for "get" will assert that the person now "has" the money. Then the demon for "have money" will pick up this new assertion and assign "it" to be the money.

The same basic argument can be constructed around another repeated example:

(7.7) Penny wanted to go to Bill's party. Mother told her that she hadn't been invited.

In the same way we substituted "have" for "get" in (7.5) we can now substitute "could not go" for "hadn't been invited", and the argument will go through as before.

We conclude from these examples that we must execute the base routines before we apply LMR, since base routines are needed to assert some new fact which in turn is needed to determine the referent. But since bookkeeping occurs after LMR it must come after the application of the base routines.

To make our ordering complete we would like to order demon application with respect to base routine application. It would seem that demons should come first, since often context will modify the meaning of a predicate, changing it to a different predicate as in (2.20). In practice however the situation is not very clear cut, and so for the moment we will just have to assume that demon application comes before base theorem application. (Though again, note that this would fall out from an assumption that base routines are really demons. This is because demons are applied in a last in - first out fashion. This is desirable since we want the most recent facts to apply first. But if we consider base routines as a form of demon, then since they were put in before the story started, they will be applied only after all the other demons

have been tried.)

What happens when one line of text becomes several assertions? There are several ways we could order application of the various DSP components on the different assertions. However, the arguments will have the same form as before, so going through them would just get repetitious. These arguments would show that demons and base routines apply to all assertions generated by a single sentence before bookkeeping applies to any of them.

7.2 An Addition to DSP

As we have things organized now, the base routines are responsible for entailment, and for putting in necessary demons. (We will not consider it now, but in chapter 9 we will see that the base routines must also take on some responsibility for predicate disambiguation.)

Note though that since LMR comes after all base routines have applied, our base routines must be able to work without needing to know the arguments to the predicates, since it is always possible that a given NP will be undecided at the time the routine is executed. This is a very serious handicap, and immediately raises the question, "Can the base routines carry out their function with this limitation?"

The answer seems to be no. Consider the following fragment:

(7.8) Bill, Harry, and Bill's dog Mutt, were outside. Harry said, "I saw George and Mutt in the park yesterday. George was doing some funny things. Mutt was doing some funny things too. He was doing tricks."

Looking at the line "He was doing tricks," we presumably get Mutt as the referent, rather than George, on the basis of LMR. (This hypothesis is strengthened by the fact that if we reverse the second and third from last lines in (7.8) we will get "he" as George.) The problem is that the action of the "do tricks" base routine, or whoever handles the phrase must necessarily be different depending on whether "he" refers to

George or Mutt. For example, one thing we will probably want to do is put out demons trying to identify a trick when one is mentioned. If it is an animal doing a trick, we would want to label as "trick" any verbal command which is obeyed. Naturally this won't do for a person. Since the tricks people do and those animals do have little to do with each other, one would imagine that there would be separate sets of demons for each. But if we don't know who is being referred to we don't know which demons to put into circulation. Another similar situation is:

(7.9) Janet went into Jack's room to find a top. In the room was Jack's ball. She knew that Jack wanted to play with it.

Again the demons looking for activities concerned with "playing" will be different depending on whether "it" in the lastsentence is ball or top, and again the only way to tell is via the LMR rule.

So far we have needed to know the objects in order to decide which set of demons to put in. There also seem to be cases where the decision whether or not to activate a particular demon depends on properties of the object being referred to. In our main example from chapter 1, we needed some demon to recognize "those paints make your picture look funny." The rule the current program has is, "If a person wants something which is owned by another person, put in a demon looking for statements saying that the object in

question is 'bad'. If encountered, mark such statements as having an ulterior motive." But this assumes that WANT-BASE knows what object is wanted, since if it doesn't it can't tell if someone else owns it. Or again, we might want the rule "If a particular object is needed, and is not around, we will put in a demon looking for the person who needs the object going to the place where the object is." If we are to know if an object is around, we must know which object we are talking about.

The conclusion is that our base routines cannot do all their work without knowing the objects. But how are we to reconcile this fact with our earlier conclusion that base routines must be able to accept undecided NPs in the hope of finding their referent?

Looking back, the only part of base routines which we tied up in reference problems was related to entailment of assertions. Those examples which required knowing the referents all dealt with the activation of demons which would have no effect until later lines of the story came in. These are activities that do not help determine reference while we are still processing the current line. So perhaps we can divide each base routine into two parts. Section 1 of the base routines will be evaluated before we apply LMR. It will be in charge of entailment of further assertions, and any other tasks which might help determine reference. (Asserting

demons with LOOK-BACK also can help determine reference.) Section 2 will be evaluated after the LMR (and presumably bookkeeping) have been applied to the assertion and hence will be able to assume that all referents are known. Its task will be to put in demons, and any other work which is not done by section 1.

7.3 Bookkeeping Again

We are now in a position to take another look at bookkeeping, and take up an issue we skipped earlier. The problem is when and where bookkeeping gets done. Up to this point we have assumed that it is done by a separate program with help from fact finders, and have further argued that it should be done pretty much after everything else, with the exception mentioned in 7.2. There is however, another interesting possibility. Perhaps bookkeeping should be done by the base routines, in particular, by part 1 of the base routines.

There is naturally an immediate objection. Haven't we already shown that it must occur after the base routines? Not completely. We saw in examples (7.6) and (7.7) that certain things in base part 1 had to happen before LMR, and hence before bookkeeping. (In particular, bookkeeping could only occur after the entailment of other assertions, like "give" entailing "have".) However, perhaps bookkeeping could be applied after entailment, but still within base part 1. As we shall see, the evidence is still against this, and for this reason I have rejected the proposal discussed in this section.

Then why am I proposing it at all?. The reason is that at one time I though I had two good objections, and the fact that one has disappeared has made me wonder.

The objection which disappeared is this: If we look just

at the problem of updating HAVE assertions, suppose we do give the responsibility to the base routines. Then GIVE-BASE, TRADE-BASE, etc., will all have to contain some duplicated code which looks for previous HAVE assertions in order to update them. This seems a waste if, as it would seem with the independent bookkeeping proposal, we could do without. But I have found in every case that I have examined, that even with independent bookkeeping, most of the code just mentioned would have to remain in the base routines. In particular, GIVE-BASE still has to look for the "giver" having the object, which is the assertion which must eventually be updated. Now I am not a great believer in coincidences, and if all these base routines need to be checking for updates anyway, then perhaps they should actually be doing the updates. But let's take a closer look.

Let's first look at "give." Incidentally, everything said about "give" will be equally true about "trade," and "get," and, to a lesser extent about "lose," "drop," throw," etc.

The claim is that if we assert that Bill gave the kite to Jack, GIVE-BASE will have to look up the fact that Bill has it. There are several reasons for this. If we don't know the fact, then we will want to assert it. This way, we will be able to answer the question "Who had the kite before Jack?" Secondly, as is well known, "have" and hence "give" are

ambiguous. In particular the kind of "give" is dependent on the kind of "have" relationship between Bill and the kite. So for example "trade" is almost always an ownership change, but not in:

(7.10) Jack and Janet rented horses for the day. At noon Jack suggested they trade.

In (7.10) we understand "trade" as a change of control over the horses since the people don't own the animals. For other examples, see chapter 9. Finally, in 5.3, we saw how consistency checks are needed to "unassume" earlier conclusions which had an ASSUME tag on them. In this case making sure that no one other than the "giver" has the object is a consistency check.

Each of these cases implies that GIVE-BASE must look to make sure that the "giver" really has the "given". Since GIVE-BASE will make this check anyway, why not also do the updating there?

There still seem to be reasons against it. In section 7.1 we saw that we had to apply LMR before we could do bookkeeping. Naturally, if the base routine is going to do bookkeeping, at some point LMR will have to be applied in those cases where it is needed. Let us suppose that we have some method which will postpone LMR until we must have it in order to do some bookkeeping in the middle of some base routine. However, it seems that no matter how clever we are about it, we are going to run into trouble. Consider:

(7.11) Jack and Bill were outside. "I like your top," said Jack. "This is my old top," said Bill. "Would you like it? I have a new top in my pocket." So Bill gave it to Jack.

One fact before we discuss the example. Since "give" and "trade" need to know the state of the "have" relation between the giver and the object to be given, this check, and presumably the updating which we are suggesting would naturally go along with it, would have to occur before we ever get around to asserting the new "have" assertion.

Now in the above example, unless some demon catches the last line before we apply LMR we will get the wrong referent for "it". In fact, there will be a demon, which we will encounter in the next chapter. But this demon will be looking for HAVE since it must work equally well for "trade", "give" and "get". But if this is the case, then LMR will be applied before we ever get to the demon (since LMR will be applied even before we assert the HAVE assertion). Hence we would not get the correct referent for (7.11).

Another example:

(7.12) When Jack got home Mother said, "Where is your top?" Jack said "I lost it. I was playing with my top at the lake. I was picking up a shell and I dropped it in the water."

Again, if we adopt our suggestion we will seemingly get the wrong referent.

Note that with our current scheme in (7.12) it is quite possible for DROP-BASE to require that the object be held by

Jack only to note that two objects, the top and the shell, fit this description. It could then leave it that way, and eventually a demon would make the decision. However, once again, if the base does the updating, it has to decide immediately which "it" is.

As I mentioned in the beginning, it seems strange that while the base routines do not do bookkeeping, they must be aware of it. It may be that this is evidence that the base routines should do updating, and that the reference problem should be solved with a much more complex organization of DSP. However, what such an organization might look like I do not know.

8 Some Details of DSP

This chapter is mainly devoted to justifying some of the decisions made in chapter 4. This includes such issues as why we specify demons by binding some of their variables at "assert time", or why we want demons to have the ability to self destruct.

8.1 ASSERT and Related Issues

When we introduced ASSERT in chapter 4 we noted that besides entering data into the data base, it also put the assertion on the TO-BE-DONE list. Thirdly, ASSERT would give the new assertion an assertion number if it did not have one. (Note 16)

Temporary Assertion Numbers

What we did not point out, however, is that the number assigned is really only a temporary assertion number. It is temporary because we might already know the fact, in which case, rather than have two copies of it floating around, we will give the new copy the same number and just not bother to put it into the data base at all. In chapter 2 we stated that bookkeeping would look for this redundancy, hence the assertion number might only last until the assertion finally reached bookkeeping, which we have shown comes fairly late in DSP.

But why not try to discover this redundancy immediately? Then, if it were already in the data base, we could ignore the new assertion completely, hence saving any time we might have spent processing it. Presumably, if it weren't in the data base, we could then assign a permanent assertion number. The argument against this is similar to those we encountered in 7.1. Consider:

(8.1) Jack, Bill, and Fred were outside Jack's house. Jack left to go to the park. Fred and Bill played ball for a while, and then Fred left also. Soon Bill's mother came by. She asked, "Where did Fred go?" Bill said, "He went to the park."

Given the scheme we proposed in chapter 2, we will get the correct referent for the "he" either on the basis of LMR, or via some demon which is activated by the question. Presumably this demon is on the lookout for an answer to the question. We can actually show that it must be the latter since we could also replace the last two lines of (8.1) with:

(8.2) Bill's mother said, "Where did Fred go? I thought I saw him here. I know Jack was here." Bill said, "He went to the park."

In (8.2) we understand "he" as "Fred" even though Jack was the last mentioned male. This suggests that a question takes strong priority in establishing reference, which is best accounted for in our system with a demon.

But no matter how we establish that the "he" refers to Fred, we cannot do a GOAL on the assertion immediately to see if we know this fact already since the GOAL will find that we

do know the fact, but only by assigning "he" to Jack rather than Fred, which, naturally, is unacceptable. (Remember, at this point in story (8.1) or (8.2) we know that Jack is at the park, but we know nothing about Fred's location.) This means that such a check must be postponed until whatever rule is operating in (8.1) and (8.2) has had a chance to work. Since the redundancy check is a routine activity which should be done to all entering assertions it seems most natural to include it in bookkeeping. Hence, the assertion numbers we assign to an incoming assertion are temporary.

When Do the New Assertions Go Into the Data Base?

There has been a tacit assumption in all this, that we really "assert" the incoming assertions before we apply DSP. That is to say, there is some question as to whether we put these items in the data base before or after the line has been processed by DSP. We have been assuming that it is done before. Actually, the question really is not whether or not we assert the items. The same effect could be accomplished by a theorem which establishes a fact by trying to find it on TO-BE-DONE. The crucial question is, when we are interrogating the data base, to what extent should our current assertions "look identical" to ones asserted in earlier sentences?

The answer is that the new assertions should look pretty

much the same. For the first example, consider PB-FOR-MONEY, from chapter 6. This was a demon which looked back or forward to see if the person needed money, which would in turn explain why he got the PB. If we are working on an example where the demon is looking forward and finds the new line "Janet needed some money," the fact that "some money" is MONEY will be a new assertion. If we did not put new lines in the data base until after the line were processed, our demons would not find the MONEY assertion, and hence not work properly.

Another example concerns the PB-SHAKE demon which we decided would not be activated until we had a "have PB" assertion. But suppose we opened a story with:

(8.3) "Janet was shaking her PB."

We would want to conclude that she was trying to get some money. Our current scheme would indeed produce this result, in the following fashion. "Shake PB" would entail "have PB", and this in turn would activate (using LOOK-BACK) PB-SHAKE. PB-SHAKE would then pick up the "shake" assertion and assert that Janet was trying to get money. This assumes however that the main assertion was in the data base already (as opposed to just being on the TO-BE-DONE list).

Hence it would seem that in most cases we want new and old assertions to "look" the same as far as GOAL is concerned. This is somewhat important since we cannot be sure that all the NPs will be decided and there may be variables in the

assertion. Neither Planner nor Micro Planner allows variables to operate in this fashion.

8.2 The Assignment Problem

When we first introduced our formalism in chapter 4 we noted that while our demons talk about "a person X" or "an object Y," our stories talk about Jack and his piggy bank. Hence it is necessary to give the variables in the demon particular assignments. Let us review the assignment problem with a new demon which is to account for stories like:

(8.4) Janet offered to trade her pencils for Jack's paints. Jack said "I want my paints."

If we were now to ask, "Will Jack trade?" the answer would be no. We might account for this fact with a demon which says "If a person has been offered a trade and he makes any statement which implies that he intends to hold on to the object in his possession, assert that he will not trade." This might look like:

```
(DEMON TRADE-LIKE-MINE
   (PERSON OBJECT OTHERPER OTHEROBJ N N1)
   (?N HAVE FUT ?PERSON ?OBJECT)
```

This is the pattern the demon is looking for. Statements like "like (or want) the paints" are translated into "like (or want) to have the paints".

```
   (GOAL (? TRADE FUT ?OTHERPER ?PERSON ?OTHEROBJ ?OBJECT))
```

Make sure that the person has been offered a trade for his object.

```
   (ASSERT (?N1 TRADE FUT NOT ?PERSON ?OBJECT ?OTHERPER
                                              ?OTHEROBJ))
```

Assert that he won't trade.

```
   (ASSERT (? RESULT ?N1 ?N)))
```

Assert that the reason is the current statement.

As written, TRADE-LIKE-MINE assigns its variables with the line:

```
(8.5)    (GOAL (? TRADE FUT ?OTHERPER ?PERSON ?OTHEROBJ
                                              ?OBJECT ))
```

This has the effect of checking that the person who likes the object is the person who was offered a trade for it. In this way our demon will not apply to the last line in:

(8.6) Janet offered to trade her pencils for Jack's paints. Jack said "I like those pencils."

In this case the GOAL will fail since it would be looking for:

```
(8.7)    (? TRADE FUT JANET1 JACK1 ?OTHEROBJ PENCIL1)
```

which means that Janet offered to trade her paints for Jack's pencils, which is not what happened. Also note that the GOAL did other things for us. In order to make the "won't trade" assertion, we need to know exactly what other person-object

pair is involved in the trade. We will get hold of them because ?OTHERPER and ?OTHEROBJ will be assigned appropriately. We will call this method of handling the assignment problem the "GOAL method".

But this demon will only be activated when a trade has been offered. So in some sense, the GOAL in our demon is redundant. We already know that TRADE is in the data base, we need only make sure that we have the objects right. We saw this same situation in chapter 4 where our PB demon had to look for the SHAKE assertion which activated it. We suggested there that one way we might get around this redundancy is to put the assignments in the demon at the time the demon is activated. That is, when we assert the demon, we will assign the variables. Hence for our TRADE-LIKE-MINE demon, ?PERSON ?OBJECT ?OTHERPER ?OTHEROBJ would be assigned to Jack, paints, Janet, and pencils, respectively. This can be done in Micro Planner in much the same way it is done in LISP though the feature is little used in either language. The variable binding list, rather than just giving the name of the variable, can also give its initial assignment. So our demon would look instead like TRADE-LIKE-MINE2 where the variables have been assigned as if we were in example (8.4).

```
(DEMON TRADE-LIKE-MINE2
   ((PERSON 'JACK1)(OBJECT 'PAINT1)(OTHERPER 'JANET1)
                                   (OTHEROBJ 'PENCIL1) N N1)
   (?N HAVE FUT ?PERSON ?OBJECT)
   (ASSERT (?N1 TRADE FUT NOT ?PERSON ?OTHERPER ?OBJECT
                                                ?OTHEROBJ))
   (ASSERT (? RESULT ?N1 ?N)))
```

I call this process "specification". So we have two binding methods, the GOAL method and the specification method. One immediate advantage of the specification method is that we no longer need the GOAL in the demon since we already have the correct arguments. (We saw a similar saving in the PB example of chapter 4.) And there are other advantages to the scheme. Consider an example like:

(8.8) Jack, Bill, Joe, and Jack's dog Tip were outside. "I like Tip." said Bill. "I will give you my pocket-knife for him." Joe said, "I like Tip too. I will give you my pogo stick." Jack said, "I want Tip."

In (8.8) Jack's reply should be taken as a refusal of both offers. However, if we used TRADE-LIKE-MINE we would only find the second offer, and the first one would remain unnoticed. With specification we would have two separate instances of the TRADE-LIKE-MINE demon, one referring to Joe's offer and one referring to Bill's. Hence we would interpret Jack's comment as applying to both. It might be possible to change the GOAL method in some fashion so that it would work in this instance, but I cannot think of any reasonable way.

Furthermore, as our model gets more complete, we would hope to be able to use the same demon in many situations. It

would seem that requiring the GOAL in the demon to find the bindings would restrict its use. In particular, TRADE-LIKE-MINE would be restricted to "trade" situations. To see how this might create problems, note that we will need a demon to account for the following situation:

(8.9) Jack had a kitten he could not keep. He offered to trade it to Bill for some money. Bill looked at the kitten. When Bill left he had the kitten.

Of course, if we asked how Bill came to have the kitten, the obvious answer is that he traded some money for it to Jack. We need a demon then, which simply says that if a person who has been offered a trade is later seen with the object, it means that the trade took place. This might look like (using the binding via GOAL):

```
(DEMON TRADE-WILL-HAVE
   (PERSON OBJ N N1)
   (?N HAVE ?PERSON ?OBJ)
   (GOAL (?N1 TRADE FUT ? ?PERSON ?OBJ ?))
   (ASSERT (? T-RESULT ?N ?N1)))
```

However, note that we would like to use this same demon in a situation where one person is just giving an object to the other. That is, even if we aren't told that the "give" actually took place, if we see the "receiver" with the object, then we can assume it did. Of course with the specification method there is nothing to prevent the above demon's being used in both cases, since the GOAL will not be there. Hence, the specification version of the same demon will be named WILL-HAVE.

It is for these reasons that I have adopted the specification method.

There is one point about the specification method which we did not cover in chapter 4. If D-ASSERT (the function which assertes demons) is called by a demon, the new demon will not be activated immediately, but rather put on a list to be handled just before the program goes on to a new sentence. The reason is that all the objects may not be known at that moment, due to normal referential ambiguity. By waiting until the sentence has been completely processed we can be assured that all the noun phrases have been assigned referents.

8.3 Demon Destruction

Update Destruction

A consideration mentioned in chapter 2 which hasn't been mentioned since is the fact that we don't want demons outliving their usefulness. Let's give a few examples of how this might happen.

We needed TRADE-WILL-HAVE to account for (8.9). However, we can quickly see that the demon can cause trouble if we don't watch out. So we can have:

(8.10) Jack had a kitten he could not keep. He offered to trade it to Bill for some money. Bill said that he liked the kitten, but had no money. Jack said that he would just give Bill the kitten. When Bill left he had the kitten.

In this story we do not want to assert that the trade took place. Let's look at another example which is somewhat similar.

(8.11) Janet wanted to trade her pencils for Jack's paints. "They are good pencils," said Janet.

Presumably in a trade context we would assume that the comment was inspired by Janet's wanting to trade. Again, we seem to need a demon, which we will call TRADE-GOODMOUTH. And again we must be careful in how we use it, as exemplified by the following:

(8.12) Janet wanted to trade her pencils for Jack's paints. He liked the pencils and so he traded. Janet said "They are good pencils."

Now, while Janet might be reassuring Jack that he made a wise choice, we cannot say that she said it to bring about the trade.

In both (8.10) and (8.12) we have a demon and a line which under other circumstances would activate it. However, due to other factors in the story we do not want the demon to be successfully executed. Furthermore, note that the demons are of no further use in the stories since the situations they are designed to catch no longer can occur. (Naturally, we could have another trade offer in the story in which TRADE-GOODMOUTH or WILL-HAVE plays a part, but the demon would be re-activated when and if this second offer came about.) Given this, we would like to "deactivate" or "destroy" the demons. The question is, under what circumstances do we want to do so?

What both of these situations have in common is that the possible future on which they were based ceased to be "true" in the sense that it had been updated. In particular, in (8.10) we had:

(8.13) (N4 TRADE FUT JACK1 BILL1 KITTEN1 MONEY1)

which activated the demon. But the statement by Jack that he would "just give" Bill the kitten says that N4 is no longer a possible future. Exactly how this is to be recognized is not obvious, but we can certainly say that after the "just give" statement Jack no longer intends to trade the kitten away.

In (8.12) we had a TRADE FUT statement, but before the "good" line came along the trade actually took place, so it was no longer a possible fututre. In passing, we might note that this update is somewhat different from those we have seen earlier. The usual or NEG update says the this fact was true at one time, but is now false. Here we have a TENSE update. Technically we are still saying this assertion was once a possible future, but it is no longer. However, TENSE updates have different properties than NEG's. In particular we will want a rule like "If X is a TENSE update of Y, and we know that Z was the reason for Y, then Z is the reason for X". This rule also applies to "specification updates" (to be mentioned in 8.4) but it clearly does not apply to NEG updates.

While we have already seen reasons for preferring the specification technique for variable assignment over the GOAL technique, at first glance the destruction problem might make us wonder at the wisdom of our decision. In some sense the GOAL method would seem to accomodate destruction in a very natural way. So when using TRADE-WILL-HAVE we want to prevent it from operating when the TRADE assertion is no longer a valid possible future. We have already established a way of preventing updated assertions from being taken as current truths, ($TRUE). We could possibly use $TRUE also to prevent the demon from working. The trouble is that while it will

prevent the demon from working, it is not clear how we can make it destroy the demon under appropriate circumstances. For example, we can't just say "if the GOAL fails, destroy the demon", or else in example (8.6) we would destroy the demon when we shouldn't. (In 8.6 the demon would be called in a situation where it was not applicable, but there was still the possibility that it might be needed at a later time.) What we would have to do is first try the goal, and then if it succeeds, see if it has been updated. But at this point it is not noticeably simpler than what we would have to do to accomodate the specification method to destruction. In the former we would have:

(8.14) (GOAL (?N1 TRADE FUT ? ?PERSON ?OBJ ?))
(DESTROY? ?N1)

inside the demon. With the latter we would have as the first line of the demon:

(8.15) (DESTROY? ?NOLD)

where ?NOLD is specified to be the assertion number of the original FUT assertion. DESTROY?, of course, looks at the assertion with that assertion number, and if it is updated, makes the demon inactive. (The demon which becomes inactive is the demon we are currently executing. There is no contradiction in this since deactivating a demon only insures that we will not be able to get at that instance of the demon in the future.) (Note 18)

Time Destruction

There is a need for other kinds of destruction also. It can easily be the case that a given demon is never executed. For example, we can have a trade situation where no one ever makes a comment about how good his object is, so TRADE-GOODMOUTH will never get executed. However after we have gone sufficiently further in the story, the demon is irrelevant, since the trade which asserted it is either no longer under consideration or forgotten completely. In the case that we have explictly updated the possible future, if we ever enter the demon, it will be destroyed. But, as likely as not, no assertion will ever match the demon pattern, so we will never get to the DESTROY?. This would mean keeping unneeded demons around, which would involve overhead expense. There is also the distinct possibility that such demons would interact with sentences occurring much later in the story giving incorrect interpretations. However, since I have not been dealing with long stories I have no examples of this.

What we want is a way to remove demons automatically after a long time has gone by. (We refer here to "reader" time rather than "story" time. What will actually be counted is the number of assertions which have gone by.) The easiest way to do this in Micro Planner is to use a filter when we look for applicable demons. Basically this filter would check the demon under consideration and find the date it was

activated. The date can be found by getting the specification for ?NOLD which activated it. (All assertion numbers have the number of assertions from the beginning of time to the time they were created on their property list. This way we can tell approximately how old a given assertion is by comparing its number to the current number.) If the assertion is over a given age it will be destroyed. There are several modifications which we might want to make on this scheme. The most obvious is that we might give different kinds of demons different acceptable ages before destruction.

8.4 More on Bookkeeping

In the course of discussing other problems, we have been accumulating tasks which we said should be done by bookkeeping.

Bookkeeping should determine if the assertion is already in the data base as mentioned in 8.1. All we need do here is remove the temporary assertion number, replacing it with a variable and do a GOAL. It will look like:

(8.16) (GOAL (?N AT JACK1 HOUSE1)$TRUE)

We use $TRUE to make sure the fact is currently known to be true.

However, note that we did not use a $DEDUCE in the above GOAL. That is to say, we do not use fact finders when trying to show that a given fact is already in the data base. The reason is quite simple. Fact finders are often used to establish a fact X which is not absolutely certain to be true, but which is not contradicted in the data base. The idea is that unless the story is grossly deceptive we may assume X is true. However, if we are told at some later time that X is true, we would want to keep the information since it is more reliable than our fact finder.

To see how this could affect us, consider the fact finder which we mentioned in 2.4, which we will write more explicitly in Planner here.

```
(DEMON OWN-FF
   (PERSON OBJECT N1 N2)
   (? OWN ?PERSON ?OBJECT)

        To show that ?PERSON owns ?OBJECT.

   (NOT (GOAL (? OWN ? ?OBJECT)$TRUE))

        If we know that anybody owns it then it can't
        be ?PERSON or else we wouldn't need this theorem.

   (GOAL (?N1 HOLD ?PERSON ?OBJECT)$TRUE)

        Find that the person was holding it.

   (NOT (AND (GOAL(?N2 HOLD ? ?OBJECT)$TRUE)
             (GOAL (? BEFORE ?N2 ?N1)))))

        Make sure no one had it before him..
```

If we used this theorem in establishing identity, what would N5 be identical to?

```
(N5 OWN JANET1 TOP1)
```

The only possibility, though it would be very weak, is:

```
(N6 HOLD JANET1 TOP1)
```

Already this seems unnatural. But things get worse. First suppose we had:

(8.17) Janet and Bill were outside. Janet was holding a top. The top belonged to Bill.

We would not want the third line in (8.17) to update the second. Janet is still holding the top. On the other hand:

(8.18) Janet and Bill were outside. Janet was holding a top. "Isn't my top pretty," said Janet. The top belonged to Bill.

Now consider what will happen during the processing of (8.18) if we use fact finders to determine whether a fact is already in the data base. When we get to the third line of (8.18) we

will see that we can already say that Janet owns it since she "introduced" it to the story. Hence we will not re-enter the fact. When we get to the fourth line we will not see that there is any contradiction in the story for the same reason we did not find a contradiction in (8.17). In fact, what would happen is that from the fourth line on, the program would just assume that Bill owned the top. But this is not what we want. We must note that there is a contradiction between what Janet said and the fact which the narrator says is true.

Bookkeeping must also determine tense updates as mentioned in 8.3. The GOAL will look like:

(8.19) (GOAL (?N AT ?TN JACK1 HOUSE1)$TRUE)

If the original tense was present, and the new one is past, or it was future, and we now find a present or past, we will mark the old and new assertions as a tense update pair. Actually, the GOAL will have to be complicated slightly since we must also take into account the possibility of partial updates as mentioned below.

Another function of bookkeeping is looking for normal negation updates. The GOAL will be:

(8.20) (GOAL (?N AT NOT JACK1 HOUSE1)$TRUE $DEDUCE))

Note that in this case we want the $DEDUCE since we will need fact finders to establish the negation in most cases. Again we will have to take account of partial updates.

Partial updates are a further job for bookkeeping. We

mentioned in a note to chapter 4 that one consequence of allowing DGROUPS to appear as a single object in an assertion is that we will be faced with situations where an old assertion says that both Jack and Bill are at the store, where the new one says that Jack left. We do not want to lose the fact that Bill is still at the store, but we must mark the old assertion as updated. In the case of negative updates the form given in (8.20) will also find partial updates since one theorem it will call will be GROUP-FIND-FF which attempts to prove a fact about an individual by proving it about a group he is in. For tense updates we will need to put in an explicit (USE GROUP-FIND-FF) in the GOAL statement. (USE recommendations in a GOAL are a way to tell the GOAL to use a particular theorem to establish the GOAL.) We will consider only the process for NEG updates, though virtually the same things happen for tense.

When the GOAL comes back with an assertion to be updated, we look to see if it contains a group with the person in it, rather than the person himself. If so, we update the old assertion and create a new assertion. This new assertion is the same as the old one except it has instead of the old group, a new group which is the old one, minus the person mentioned in the new assertion. Actually, the "person" mentioned in the new assertion may be a group which is a sub-group of the group in the old and now updated assertion.

There is one kind of update we haven't mentioned, a specification update. This is for situations like:

(8.21) Jack and Bill were in the house. Jack said "Will you trade your kite to me, Bill? Will you take my paint set?"

The first line should suggest a possible future concerning a trade. As we saw earlier (8.8), the mere presence of a second trade offer does not invalidate the first, but in (8.21) we interpret the third sentence as a further specification of the second, and we would not want demons from both being operational. We could handle this either by physically removing the former assertion, or by changing it and calling the second a repeat of the first, or we could say that the second updates the first. From the standpoint of bookkeeping, it really doesn't matter very much which we do; its responsibilities are the same.

9 Problems in Ambiguity, and Why Characters Ask Questions

This chapter contains the beginnings of two separate theories. The first four sections of this chapter are concerned with the problem of disambiguating the word "have," a problem which cannot, in most cases, be solved with selectional restrictions. The discussion will be concerned with whether an adequate theory will be compatible with the model presented in this thesis.

The last five sections are concerned with the problem of answering questions about why a character in a story asks a question. The knowledge that enables one to answer such a question is a kind of knowledge whose organization seems quite different than that of, say, piggy banks.

9.1 Information for Deciding Meanings of Have

"Have" has many meanings. To name a few, there are "own", "hold", "immediate control", "have as part" (as in "he has red hair"), and even a vague association defined by the arguments to "have" as in "he has a lawyer" or "she has a home".

In this section we will concentrate on the factors which come into play in distinguishing between "own", "immediate control" (I-C) and the "vague associational have". I choose these three because they are the primary ones which come up in children's stories, and I have had the most experience with their intricacies. In particular I am using I-C rather than "hold" since if we look at some everyday situations, we find I-C tends to be more important. If Jack was using the can of paint, which is standing next to him, then he has I-C over it. So, if a given activity requires a certain object, it always requires that the person have I-C relation with the object, but seldom that he be holding it. To eat cake does not require holding the cake; it could be on a plate on the table and painting a picture does not require holding either the picture or the paint, but certainly requires I-C over them. Baking a cake requires I-C over flour, a cake pan, etc, but at any given time, one is unlikely to be holding a particular one of them. Even in the cases where "hold" is technically necessary, in practice it is not wise to insist that it is

true. For example, we might be told that Janet was painting a picture with a paint brush when really she had temporarily put the brush down. The idea is that stories generally give the broad outlines and not the details. I-C is a broad outline descriptor; in contrast, HOLD gives detail. So when we say that an action requires an object, we are usually referring to the I-C relation. I-C is also useful in cases like "Where is my dog Tip?" "Oh, Tom has him." We don't mean that Tom owns Tip, or that Tom is holding Tip, but the I-C relation is just what we need.

But if "have" has these three meanings, so do "give" "get" and "trade". The following sets of sentences and fragments give examples where they are used all three ways. The correct meaning is given after each sentence.

(9.1) Janet went into the house and got a spoon. (I-C)
(9.2) Janet went to the store and got a ball. (OWN)
(9.3) That was how Fluff got a home. (vauge)

(9.4) Bill wanted to look at Jack's knife, so Jack gave it to him. (I-C)
(9.5) When Jack got a brand new knife, he gave the old one to Bill. (OWN)
(9.6) We will give Fluff a home. (vague)

(9.7) Jack and Janet rented horses for the day. At noon Jack suggested that they trade horses. (I-C)
(9.8) Jack traded his top for Bill's knife. (OWN)
(9.9) Jack and Bill wanted to trade parents. (vague)

The logical assumption throughout this thesis has been that "trade," etc., all produce HAVE assertions. By retaining the assumption here the related ambiguities in "trade", "give", and "get" are all reduced to the ambiguity in the HAVE

assertion they each produce.

Let us try to isolate some factors which come into play in determining which interpretation is assigned to a particular HAVE.

1) The nature of the objects being "had". "Get a home" means the vague "have" most of the time in children's stories because homes are seldom bought or sold, and it could not mean I-C because houses are too large to have immediate comtrol over. In most cases this type of information only distinguishes between the vague "have" on one hand, and I-C and OWN on the other. This is simply because most objects which a child can have I-C over he can also OWN. One exception is "report card", and as expected

(9.10) "Jack got a reportcard."

is interpreted is I-C.

2) "Context." In "Janet went to the store. She got a ball." it seems certain that the "store" context is telling us that "get" means either "get-own" or perhaps "buy". Another example was (2.20), where in the context of trade we assumed that the "have" in "give" meant OWN. But when Bill gave Jack his pogo stick inorder that Jack could show it to his father, the give was I-C. One interesting case is when the "have" statement is preceded by a "where" question as in, "Where is Tip? Jack has him." The interpretation is clearly I-C.

Presumably we will handle such situations by having "where" questions put in a demon looking for "have" statements as answers.

3) "Specificity." The more "knowledge" the speaker has about the HAVE assertion, the more likely it is to mean I-C. (I use the word "knowledge" in a metaphoric sense, as we shall see.) So we have:

(9.11) Last week Janet got a top.
(9.12) Janet just got a top.

While (9.11) is clearly OWN, (9.12) is more ambiguous. There are other cases which are quite similar. Suppose Jack was not around when:

(9.13) Penny said, "Jack has a ball."

We will almost always interpret this as OWN. On the other hand, if Jack is there, the statement is ambiguous.

Or again, consider:

(9.14) Jack has a ball

vs.

(9.15) Jack has the ball.

The former is much more likely to mean OWN than the latter. Formulating the rules for this kind of information will be quite difficult. If we try a "conservative" rule, such as "If the NP is indefinite, and the 'haver' is not there, then the HAVE means OWN" But we have situations like:

(9.16) Bill was wondering what happened to his ball. Janet said "I saw Jack at the park. He had a new ball."

In this case the interpretation has something to do with the time frame. I.e., we understand that Jack had the ball while Janet was watching him, which clearly means I-C.

4) What we know to be true. This is most commonly helpful with "give" and "trade". For example we saw in (9.9) that even "trade" is forced to mean I-C when both parties clearly don't own the things being traded. We can see a similar phenomonon in:

(9.17) Jack and Janet were playing with a set of paints. Jack offered to trade the red paint for the yellow.

(9.18) Jack and Janet were reading the Sunday Times. Jack offered to trade the Arts and Leisure section for the Week in Review.

In each of these cases the objects being traded form a natural set, so that we would assume that whoever owned one owned the other. Hence we cannot have Jack owning one of the objects and Janet the other, so we must mean I-C.

There are some cases where this applies to "have" also. For example in:

(9.19) Janet and Bill were outside. They wanted to play ball, but didn't have a ball. Jack came along holding his blue ball. Janet said "Bill, Jack has a ball."

First note that the statement "but didn't have a ball" is understood as I-C since Janet and Bill need I-C in order to play with the ball. Presumably this would be handled by the

demon which deduces the fact that the reason they weren't playing was because they did not I-C a ball. However, the line "Jack has a ball." cannot be accounted for in this manner, since the same story, minus the fact that Jack was holding a ball, would allow the last line to be ambiguous. That is, Janet could just be saying that Jack owns a ball. Perhaps she hopes he will go back to the house and get it. Clearly then, the fact that we already know that Jack is holding a ball is what compels the I-C interpretation of the last line.

While information given earlier in the story is a special case of "context" we should distinguish "what we know to be true" from the type of information just discussed in (2). The information in (2) was handled by demons since it seemed most reasonable that "store" was looking for GET (in order, at least, to assert how the "getting" took place). However, it does not seem reasonable that "Sunday Times" in (9.18) is looking for TRADE. Rather, the base routine for TRADE should look to see what the most likely relation is between Jack and Janet, and the Sunday Times. So, the distinction between (2) and (4) is between processes handled by demons, and those handled by base routines.

9.2 Where Information Is Placed, and Some Implications

Without going into details, we can see that the theory outlined in chapter 2 has other things to say about how such information as outlined in (1) through (4) should be handled. First, we know that the choice of predicate must be made before we do bookkeeping, so that we will know which previous facts must be updated. That is, when Mary gives Jack her top do we want to update the fact that Mary owns the top, or that she has I-C over the top, or both? (Actually, as we shall see in 9.4 there are cases where we cannot decide which interpretation of "have" is correct without reading more of the story. But if we can decide without reading more, then our model should make the decision before bookeeping.)

Type (1), nature of the objects being had, could be applied either in internal translation, before DSP gets the assertion, or by HAVE-BASE. There are some problems with having internal translation do the work, since we might have a story which revolved around buying and selling report cards, perhaps in order to fool one's parents. While we would expect our base theorems to be able to take such context into consideration, it would seem beyond the "filtering" capacity of the semantic phase. Also note that this might be an argument in favor of applying demons before base routines, since we could then expect our HAVE to be marked as OWN since we were expecting ownership of report cards to be important.

Type (2), context, as we have already pointed out is demon application. In the case of statements like "Jack has a ball", demon application will occur before any base routines look at the assertion. In "Jack gave a ball to Bill" we would first have the application of GIVE-BASE which would entail HAVE and the demons would apply to the latter.

Type (3), "specificity" would probably be applied by HAVE base. This seems most reasonable since this kind of information is applicable to the entire "have" family.

Finally, type (4) information should be applied by the respective base routines. We have already argued that demons would be inappropriate, and since the way previous facts are to be used will differ in the case of each predicate, it seems most natural to have the base theorems for each predicate apply the information rather than HAVE applying the information for all. For example, while "have" just checks to see if we already know that the "haver" I-C's or OWN's the object (9.19), "give" and "trade" also check the status of the HAVE relation prior to the exchange (9.17).

Given this placement of information one might expect interference between reference determination and predicate determination. We have already established that the predicate should be determined before we do bookkeeping. Note then that type (1) information, nature of the object and type (4), what is known, presuppose we know what the object is. But of

course, since the "last mentioned" rule has not applied yet, we cannot be sure that the objects will be known at this time.

The really interesting fact is that, as far as I can tell, not only is there no interference, but predicate determination seems to aid reference determination. The evidence is not as strong as I would like, but it is suggestive. Consider:

(9.20) The rock hid the stick. Jack went to get it.

In (9.20) we understand "it" as referring to the stick rather than the rock. Furthermore, we could change the first sentence to "The stick was hidden by the rock," and we would get the same effect. This seems to indicate that we are biased towards objects a person can "own" or "control". Naturally, "it" would tend to be bound to such objects in the course of applying type (1) information.

There is also evidence that type (4) information also plays a helpful role in reference. Consider:

(9.21) Bill and Fred were in the park. Jack ran into the park holding his bat. Bill was saying to Fred, "We need a bat. We need a ball too. Look, Fred, there is Jack. He has one."

While I do not find this the best of all stories, the fact that I tend to understand the final "one" as referring to "bat" is, of course, significant.

9.3 Using Demons to Determine Meaning

Before we can give explicit examples of demons determining meaning, we must decide on a notation for HAVE and its various meanings.

Presumably the incoming internal representation will have unadorned HAVE's since a particular meaning has not yet been chosen. We will select a particular meaning by placing a tag on the assertion number. A demon will specify what meaning it "prefers" by placing a tag on the previously "bare" HAVE assertion. Consider:

(9.22) Jack and Bill were at the park. They wanted to play baseball. Jack got a ball and returned to the park.

We will need some demon which will answer the question, "Why did Jack get the ball?" It might look something like:

```
(DEMON NEED-BALL
   (PERSON NOLD BALL N)
   (?N HAVE ?TN ?PERSON ?BALL)
```

This demon would be associated with baseball.

```
   (DESTROY? ?NOLD)
   (GOAL (? IS ?BALL BALL)$TRUE $DEDUCE)
   (ASSERT (? REASON ?NOLD ?N)))
```

Now this demon does not specify one crucial fact; the HAVE must be I-C. We could specify this with a restriction on the assertion number (which we have already said will receive the tags) which might look like

```
((?R ?N I-C-F) HAVE ?TN ?PERSON ?BALL)
```

I-C-F is a function which does the following:

If ?N has an OWN tag, causes the match to fail.
If it has an I-C tag, allows the match to go on.
If it has no tag at all, will place an I-C tag on it.

Naturally, should the demon fail, presumably in this case because the first GOAL failed, this tag must be removed.

In much the same manner we want to account for (9.2) where "store" implies that the "get" means "own". Ignoring the possibility of somehow including "buy" in the situation we might use:

```
(DEMON GET-OWN
   (PERSON NOLD N)

        PERSON and NOLD will be specified

   (($R ?N OWN-F) GET ?PERSON ?)
   (DESTRUCT? ?NOLD)
   (ASSERT (? REASON ?NOLD ?N)))
```

In this case the demon is looking for GET rather than HAVE since it seems at first glance that statements about "giving" or "trading" are not subject to the rule which takes "get" as meaning "own".

9.4 Ambiguous Situations

There are cases where we cannot decide which meaning of HAVE is meant. Consider:

(9.23) Mary and Bill were at the park. They wanted to play ball but couldn't because they didn't have a ball. Jack came by and Mary said "Bill, Jack has a ball."

If the next line were

(9.24) Jack threw his ball to Janet.

we would understand the HAVE in (9.23) as I-C, and if it were

(9.25) Janet said, "Jack, will you go home and get your ball?"

we would understand it as OWN. Presumably, in ambiguous situations there won't be any tags on the HAVE. When we get to the line (9.24) we need to assume that Jack is holding the ball, i.e., has I-C over it, in order to account for the fact that he is able to throw it. Presumably the GOAL (it will be in THROW-BASE) which is to establish this fact will use I-C-F. Then, as a matter of course, it will put the I-C tag on HAVE. Hence we have accounted for the situation in (9.24).

We can assume that (9.25) will imply that Jack is not holding the ball since we need "will get" = "not have now" for independent reasons (see section 6.8). This deduction is not a firm one. (Jack asked Janet if he could use the kite she was holding. "I need it by tomorrow," said Janet. "You will have it tonight," said Jack.) Hence it is not clear how the details will work.

9.5 The Problem of Spread Out Knowledge

In chapter 5 we looked at knowledge about piggy banks. This body of knowledge seems both reasonably "cohesive" and reasonably "isolated." We could point to a few demons, along with a base routine or two, and say "That's what we know about PB's." Furthermore, there were only a limited number of situations which called for access to that knowledge. In fact, we postulated that our information about PBs only needed to be made available when we saw someone getting, or thinking of getting a PB.

There are other bodies of knowledge which don't seem to have such properties. Consider the concept of "mistake." What constitutes a mistake depends on what a person is doing. Furthermore the response to a mistake (or its significance) will vary in much the same way.

One might then ask if there really is any substantial knowledge about "mistakes" in general. If there is, it will probably be in the form of generalizations concerning the relation between the facts of a domain and its "mistakes". So perhaps there are a limited number of basic kinds of mistakes that account for most situations.

Many other topics seem to be quite similar to "mistake" in that they are not as easily "localizable" as was piggy banks, like "forgetting," "pretending," "friendship," "authority," to name a few. The topic we will examine next in

some detail is the question, "why do people ask questions?" We will see that in this case at least, there are useful generalizations which can be made about "why people ask the questions they do".

9.6 What a Solution to the "Question" Problem Might Look Like

One of the most common reasons we give for asking questions is "curiosity". In this age we all recognize that "curiosity" is just a cover for deeper motives. Such motives are not in the domain of children's stories, however. What we are interested in are motives which are closer to the surface. Suppose we have:

(9.26) Jack wanted to talk to Daddy. He went up to Janet and asked "Where is Daddy?"

He asked because he wanted to talk to Daddy. Questions like (9.26) related directly to activities and situations in the story. It is this kind of question we will examine.

Now it should be clear that we could put in a demon explicitly looking for a question as in (9.26). This would correspond to the idea that each domain of knowledge must handle its own questions. However, it would be nice if we could find generalizations between domains so separate demons wouldn't be necessary. The rest of this section will be devoted to showing that this is a reasonable hope, and establishing a modicum of extra baggage needed to cover a large proportion of questions. Our plan is to find some way to modify demons in such a way that not only will they perform their normal deduction duties, but they will also serve to recognize the logic behind a character's questions.

Let us start with:

(9.27) Jack wanted to talk to Daddy. He went to the hardware store. Daddy was in the store looking at tools.

Question: "Why did Jack go to the hardware store?" What we hope to find is some relation between the normal deductions concerning "talking to a person", as in (9.27), and questions concerning "talking to a person" as in (9.26).

When we are told that Jack wants to talk to Daddy, we note that he is not with Daddy, so we put in a demon which looks for Jack going someplace. The demon checks to see if Daddy is there, and if so asserts that the reason Jack went there is to talk to Daddy. Unfortunately in (9.27) this doesn't work since we don't know that Daddy is in the store until after we are told that Jack went there. So we need a second demon, this one looking for Daddy's location. If it finds it the demon will see whether Jack just went there, and if so assert why. The pattern for this demon will be something like:

(9.28) (?N AT PR ST DADDY1 ?PL)

(Since the rest of this chapter deals with questions we will revert to including the type and tense markers in the assertion (PR (present tense) and ST (statement) in (9.28)). But note that if we generalized (9.28) to accept questions also, it would match the question which we had Jack asking Janet in (9.26), i.e., "Where is Daddy?" So we have found the begining of a connection between (9.26) and (9.27). There is

a single demon whcih can be written so as to pick up both.

But finding a demon which could pick up both a fact needed to make deductions and a question does not mean that we have found a meaningful generalization. We must insure that the demon works properly in both cases. If to do this we must double the size of our demons, we have only disguised two demons as one. That is, we might go from

```
(9.29)   (DEMON (name) (variables) (pattern) (exp 1)
                                              (exp 2)
                                               ...
                                              (exp n))
```

to

```
(9.30)   (DEMON (name)
                (variables)
                (pattern)
                (COND ((EQUAL ?TYPE 'STATEMENT)
                       (exp 1)
                        ...
                       (exp n))
                      ((EQUAL ?TYPE 'QUESTION)
                       (exp 1')
                        ...
                       (exp n'))))
```

Instead of having two demons, we will have one which is twice as long. What we really want is to show that only a small modification to the deduction demon is necessary to accomodate it to questions. To see if this is possible, let's look at the LOC-OF demon in more detail. We will specify it to reflect the "Jack wanting to talk to Daddy" situation.

```
(DEMON LOC-OF
   ((NOLD 'N10)(PERSON 'DADDY1)(PERSON2 'JACK1) OLDN TYPE
                                                LOC NLOC)
   (? ($R ? LOCATION) PR ?TYPE ?PERSON ?LOC)

        Look for Daddy at (in, on, etc.) some location.

   (DESTRUCT? ?NOLD)

        Make sure Jack still wants to talk to Daddy.

   (GOAL (($R OLDN LAST5) GO ? ST ?PERSON2 ?NLOC) $TRUE)
   (GOAL (?NLOC AT ? ST ?PERSON ?LOC) $TRUE $DEDUCE)

        See if Jack has, is, or will, go there. (Note 19)

   (ASSERT (? RESULT ?OLDN ?NOLD)))

        Assert that you know why.
```

The problem is that our understanding of the question "Where is Daddy?" does not depend on Jack's going to the specified location. So, at least in this simple instance, it doesn't seem that our question analysis will use much of the program written for deduction. Furthermore, I would like to suggest that this will generally be the case. If we know that Jack is at the beach, and we are told that the weather has turned cold, we might look for Jack going home, or perhaps packing up to go home, or the like. If Jack asks Janet if she wants to go to the beach, and she asks if it will be cold, we are not interested in any of these things.

There is another possibility which would constitute a solution to the "question" problem. If understanding why a question was asked were based on only a few facts, which were the same for most questions, then every demon, after seeing

that it had caught a question, could call a single routine to process the question. We shall see that this seems to be the case.

9.7 Necessary Subgoals and Reminders

Consider:

(9.31) Jack and Janet were outside the house. Jack said "I want to talk to Daddy." Janet said "Where is Daddy?"

(9.32) Jack was going to the beach. Janet asked him if she could go along. Jack said that he would have to ask Daddy. Janet asked "Where is Daddy?"

In (9.32) but not in (9.31) we recognize Janet's question as motivated by the possible future "Jack talk to Father". The conclusion seems to be that the questioner must desire the possible future. With this in mind, let us make a first try at writing the necessary function.

```
(SUB-ROUTINE QU-HANDLE
   (NOLD PERSON NOLDP NSD)
   (?NOLD)

        Takes one argument, ?NOLD, the possible future.

   (GOAL (? WANT PR ST $E (GETSPK) ?NOLD)$TRUE $DEDUCE)

        $E simply says to evaluate the next expression
        and use its value in the pattern.  This GOAL
        asks if we can establish that the speaker (found by
        (GETSPK) ) wants the relevant possible future.

   (GOAL (?NDS SAID PR ST $E (GETSPK) $E (CURRENT)) $TRUE
                                                    $DEDUCE)

        This last line is simply to get the assertion
        number (?NDS) of the current SAID assertion.
        We then assert that the person said the question
        because of ?NOLD, the possible future.

   (ASSERT (? RESULT ?NSD ?NOLD)))
```

This will work for such questions as:

(9.33) Where is the baseball?
(when the speaker wants to play ball.)
(9.34) Will you go there?
(after the last question is answered.)
(9.35) How can I get the paints?
(9.36) Do you have a ball?
(9.37) Do you have any kleenex?

This, of course, is with the provision that the questions are asked in the appropriate context, and we have a demon which could handle the normal statement form deduction. That is, in (9.37) the person would have a cold, or would be about to sneeze, and we would have a demon which could answer the question "Why did Janet get the kleenex" in the story:

(9.38) Janet was about to sneeze. She got some kleenex.

In general, our routine QU-HANDLE will work for situations where something is wanted, because it is a necessary subgoal (NSG) of another situation, and we shall call this type of question the NSG type.

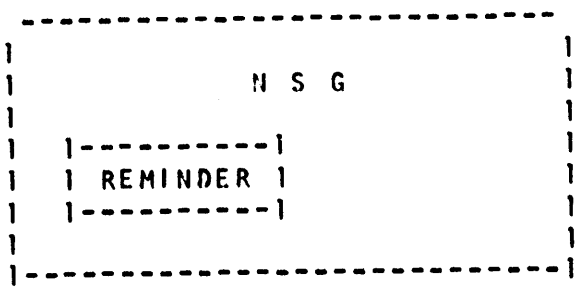
NSG is not the only type. Consider:

(9.39) Jack was about to leave the house. Mother asked him if he had his keys.

The reason Mother asked was to remind Jack that he should have his keys because he was about to leave the house. We must distinguish between this and the NSG case for two reasons: we used the word "remind" in describing Mother's motivation, and it was not necessary to establish that Mother had any stake in Jack's going out, as we did for NSG types. The distinguishing feature seems to be that having one's keys is not necessary for leaving the house in the sense that having a baseball is

necessary for playing with it. Or, to put it another way, you don't have to remind Jack that he should have a baseball, when he is about to go out and play with it. It should be noted that not-strictly-necessary-sub-goals (or REMINDERS as we shall call them) are, from our point of view, a proper subset of NSG types, since if Jack asked "Where are my keys?" we would handle it like a NSG. That is, we would say that he asked because he was about to leave the house.

To illustrate the situation graphically, we have:



The new version of our function appears on the next page.

```
(SUB-ROUTINE QU-HANDLE2
   (NOLD PERSON NOLDP NSD QU-TYPE)
   (?NOLD ?QU-TYPE)

        Takes two arguments, the statement that a
        person wants something and the kind of question.

   (COND ((GOAL (? WANT PR ST $E (GETSPK) ?NOLD)$TRUE $DEDUCE)
          (SETQ ?QU-TYPE 'NSG))

        If a reminder is used in an NSG situation it
        becomes an NSG type question.

         ((EQUAL ?QU-TYPE 'REMINDER)))
   (GOAL (?NSD SAID PR ST $E(GETSPK) $E (CURRENT)) $TRUE
                                                   $DEDUCE)
   (PUTPROP ?NSD ?QU-TYPE 'QU-TYPE)

        Mark the question type on the SAID statement.

   (ASSERT (? RESULT ?NDS ?NOLD)))
```

9.8 "Information For Decision" Type Questions

In an earlier example, which we didn't go into in any depth, we basically had:

(9.40) Jack asked Janet if she wanted to go to the beach with him. Janet said "Will it be warm?"

A little thought indicates that this example falls into neither category we have considered so far. Janet has not yet decided that she wants to do it (so it can't be NSG), nor is she reminding Jack of anything. The idea here is that Janet has to make a decision, and she wants more information in order to make it. It is not necessary for Jack to have actually asked Janet. He might have said "I would like you to go to the beach with me," in which case the situation would be the same. Nor does the person asked need to be involved in the proposed action, as in "Mother, may I go to the beach?" said Jack. "Will it be warm?" asked Mother. The common denominator is that the asker must make a decision. The decision can either be forced by a "may" question, or implied by a possible future in which the askee will participate such as, "I want you to go to the beach with me."

We have seen that all REMINDER questions are also NSG 's. This prompts us to ask about the relation between IFD questions (Information For Decision) and the two previously defined types. Consider:

(9.41) Jack was about to leave the house. He asked Janet "Do you want to come with me?" Janet said "Where are your keys?"

The story does not make a lot of sense. The best we can do with it is assume Janet either did not hear, or chose to ignore Jack's question. Note that if we really want to stretch the matter we could interpret Janet's question as IFD, since having keys is useful for going out, and Janet's decision concerns going out. But most readers would regard this as over-stretching the meaning. The conclusion, based only on this one example (so far) is that REMINDERS cannot be IFD's.

"Where will you go?" is clearly IFD, but it can't be REMINDER. The same goes for "When will you go?" These also seem to suggest that IFD's can't be REMINDER's. But our complete separation of IFD's and REMINDERS is not absolutely hard and fast.

(9.42) "Do you want to go to dinner with me, Janet?" asked Jack. "How much money do you have?" asked Janet.

The last question is ambiguous. If we know Janet is a grasping little girl, we will interpret this as IFD; if Jack and Janet have an easy-going relationship we will call this REMINDER. However, (9.42) strikes me as a strange case. It is the first case we have seen where the questioner's motivation is clearly ambiguous. Then until further evidence is gathered I will continue to assume that IFD's and REMINDERS are mutually exclusive.

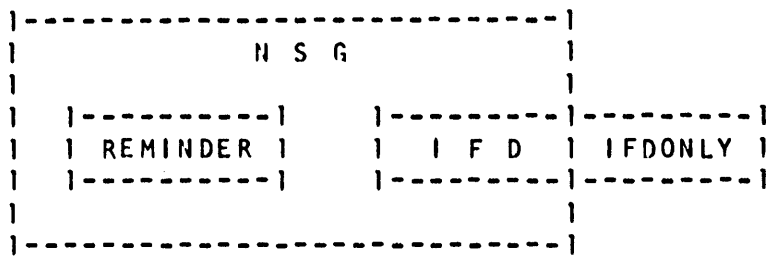
Now, IFD's can be NSG's as indicated by:

(9.43) "I have to talk to Daddy, do you want to come with me?" asked Jack. "Where is Daddy?" asked Janet.

The last question is one we have already seen in the guise of an NSG type. But not all IFD's are NSG's.

(9.44) Jack wanted to go to the beach. "Will it be warm?" he asked.

The last question, if it can be understood at all, implies that Jack is trying to decide if he will go. That is, even following a statement which implies that Jack has already decided to go, the question implies doubt. If the question could also be NSG, we could have understood it as "He asked because he was going to the beach," without any reference to doubt. So there are really two kinds of IFD types, those which can be NSG 's which we will still call IFD and those which can't, which will be called IFDONLY. To make sure we have the distinction clear, in the context of "'Will you play ball with us?'" consider the questions "When will you be playing?" and "Who will be playing". If the first is asked before the asker has decided it is IFD, if after, NSG. The second can only be IFD, since if it is asked after the asker has decided, it can only be chalked up to curiosity. So we can diagram our types as:



To say that a person asked a question because he needed information for a decision implies that we know that he has a decision to make. How can we determine this? To put this another way, we want to determine if the speaker is making a decision about some possible future. Clearly the questioner cannot have already decided to participate in the possible future, or else there would be no decision to be make. What other conditions should we look for in trying to decide that a decision has to be made?

1) The speaker is part of the possible future referred to in the question. So in "Will you go to the store?" "When?" we understand the last question as referring to the possible future (questioner possibly going to the store). We say that the questioner is directly involved in the possible future.

2) The questioner must decide on giving permission (but the questioner will not be involved in the action). "May we go to the beach, Mother?" "When will you go?"

3) The questioner is not involved in the possible future referred to in the question, but the question is whether the questioner will do it also, as in "Do you want to play

baseball with us?" or "When will you play?"

4) There are other troublesome cases which we will mention later, but which we will not worry about.

A routine incorporating these ideas is given on the next page.

```
(SUB-ROUTINE QU-HANDLE3
   (NOLD QU-TYPE NOLDP N P NSD)
   (?NOLD ?QU-TYPE)
   (COND ((GOAL (? WANT PR ST $E (GETSPK) ?NOLD)$TRUE $DEDUCE)
          (NOT (EQUAL ?QU-TYPE 'IFDONLY))

                    NSG is as before, but we do not
                    accept IFDONLY's

          (SETQ ?QU-TYPE 'NSG))
         ((EQUAL ?QU-TYPE 'REMINDER))
         ((NOT (MEMQ ?QU-TYPE '(IFD IFDONLY))))

                    I.e., the rest only applies to IFD's.

         ((GOAL (? WANT PR ST $E (GETSPK) ?NOLD)$TRUE $DEDUCE)
          (FAIL THEOREM))

                    If the possible future is desired
                     then IFD does not apply

         ((INVOLVED-IN (GETSPK) ?NOLD))

                    Covers Case 1 above.

         ((GOAL $E(CREATE (?NOLD ?N) (4 QU)) $TRUE $DEDUCE)

                    Case 2.  CREATE makes up a new
                    assertion based on ?NOLD, but
                    with ?N in the assertion number
                    position and QU in the fourth position.

          (GOAL (? LET PR ST $E (GETSPK) ?N) $TRUE $DEDUCE))
         ((GOAL $E (CREATE (?NOLD ?)(5 ?P))$TRUE $DEDUCE)

                    Case 3.

          (OR (EQUAL ?P $E (GETSPK))
          (GOAL (? MEM $E (GETSPK) ?P)$TRUE $DEDUCE))))
   (GOAL (?NSD SAID PR ST $E (GETSPK) $E (CURRENT)) $TRUE
                                                    $DEDUCE)
   (PUTPROP ?NSD ?QU-TYPE 'QU-TYPE)
   (ASSERT (? RESULT ?NSD ?NOLD)))
```

9.9 Further Problems

So far, QU-HANDLE has only been called from demons. That is, we have never called it from a base routine. It seems quite clear that at least for IFD's we will have to do so. Consider "Will you go to the store?" "When?". The second question is handled from the base routine for time reference. That is, when we mention a possible future, we don't always put in a demon looking for statements talking about when they will be done. Instead, should we get such a statement, we look back to decide what it is referring to. So the base routine finds "you possibly go to the store" and hands that off to the question routine as ?NOLD . Now in principle there is no difference, no matter where the information comes from. But the fact that we have now included all base routines as possibly having to call QU-HANDLE should cause us to wonder exactly what demons and base routines can be the basis for legal questions, and what type of questions they will be.

I don't know the answer to this question. Nor is it the only remaining problem. A few others:

In the last section we tried to outline the necessary relation between a possible future and the person who, we suspect, is asking an IFD type question. The "troublesome" cases mentioned there are cases like:

(9.45) Jack and Bill were outside. Bill said, "Will you watch my dog while I am at the store?" "How long will you be at the store?" asked Jack.

Note that the questioner is not going to the store, so the possible future has nothing to do directly with Jack. It is the time concurrence which makes the question relevant.

Another problem. There may be some overlap in what we have discussed as clearly distinct categories. So, if you are driving a car, and someone says "How much gas do we have?" is he reminding you that you might need gas, or getting information to decide if you should stop at the next station. This may indicate that our definitions need further sharpening, but more likely we need to recognize and know how to handle the doubtful cases which arise.

A kind of question motivation very different from what we have seen seems to be involved in:

(9.46) Mother was standing in the yard. The family dog came from behind the garage, covered with paint. Jack and Janet were following the dog. "What did you do to the dog?" asked Mother.

If we asked "Why did mother ask that?" a reasonable answer would be "Because the dog was covered with paint." Presumably the paradigm here is "any question about an unusual event was caused by the event." All we have to do is define "unusual event" and we are all set.

10 A Summing Up

10.1 Looking Back

In trying to summarize this thesis we might look at some of the threads which wander through many of the chapters.

The reader of a children's story must have a broad range of knowledge available to be able to understand the story. At any given point however, only a small portion of that information is relevant. To cut down on the information which could be used to make deductions we devised a "topic" rule, which simply says that every piece of information has one (or more?) "topic concepts". A fact cannot be made available until its topic concept appears in the data base. So situations persistently arise in which the significance of a line like "Jack has a top" depends strongly on what happened earlier in the story. In our model this implies that the information which interprets the line has a topic concept which appeared earlier. In example (2.19), where Janet is thinking of getting Jack a top, "Jack has a top" implies that she should not do it. The information set up by earlier events gives the new line its interpretation. Hence we represented the information as a demon which is on the lookout for a particular kind of event in the story.

Before we could use demons, however, there were several problems which had to be ironed out. A demon should embody a

somewhat general fact about a situation independent of the particular people or objects involved, so we need machinery to specify demons to specific situations. Again, if demons are to represent the current relevant information, we need machinery for removing demons when no longer needed. First order solutions to some of these problems were proposed in chapters 4 and 8. The major problem, constructing appropriate demons, was examined in chapter 5 and to a lesser extent in chapter 9.

Closely related to the idea of demons is the idea of "possible futures". Simply, a possible future is an assertion about something which might occur in the future. These include not only straight futures, ("I will do X") but also statements about "want" or "must" or "can". Possible futures are the major, but not sole source of the topic concepts which introduce information which will not be used until later in the story.

Another theme, not as prominent as demons, is the idea that our program should make deductions as it goes along. We originally argued that point in chapter 2, and much of the rest of the thesis is directly dependent on this assumption, in particular the discussion of reference in chapters 3 and 6.

"Deduction on the fly" is also an underlying basis for other decisions in the thesis. For example, this is why we need the distinction between demons and fact finders. There

are a huge number of deductions which could be made about a story. We don't want to make all of them, so we need some way to distinguish those deductions which are worth making as we go along from those which aren't. In our model, the first become demons, the second, fact finders. Naturally, in delayed deduction models this problem does not come up because the deductions one makes are those needed to answer the "user" questions.

Probably the most well developed idea running throughout the thesis is the relation between complex deductions and the reference problem. The topic first came up in chapter 2 as an argument for "understanding" as we went along in the story, rather than waiting for a question before anything was done. It came up again as the main point of chapter 3 where we showed that our model could very naturally account for many situations where past portions of the story influenced reference decisions. What we found was that in the course of trying to fill in the blanks of the story, (such as establishing the causal relations between lines) the model would naturally have preferences about referents for undecided noun phrases. But our method of handling reference forced us to be very careful in organizing the model, and this was the basic impetus for chapter 7 where we showed that the ordering of the basic processes within the model made a difference in our ability to handle reference problems. Reference also was

the topic of chapter 6 where we found that with slight modifications our reference procedure would handle most cases of undetermined noun phrases (like "a ball" in "I want a ball") and over-specified noun phrases (where the noun phrase gives more detail about the object than we currently know). Also, our analysis of ambiguity in chapter 9 was influenced by the work on reference.

10.2 Looking Forward

Throughout this thesis we have constantly run across problems which will require further attention. Some of these problems are linguistic in nature. How does one determine sentential references, as in "It took place in Joe's house"? While a complete answer will depend on how our knowledge of the world is organized, such questions are more about "language" than about "real world knowledge" as we have been discussing it. Or again, how significant is the problem of syntactic ambiguity? Do we need methods similar to those for reference ambiguity? What is the semantic significance of sentential connectors such as "but", "and", "so"? Questions like these need to be answered, and at first glance it would seem that given a rough model like the one proposed in this thesis, they can be answered independently of more detailed information about how our knowledge of the world is organized.

My main interests lie in trying to find how our common sense knowledge should be organized. The way to proceed, it seems to me, is by continuing to organize small "micro worlds" of knowledge as we did with piggy banks. I expect to see several kinds of results come out of such an effort. The most obvious is that as we examine more domains more and more generalizations will begin to appear. We will, for example, be able to collect examples which amplify, or perhaps refute, some of the tentative generalizations we made in chapter 5 by

looking only at piggy banks. Then, given more partially built up sub-worlds we will be able to give more serious considerations to more global problems, like the effect of tense on the deduction process, problems of jumping to conclusions, or the concept of making mistakes. These problems all share the property that by their very nature they enter into most sub-worlds.

Another purpose of studing micro worlds is to test our assumptions about the basic structure of our model. In section 2.5 we looked at some of the assumptions we had made up to that point, but there are many more. For example, our model only interprets the story as it happens. In chapter 5 we briefly looked at one problem in trying to answer questions about what is likely to happen later in the story, but otherwise we only considered questions which interpreted actions which were mentioned in the story. However, there are other kinds of questions which depart from our "interpretation" limitation, such as, "What would you do if your were Jack?", and "How would you get some money if you needed it now?" These questions point out the fact that a child's knowledge is not there only to enable him to read stories. He has it to be able to get about in the world. So in some sense these "extra-story" questions are the most "natural", yet they are questions which our model cannot handle. However, such questions would involve us in the

storage of "personal information" (e.g., the fact that I keep my money in an old shoe.) It is an open question whether such "personal knowledge" is organized the same way as the "every day knowledge" we have been discussing. In fact, it is an open question whether a hard distinction can even be drawn between the two.

Another aspect of our model which we have not challenged is the unstructured data base of knowledge about the story (as opposed to our general "every day knowledge"). When we place assertions in the data base they are filed away according to their "syntactic" structure, i.e., what symbols appear in the assertion and in what order. We might ask how useful it would be to have a more "semantically" organized data base? What would such a data base look like?

The model we have proposed is somewhat "local". Demons look for small pieces of information which they then try to put together into somewhat larger patterns. But I certainly never get to "very large patterns" such as the "topic" of a story. Is this a serious defect in the model? How useful would it be to have available the model's best guess at the topic of a story? There is no question that people can compute "topic", if only to answer a question like "What would be a good title for this story?" The question is whether "topic" is important in the understanding process. So further study might indicate that "topic" can help decide which demons

should be asserted and which deductions made.

When we look at more complex stories there are other "larger patterns" which we must consider. What are the narrator's (or author's) biases, or attitudes about the story? It is interesting to consider whether questions such as those encountered in English literature courses can be answered by "local" demons. Do we have specific information about "detecting authors' biases" just as we have information about "piggy banks", or will other techniques prove to be necessary?

NOTES

1) Since my system is "understanding" as it goes along, it is not really necessary to ask a question in order to make sure it knows the answer. Instead one can just look in the data base to see if the answer is there. So when I say "To determine what the model has understood we will ask it questions," I do not really mean that such a behavioristic test is necessary. Rather, I mean that the ability to answer questions was what I had in mind as the definition, so to speak, of understanding for the system.

2) Another probable constituent is what might be called "conversational" information. Under this heading would fall the well known rule that a pronoun is more likely to refer to the subject of the last sentence than the object. For more details see the last section of chapter 8.

3) The backup hypothesis is aimed at those noun phrases which can be assigned a referent confidently before going on to read the rest of the story. There are cases, however, where we simply make a mistake, and we are only able to correct it when we have read more of the story. In such cases we will have chosen a referent which indeed gets us into trouble, and we will have to go back and correct it. However, we should not

confuse the "mistake" situation with the backup hypothesis. When we make a mistake, we are conscious of going back and revising our understanding of the story. In normal reference decisions, on the other hand, we don't even seem to be aware that the NP has more than one possible referent, much less that we have been backing up and changing our interpretation.

4) Actually, things are somewhat more complicated. When a Planner theorem fails for any reason, the program tries to go back and "fix things up" by seeing, for example, if there are any other assertions in the data base which also match an earlier goal. This way, if the GOAL binds any variables, we may get a new value for the variable, which may allow the theorem to go through the second time. See chapter 4.

5) We can see that we need DGROUPS because of sentences like

> "Look, Janet" said Jack. "See the paints and pencils Daddy got for us!"

We do not know who will be getting what, or if the paints and pencils will be shared. Until the line is disambiguated we must leave it in some ambiguous form, and this requires DGROUPS or their equivalent.

However, it should be noted that using dgroups is somewhat costly in terms of deduction, since then, if we want to prove that F is true of X, we must check to see if there is some dgroup D such that F is true of D and X is a member of D.

Also, dgroups will complicate updating. If we have a single assertion which says that Jack and Bill are at the store, and we are then told that Jack left, we must update the old assertion. At the same time we want to keep the fact that Bill is still at the store, so that will have to be reasserted separately.

6) For example, we cannot ask a "how" question about a state, as in:

*How
does Jack have the ball.
Why

Also, when the subject of the question is a non goal directed object we can use T-RESULT links to answer "why" questions, as in:

*How
did the top fall?
Why

Jack dropped it.

7) One might consider an alternative to using ASSUMPTION tags; don't assert the assumed fact, but put in a DEMON which will handle questions based on the assumed fact. In our particular example, the theorem would answer the question "Does Janet have the money?" This scheme seems to offer the possibility that we can answer the question "Does Janet have money?" without really asserting that she does. To put it another way, we don't commit ourselves to saying she has the

money unless we are specifically asked. But this dosen't solve enough problems. Note that our "tag" scheme has two components to the solution. First we need to say how we will answer the question under normal circumstances (by having the assertion in the data base). Then we need to specify how we will decide that we were wrong (by allowing updates of assumed assertions to erase the assumed assertions). However, our new scheme does not have any mechanism for deciding we were wrong. We could add a mechanism to remove the theorem when a later statement contradicts it, but then we really have our tag method, only complicated by the fact that we are using theorems rather than assertions. So the tag method seems better.

8) The other example of jumping to conclusions is harder to correct. In essence we want to say that "find" has as a T-RESULT Janet's having the nickel. But the uniqueness of T-RESULTs would then force us into denying that Janet got the money from the PB. The problem with this analysis is that the find could occur after the "get from PB" in which case the two are clearly not mutually exclusive. This leaves us with two unsolved problems; 1), the general problem of determining the order of past tense events, and 2), determining what information tells us that the find did not occur after the "get from PB".

9) In fact, there are three unassigned variables in the pattern. That the assertion number is unassigned is usual. We discussed in 4.1 why our asserting function should assign temporary numbers in such cases. That the tense is initially unassigned is more of a problem. It is clear that the assertion we want to end up with is FUT, in that we have established that Janet needs money, not that she has some. And in fact, in the course of BP-FOR-MONEY the tense will be assigned to FUT by the expression (EQUAL $?TN 'FUT). But this seems almost a coincidence. I have no idea about what is really going on.

10) Another problem with this demon as stated is the fact that I-C is an indicator of location, but by our standard ordering of argument to predicates, the object in an I-C relation is the last argument, while the argument in any other location statement will be the second to last argument. This could be corrected by changing either I-C or the various location statements. Off hand there seems to be no reason not to do so.

11) Actually, these examples by themselves do not commit us to denying the "command" rule, since it is possible to use other facilities in transformational theory (which is the

basis for all the linguistics papers mentioned) to bring the "command" theory and this most recent data into accord. However, Lakoff in his paper (Lakoff 68) gives further examples to eliminate these other posibilities.

12) Actually this is not quite true. One other possibility is that we have a phrase like "the yellow one" which refers to a particular object, even though we didn't previously know it was yellow. We will not worry about this case. An even more complicated problem would be NP's which were "wrong" in the sense that the name the person assigned to the object was not correct for some reason. So a child might call an orange ball an orange, either thinking that it was an orange, or perhaps even because the distinction between the two was not completely clear in his mind. Such problems are also beyond our scope.

13) If we are to take this approach we no longer have the simplicity of our original double negation technique. One possible escape is to say that what happens is that the double negation technique changes imperatives into statements for the purpose of deriving counter-examples. While this possibility would work for the examples given so far, it would fail on

> Jack and Janet were in the house. Jack was holding some paints and pencils. When Janet walked into the room she did not see them. Jack said "Janet, see the paints and pencils Daddy got for us."

We would, in fact, be able to derive the fact that Janet did not see the paints and pencils, since we said exactly that. We would thus eliminate them from consideration, giving a clearly wrong result. The reason is that we treated the imperative as a statement and tried to prove the negative of the statement.

14) We might argue instead that we should have a fact-finder which says "if you want to prove that X did not burn Y, show that Y still exists". Then this fact-finder would allow us to prove that "the yellow ball" could not be the one Jack is using, without resorting to full DSP treatment. However, unless we show that such a F-F is needed for independent reasons, the solution is ad hoc. If we try to ask, "Did Father burn Jack's ball?" in the middle of some story where Jack has been playing with a ball but Father hasn't burned anything, it seems very unnatural, though the question is answerable. If father has burned something, then the question can be answered without such a fact-finder. We would simply say, "No, Father burned the ...". Then, if we had such a fact-finder, we would be predicting that such a question would be natural under any circumstances. If, on the other hand, we want to account for our ability to answer such questions, even when they seem silly and irrelevant, we might postulate that we assert that Father did burn the ball, and then note that

this implies that the object is no longer usable, contrary to what we already know. If we postulated that questions which force us to resort to such a mechanism are "unnatural", our model would not only exhibit the correct behavior, but would even explain our reactions to certain types of questions. Whether or not this suggestion will go through, it seems certain that the fact-finder proposed is untenable.

15) It was Bob Moore (office mate and fellow MIT graduate student) who convinced me that a rule like this is necessary, and that (6.68) was not sufficient evidence against it.

16) The necessity of assigning a number immediately is not completely obvious, since as we will see later, the system allows assertions with unassigned variables. However, it is not possible to leave assertion numbers unassigned. Suppose we needed to go from (N1 RESULT N2 N3) to the assertion represented by N3. If, rather than N3 we had an unassigned variable, locating N3 would be most difficult. In fact, all assertions with unassigned variables would match equally well since N1 above gave no indication of what N3 looked like.

17) There is one final point which should be made explicit. If we are going to put assertions into the data base before applying DSP, and if we use the normal data base mechanism in

Micro Planner (which seems to be the simplest way) then we will have to allow assertions with variables. This is contrary to currently standard Micro Planner.

18) Since every demon will have as its first line DESTROY? in a system specifically designed for our use, this could be made part of the "definition" of a demon.

19) Those familiar with Planner might be somewhat confused about a GOAL statement which specifies the assertion number of the assertion to be used, while at the same time it allows theorems to be used. Presumably one might think that if we know the assertion to be used, then either it matches the pattern or it doesn't. The reason this is not the case is because we will want to allow an assertion saying "Jack is IN the house" to satisfy our goal which is asking, "Where is Jack AT?". To do this we will have a theorem which says, in effect, if you want to know the location of a person, find an assertion about him being IN, ON etc. In this case we have also told this theorem the assertion number of the IN, ON etc. assertion to be used.

REFERENCES

(Alexander 71) Alexander, B. "A Question-Answering Program for Simple Kernel Sentences," Technical Report No. NL-5. Computer Assisted Instruction Laboratory, University of Texas at Austin, 1971.

(Black 68) Black, F. "A Deductive Question-Answering System," Semantic Information Processing. Ed. M. Minsky, Cambridge, Mass.: MIT Press, 1968, pp. 354-402.

(Chomsky 65) Chomsky, N. Aspects of the Theory of Syntax. Cambridge, Mass.: MIT Press, 1965.

(Craig et al. 66) Craig, J. A. et al. "DEACON: Direct English Access and CONtrol," Proceedings of the 1966 Fall Joint Computer Conference. Baltimore, Md.: Spartan Books, pp. 365-380.

(Dewar, et al. 69) Dewar, H., Bratley, P., and Thorne, J. P. "A Program for the Syntactic Analysis of English Sentences," Communications of the ACM, XII, No. 8 (August, 1969), pp. 476-479.

(Dougherty 69) Dougherty, R. C. "An Interpretive Theory of Pronominal Reference," Foundations of Language 5, pp. 488-519.

(Evans 64) Evans, T.G. "A Heuristic Program to Solve Geometric-Analogy Problems," Proceedings of the 1964 Spring Joint Computer Conference. Baltimore, Md: Spartan Books, pp.327-338, (An expanded version in (Minsky 68)).

(Feigenbaum and Feldman 63) Feigenbaum, E. A., and Feldman, J. (eds.). Computers and Thought. New York: McGraw-Hill Book Company, Inc., 1963.

(Fodor and Katz 63) Fodor, J. A., and Katz, J. J. "The Structure of a Semantic Theory," Language, XXXIX (April-June, 1963), pp. 170-210.

(Green 69) Green, C. C. "Application of Theorem Proving to Question Answering Systems," Memo AI-96. Stanford Artificial Intelligence Project, 1969.

(Hewitt 69) Hewitt, C. "Planner: A Language for Proving Theorems in Robots," Proceedings of the 1969 International Joint Conference on Artificial

Intelligence. Ed. D. E. Walker and L. M. Norton. 1969, pp. 295-302.

(Jackendoff 71) Jackendoff, R. S. "Modal Structure in Semantic Representation," Linguistic Inquiry, II, No. 4. (Fall 1971), pp. 479-514.

(Karttunen 71) Karttunen, L. "Discourse Referents," Reproduced by the Indiana University Linguistics Club, 1971.

(Lakoff 68) Lakoff, G. "Pronouns and Reference Parts I and II," Reproduced by the Indiana Universtiy Linguistics Club, 1968.

(Lakoff 71) Lakoff, G. "On Generative Semantics," Semantics: An Interdisciplinary Reader in Philosophy Linguistics and Psychology. D. D. Steinberg, and L. A. Jakobovits Eds.. Cambridge: Cambridge University Press 1971.

(Langacker 69) Langacker, R. "On Pronominalization and the Chain of Command," Modern Studies in English. Eds. D. A. Reibel and S. A. Schane. Englewood Cliffs, N.J.: Prentice Hall, 1969, pp. 160-186.

(Lees and Klima 63) Lees, R. B., and Klima, E. S. "Rules for English Pronominalization," Language, XXXIX, No. 1 (1963), pp. 17-29.

(McCarthy and Hays 68) McCarthy, J., and Hays, P. "Some Philosophical Problems from the Standpoint of Artificial Intelligence," Memo AI-73. Stanford Artificial Intelligence Project, 1968.

(McCawley 1970) McCawley, J. P. "English as a VSO Language," Language, XLVI, pp. 286-299.

(McKee et al. 66) McKee, P., et al. Up and Away. Boston: Houghton Mifflin Co., 1966.

(Minsky 61) Minsky, M. "Steps Towards Artificial Intelligence," Proceedings of the Institute of Radio Engineers, XLIX (January, 1961), pp. 8-30. Also in (Feigenbaum and Feldman 63).

(Minsky 68) Minsky, M. (ed.). Semantic Information Processing. Cambridge, Mass.: M.I.T. Press, 1968.

(Quillian 66) Quillian M. R. "Semantic Memory," Semantic Information Processing. Ed. M. Minsky. Cambridge, Mass.: M.I.T. Press, 1968, pp. 216-270.

(Quillian 69) Quillian, M .R. "The Teachable Language Comprehender: A Simulation Program and Theory of Language," Communications of the ACM, XII, No. 8. (August, 1969), pp. 459-475.

(Ross 69) Ross, R. J. "The Cyclic Nature of English Pronominalization," Modern Studies in English. Eds. D. A. Reibel and S. A. Schane. Englewood Cliffs, N.J.: Prentice Hall, 1969, pp. 187-200.

(Sandewall 72) Sandewall, E. "PCF-2, A First-Order Calculus for Expressing Conceptual Information," Working Paper, Uppsala University Department of Computer Sciences, 1972.

(Selfridge 59) Selfridge, O. G. "Pandemonium: A Paradigm for Learning," Proceedings of the Symposium on Mechanisation of Thought Processes. 2 vols. National Physical Laboratory, Teddington, England, London: H.M. Stationary Office, 1959, pp. 511-529.

(Shank and Tesler 69) Shank, R. C. and Tesler, L. G. "A Conceptual Parser for Natural Language," Proceedings of the 1969 International Joint Conference on Artificial Intelligence,. Eds. D. E. Walker and L. M. Norton, 1969, pp. 569-578.

(Simmons et al. 68) Simmons, R. F., Burger, J. F., and Schwarch, R. M. "A Computational Model of Verbal Understanding," Proceedings of the 1968 Fall Joint Computer Conference. Washington, D.C.: Thompson Book Co., 1968, pp. 441-454.

(Simmons 70) Simmons, R.F. "Natural Language Question Answering Systems: 1969," Communications of the ACM, XIII, No. 1 (January, 1970), pp. 15-30.

(Simmons 70a) Simmons, R. F. "Some Semantic Structures for Representing English Meanings," Technical Report No. NL-1. Computer Assisted Instruction Laboratory, The University of Texas at Austin, 1970.

(Steinberg and Jakobovits 71) Steinberg, D. D., and Jakobovits, L. A. Semantics: An Interdisciplinary Reader in Philosophy Linguistics and Psychology. Cambridge: Cambridge University Press 1971.

(Sussman et al. 71) Sussman, G.J., Winograd, T., and Charniak, E. "Micro Planner Reference Manual," AI Memo No 203A. MIT Artificial Intelligence Laboratory, 1971.

(Tharp and Krulee 69) Tharp, A. L., and Krulee, G. K. "Using Relational Operators to Structure Long-Term Memory," Proceedings of the 1969 International Joint Conference on Artificial Intelligence. pp. 579-586.

(Vendler 67) Vendler, Z. "Singular Terms," Linguistics in Philosophy, Ithaca, N.Y.: Cornell University Press, 1967. Also in (Steinberg and Jakobovits 71).

(Weissman 67) Weissman, C. LISP 1.5 Primer. Belmont California: Dickenson Publishing Company Inc., 1967.

(Winograd 71) Winograd, T. "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language," MAC TR-83. Project MAC, MIT, 1971.

(Winston 72) Winston, P. H. "Summary of Selected Vision Topics," Vision Flash 30, MIT Artificial Intelligence Laboratory, 1972.

(Woods 68) Woods, W. A. "Procedural Semantics for a Question Answering Machine," Proceedings of the 1968 Fall Joint Computer Conference. Washington, D.C.: Thompson Book Co., 1968, pp. 457-471.

(Woods 69) Woods, W. A. "Augmented Transition Networks for Natural Language Analysis," Report No. CS-1 to the National Science Foundation, Aiken Computation Laboratory, Harvard University, 1969.

APPENDIX A - Actual Input for Example in Chapter One

The input is divided first into pages (corresponding to the pages in the original) and then into sentences. So we have:

```
(       ( sentence 1 )              the first page
        ( sentence 2 )
                .
                .
                .
        ( sentence n )      )

(       ( sentence 1 )              the second page
                .
                .
                .           )

        etc.
```

Within each sentence there are three sections, the last two sections being optional. The first section is a list of the assertions which represent the original sentence. The second section is a list of tags. The only tags are:

SD (person) - Who said the sentence (only included if the original English explicitly mentioned who the speaker was.)
TO (person) - The person to whom the sentence was addressed (again, only if explicitly stated)
"1 - Indicates that the sentence started with an open quote
"2 - indicates the presence of a closed quote (if a sentence has a SD marker and both an open and closed quote then both quote markers may be deleted.)

The last section is just a copy of the original English sentence.

So an individual sentence looks like:

```
(       (       ( assertion 1 )
                ( assertion 2 )
                        .
                        .
                        .
                ( assertion n )       )

        ( tags )
        ( English )       )
```

To understand the syntax within an assertion one should first read section 4.1. That describes the internal format, which is almost identical to the input format. However there are a few additions necessary for input. Since we naturally don't know the assertion number for an incoming assertion, its place is taken by either a Micro Planner variable (in the case that the assertion is embedded and another assertion of the same sentence refers to it,) or a "?" when no other assertion in the same sentence refers to it. So we have:

```
(? BEFORE ?A1 ?A2)
(?A1 KICK JANET JACK)
(?A2 KICK JACK JANET)
```

Also we don't know the internal symbols for the objects in the story, so the objects in the assertions are usually represented by the noun phrases used to describe them in the original English. To distinguish a description of an object from an actual part of the assertion, the descriptions have parentheses around them. So if we really wanted to say "Jack kicked Janet" we would have (including tense and type markers):

```
(? KICK PR ST (JACK) (JANET))
```

The program has only limited abilities with respect to understanding the syntax of noun phrases. Some typical NP's are

```
        "the red pencil"  -  (THE RED PENCIL)
       "the red pencils"  -  (THE RED PLM PENCIL)
        "Jack and Janet"  -  (DGROUP (JACK) (JANET))
                  "me"    -  (ME)
            "my paints"   -  ((ME) 'S PLM PAINT)
 "the paints I gave you"  -  (THE PLM PAINT SUBCL
                               (? GIVE PAST ST (I) (YOU) ***))
```

Any NP which will not fit into these molds has to be expressed in terms of the actual assertions we will need to have in order to describe the object in the data base. The only object in this story which required this was "toy cat" which is represented by:

```
((NIL NIL NIL) (? IS PR ST ?CUR TOY)
               (? SHAPE PR ST ?CUR CAT)))
```

This says, ignoring the three NIL' s in the front, that the object is a toy in the shape of a cat.

Sometimes a single NP will be a part of more than one assertion. In such cases the second occurrence of the NP will have a * as its first element, which tells the program that this NP is identical with one it has already seen. Note that the program first looks at the tags before it looks at the assertions, so that any NP which first occurs in the tags (as in SD (JACK)) will be a repeat in the assertions. In the case of long repeated NP's it is only necessary to give enough to distinguish the NP from all others, starting from the

beginning of the NP. So for example if we only have one dgroup in a sentence we can repeat it by saying (* DGROUP)

This should be enough to understand the representation of the story.

```
((((? IN PR ST (DGROUP (JACK) (JANET)) (THE HOUSE)))
  (JACK AND JANET ARE IN THE HOUSE))

 (((? HOLD PR ST (JACK) (DGROUP (A BOX OF PLM PENCIL)
                                (A BOX OF PLM PAINT))))
  (JACK IS HOLDING A BOX OF PAINTS AND A BOX OF PENCILS))

 (((? SEE PR IMP (YOU)
      (DGROUP (THE PLM MODM PAINT)
              (THE PLM MODM PENCIL)
              SUBCL
              (?A3 GET PAST ST (FATHER) (* DGROUP))
              (?A2 GOAL ?A1 ?A3)
              (?A1 GIVE FUT ST (* FATHER) (US)
                               (* DGROUP)))))
  (SD (JACK) TO (JANET))
  ("JANET SEE THE PAINTS AND PENCILS THAT DADDY GOT FOR
   US"SAID JACK))

 (((?A1 GO PR ST (JANET) (UNM))

     The sentence does not say where Janet went, so the
     place where this information should be is filled with
     (UNM) which just means "unmentioned."

   (? GOAL ?A2 ?A1)
   (?A2 SEE PR ST (* JANET) (THEM)))
  (JANET WENT TO LOOK AT THEM))

 (((? GET PR QU (ME) (THE PLM PAINT)))
  (SD (SHE))
  ("ARE THE PAINTS FOR ME?" SHE ASKED))

 (((? POSSESSIVE PR ST (ME) (THE PLM PAINT)))
  (SD (JACK))
  ("THE PAINTS ARE MINE " SAID JACK))

    The original was "No, the paints are mine." Handling
    "no" is not a particular problem, at least in this case
    where the "no" negates the very last line. In fact, the
    reason for omitting the "no" was to make sure that the
```

program would realize that the answer was "no".

```
(((? GET PR ST (YOU) (THE PLM PENCIL)))
 ("2 TO (JANET))
 ("THE PENCILS ARE FOR YOU JANET))
```

Two lines of the story are skipped at this point. One was repetition and the other was "When you get to be as big as I am, you will get paints, too." which I had no idea how to handle.

```
(((? WANT PR ST (JANET) ?A1)
  (?A1 HAVE FUT ST (* JANET) (THE PLM PAINT)))
 (JANET WANTED THE PAINTS))

(((? DISCOVER FUT ST (I) ?A1)
  (?A1 CAN FUT ST (* I) ?A2)
  (?A2 DISCOVER CAN ST (* I) (UNM))
  (? WAY PR ST (* UNM) ?A3)
  (?A3 GET CAN ST (* I) (THEM)))
 ("2)
```

This line is a throw away in the sense that the program does not really understand it at all. Any line mentioned here but not in chapter one is either a repetition or a throw away.

```
 (I WILL SEE IF I CAN FIND A WAY TO GET THEM " SHE SAID
  TO HERSELF)))

((((? BEGIN PR ST (JACK) ?A1)
   (?A1 PAINT PR ST (* JACK) (A PICTURE OF (A RED
                                            AIRPLANE))))
  (JACK BEGAN TO PAINT A PICTURE OF A RED AIRPLANE))

(((?A1 GO PR ST (JANET) (UNM))
  (? GOAL ?A2 ?A1)
  (?A2 SEE PR ST (* JANET) (IT)))
 (JANET WENT TO LOOK AT IT))

(((? RESULT ?A2 ?A1)
  (?A1 USE PR ST (UNM) (THOSE PLM PAINT))
  (?A2 FUNNY PR ST ((YOU) 'S AIRPLANE)))
 (SD (SHE))
 ("THOSE PAINTS MAKE YOUR AIRPLANE LOOK FUNNY " SHE SAID))

(((? COULD PR ST (YOU) ?A1)
  (?A1 USE-IN FUT ST (* YOU) (THESE PLM PENCIL) ?A2)
  (?A2 MAKE FUT ST (* YOU) (A GOOD PICTURE OF
                            (A RED AIRPLANE))))
```

```
 ("1)
 ("YOU COULD MAKE A GOOD PICTURE OF A RED AIRPLANE WITH
  THESE PENCILS))

(((? LET FUT ST (I) ?A1)
  (?A1 HAVE FUT ST (YOU) (THE PLM PENCIL)))
 (I WILL LET YOU HAVE THE PENCILS))

(((? HAVE FUT ST (I) (THE PLM PAINT)))
 (I WILL TAKE THE PAINTS"))

  Skipped over the line "Jack looked at the pencils."

(((? NO))
 (SD (JACK) TO (JANET))
 ("NO THANK YOU JANET" SAID JACK))

(((? WANT PR ST (I) ?A1)
  (?A1 PAINT FUT ST (* I) (MORE PLM PICTURE)))
 ("2)
 ("I WANT TO PAINT MORE PICTURES"))

(((?A1 LOOK PR IMP (YOU) (THESE PLM PENCIL)))
 (SD (JANET))
 ("BUT JUST LOOK AT THESE PENCILS " SAID JANET))

(((? WANT FUT ST (YOU) ?A1)
  (?A1 HAVE FUT ST (* YOU) (THEM)))
 ("2)
 ("YOU WILL LIKE THEM"))

(((? MORE PR ST ?A1 (UNM))
  (?A1 NUM PR ST
       (PLM COLOR SUBCL
            (? IN PR ST ***
               ((ME) 'S BOX OF PLM PAINT)))
       (UNM)))
 (SD (JACK))
 ("I HAVE MORE COLORS IN MY PAINT BOX " SAID JACK))

(((? WANT PR ST NOT (I) ?A1)
  (?A1 HAVE FUT ST NOT (* I) (THESE PLM PENCIL)))
 ("I SD (JANET) TO (* JANET))
 (JANET SAID TO HERSELF "I DO NOT WANT THESE PENCILS))

(((? CAN PR QU (I) (WHAT))
  (? RESULT ?A1 (* WHAT))
  (?A1 GET FUT ST (* I) (THOSE PLM PAINT)))
 ("2)
 (WHAT CAN I DO TO GET THOSE PAINTS?")))
```

```
((((? SOON PR ST ?A1)
    (?A1 GO PR ST (JANET) BACK)
    (? CONCURRENT ?A2 ?A1)
    (?A2 WITH PR ST (* JANET) (SOMETHING))
    (? GOAL ?A3 ?A2)
    (?A3 GIVE FUT ST (* JANET) (JACK) (* SOMETHING)))
   (SOON JANET CAME BACK WITH SOMETHING FOR JACK))

  (((? HOLD PR ST (* SHE)
       ((NIL NIL NIL)
        (? IS PR ST ?CUR TOY)
        (? SHAPE PR ST ?CUR CAT)
        (? CAN FUT ST (* JACK) ?A1)
        (?A1 PUT FUT ST (* JACK) ?A2)
        (?A2 IN FUT ST ((YOU) 'S MONEY) ?CUR))))
   (SD (SHE) TO (JACK))
   ("JACK HERE IS A TOY CAT TO PUT YOUR MONEY INTO " SHE
    SAID))

  (((? TRADE FUT ST (I) (YOU)
       (DGROUP (THIS CAT) (ALL (ME) 'S PLM PENCIL))
       ((YOU) 'S PLM PAINT)))
   ("2)
   ("I WILL GIVE YOU THIS CAT AND ALL MY PENCILS FOR YOUR
    PAINTS"))

  (((? HAVE PR ST NOT (* JACK) (MONEY)))
   (SD (JACK))
   ("I HAVE NO MONEY " SAID JACK))

  (((? WANT PR ST NOT (I) ?A1)
    (?A1 HAVE FUT ST NOT (* JACK) ((NIL NIL NIL)
                                   (? IS PR ST ?CUR TOY)
                                   (? SHAPE PR ST ?CUR
                                                  CAT))))
   ("2)
   ("WHAT DO I WANT WITH A TOY CAT?))

  (((? GIVE-UP FUT ST NOT (JANET)))
   (STILL JANET DID NOT GIVE UP))

  (((? MUST PR ST (I) ?A1)
    (?A1 GET FUT ST (* I) (SOME MONEY))
    (? GOAL ?A2 ?A1)
    (?A2 GIVE FUT ST (* I) (JACK) (* SOME MONEY)))
   (SD (SHE) TO (* SHE))
   ("I 'LL HAVE TO GET SOME MONEY FOR JACK " SHE SAID TO
    HERSELF)))
```

```
((((? SOON PR ST ?A1)
    (?A1 GO PR ST (SHE) BACK)
    (? CONCURRENT ?A2 ?A1)
    (?A2 WITH PR ST (* SHE) (IT)))
   (SOON SHE CAME BACK WITH IT))

  (((? HOLD PR ST (* SHE) (SOME MONEY))
    (? CAN PR ST (* JACK) ?A1)
    (?A1 PUT FUT ST (* JACK) ?A2)
    (?A2 IN FUT ST (* SOME MONEY)
          ((NIL T NIL) (? IS PR ST ?CUR TOY)
                       (? SHAPE PR ST ?CUR CAT))))
   (SD (SHE) TO (JACK))
   ("JACK HERE IS SOME MONEY TO PUT INTO THE TOY CAT " SHE
    SAID))

  (((? HAVE FUT ST (YOU) (DGROUP (SOME MONEY)
                                 ((NIL NIL NIL)
                                  (? IS PR ST ?CUR TOY)
                                  (? SHAPE PR ST ?CUR CAT))
                                 (ALL THE PLM PENCIL))))
   ("1)
   ("NOW YOU WILL HAVE SOME MONEY A TOY CAT AND ALL THESE
    PENCILS))

  (((? SEE PR IMP (YOU) ?A1)
    (?A1 NUM PR ST
          (THE PLM THING SUBCL
               (?A2 TRADE FUT ST (UNM) (* YOU) ***
                                 ((YOU) 'S PLM PAINT)))
          MANY))
   ("2)
   (JUST SEE HOW MANY THINGS YOU WILL GET FOR YOUR
    PAINTS1"))

  (((? LAUGH PR ST (JACK)))
   (JACK LAUGHED AND SAID "TAKE YOUR THINGS AND GO AWAY))

  (((? TAKE PR IMP (YOU) ((YOU) 'S PLM THING))
    (? GO PR IMP (* YOU) AWAY))
   ("I SD (JACK)))

  (((? WANT PR ST (I) ?A1)
    (?A1 PAINT FUT ST (* I) (PLM PICTURE)))
   ("2)
   (I WANT TO PAINT PICTURES")))
```

APPENDIX B - Special Words and Abbreviations

This gives pointers to the definitions of various terms used in the thesis. For the moment the pointers are to the sections numbers, in the final version the page numbers will be given. When more than one number is given we will give the first occurance of the term followed by the location of the best discription of the term.