MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

Artificial Intelligence
Memo #265

August 1972
Reprinted January 1974

INFANTS IN CHILDREN STORIES - TOWARD A MODEL OF NATURAL LANGUAGE COMPREHENSION

Garry S. Meyer

ABSTRACT

How can we construct a program that will understand stories that children would understand? By understand we mean the ability to answer questions about that story. We are interested here with understanding natural language in a very broad area. In particular how does one understand stories about infants? We propose a system which answers such questions by relating the story to background real world knowledge. We make use of the general model proposed by Eugene Charniak in his Ph.D. thesis (Charniak 72). The model sets up expectations which can be used to help answer questions about the story. There is a set of routines called BASE-routines that correspond to our "real world knowledge" and routines that are "put-in" which are called DEMONs that correspond to contextual information. Context can help to assign a particular meaning to an ambiguous word, or pronoun.

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.

INFANTS IN CHILDREN STORIES - TOWARD A MODEL OF NATURAL LANGUAGE COMPREHENSION*

Abstract

How can we construct a program that will understand stories that children would understand? By understand we mean the ability to answer questions about that story. We are interested here with understanding natural language in a very broad area. In particular how does one understand stories about infants? We propose a system which answers such questions by relating the story to background real world knowledge. We make use of the general model proposed by Eugene Charniak in his Ph.D. thesis (Charniak 72). The model sets up expectations which can be used to help answer questions about the story. There is a set of routines called BASE-routines that correspond to our "real world knowledge" and routines that are "put-in" which are called DEMONs that correspond to contextual information. Context can help to assign a particular meaning to an ambiguous word, or pronoun.

The problem of formalizing our real world knowledge to fit into the model is the prime problem here. I discuss a first level attack on formalizing the information about infants and then baby bottles. The contrast between the two leads me to suggest that the same methods can not be used successfully for both inanimate objects and animate objects. Finally I outline how a finite state model of infant behavior can be used to better understand infants in children stories.

Thesis Supervisor: Marvin L. Minsky, Professor of Electrical Engineering

---

*This memo reproduces a thesis of the same title submitted to the Department of Electrical Engineering, Massachusetts Institute of Technology in August, 1972, in partial fulfillment of the requirements for the Degree of Master of Science.

Acknowledgement

The author would like to thank Professor Marvin L. Minsky for supervising this thesis, and for the help he provided in the form of many useful discussions. I would also like to express my sincere gratitude to the many people who provided encouragement and aided me in this work: to the members of the AI Project at M.I.T.; in particular to Eugene Charniak, Jerome Lerman, Ira Goldstein, R. Bruce Roberts, Drew McDermott, Professor Seymour Papert, and Terry Winograd.

I would also like to thank publicly my wife, Ellen, who contributed several ideas for the sections on infants, as well as for providing general support and encouragement.

Table of Contents

Introduction:

We are interested in creating a model of infant behavior, just complex enough to allow us to understand references to infants in children stories. There are two principal reasons why we believe it is useful to assemble such a model or "micro-world" of knowledge.

1. They are needed to extend "language understanding" programs like Winograd's.

2. They may serve as a starting point toward proposing how human knowledge about such subjects is organized, particularly in children.

Our feeling is that for this purpose infants (and indeed most animals) can be considered as "many animals in one". That is, we can use a finite state model in which at one time he is a sleeper; in another state a drinker; another an eater, etc. In the section on the relation of this thesis to other work we will discuss briefly some of the origins for such a model. In the following pages we shall discuss a general model for children story comprehension and a first attempt at formalizing infants and the related topic of "Baby Bottle." We will then indicate the direction in which our finite state model will do.

The purpose of this paper is to discuss my efforts to formalize some knowledge about a particular subject. I will use the framework of Eugene Charniak's model of children story comprehension (Charniak 72). In doing so I will also note certain extensions which in the very least will make it easier to build a children story comprehender, and which may in fact be necessary if we want to comprehend general discourse. Initially I chose to attack the problem of understanding infants in children stories. Later I decided to work on Baby Bottle for reasons which will become obvious to the reader. Before describing my work I will briefly describe the Charniak model as I understand it.

<u>Relation to Other Work</u>:

The impetus for this work stems mainly from the work of Eugene Charniak. In his thesis (Charniak 72) he describes a model directed toward understanding natural language in the children story context. Aside from this work, very little work has been done in creating computer systems aimed at understanding stories. Most language understanding systems, SIR (Raphael 64), QA3 (Green 69), and SHRDLU (Winograd 71) do not deal with very broad universes. Winograd's is limited to the world of toy blocks. None of them understand very much about things like human motivation or human desires.

I will not attempt to describe in detail or evaluate much of the work in Artificial Intelligence relating to language understanding. For those who would like to see such a discussion I can recommend Winograd's paper (Winograd 72) which contains a very good evaluation of research in A.I. on natural language processing, semantics, and theorem proving.

Since our attack on natural language understanding is founded on the idea of working on children's stories first, we are concerned about the knowledge that children have that allows them to understand simple stories. There is a vast literature on human infancy (Kessen 70) but almost nothing on how children think about infants, or what facts they know. Piaget discusses what he believes children know about the birth of an infant (Piaget 69). But no one seems to get at what the models are that children use for understanding infant behavior.

Several people have made or used a finite state model to explain animal behavior (McCulloch, Kilmer, Blum). In their work they postulate an integrated, stable physiological pattern for each mode of behavior, where each mode generally subsumes many particular kinds of behavior. Their evidence for this model derives mainly from neuroanatomy, phylogenetic arguments, experiments with the reticular formation (severe damage, electrical stimulation, etc.) ethological and physiological arguments. The more recent work of Kilmer and McLardy deals with how different acts are determined within the modes. They define "acts" as "well-motivated, species-typical, intramodal behaviors that are based on an animal's instincts but which are usually decided on partly as a function of reinforced past individual experience" (Kilmer 72). Their hypothesis is that one of the main functions of a sector of the hippocampus in mammals "is to help select act decisions and activate them via signals over the precommissural fornix into the motor circuits of the medical forebrain bundle." The model is aimed at determining how a mode and in turn a particular act are instituted. They do not seem to be especially concerned with what the actual modes and acts are, except in so far as the total number must be reasonable for their model.

Piaget and Infants in Children Stories:

Piaget is a rich source of information on babies and the development of intelligence in the child. In his model children go through four stages of development. The first two of these are relevant here. They are: 1. sensori-motor (ages 0 to 2), and 2. pre-operational (ages 2 to 7). These age brackets are not fundamental to his theory but simply approximate cuts. In the first stage the child learns to coordinate his actions with what he perceives or with other actions and to use certain elementary schemata (sucking, following objects with his eyes, shaking). In the second stage the child learns to represent the world through symbols and signs but is not at the level of concrete operations (idea of conservation of number, matter, weight, etc.).

Piaget's work brings out the fact that the behavior of babies change with time. The implication of some of his work is that a knowledge of the history ("life-history") of a child is necessary to understand the child at any particular time. If this is true then it is hard to see how anyone could understand a child. The thing to note, I think, is that the concept of a "life-history" is important. This fact together with the fact that the state (determined by very recent events) of the child is ever changing implies that the problem of understanding discourse about infants or with infants in it can only be solved in a system which can handle these changes and understand "where the baby is at." We are forced, in a story understanding program, to maintain a history (recent events and not "life-history") because one can always ask questions of it like "How did the infant get into the present state?" or "What happened to the child before some particular act occurred?" So although what we mean by "state" is a finite summary of the history we are also forced to maintain the history so that we can answer questions about it. It seems clear to me that in the ultimate understander states and histories will play an important role. We could add a tag on every assertion that indicates the state. In some sense however, the set of currently active (ASSERTed) DEMONs is the state in the Charniak model. Also much of the history type of information can be gotten from the assertion numbers which are ordered in terms of input to the understander. The Charniak model does not have state markers, but they are a possible extension. This may be very important if we can show that we need alternative worlds existing in parallel. We could then use such state markers together with a filtering mechanism to distinguish various co-existing worlds. The notion of a time line, if properly developed, may prove sufficient to handle many problems, like tenses, hunger, saving, etc.

PLANNER

This chapter is intended to provide enough information about the language PLANNER (Hewitt 69, 72) so that the codes used later will be understandable. A knowledge of LISP (McCarthy 62, Weissman 67, Winograd 72) will prove helpful.

For a better (more complete) description of MICRO-PLANNER, the currently working subset of PLANNER, you can see the MICRO-PLANNER reference manual (Sussman et al. 72). PLANNER is a procedural language oriented toward the accomplishment of goals which may in turn be broken down into subgoals. In contrast to other languages, when a goal is activated in PLANNER it can be satisfied by any one of a number of assertions in the data base or by any number of theorems. Theorems can be activated or referenced by pattern and need not be called by name. So you do not have to explicitly call some procedure, but you can simply allow the system to accomplish the desired goal in what ever way it can. If a failure occurs as a result of some decision, a backup facility is provided so that another possibility can be tried.

(We have simplified some notations. All MICRO-PLANNER functions start with "TH" but we will delete the "TH" in our presentation so that our MICRO-PLANNER codes will be more readable. Because of this decision there may be some question as to whether a particular function is a LISP function or a PLANNER function; we assume that the function is a PLANNER one unless we specifically noted otherwise.)

Perhaps the easiest way to understand how PLANNER works is to look at some examples in the form of a console session. The ">" (greater than sign) indicates a response to some line that we typed in. Probably the most basic function is ASSERT. If we say

```
(ASSERT  (MARRIED JOHN MARY))
> ((MARRIED JOHN MARY))
```

The system has put the item (MARRIED JOHN MARY) into the data base. This item is called an assertion.

The GOAL function can be used to see if an assertion exists or can be deduced.

```
(GOAL  (MARRIED JOHN MARY))
> ((MARRIED JOHN MARY))
```

In this case the system has found the assertion in the data base. Later we will see how the system can deduce facts. If we asked it whether someone else, say Bill was married to MARY?

```
(GOAL (MARRIED BILL MARY))
> NIL
```

The response NIL means "false", meaning that it could not find that the assertion (MARRIED BILL MARY) was true.

The language has no knowledge of what the symbols MARRIED, BILL, MARY, etc. really mean. This is clear if we say

```
(GOAL  (MARRIED MARY JOHN))
> NIL
```

We know that if MARY is married to JOHN, then JOHN is married to MARY. This fact can be expressed in a PLANNER theorem. The CONSE type theorem is used for establishing goals. CONSE stands for consequent theorem. We could write

```
(CONSE MARRIAGE1 ()
                (MARRIED MARY JOHN)
(GOAL (MARRIED JOHN MARY))
> MARRIAGE1 (defined and asserted)
> (MARRIAGE1)

(GOAL (MARRIED MARY JOHN) $T)
> (MARRIED MARY JOHN)
```

So now we have a theorem that states that if we want to prove (MARRIED MARY JOHN), try (MARRIED JOHN MARY). Note that we have added a second argument to the GOAL. TBF means filter so that only those theorems that satisfy the criteria are tried. TRUE means always true, so we will try every theorem that matches the pattern. We could alternately have had any LISP function of one argument in the place of TRUE. Since (TBF TRUE) is used so often, it is abbreviated $T.

Now lets say we asserted

```
        (ASSERT  (MARRIED GARRY ELLEN))
        > ((MARRIED GARRY ELLEN))
```

and we did a

```
        (GOAL  (MARRIED ELLEN GARRY))
        > NIL
```

This is because our marriage theorem is only invoked when we want to prove (MARRIED MARY JOHN), we can make the theorem general by introducing variables. A variable is proceeded by a "$?" to distinquish it from literals. In our exposition we will abbreviate this convention by removing the "$" and using the "?" only. When we write theorems that contain variables they must be bound, so we must declare the variables we are going to use. This is the purpose of the list following the theorem name.

```
        (CONSE MARRIAGE2  (X Y)
              (MARRIED ?X ?Y)
          (GOAL  (MARRIED ?Y ?X))
                                             )
        > (MARRIAGE2 defined and asserted)
        > (MARRIAGE2)

        (GOAL  (MARRIED ELLEN GARRY)
                                (TBF TRUE))
        > ((MARRIED ELLEN GARRY))
```

The system first checks the data base and finds that there is no assertion (MARRIED ELLEN GARRY). It next looks for theorems which match the pattern (MARRIED ELLEN GARRY) and finds that the pattern of MARRIAGE2 matches with ?X being set to ELLEN and ?Y set to GARRY. The goal (MARRIED ?Y ?X) is then executed as (GOAL (MARRIED GARRY ELLEN)) which succeeds because it is in the data base.

If there were many theorems that had the pattern (MARRIED ?X ?Y) then the system would keep trying them until one succeeded. We can also tell the system to try a particular theorem or several particular theorems.

```
(GOAL  (MARRIED ELLEN GARRY )
                    (USE MARRIAGE2))
> (MARRIED ELLEN GARRY)
```

Now that we have seen how to put assertion in the data base and how to define theorems, we show how to get rid of them.

```
(ERASE  (MARRIED GARRY ELLEN))
> ((MARRIED GARRY ELLEN))

(GOAL  (MARRIED GARRY ELLEN))
> NIL
```

However if we did:

```
(ERASE MARRIAGE1)
> (MARRIAGE1 ERASED)

(ERASE MARRIAGE2)
> (MARRIAGE2 ERASED)

(GOAL  (MARRIED MARY JOHN)$T)
> NIL
```

```
        (GOAL  (MARRIED MARY JOHN)
                        (USE MARRIAGE2))
        > (MARRIED MARY JOHN)
```

This is true because when we erase (un-assert) a theorem, what we really mean is removal of that theorem's pattern from the list of patterns available to be matched. However, the theorem still exists and can be called by name. So by define we mean storing the theorem by name, and by assert we mean storing by pattern.

So far only relevant items appeared in the data base so that the right assertions were found at once. Let us add

```
        (ASSERT  (WASP  ELLEN))
        > ((WASP  ELLEN))

        (ASSERT  (MARRIED GARRY ELLEN))
        > ((MARRIED GARRY ELLEN))

        (ASSERT MARRIAGE2)
        > (MARRIAGE2 ASSERTED)
```

Our data base now contains the following:

```
        (MARRIED JOHN MARY)
        (MARRIED GARRY ELLEN)
        (WASP ELLEN)
        Theorem MARRIAGE2
```

Suppose we wanted to find a happy person and had defined this to be someone who was married (this is purely hypothetical). We could write the following theorem:

```
(CONSE  HAPPY?  (X Y)
      (HAPPY ?X)
   (GOAL  (MARRIED ?X ?Y) $T)
                         )

> (HAPPY? defined and asserted)
> (HAPPY?)
```

now if we did:

```
(GOAL  (HAPPY MARY)$T)
> (HAPPY MARY)
```

However, if we wanted a happy WASP we might write

```
(PROG  (X)
   (GOAL  (HAPPY ?X)$T)
   (GOAL  (WASP ?X) )      )
> (WASP EILEN)
```

The PROG acts like an AND function so that the procedure succeeds only if every term is satisfied.

Now what happens? The system first searches its data base for someone who is happy. Let's say, for instance, that it finds JOHN before EILEN. The goal (MARRIED ?X ?Y) will succeed assigning to ?X JOHN, now the (HAPPY ?X) succeeds, and the PROG proceeds to the new goal of (WASP JOHN) which fails because there is no such assertion in the data base or applicable theorem. On failure the system backs up to the last decision point, which was the choice of JOHN for binding to ?X in the married assertion. The system then tries another item that will match. This is like going down another branch in a tree. At some point EILEN will be picked up by using the MARRIAGE2 theorem. In this case by using the theorem MARRIAGE2 to deduce that (MARRIED EILEN GARRY) and then the theorem HAPPY? to deduce (HAPPY EILEEN).

So ?X is bound to EILEEN and the goal (HAPPY EILEEN) succeeds. Now the goal (WASP EILEEN) is tried and succeeds since it is in the data base. Our PROG succeeds returning the value of the last GOAL, namely (WASP EILEEN).

Note that if the data base contains very many assertions, the time to search for one when we do a GOAL may grow, and might become too great. More seriously, we might need to do some costly computations in order to deduce (generate) certain assertions and since we may not really need any or all of them it makes sense to wait and see if we really need them. There is also the possibility that we are working with infinite or nearly infinite sets and would not want to waste time and storage by entering any assertions before we need them. We would therefore not want to assert facts that are not very important or are not used frequently or are too many or very costly to deduce. In other words, we can make a decision as to whether we want to deduce a particular assertion every time we need it, or deduce it the first time we need it and then put it in the data base, or have it in the data base when we begin. Such decisions may depend on the problem area and the task of the program. In my work I might assert facts like (IS BILLY HUNGRY), but would not put in facts like (QUANTITY-OF-MONEY PENNY), (QUANTITY-OF-MONEY DIME), etc. We have seen an example where we avoided asserting many items by using a consequent theorem. The consequent theorem HAPPY? above, was used to avoid asserting that every married person is happy.

We have seen how the CONSE theorem is invoked when we are trying to do a GOAL, and the assertion isn't in the data base. Now lets see what can happen when we make an assertion. That is we can decide that we want to add one or more assertions to the data base because a particular assertion is made. We use antecedent theorems to do this. It has a pattern like the consequent theorem and is invoked when an assertion is made that matches this pattern. Let us write a theorem which will assert that a person is unhappy if he is hungry.

```
(ANTE RESULT-OF-HUNGER
            (X)
      (IS ?X HUNGRY)
   (ASSERT  (IS ?X  UNHAPPY))
> (RESULT-OF-HUNGRY defined and asserted)
> (RESULT-OF-HUNGRY)
```

Now if we input

```
(ASSERT  (IS BILLY HUNGRY)$T)
>((IS BILLY HUNGRY))
```

We will find (IS BILLY UNHAPPY) is in our data base.

```
(GOAL (IS BILLY UNHAPPY))
> ((IS BILLY UNHAPPY))
```

We will see later that our "DEMONs" are antecedent theorems because they are procedures which are run as a result of some new assertion.

Finally in this short description we note the last type of theorem, called ERASING. It is invoked as a result of the erasing of an item whose pattern matches that of the theorem.

```
(ERASING UNMARRIED  (X  Y)
              (MARRIED  ?X ?Y)
         (ASSERT  (UNHAPPY ?X))
         (ASSERT  (UNHAPPY ?Y)))
> (UNMARRIED defined and asserted)
> (UNMARRIED)

(ERASE  (MARRIED GARRY ELLEN)$T))
> ((UNHAPPY GARRY))
> ((UNHAPPY ELLEN))
```

What has happened is that the act of erasing the assertion (MARRIED GARRY ELLEN) has caused the theorem UNMARRIED to be run. When we entered that theorem ?X was bound to GARRY and ?Y to ELLEN, so that the two UNHAPPY assertions resulted as shown above.

The Charniak System

In (Charniak 72), E. Charniak outlines a proposed computer program that has the task of understanding childrens stories. Although by "understand" we mean having substantial ability to answer questions, this model does not wait until a question is asked but processes assertion by assertion. That is, as each new sentence is input to the system, it attempts to generate, and answer for itself, the kinds of questions most likely to be asked by a person checking to see if the story was understood. It attempts to answer such questions by relating the story to background knowledge of the real world. In many cases an event at an earlier part in the story can set up "expectations" that make it practical to understand later events without enormous logical (and not-very-logical) calculations. These expectations are implemented by invoking procedures (called THEOREMs in PLANNER, and DEMONs by Charniak) which look for what is likely to occur on the basis of what has occurred in the story to that point. A typical story fragment might be:

Daddy was going to the hospital.
Daddy said, "Mother is going to give birth soon
and we will have a new baby".

Typical questions one might ask are:

Why is Daddy going to the hospital?

Who is going to have a baby?

Is Mother likely to be at the hospital?

It is important to realize that the story does not explicitly answer these questions. The story does not say for instance "Daddy was going to the hospital because Mother was there having a baby". Even to answer very simple questions about fairly simple stories our model must know facts about the world. In the example above for instance, it must know:

Babies are usually born in a hospital.
One visits close relatives when they are in the hospital.
If a person is going to "give birth" then her family will "have" a new baby.
One will go to the hospital either to visit or to be a patient.

and so forth.

The model is claimed to be a system which contains such general information in a form which can be applied automatically to particular stories. One very important feature of this model is that it has the ability to make deductions based on assumptions and if these assumptions prove incorrect, it goes back and undoes the things that were done because of it. That is, if we are told, after reading the previous story (above), that "Mother was coming home tomorrow", we would assume that the baby was born and probably coming home too. We would then make inferences based on these assumptions. However, if we are told later in the story that the baby died we would have to go back and undo the work that we did subject to those assumptions.

Charniak puts forth the following question as a way to get at the focus of his thesis: "How does a sentence mean?" Given the focus determined by this question we now are in a position to examine Charniak's model. In the past it was customary to think about such problems as being roughly divided into three areas:

Syntax
Semantics
Inference

It must be noted that this is not a hard and fast distinction, but a first level divison of the problem in the very least; it is never clear where the boundary should be.

Focus of the Model

Charniak points out rightly, I think, that of the three, inference is the section which is "most in the dark." 'It seems clear to me that our initial problem, "construct a model which contains 'real world knowledge' " will fall under the heading of inference.' Furthermore, inference is necessary to resolve problems in areas other than "real world knowledge", i.e. resolving certain reference problems. In fact much of the discussion in his thesis is about problems of reference. The conclusion is still that we must be most concerned about the inference problem.

Internal Representation or What's Our Input

The easiest way to give an idea of how stories and knowledge are represented in the Charniak model might be to say that it is "half-way between English and Predicate Calculus." This is not very useful or accurate, but we do not want to describe the internal representation (hereafter called IR) used. We would, however, like to comment upon it. We assume that the input has already been processed into the IR. Although this assumption may be necessary at this time to limit the scope, it by no means is a "trivial" processing problem. There has been no study of how our IR compares with anything that a human might have or use, or for that matter how humans process language at all. One may ask to what extent do humans do it that way? There is no reason to think it close to the actual processing humans do; it is not an attempt at cognitive simulation. Humanoid or not, we presume that it is adequate. The really important things about an IR are that it is rich, reasonable, oriented for working within the system, and doesn't cheat by doing all of the interesting work. It allows us to express fuzzy relationships and yet limits the number of ways a fact can be expressed to a reasonable number. IR is well suited for a PLANNER based system and is reasonable.

The Model and PLANNER

The Charniak model is divided into parts: BASE-routines, DEMONS, BOOK-KEEPING-routines, and FACT-FINDERS. The BASE-routines are PLANNER ANTECEDENT THEOREMs. They represent the world knowledge needed to understand a story independent of context, that is, they are always there (asserted). DEMONS are also ANTECEDENT THEOREMs but they represent knowledge gained from context. They may be "put-in" (asserted) by the BASE-routines or currently active DEMONs as a result of context. FACT-FINDERs are invoked when we want to prove an assertion that matches its pattern. Basically they are used to establish facts which are comparatively unimportant. Without having FACT-FINDERs the number of assertions in the data base would be very great because we would explicitly have to assert many things (i.e. milk is food, juice is food, etc. and penny is money, nickel is money, etc.) The BOOK-KEEPING part does what its name implies. They must figure out updates, sometimes with the help of other sections. Charniak shows that the ordering of these parts is important. The flow of control is:

1) Apply syntax and semantics to get sentence into assertion format
2) Place assertions on the TO-BE-DONE list
3) Apply DEMONs and BASE-ROUTINEs in that order to each entry, if a new assertion is generated place it on the TO-BE-DONE list
4) Go through TO-BE-DONE list and apply strong occurrence rule and BOOKKEEPING
5) Apply second part of BASE.

Although the Charniak model uses many features of PLANNER and much of the knowledge we try to formalize appears as MICRO-PLANNER like theorems, it does not adhere strictly to the language as described in Sussman, et. al. (1971). The fact of the matter is that as written here, and in the Charniak thesis, most of the theorems would not work properly until programming changes are made in MICRO-PLANNER. Such modifications, though time consuming, will not be very difficult. While we do not want to be excessively and unnecessarily pedantic, we should, however, note some of the problem areas. One illustration is the way in which DEMONs are deactivated (erased). The statement (DESTRUCT? ?HOLD) is supposed to deactivate the DEMON in which it occurrs if the assertion number which is pointed at by HOLD has been updated (is no longer true or has been erased). This raises problems with variable binding and the failure mechanisms that PLANNER uses. It must also be pointed out here that I use several predicates, as does Charniak, that are not part of PLANNER but which must be there for our theorems to work. ASSERT and PUT-IN are two of them. ASSERT for instance, must 1) create the assertion number, 2) put a NEW-A tag on the assertion, 3) place the assertion on the TO-BE-DONE list, and 4) issue an ASSERT.

A First Look at Infants

The initial task I attempted was to write DEMONs for understanding infants in the context of the Charniak model. The first thing I did was to look at several stories that were either about infants or had them in them. From this experience, plus the knowledge gained by talking with three children, ages 2, 6, and 8, and with my wife's help, I produced the following summary of the knowledge that I think children have about infants:

> Infants are happy to sleep most of the time, which they do either in a crib, bassinet, or any comfortable and "safe" place (Mother's arms). When they are not sleeping they are either playing, eating, or relieving their bodily functions (described by children with euphemisms like: poo-poo, bunny, wee-wee, etc.). Infants usually wear diapers because they don't control their body movements very well. Dad or Mother must change the diaper after this happens. Excrement causes irritation which leads to pain and will thus cause crying. Babies also have all of the common characteristics we attribute to most humans (i.e. 2 arms, 2 legs, 2 eyes, 10 fingers, etc.), except they are proportionally smaller. Their size being bigger than a Tiny Tears doll (or any small doll for that matter) and smaller than a large stuffed animal. Infants are "new" (although not necessarily improved) so they can not do many of the things that older humans do. That is, they don't know how to walk, talk, play with most games, or dress, wash, or feed themselves. Infants cry when hungry. Crying has been shown to be inhibited both by feeding and by nonnutritive

sucking. You can reduce crying by holding a baby or by supplying a continuous auditory stimulation (singing a lullaby). Young infants are burped after feeding, this is to bring up trapped air that they may take in while feeding. To burp, hold the baby over your shoulder and pat it on the back gently. They are fed liquid food (usually milk or a formula) in a bottle or from a mother's breast. When they get a little older they move up to soft foods called "baby food." When little, they eat frequently and at regular intervals. Milk or formula must be warmed to take the chill off. When the baby starts cutting teeth, called teething, the baby gets cranky and cries easily due to its gums being tender and sore. Baby is washed by mother in a layette or small tub possibly with the help of older children or father. Infants are too little to splash and play with toys when they bathe as older children often do. Infants' skin are very tender so they are oiled and powdered. This is done after a bath and is accomplished by sprinkling a little on and rubbing it in "very" gently. They feel small and soft, and they are therefore "nice" to hold, but they wiggle a lot so you must be very careful or they may fall. Dropping a baby is considered "bad form" and can be very serious, as are other actions that cause harm. Pinching, hitting, pushing, and kicking are some of the ways in which you can hurt a baby. If you intentionally hurt a baby then that is reason for your being reprimanded. There are other ways of causing a baby to cry, like making a loud noise, taking away something it thinks it owns, or frightening the baby by holding it the wrong way. Very young infants grab onto things like your finger or eyeglasses and smile at you. They make sounds when happy, like goo-goo. They are also highly susceptible to diseases so you must stay away from them when you are sick. Babies need a lot of attention from parents, causing other older children to feel "out of it" and thus to become jealous. This jealous reaction is also caused by the fact that baby is not responsible for its actions and therefore is

> not punished for the same things that an older child might be punished for. Babies don't always know what is bad for them so you must watch them. This appears to the older child to be "unfair" treatment. Another cause for hostility is the fact that people bring presents to the new baby and not to the older children. Having a baby is a "Joyous Event" and is usually celebrated. Mother has to go to the hospital to have the baby. Pregnancy precedes giving birth.

The amount of information here is considerable larger and more complex than the knowledge about objects like piggy bank or baby bottle. Much of the information is of the form of "babies can not _____ ". This implies that children have a good idea of what they themselves can do. Many of the stories are based on "can not do" sort of facts and many are centered around the ways in which a baby acts differently from "us" (children). We can not use the information in the form that it is above. It is not clear at all how we would use a fact like "Babies feel small and soft." I have only been able to formalize a small subset of the facts above. The task of formalizing all of it in the context of Charniak's model may in fact be very difficult. I will present what little I have done here and then move on to baby bottle.

Baby DEMONs:

Suppose we are given:

(1.1) Baby was hungry. Mother gave him his bottle.

At this point one might ask what Mother was doing to the baby when she gave him his bottle. This suggests that when we enter our BABY base routine we want to check if the baby needs food or is hungry and assert the intention is to get food. Alternatively we could have had:

(1.2) Mother gave the baby his bottle. The baby wants milk.

In (1.2) we don't find the need until after the give baby statement. This implies that we want a DEMON looking for "give food".

The demon should look something like:

```
(ANTE BABY-GIVE-FOOD
     (NOLD BABY TN PER1 FOOD)
          ; ?NOLD is specified to the assertion number of
          ; the baby hungry assertion, ?BABY is specified
          ; as the person given food,
          ; ?PER1 is the person giving,
          ; TN is the tense, and FOOD is either food or bottle
     (?N GIVE ?TN ST ?PER1 ?BABY ?FOOD)
          ;this is the pattern we match
     (DESTRUCT? ?NOLD)
(GOAL (IS PR ST ?BABY BABY))
     ;make sure that ?BABY is indeed a baby
(OR
     (GOAL (IS ? ST ?FOOD BABY-FOOD))
     (EQUAL ?FOOD 'BOTTLE) )
     ;food must be either baby food or a bottle
(ASSERT (? HAVE PR ST ?BABY HB))
     ;after writing this DEMON and DEMONs
     ;for HB, I found that the
     ;"have HB" assertion above was needed
(ASSERT (?M1 FEED ?TH ST ?PER1 ?BABY))
(ASSERT (? REASON ?M1 ?HOLD))
(ASSERT (? DONE-WITH ?M1 ?FOOD))
     ;assert is similar to the PLANNER ASSERT
     ;here we are saying that PER1 feeds BABY with
     ;food FOOD (i.e. food or bottle) and that
     ;the reason for feeding is the hungry assertion.
                                   )
```

It should be noted, about the above DEMON, that there is some question as to whether we should have the FEED assertion in it, or use some FACT FINDER to deduce this information as needed. It seems that in (1.1) the author of the story is telling us that "Baby was hungry" caused "Mother to give him his bottle", whereas in the second case (1.2) he is saying that "Baby wants milk" is an "explanation" for "Mother gave the baby his bottle." This suggests that if we haven't seen a "cause" for some action, then we should look ahead a little for an "explanation."

Initially we suggested that BABY base look for a person needing food and then asserting statements about how being given a bottle or food fitted into the goal. In fact this is what the DEMON does. However it waits for the "give" statement. We can use the technique that Charniak calls

LOOK-BACK to handle the two cases, namely we can first make a GOAL out of the DEMONs pattern and do a GOAL on it. If there is a match in the data base, then this DEMON is applied to the assertion.

Let us write another DEMON, this time related to the fact that a baby needs to be watched.

(1.3) "I'm all alone at home," Jack thought.

"I have to watch Sally."

We could ask several questions like:

(1.4) Can Jack go out?
(1.5) Why must Jack watch Sally?
(1.6a) Where is Sally?
(1.6b) Is Jack really alone?

The first attempt looked something like this

```
(ANTE  BABY-NEEDS-WATCHING
     (HOLD PER BABY N N1)
          ;?HOLD is specified to the assertion number of
          ;the "only one at home" assertion, ?PER is
          ;specified to the person who is home, and ?BABY
          ;to the baby
     (?N MUST PR ST WATCH ?PER ?BABY)
          ;this is the pattern we match
      (DESTRUCT? ?HOLD)
     (GOAL (? IS PR ST ?BABY BABY))
          ;make sure variable ?BABY is indeed an infant
     (IF-NEED (?N1 NEED-TO-BE PR ST WATCHED INFANTS)
          ;if it has not already been asserted
          ;then assert that
          ;an infant must be watched
     (ASSERT (? REASON ?N ?N1))
     (ASSERT (? REASON ?HOLD ?N))
                                                 )
```

This DEMON says that if Sally is a baby in the above case, then she needs to be watched and that JACK must watch her. The system must be smart enough to know that Jack is not really alone, but means that he and the baby together are alone. It also asserts that he must watch her because he is the only one who can watch her at home (i.e. babies can't watch themselves). However there is a problem with this DEMON, if we were given

(1.7) Mom left me here with the baby.
(1.8) David came by and asked, "Will you come out and play?"
(1.9) I said, "I have to watch the baby."

We need some way to indicate that the question was answered in the negative and thus we now know that I am not going to play outside and the reason why I am not going is that "I must watch the baby". Also, we may not have an explicit question but an implied one, that is, we could replace (1.8) above by

(1.10) Jim and Mary came by and said,
"We are going to the beach."

To handle cases like this I first propose the predicate PREV-QU-IMPI which returns the assertion number of the previous question or implied question. We then ask whether the negative is implied by the "I must watch the baby" assertion. Then if we succeed we can assert that the answer to the question is no and that the reason is something like statement (1.9) or in the case of an implied question we can assert that Jack will not do what is implied and that we know this because of (1.9). Before we look at our new DEMON I would propose another addition that allows us to match on an ORed pattern so we will match on any of (1.11) - (1.13). There is a strong case for representing "has to" and "must" by the same symbol in our IR because they may be sufficiently close in meaning. My personal feeling is that "must" is stronger than "has to", so I leave them as separate entities in the IR. A third alternative would be to represent them by the same symbol but also associate a tag indicating degree of need. In the case of "watch" and "stay with" it seems that we have good reasons why we would not want to represent them by the same symbol in our IR. For example (1.13.1) and (1.13.2) mean very different things. "Stay with" has the property that its meaning depends on the relationship (age or relative degree of responsibility) between the subject and object (see (1.12) - (1.13.4)).

(1.11) David has to watch Sally.
(1.12) David has to stay with Sally.
(1.13) David must stay with Sally.
(1.13.1) I had to watch Grandmother.
(1.13.2) I had to stay with Grandmother.

(1.13.3) Grandmother had to stay with me.
(1.13.4) I had to stay with my friend Jimmy until Mother came.

Now let's look at our DEMON

```
(ANTE  BABY-NEEDS-WATCHING2
     (NOLD PER BABY N N1 N2 N3 TYPE COV IMP)
          ;?NOLD is specified to the assertion number of
          ;the "only one at home" assertion, ?PER is
          ;specified to the person who is home, and ?BABY
          ;to the baby
     (?N ($R ?IMP (OR 'MUST  'HAS-TO))  PR ST
             ($R ?COV (OR 'WATCH 'STAY-WITH)) ?PER ?BABY)
          ;this is the pattern we match
     (DESTRUCT? ?NOLD)
     (GOAL (? IS PR ST ?BABY BABY))
          ;make sure variable ?BABY is indeed an infant
     (IF-NEED (?N1 NEED-TO-BE PR ST WATCHED INFANTS)
          ;if it has not already been
          ;asserted then assert that
          ;an infant must be watched
     (ASSERT (? REASON ?N ?N1))
     (ASSERT (? REASON ?NOLD ?N))
     (SETQ ?N2 (PREV-QU-IMPL ?N))
     (COND ((IMPLY ?N2 ?N)
             (COND ((EQUAL ?TYPE 'QU) (ASSERT (?N3
                                         (ANSWER ?N2 'NO))))
                       (T (ASSERT (?N3
                                     ?PER 'WILL 'NOT ?N2))))
            (ASSERT (REASON ?N3 ?N))  )
            (T (SUCCEED)))
                                                   )
```

Let us now look at a simple DEMON which is used generally, that is it will be asserted by many bases. In the baby base we want to handle the fact that Mothers have babies at a hospital. Any other reason for being at a hospital, we presume, would put-in such a DEMON. We must make the name unique on each "Put-in" so the assertion that triggered the base will be part of the name. The DEMON will look something like this

```
(ANTE AT-HOSP-HOLD
     (HOLD PER IN HOSP)
          ;?HOLD is specified to the have baby assertion
          ;in this case, ?PER is bound to the person having
          ;the baby, and ?HOSP is a hospital.
     (?N AT ?TN ST ?PER ?HOSP)
     (DESTRUCT? ?HOLD)
     (GOAL (? IS ?HOSP HOSPITAL))
          ;check to see if HOSP is indeed a hospital
     (ASSERT (? REASON ?HOLD ?N))
          ;assert that the reason for being at the hospital
          ;is having a baby (in this case).
                                                  )
```

We have made certain assumptions about the baby base which if untrue will cause problems. We may have the following case:

(1.14) We were having a baby.
(1.15) Dad went to the hospital.

The problem here is that a member of a family can "have a baby" meaning that the Mother in that family is having the baby. The baby base is going to have to handle this problem. In the above case we want to say that Dad is going to the hospital to visit Mother (a less reasonable reply would be that Mother is having a baby). I feel that we must be able to give both answers if the questioner presses. This brings up one of the problems I faced; very often I am not sure what the correct answer to a question is, and in fact feel that more than one is appropriate. Of course, at this time we would be very happy to have a system that could give any answer that was correct.

I would now like to discuss several FACT-FINDERs (hereafter simply FF) that might prove useful. In the first case the Baby base must have some way to tell whether a person has had a baby or more simply whether "they" have a baby. It might use a FF that would tell whether some word or group of words is a baby indicator. The program would look something like this:

```
(CONSE BABY-IND-FF
     (BABY)
     (?BABY BABY-INDICATOR)
          ;this is the pattern we match
     (COND  ((GOAL (? IS PR ST ?BABY BABY))
          ;check to see if we have an assertion in the
          ;data base that says that
          ;?BABY is a baby (i.e. "Jim is a baby")
              ((EQUAL ?BABY 'BABY))
              ((EQUAL ?BABY 'CHILD))
              ((EQUAL ?BABY 'BOY))
              ((EQUAL ?BABY 'GIRL))
              ((EQUAL ?BABY 'NEW-MEMBER))
              ((EQUAL ?BABY 'NEW-ADDITION))
              ((EQUAL ?BABY 'BLESSED-EVENT))
         ;in all of the above cases we would
         ;win if were looking
         ;for something like "Mother had ?BABY".
                                                )
                                                      )
```

This FF would allow the baby base to check whether someone had a baby by looking for (? HAVE ?TN ?type ?person ?BABY) as its pattern, and by then doing a (GOAL (?BABY BABY-INDICATOR)). Of course in reality it is not all that simple, the base must not allow "Mrs. Jones had a new member." or "I have a nice boy, he's 22 years old". It must also know that if "Grandmother has a new Grandchild", then "Mother has had a new baby". It would therefore be reasonable to have a IF that knew about family relationships (a mini-theory of relations). That is, a routine that would allow us to prove that "David has a new brother or sister", if all we know is that "David's Mother just had a baby". Such a IF will have to be written, I believe.

Let's look at another IF. Suppose we want to know whether some person has fed another person. We might write:

```
(CONSE PERSON-FEED-IF
     (PER1 PER2 TN FOOD)
     (?N FEED ?TN ST ?PER1 ?PER2)
          ;this is the pattern we match
     (GOAL (? GIVE ?TN ST ?PER1 PER2  ($R ?FOOD
                              (IS-OBJ (?TYPE FOOD) (SOME)))))
          ;we want to look for person1
          ;giving person2 some food
                                        )
```

The decision as to whether to have such a IF depends on whether one wants the FEED assertion (the information in the data base). Of course one might require PERSON-FEED-IF to require the victim to be hungry, but many stories don't usually bother to mention this. Perhaps the "time-line" will generate hunger after a while or better yet our DEMONs for "hunger" will take care of checking food consumption and activity to see if a person is hungry when it is asked.

There is a major problem with this DEMON, namely, we might have "Ellen was going to the circus and wanted to feed the elephants." / "Mother gave Ellen some peanuts." It is clear that in this case Mother is not feeding Ellen peanuts but giving them to her so she can feed the elephants. To take care of cases like this we can modify our FEED-IF. We will work on the assumption that things like "going to the circus to feed the elephants" will put in DEMONs that will assert that "Ellen was given the food (peanuts) as a result of wanting to feed the food to the animals. Our new IF now looks like this:

```
(CONSE PERSON-FEED-IF-2
     (PER1 PER2 TN FOOD N N1)
     (?N FEED ?TN ST ?PER1 ?PER2)
          ;this is the pattern we match
     (GOAL (?N1 GIVE ?TN ST ?PER1 ?PER2 ($R ?FOOD
                                (IS-OBJ (?TYPE FOOD) (SOME)))))
          ;we are looking for person1 giving
          ;person 2 some food
     (NOT (GOAL (? RESULT ?N1 ?)))
          ;we do not want to find that the food
          ;was given for another reason.
                                              )
```

This discussion of feeding leads to the last DEMON I wrote pertaining directly to babies. Our first DEMON said that if we know that a baby is hungry then look for someone giving him food and assert that the baby got the food because it was hungry. Now I would like to examine the case where "Baby is hungry" asserts the feed statement, puts-in (asserts) the BABY-GIVE-FOOD DEMON, and puts the baby into a negative state. We might then make this the pattern of proposed DEMONs like BABY-CRY, BABY-MAKES-NOISE, etc. I feel that creating this tag "neg-state" is useful because there are many cases where the reason for the baby's discomfort is unimportant and the baby's state is really the key. Now let us look at the BABY-HUNGRY-DEMON which we have written:

```
(ANTE BABY-HUNGRY
     (NOLD BABY TN FOOD N1 N2 (PER 'SOMEONE))
     (?N IS ?TN ST ?BABY HUNGRY)
     (DESTRUCT? ?NOLD)
     (ASSERT (?N1 IS ?TN ?BABY NEG-STATE))
          ;assert that the baby is in a negative state
     (ASSERT (? REASON ?N1 ?N))
          ;and assert that the reason is baby is hungry
     (COND ((EQUAL ?TN 'PAST)
                (ASSERT (?N2 FEED PAST ST ?PER ?BABY)))
             ((ASSERT (?N2 FEED FUT ST ?PER ?BABY)))
          ;someone will feed the baby
                                                  )
     (ASSERT (? SUB-ACT ?NOLD ?N2))
     (COND ((EQUAL ?TN 'FUT) (ASSERT (? RESULT ?N2 ?N)))
             ((ASSERT (? T-RESULT ?N ?N2)) )
     (PUT-IN (BABY-GIVE-FOOD ?N ?BABY))
                                                    )
```

Infants into Baby Bottles:

As I mentioned earlier, I was not very happy with my progress in dealing with infants. It is true that I have written some half dozen THEORIEs which capture some of the knowledge about infants in children stories, but I am pessimistic about the goal of understanding all or most of the knowledge listed at the beginning of this chapter, in the context of the model discussed so far. Although it is possible to write programs which incorporate more of this information, I do not feel that this would be a fruitful approach at this time. So being in the context of hungry babies I decided to work on baby bottles, feeling that in this simpler area I might gain a better understanding of the DEMON writing process.

Baby Bottle:

Again the first step was to write down a summary of the information that I think children know and probably use when they understand stories.

Description of Baby Bottles:

> Baby bottles (hereafter called BB) come in a standard size and shape. They are a liquid container with a nipple on the end. Infants drink food from BBs. This food may be milk, water, juice, soda, etc. In the case of milk (or formula) the BB must be warmed to take the chill off the milk. The temperature of the milk should be tested (usually a few drops from the BB on the wrist). Mother or Dad prepares the bottle for baby in most cases. When the baby is hungry it will often look for its BB or cry out until it gets its BB or some other food. To get food out of the bottle you need to place the nipple in your mouth and tilt the bottle up so that the liquid comes out. You may also suck the nipple to help get the food out. Squeezing the bottle also helps, especially if it is the soft plastic type. Generally the heavier the bottle the more food that it contains. Many bottles are see-through so the liquid-level is apparent by inspection. You can also shake bottles to tell how much liquid they contain. If you hear a sloshing sound that generally indicates that there is food inside, if you don't hear anything then it is probably empty. One assumes that after the food comes out it is ingested by the person holding it or by the baby if someone is holding both a baby and the bottle. To put food in you need both the liquid and the bottle. The nipple part screws off the top of the BB so that it can be filled by pouring

in liquid. BB are also considered toys and can be owned by a baby. This ownership extends to the liquid contents of the BB. It is considered bad form for an older child to take and/or hide a baby's BB.

Baby Bottle DEMONs:

Here we have much of the knowledge about BB that I think is necessary. In this case we can note that, although not very small, the description is not all that large. It is most certainly less complex than infant was. The important thing about BB is that they are a physical object with a single primary utility, that is, they are used to feed babies. Although they may be used as toys or pacifiers this is of secondary importance. Our first DEMON will try to capture the fact that BB contain food that infants try to get out. Suppose we have the following story fragment:

(2.1) David sucked on his bottle.
(2.2) Finally he began to get milk.

How can we answer questions like:

(2.3) What is in the bottle before David sucked?
(2.4) How did David get the milk?
(2.5) Where was the milk before David began to get it?
(2.6) Is the milk still in the bottle?

The story fragment above does not explicitly answer any of these questions. For instance it does not say, "David sucked milk from his bottle." What we need then is a DEMON that can infer this information. The DEMON that would help us answer these questions looks something like:

```
(ANTE BB-HAVE-FOOD
     (HOLD BOTTLE PER N1 FOOD N)
     (?N HAVE PR ST ?PER ?FOOD)
          ;this is the pattern we match
     (DESTRUCT? ?HOLD)
     (GOAL (? IS PR ST ?FOOD LIQ-BABY-FOOD))
          ;make sure that FOOD is a liquid baby food
     (GOAL (? IS PR ST ?PER BABY))
          ;make sure that PER is a baby
     (IF-NEED (?N1 GET-FROM FUT ST ?PER ?BOTTLE ?FOOD))
          ;IF-NEED does an ASSERT if the statement
          ;is not found in the data base
     (IF-NEED (? IN PAST ST ?FOOD ?BOTTLE))
     (IF-NEED (? IN PR ST NOT ?FOOD ?BOTTLE))
     (ASSERT (? T-RESULT ?N ?N1))
          ;T-RESULT is a predicate that means
          ;that the second assertion (pointed
          ;at by the second argument) is a
          ;"trivial" result of the first assertion
          ;(pointed at by the first argument).
     (ASSERT (? SUB-ACT ?HOLD ?N1))
                                        )
```

This DEMON would have been put in by a HAVE-BB-BASE routine. This DEMON was aimed at understanding the following story fragments:

(2.7) David got his bottle.
He needed some milk.
David sucked on the BB.

(2.8) It was feeding time.
      Jimmy was given his bottle.
      He started to drink.

(2.9) Ellen was hungry.
      Mother gave Ellen her bottle.
      She got some milk.

The DEMON must assert the "get-from" intention in the above case but in the previous case (2.1 - 2.2) it will not have to do this but simple "put-in" the BB-HAVE-FOOD DEMON. This DEMON will also put-in several other DEMONs that are needed if the suck assertion results in the emptying of the BB. Our SUCK DEMON looks like this:

```
(ANTE BB-SUCK
     (BB PER N N1 N2 (FOOD 'SOME-BABY-FOOD))
     (?N SUCK PR ST ?PER ?BB)
          ;this is the pattern we match
     (IF-NEED (?N1 GET-FROM FUT ST ?PER ?BB ?FOOD))
     (IF-NEED (? HAVE PR ST ?PER ?BB))
          ;make sure that the baby has
          ;the baby bottle
     (ASSERT (?N2 SUB-ACT ?N ?N1))
          ;sucking is a sub-action of getting
     (ASSUME ?N1 ?N2)
     (PUT-IN   (BB-OUT-OF ?N ?PER ?BB)
               (BB-SEE-LEVEL ?N ?PER ?BB)
               (BB-NO-SEE-LEVEL ?N ?PER ?BB)
          ;put-in the necessary DEMONs.
                                                       )
```

Now let us look at a very simple DEMON which was put-in by the previous DEMON but which would in fact be put-in by any DEMON that might result in the BB being out-of food. It must look something like:

```
(ANTE BB-OUT-OF
     (NOLD PER BB FOOD N N1)
          ;in the above case NOLD would be the assertion
          ;number of the "suck BB" assertion
     (?N OUT-OF PR ST ?FOOD ?BB)
          ;this is the pattern we match
     (DESTRUCT? ?NOLD)
     (ASSERT (? RESULT ?N ?NOLD))
     (IF-NEED (? EAT PR ST ?PER ?FOOD))
                                                                )
```

One of the facts that we know about BB is that we can tell how much they contain by examination, that is we can see the level if we look. In this we can infer that there is something in the BB and from what this has resulted. The way the model indicates negation (with a NOT inserted in the assertion) makes it necessary for us to have a second DEMON for "no-see", although the information content should really be handled by one DEMON. Now lets look at our two DEMONs:

```
(ANTE BB-SEE-LEVEL
     (NOLD PER BB  N TN TG)
     (?N SEE ?TN ST ?TG ?PER)
          ;as in "She could see it was filled"
     (DESTRUCT? ?NOLD)
          ;this DEMON may have been put in by a "suck" type
          ;DEMON or by a "HAVE" BB routine
     (COND
        ((EQUAL ?TG LEVEL))
          ;as in "He saw it was filled to the 5 oz. level."
        ((GOAL (? IS ?TN ST ?TG LIQUID-FOOD)))
          ;as in "The milk could be seen."
        ((GOAL (? IS ?TN ST ?TN ?))
           (OR (GOAL (? IN ?TN ST ?TG ?BB) $S $T)
              (FAIL THEOREM)))
              (T))
          ;as in "It was seen."
```

```
    (COND ((GOAL (? IN PR ST ? ?BB) CS ST)
           (ASSERT (? IN PR ST ?TG ?BB)))
           (ASSERT (? RESULT ?M ?MOLD)))
                                                        )
```

and

```
(ANTE BB-NO-SEE-LEVEL
     (MOLD PER M M1 M2 TG )
     (?M SEE PR ST NOT ?PER ?TG)
     (DESTRUCT? ?MOLD)
     (COND ((EQUAL ?TG ?BB) (SETQ
                      ?TG (MAKE-OBJ 'SOMETHING))))
          ;the level in the BB could not be seen
        ((GOAL (? IS PR ST ?TG LIQUID-FOOD))))
          ;as in "David didn't see any milk
     (ASSERT (?M1 IN PR ST NOT ?TG ?BB))
     (ASSERT (?M2 EMPTY PR ST ?BB))
          ;now we know that the BB is empty
     (ASSERT (T-RESULT ?M2 ?M))
          ;we also know how we know that it is empty.
                                                      )
```

There are some facts which might more profitably be included in our knowledge of any container, such as, when someone puts something into the container, then there is that something in the container. In the BB case we might have something like:

(2.10) Mother got the BB. She put some milk in it.
or
(2.11) Mother gave Dad some milk. He put it in the bottle.

We need to assert that the milk is in the bottle in the examples above. We also need to assert the relationship between this act and the "get BB" assertion. At some later time we will have to consider whether we should have a PUT-IN-BASE routine that will generalize this idea. But getting back to the present case of BB we might write a DEMON which would look something like:

```
(ANTE BB-FOOD-IN
     (NOLD BB PER TN N N1)
     (?N PUT-IN ?TN ST (SR ?FOOD (IS-OBJ
                              (LIQ-BABY-FOOD) (SOME))) ?BB)
     (DESTRUCT? ?NOLD)
          ;?NOLD will point to the "get BB" assertion
     (ASSERT (?N1 IN PUT ST ?PER ?BB ?FOOD))
     (ASSERT (? SUB-ACT ?NOLD ?N1))
     (COND ((EQUAL ?TN 'FUT) (ASSERT (? RESULT ?N1 ?N)))
           ((ASSERT (? T-RESULT ?N ?N1)))
                                                      )
```

When we have a statement like "He put A in B.", the fact expressed by this DEMON is true for most choices of A and B. This is evidence that we should have this fact in a base routine, like PUT-IN-BASE. This routine will have to be complex enough to handle cases where the "general fact" is not true. Examples would be: "putting air in a flat tire", "putting a highly volatile liquid in an open container", "putting water into a sink", etc.

Now for the last BB DEMON that I wrote. I am trying to capture the fact that if a baby has the bottle that someone else just got, we can infer that the person gave it to the baby. The baby having the bottle in this case also implies that it will try to get food from it. There is also a question in my mind as to whether this DEMON belongs to baby base or here in BB. It may interact with the BABY DEMON on page 32, BABY-GIVE-FOOD. The DEMON should handle, in any case, the following story fragments:

```
(2.12)  Mother got the BB.  She gave it to the baby.
(2.13)  David got his BB.
(2.14)  Father got David's BB.
       "I want to give him his food", he said.
```

Now for the DEMON:

```
(ANTE BB-WANT-FOOD
     (HOLD PER1 PER2 BB FOOD N N1 N2 TN)
          ;?HOLD points to the "get BB" assertion
     (?N HAVE ?TN ST ?PER2 ?BB)
```

```
(DESTRUCT? ?NOLD)
(GOAL (? IS PR ST ?FOOD LIQ-BABY-FOOD))
(GOAL (? IS PR ST ?PER2 BABY))
(COND ((NOT (EQUAL ?PER1 ?PER2))
                     (IF-NEED (?N2 GIVE FUT ST
                          ?PER1 ?PER2 ?FOOD))
                  (ASSERT (? SUB-ACT ?NOLD ?N2)) )
(ASSERT (?N1 GET-FROM FUT ST ?PER2 ?BB ?FOOD))
(ASSERT (? SUB-ACT ?N1 ?N))
(ASSERT (? SUB-ACT ?NOLD ?N1))
                                    )
```

Finite State Model:

We now try to construct a model in which modes of behavior or self commands are used to formalize our knowledge listed in the paragraph we used earlier to describe infants. Our assumption is that there is some "abductive organ" that commits the entire organism to any of a number of incompatible modes. It will not surprise us if further research indicates that this assumption is not true physiologically for all infant behavior because we are concerned here only with how adequate is such a model for understanding infants in children stories. We hypothesize that a model of how children think infants work would place the infant in any one of a finite number of modes. The modes listed in Table A are a first attempt at specifying them. However, to handle all of the information we need, we will have to handle such knowledge as comes from facts about what an infant can not do. For example, we might have the following story:

(6.1) Daddy came home from the hospital with
      Mother and the new baby.
(6.2) Grandmother made coffee and cake for everyone,
      everyone that is, except the baby.

We could ask:

(6.3) Why didn't she make coffee and cake for the baby?

The answer, because it can't eat them, comes from the fact that new babies can't eat coffee and cake. We could represent this by saying that babies do not have a "solid-food eating" state. In the same sense we know that states like "going to school", "playing street games", "feed oneself", etc. will be of importance in our system because infants can not be in them and older children do enter them often. As noted earlier much of the information needed to understand infants was of this form. This corresponds to well defined modes that an infant can not achieve. Table B contains a list of such modes. Entries that have a star next to them are those that are probably irrelevant to understanding infants in children stories.

Table A

fear, anxiety (unstability)
sleep - rest
eat - drink
unhappy - cry
happy - smile
search - explore
urinate - defecate
locomotion
sit - stand - lie
disinterest (boredom)
sucking
hunger

Table B

clean - groom
get food - hunt - work
eat solid food
fight
mate* - fornicate*
give birth*
mother young*
build home*
anger
talk
greed - envy
laughter -euphoria
hibernate*
surrender
play

These tables indicate the level of behavior at which our model is aimed. Clearly, neurophysiological reactions, clinical reactions, etc., are not directly relevant to our understanding of infants behavior in children stories. Theories about which neurons have just fired and the paths of signals are not useful for understanding why someone does or does not do something in a story. We will now have to consider what particular behavior is possible for the infant in each mode.

Kilmer and McLardy use the term "act" to represent that behavior which takes place within a mode. A few examples will serve to illustrate what we mean by "act". If a baby committed to the searching mode sees two different toys, whether he goes to the first or second is an act decision. If he comes to any object, the decision as to whether to explore that object or continue his search is an act decision. A young infant seems to have the property that there are relatively few acts associated with each mode. This can be contrasted with say a grown wolf who, in the hunting mode may be "stalking, lying in ambush, driving a herd of prey, running to the attack, rejoining his pack, digging for mice, and so on." Now let us look at a more detailed finite state model of infants.

Finite State Infants

We will list the states, followed by rules for entering and leaving the state. We will then list some of the acts which occur within each state.

Sleep-deep

If no sleep (4 or 5 hours) and "happy" -> "sleep-deep". If "sleep-deep" and long time goes by (8 to 10 hours) -> "sleep-moderate", "sleep-light", "awake". "Sleep-deep" and very loud noise -> "crying-discomfort". "Sleep-deep" and low level talk -> "sleep-moderate". "Sleep-moderate" and gentle rocking -> "sleep-deep". Acts: We don´t really know if there are different ways for an infant to sleep (deeply, or anyway for that matter). We might however, consider the location where the infant is sleeping as an act decision, with the alternatives being crib, bassinet, carriage, etc.

Sleep-moderate

If "sleep-moderate" and low talk -> "sleep-light". Acts: Same as for "sleep-deep".

Sleep-light

If "sleep-moderate" and low talk -> "sleep-light". If "sleep-light" and loud noise -> "crying-discomfort". If "sleep-light" and noise -> "awake". Acts: Same as for "sleep-deep".

Eating

"Hungry" and has bottle with milk -> "eating". "Hungry" and Mother's breast -> "eating". "Hungry" and someone spooning baby food into babies mouth -> "eating". "Bottle empty" -> "happy" (stomach full). "Eating" and bottle empty -> "hungry", "crying-hungry" (The choice depends on degree of hunger of the infant and the amount of food in the bottle when he started. Note: We could handle this by making several hungry states, and specifying the "state" of the bottle, namely "full", "half", etc.). Acts: Sucking on nipple of bottle, sucking on mother's breast, squeezing plastic bottle.

Crying-irritation

Urinating -> "crying-irritation". Bowel movement -> "crying-irritation". Someone changes diaper and powders -> "happy". Pick-up and continuous auditory stimulation -> inhibits crying and enters "happy" for limited amount of time. Acts: Wailing, yelling, movement of body and limbs.

Crying-discomfort

Punching, teething, kicked, shock, pushed, overtired, picked up the wrong way, loud noise -> "crying-discomfort". Holding, soothing, non-nutritive feeding, auditory stimulation -> "happy". Acts: Same as "crying-irritation".

Crying-hungry

No food and 4 or 5 hours -> "crying-hungry". Bottle with enough food -> "happy". Non-nutritive sucking or bottle without enough food -> inhibits crying for a period of time. Acts: Same as "crying-irritation".

Anxious-unstable

Any low level (small amount) crying stimulus -> "anxious-unstable". Unfamiliar surroundings, strange faces, mother leaving alone -> "anxious-unstable". Soothing noise, mother returning, pick-up -> "happy". Lack of sleep -> "anxious-unstable". Acts: Frown and look unhappy, prepare to cry.

Happy

Full stomach and clean diaper and love and affection -> "happy". Any of: urinating, BM, pain, hunger, loud noise -> type of crying (irritation, discomfort, or hungry). "Happy" and toys -> "play". Acts: Smiles, contented sounds.

Play

The same "bad" stimuli listed for changing from "happy" to a type of "cry" apply here also. Take away toy -> "anxious-unstable". Acts: Follow objects with eyes, crawl, grasp objects, look or stare, shake or taste objects.

We shall also list some of the acts that occur with the baby bottle, depending on whether its state is full or empty.

Bottle full:

Drink till empty (baby contented).
Drink till empty (baby crying for more).
Drink partially and baby sleeps.
Drink till empty and baby sleeps.
Baby drops bottle.
Baby throws bottle.
Baby spills milk on himself and environment.
Baby plays with bottle.
Unscrews cap and pours our milk.

Bottle empty:

Baby crys.
Baby throws it away.
Baby sucks on it anyway.
Drop on floor.
Play with it.

Possible Extensions?

One of the deficiencies of the Charniak model is the implication that we have to tell it things in languages like PLANNER in order to add to its knowledge. One might propose that the solution is to write code for translating declarative statements into PLANNER assertions and theorems. In this way, it is hypothesised, we can add to the program's knowledge of an area. However, there are several reasons why this is not sufficient. As the program "understands" a story it is inevitable that it will run into phrases and words that it does not know. This is especially true of children stories as they often try to teach children new things. For example almost all of the stories that I have seen with infants in them try to teach the reader what one must do if they have a younger brother or sister. This teaching nature of children stories makes me feel that learning should play a part in our "understander". The Charniak system does not learn and it is not clear how we would go about making it learn. There are several reasons why learning is important, most of them are obvious and need no mention.

Stories are not perfect. A story that was written by a human is likely to be imperfect, after all, humans are fallible. When we read real life stories we often find steps missing, inconsistencies, jumps in the time frame, errors due to the author's lack of knowledge, and out-and-out lies. The system must have some way to understand these things for what they are, while still refusing to "believe" occurrences of them. Drew McDermott, in his Masters thesis proposal, proposes an approach to solving some of these problems. Charniak can not be faulted for not trying to solve the "adding information" problem, but we must still ask whether his model is suited for its solution. At the present stage of development of the Charniak model we are not forced to worry very much about this problem, but eventually we must consider the advantages and costs of adding new information.

In its present form it seems that it is not very well suited for making a learning system. This means simply that we are forced to do all of the work when we want to add knowledge to the system. We should point out that the system does "learn" when it reads a particular story, that is, if the story says that "Sally is an infant", it will act accordingly. If we asked the system whether Sally was a baby it would answer "yes", however once it has finished with that story and has gone on to another one, there is no carry over of information about anything. It assumes that there is no relationship between characters or events from one story to the next. So even though it might "learn" in one story, it does not attempt to remember past the immediate story. We see here that the real problem of learning is not making use of immediately useful information, but deciding what is important or general enough to keep and what should be flushed. One could conceive, possibly, of a system that was programmed to understand infants which could then program itself to understand toddlers after being given an explanatory story. To be able to do this would mean that we have some way to distinguish between facts that we want to maintain as part of our "improved" system (i.e. Toddlers can eat solid food if it is cut up into small pieces for them.) and facts which should be flushed after understanding the points that they illustrate (i.e. Jane ate some of the rice and a few pieces of the steak after Mother cut them for her.) Since we don't know how to decide such matters (we know so little about learning), the best strategy at this time is probably to ignore this problem now. Part of the problem may evaporate when we know more about DEMON construction and DEMON DEMON interaction but this remains to be seen.

What Have We Learned:

The primary question we must ask ourselves now is "What have I learned about the process of writing DEMONs?". Another question one may pose is "To what extent was it easier to write a DEMON after writing a few?". I am afraid that I was not able to find any significant progress with latter DEMONs over earlier ones. However, this is because the latter DEMONs that I worked on here were about very different sorts of things. If I were to sit down after the experience described here and try to write DEMONs for toddlers or milk bottles then the answer to the question posed earlier would have been very different. It seems to me that there are classes of DEMONs that have similar structure. So, it is easier to write DEMONs when they fit into the same class, but it also seems that when we are forced to work on very different sorts of knowledge we find the task of writing new DEMONs no easier then when we started.

This points to classification as a starting point which is the lowest level of the larger problem of structuring knowledge. Containers of hard objects, liquid containers, etc., all have something in common which makes the writing of, say, milk-bottle or cookie jar DEMONs slightly easier after writing DEMONs for PB (piggy bank) or BB. Perhaps this indicates that we have written our DEMONs on the wrong level of generalization. There is some reason to expect that we will need several types of DEMONs. We probably need a new type for acts of nature or physical facts. In fact what we may need is a classification scheme for DEMONs. When we examine how other people write DEMONs for many different topics we may have clues as to the level of generalization we want.

I believe that we need better structures for knowledge and that writing DEMONs is a reasonably good strategy for getting ideas about structuring knowledge. At this time we aren't even good at classification. We don't know very much about the DEMON writing process at this time. More work will have to be done writing DEMONs.

References:

(Almy 55) Almy, M., Child Development, Henry Holt, N. Y.: 1955.

(Bar-Adon et al. 71) Bar-Adon, Aaron and Werner Leopold (eds.) Child Language : A Book of Readings, Prentice-Hall, N.J.: 1971.

(Bever 70) Bever, T. "The Cognitive Basis for Linguistic Structure" in J. Hayes (ed.) Cognition and the Development of Language, Wiley, N.Y.: 1970.

(Black 68) Black, F. "A Deductive Question-Answering System," in Minsky (ed.) Semantic Information Processing (M.I.T. Press, Cambridge, Mass., 1968), pp. 354-402.

(Bobrow 64) Bobrow, D. G. "Natural Language Input for a Computer Problem Solving System," MAC-TR-1, MIT Project MAC, 1964.

(Buhler 35) Buhler, C. From Birth to Maturity, Routledge and Kegan Paul Ltd., London, 1935.

(Burger et al. 68) Burger, J. Schwarcz, Robert and Robert Simmons "User´s Guide and Program Description for Protosynthex III, SDC Corporation, TM 4068, Santa Monica, California: 1968.

(Charniak 69) Charniak, E. "Computer Comprehension of Calculus Word Problems," Proceedings of the 1969 International Joint Conference on Artificial Intelligence, pp. 303-316.

(Charniak 72) ____________ "Toward a Model of Children Story Comprehension", March 1972 draft of Ph.D dissertation, M.I.T., 1972.

(Chomsky 69) Chomsky, Carol The Acquisition of Syntax in Children from 5 to 10, M.I.T. Press, Research monograph number 57, Cambridge, Mass.: 1969.

(Chomsky 57) Chomsky, N. Syntatic Structures, The Hague: Mouton and Co., 1957.

(Chomsky 64a) ____________ Aspects of the Theory of Syntax M.I.T. Press, Cambridge, Mass.: 1964.

(Chomsky 64) ___________ "Current Issues in Linguistic Theory" in Fodor and Katz (eds.) The Structure of Language, pp. 50-79, Prentice-Hall, N.Y.: 1964.

(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, (A more complete 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.

(Feigenbaum 63a) Feigenbaum, E. A., "The Simulation of Verbal Learning Behavior" pp. 297-309 in (Feigenbaum and Feldman 63), 1963.

(Fodor and Katz 63) Fodor, J. A., and Katz, J. J. "The Structure of a Semantic Theory," Language, Vol. 39, (April-June 1963), pp. 170-210.

(Fodor and Katz 64) ___________________________ (eds.) The Structure of Language, Prentice-Hall, N.J.:1964.

(Gesell 48) Gesell, A. Studies in Child Development Harper and Row, N.Y.: 1948.

(Green 69) Green, C., "Theorem Proving by Resolution as a Basis for Question-Answering Systems", in Machine Intelligence 4, edited by B. Meltzer and D. Michie, 1969.

(Grice 71) Grice, H. P. "Meaning" in Steinberg and Jakobovits (eds.) Semantics, Cambridge University Press, Cambridge: 1971.

(Hailman 69) Hailman, J.P. "How Instinct is Learned" Scientific American 221, 6 pp. 98-106, Dec. 1969.

(Hewitt 69) Hewitt, C. "Planner: A Language for Proving Theorems in Robots," in Proceedings of the 1969 International Joint Conference on Artificial Intelligence.

(Hewitt 72) __________ "Descripton and Theoretical Analysis (using schemas) of PLANNER: A language for proving theorems and manipulating models in robots", Ph.D thesis reprinted as M.I.T., A.I. Lab TR-258, Cambridge, Mass.: 1972.

(Iberall 67) Iberal, A., and W. S. McCulloch, "Behavioral model of man - his chains revealed." Currents in Modern

Biology 1, 337, 1967.

(Jones 71) Jones, T. "A Computer Model of Simple Forms of Learning", Project MAC, TR-20, 1971.

(Kessen 70) Kessen, W., M. M. Haith, and P. H. Salapatek "Human Infancy: A Bibliography and Guide" in Carmichael's Manual of Child Psychology, Vol. I, Third edition, edited by P. Mussen, John Wiley, N.Y., 1970.

(Kilmer 69a) Kilmer, W. L., J. Blum, and W. S. McCulloch, "The Reticular Formation - Part I - Modeling Studies of the Reticular Formation," Division of Engineering Research, Michigan State University Report AFOSR-1023-67B.

(Kilmer 69b) ________________________, "Part II - The Biology of the Reticular Formation."

(Kilmer 69c) ________________________, "A Model of the Vertebrate Central Command System", Int. J. of Man-Machine Studies, Vol. 1, 279-309, 1969.

(Kilmer 72) Kilmer, W. L. and T. McLardy "A Model of the Brain's Hippocampal Computer", to appear in Cybernetics, Artficial Intelligence, and Ecology (ed. Robonson and Knight), Spartan Books, N.Y.

(Manwell) Manwell E., Always Growing

(McCarthy et al. 62) McCarthy, J, Abrahams, P., Hart, T., and M. Levin Lisp 1.5 Programmer's Manual M.I.T. Press, Cambridge: 1962.

(McCulloch 65) McCulloch, W. S., Embodiment of Mind Cambridge, Mass.: M.I.T. Press, 1965.

(Minsky 61) Minsky, M. L. "Steps Toward Artificial Intelligence," Proc. IRE, vol. 49, no. 1, 1961, also in (Feigenbaum and Feldman 63).

(Minsky 68) ____________ (ed.) Semantic Information Processing Cambridge, Mass.: M.I.T. Press, 1968.

(Minsky 70) ____________ "Form and Content in Computer Science", JACM, January 1970.

(Moses 67) Moses, J. "Symbolic Integration," MAC-TR-47, Project MAC, MIT, 1967.

(Moses 71) __________ "Symbolic Integration: The Stormy Decade," Communications of the ACM, Vol. 14, No.8, (August 1971), pp. 548-560.

(Newell and Simon 72) Newell, A. and H. A. Simon Human Problem Solving, Prentice Hall, N.Y.: 1972.

(Piaget 54) Piaget, J., The Construction of Reality in the Child, Translated by M. Cook, N.Y.: Basic Books, 1954.

(Piaget 55) __________, The Language and Thought of the Child, Translated by M. Gabain, N.Y.: The World Publishing Co., 1955.

(Piaget 56) __________, The Origin of Intelligence in the Child, Translated by M. Cook, N.Y.: International University Press, 1956.

(Piaget 69) __________, The Child's Conception of the World, Translated by J. and A. Tomlinson, Totowa, N.J., 1969.

(Quillian 66) Quillian M. R. "Semantic Memory," in Minsky (ed.) Semantic Information Processing (M.I.T. Press, Cambridge Mass., 1968) pp 216-270.

(Raphael 64) Raphael, B., "SIR: A Computer Program for Semantic Information Retrieval", Ph.D dissertation, M.I.T., 1964.

(Simmons et al. 68) Simmons, R. F., Burger, J. F., and Schwartz, R. M. "A Computational Model of Verbal Understanding," Proceedings of the 1968 Fall Joint Computer Conference, Washington DC: Thompson Book Co, 1968.

(Steinberg and Jakobovits 71) Steinberg and Jakobovits (eds.) Semantics: an interdisciplinary reader, Cambridge University Press, Cambridge: 1971.

(Strangle) Strangle, M., A Brand New Baby

(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.

(Sutro ?) Sutro, L. and W. I. Kilmer, Draft of chapters 4 and 5 (dated June 1972) from Design of Robots, To be published by M.I.T. Press.

(Tinbergen 69) Tinbergen, N. *The Study of Instinct*, Oxford University Press, N.Y.: 1969.

(Weissman 67) Weissman, C. *Lisp 1.5 Primer*, Dickenson Press, Belmont, California: 1967.

(Weizenbaum 67) Weizenbaum, J. "Contextual Understanding by Computers", C.A.C.M., 10, pp. 474-480, 1967.

(Werner 48) Werner, L. "Semantic Learning in Infant Language" in *Child Language*, Bar-Adon et al. (eds.).

(Woods 68) Woods, W. A. "Procedural Semantics for a Question Answering Machine," *Proceedings of the 1968 Fall Joint Computer Conference*, Washington DC: Thompson Book Co, 1968

(Woods 69) ____________ "Augmented Transition Networks for Natural Language Analysis," Report No. CS-1 to the National Science Foundation, Aiken Computation Laboratory, Harvard University, 1969.

(Winograd 71) Winograd, T. "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language," MAC TR-83 Project MAC, MIT, Cambridge, Mass., 1971.

(Winograd 72) ____________ "Understanding Natural Language", *Cognitive Psychology* 3,1, Academic Press, N.Y.: January 1972.