A CASE-DRIVEN PARSER

by

Brock H. Taylor

B.Sc., University of British Columbia, 1973

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department

of

COMPUTER SCIENCE

We accept this thesis as conforming

to the required standard.

The University of British Columbia

May 1975

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
Vancouver 8, Canada

Date April 23, 1975

ABSTRACT

A parser for English sentences using case and an augmented transition network is described. The interaction between syntax, semantics, and case is examined with special emphasis on their role in this particular system. The thesis concludes with a re-affirmation of the inseparability of syntax and semantics, and with cautious optimism for the future of case in computational linguistics.

TABLE OF CONTENTS

CHAPTER 1 1
SECTION 1.1: Introduction 1
SECTION 1.2: A Brief Look at Some Relevant Case Systems 2
SECTION 1.3: A top-level description 6

CHAPTER 2: CASES 9
SECTION 2.1: List of Cases 11
SECTION 2.2: Determination of Case 18
SECTION 2.3: Verb Specific Cases 21
SECTION 2.4: The Structure Returned 23

CHAPTER 3: THE DICTIONARY 25
SECTION 3.1: Verb Definitions 26

CHAPTER 4: AN EXHAUSTIVE EXAMPLE 32

CHAPTER 5: SOME DETAILS 45
SECTION 5.1 Anaphoric references 45
SECTION 5.2 Particles 49
SECTION 5.3 Ergative Verbs 51
SECTION 5.4 Some Obvious Extensions 52

CHAPTER 6: CONCLUSION 53

BIBLIOGRAPHY 57

APPENDIX I: Some Examples 59

APPENDIX II: Some Non-examples 74

APPENDIX III: A Description of What the System Returns 75

APPENDIX IV: The Dictionary 77

APPENDIX V: The Code 95
The Case Analysis Code 96
The ATN Grammar 129
Some Auxiliary Routines 137
The ATN Parser 138

TABLE OF FIGURES

FIGURE 1: An Example From Schank 4

FIGURE 2: The Dictionary Entry for the Verb "to order" 28

FIGURE 3: An ATN Parse of Sentence (4.1) 33

FIGURE 4: Incorrect Initial Structure for Sentence (4.1) ..37

FIGURE 5: The Structure for "the man beside the window" ...40

FIGURE 6: Correct Initial Structure for Sentence (4.1) 42

FIGURE 7: The Final Structure for Sentence (4.1) 44

Each man
has a way to betray
the revolution
This is mine

CHAPTER 1

I fought in the old revolution
on the side of the ghost and the king
Of course I was very young
and I thought that we were winning
I can't pretend I still feel very much like singing
as they carry the bodies away

-L. Cohen

SECTION 1.1: Introduction

The work described in this paper was largely inspired by Roger Schank's [11] work with Conceptual Dependancy Networks, although it is not intended to be an implementation of his ideas. It is a parser which uses the notion of case in its breakdown of a sentence, and which labels sentence parts with case names in the structure that it returns. The system is written in LISP and it runs at the University of British Columbia on an IBM 370-168 under the MTS operating system.

Case is a much more intuitive approach to language than transformational grammars are. It not only demands the integration of semantics into the parser, but it makes it obvious that this is where the semantics belong. Case at least partially unifies these long separated components of language.

SECTION 1.2: A Brief Look at Some Relevant Case Systems

Since Fillmore [6] wrote his paper "The Case for Case" in 1968 there has been a strong movement in computational linguistics towards a case analysis of the English language. Very briefly, Fillmore introduced a notion of deep case in which a simple sentence is composed of a verb with certain properties such as tense, aspect, etc., and a set of noun phrases labelled with case names. Fillmore does not give an exhaustive list of the deep cases in English, but he does list a few cases that he views as essential. They are: AGENTIVE, DATIVE, INSTRUMENTAL, FACTITIVE, LOCATIVE, OBJECTIVE, and BENEFACTIVE. Verbs are classified by their case environments, that is, by the set of cases they accept.

In 1972 Celce-Murcia [3] proposed a case system of five cases: CASUAL-ACTANT, THEME, LOCUS, SOURCE, and GOAL. Simmons [13] has written a system based on Celce-Murcia's theory. He classifies all verbs into what he call "paradigms" according to the way the five cases may be arranged with the verb in the surface structure. He gives the example of the verb "to break". It belongs to the "ergative" paradigm which allows four case patterns in the surface structure in the active voice:

(1.1) John broke the window with the hammer.

(1.2) John broke the window.

(1.3) The hammer broke the window.

(1.4) The window broke.

They can all be generated from the structure:

break: CASUAL-ACTANT1: John
THEME: window
CASUAL-ACTANT2: hammer

with the appropriate deletions and re-arrangements allowed ergative verbs. If enough verbs can be so classified, this provides a simple, yet apparently powerful mechanism for generating language. Using this scheme for parsing, however, leads to the usual problems case systems have. In sentence (1.4) what is the CASUAL-ACTANT1, the AGENT in my terminology? What is the CASUAL-ACTANT2, my INSTRUMENT case? The system may be able to tell from context, or to make an intelligent guess. I do not mean to imply that these are problems particular to Simmon's work. They are not. His paradigm idea is very useful in that it limits to four the number of possible arrangements of these three cases for ergative verbs.

The basic problem with using only five cases is that information which is fairly readily available in the surface structure tends to become obscured, if not lost. In sentence (1.1), for instance, we can infer a lot of information not explicit in the sentence. For instance, we guess that John

performed some act which caused the hammer to strike, and thus to break through the pane of glass in the window. He probably held the hammer in his hand and struck the window with its metal head, but he could have thrown it through the window, he could have struck the window sill or frame thus causing the pane to break, etc.

It is this more general picture that Schank tries to build up with his conceptual dependancy nets. Schank employs fourteen verbal primitives, or ACTs, and at least four cases (OBJECTIVE, RECIPIENT, DIRECTIVE, and INSTRUMENTAL) to build up a structure of primitives which captures the "meaning" of the sentence. I say "at least four cases" because other cases such as AGENT are implicit in his structures.

(1.5) John grew the plants with fertilizer.

For instance, from sentence (1.5) he builds the structure in figure 1.

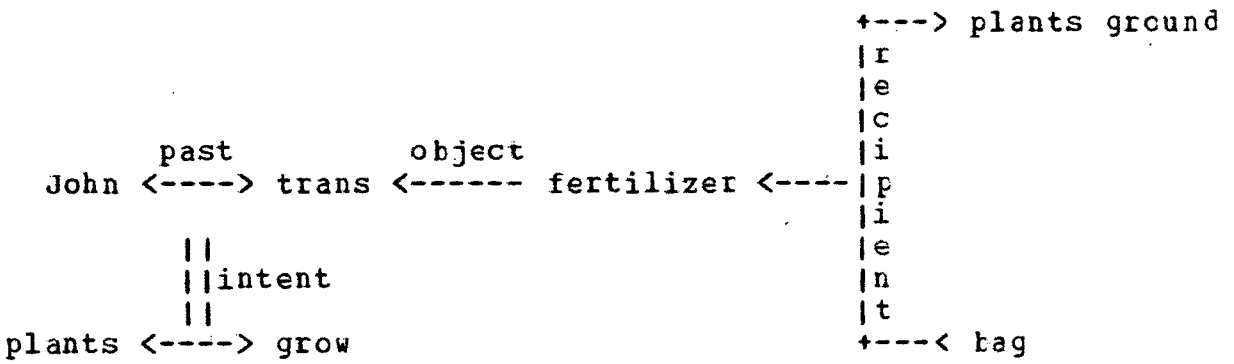

Figure 1 - An Example from Shank

Notice that "fertilizer" appears to be in the instrument case in the surface structure, but it is in fact the objective

case of the ACT "trans". Schank's structures capture more of the meaning of the sentences than Fillmore's deep structure do.

Currently at MIT Martin [9] is building a case system for the MAPL project. He has proposed a set of some thirty cases.

SECTION 1.3: A Top-level Description

The basic objective in this thesis was to explore the use of case in a system which parsed English sentences into a structure like Schank's conceptual dependancy networks. A loose and incomplete set of verbal primatives were used instead of Schank's fourteen primative ACTs because they seemed more intuitive. Verbs like "buy", "sell", "give", "take", "steal", etc., are represented by "trans"; verbs like "go", "come", "run", and "leave" by "move"; and so on.

An important feature of case which Simmons does not acknowledge is that Fillmore's "case frames" can be used to select, from the multiple meanings of a verb, its intended meaning in the sentence. The procedural definition of a verb in my dictionary consists of a list of case frames which distinguish verb meanings.

(1.6) John stole the jewel.

(1.7) John stole into the building.

For instance the two meanings of the verb "to steal" in sentences (1.6) and (1.7) are distinguished by the two case frames for the verb. The case frame for the meaning used in (1.6) demands a PATIENT, while the one for (1.7) requires a locative phrase with an adverbial preposition. This notion is used in a similar way by Schank.

(1.8) He stole.

Sentence (1.8) is ambiguous to my system since the case frames for the verb cannot tell which meaning is intended. The system must resort to context, as you and I do, to resolve the ambiguity. This mechanism has not been programmed, so currently the system will return the first meaning it succeeds on, which, after several failures to find a patient or a locative phrase, will be "to thieve".

The rest of this chapter will be devoted to a simple description of the control structure in which these case frames operate.

The system takes the sentence and tries to fit it into one of the frames, that is, it tries to find which meaning is being used in the sentence by looking at the sentence structure, its parts, and their semantic categories. If it goes through all of the meanings without finding one that fits the sentence, it weakens its tests on the semantic categories, and then tries all of the meanings again.

(1.9) The turnip played a wild dance on the violin.

In sentence (1.9), for instance, the system will at first fail to find an agent because all of the case frames for "to play" demand an animate, if not human, agent. These tests will be weakened, however so that on the second, or third time through the list of case frames a meaning will be selected. This process of going through the case frames again with weaker tests is refered to as the top-level back-up, although it is not really back-up in the usual sense, since decision points are not

stacked and backed-up through on failure.

This was done for a specific reason. Since the case frames are tried in order, the ordering can be used to help disambiguate the verb. In defining the second case frame for a verb, for instance, it is often useful to use the fact that the first case frame has failed.

If the system backs up into this second frame with a weaker test, however, it may not be true that the first case frame would fail on the weaker test.

Another reason for not backing up in the usual sense, is for efficiency. Since the case frames are tried in order, obviously the most commmon verb usages should be put first in the list. If all of the meanings fail beacuse the tests are too severe, one still wants to try the most common meanings first with the weaker tests.

Once the system has succeeded on a particular meaning of the verb it will proceed to look at the rest of the sentence not already looked after by the framing process. Here a genuine back-up mechanism is employed. If the system thinks it has the correct case for a particular sentence part, it stacks a form to put the corresponding sub-structure into the main structure of the parse. If the rest of the sentence is handled correctly, when the stack is unwound at the end, all of the forms so stacked are EVALed, and these sub-structures are put in. If, on the other hand, the system fails later on in the parse, it will back up to this decision point and look for another way to handle this particular sentence part.

CHAPTER 2

They will never ever reach the moon
at least not the one that they're after
It is floating broken on the open sea
and it carries no survivors

-L. Cohen

CASES

The question of how many cases we need to describe English is a contentious one. As mentioned above, Fillmore is vague on the issue, whereas Celce-Murcia claims that five cases will do. The purpose of this work is to capture as much of the meaning of a sentence as is possible, and to make it explicit in a formal structure. Case, then, is an explanation of the semantic function of a sentence part, so a fairly large number of cases have been used, one for each of these "semantic functions". We are not adamant about our set of cases. The system is flexible and structured enough, that the addition or deletion of cases is a simple operation.

The system was originally designed with Martin's thirty-odd cases as its basis. There are currently twenty-four general cases implemented, plus numerous verb-specific cases. Some of Martin's cases have been dropped completely, some have simply not been implemented yet, and several have been added.

The following list of these twenty-four cases should be

fairly self-explanatory. With each case is a set of examples (mostly from Leonard Cohen, if they seem strange) with the case in question underlined. Those cases which are flagged by certain prepositions have these prepositions listed.

SECTION 2.1: List of Cases

Agent

The man in white says he has no friends.

The trees are burning in your promised land.

I got wiped out by several charismatic holy men.

Love calls you by your name.

Patient

He trained a hundred women just to kill

an unborn child.

You can read their address by the moon.

I am punished by your solitary truth.

I washed my eyelids in the rain.

Location

A scalpel blade lay beside my silver spoon.

Your letters all say that you are beside me now.

I sleep beneath the golden hill.

In city and in forest our steps will always rhyme.

I have lain by this window long enough.

Flagged by at, in, on, by, near, beside, before, beneath, upon.

Time

I love you in the morning.

If by chance I wake at night and ask you who I am.

I suppose that he froze when the wind took your clothes.

We all should study etiquette before we study magic.

She fell in love for us in 1961.

Flagged by at, in, on, by, near, during, before, after, when, while, around.

Recipient

She said that you gave it to her.

They brought me their comfort.

Flagged by to.

Source

We will come from the shadows.

Thanks for the trouble you took from her eyes.

Taking from his wallet an old scheduel of trains, he'll say, "I told you when I came I was a stranger."

Flagged by from.

Destination

I will let you take me home.

Into this furnace I ask you now to venture.

She leads you to the river.

Take me to the slaughterhouse.

Flagged by to, into, onto, toward, towards.

Duration

You have lived for years for nothing.

He will sleep for one night with the woman
he plans to abandon.

They played their game in two hours.

Flagged by for, in.

Instrument

She touched your perfect body with her mind.

Cover up your face with soap, now you're Santa Clause.

The infant with his cord is hauled in like a kite.

Flagged by with.

Manner

They left in a hurry.

I walk with my mouth shut into the drifting world.

I choose the rooms that I live in with care.

I am eating a juicy peach by lamplight.

Flagged by in, with, by.

Path

We started up the mountain.

Through the graves the wind is blowing.

I journeyed down a hundred steps.

We poured across the border.

Flagged by by, about, along, up, down, around, across.

Method

I do not intend to save you any work by keeping silent.

Some men find strength by going their lonely ways.

Who would have foretold that the heart grows old from touching others.

Flagged by by, from.

Expected-effect

I am only whispering to my tomahawk so the image itself will reduce you to scorn.

They blamed Mitchell so the people would not suspect them.

Flagged by so.

Quantity

What is grey and comes by the quart?

We produce oil by the barrel and sell it by the cup.

Flagged by by.

Exchange

I bought it with a nickle and I sold it for a dime.

He wants to trade the game he plays for shelter.

You could not trade them for priesthood.

They are waiting to see a signal that some killer will be lighting for pay.

Flagged by with, for.

Beneficiary

I fought every man for her until the night was over.
I sing this to the crickets, I sing this
for the army.
I forgot to pray for the angels,
and so the angels forgot to pray for us.

Flagged by for, to, before.

Co-agent

I went with the one who found me.
I lived with a child of snow.
Nancy wore green stockings, and she slept
with everyone.
Oh, come with me my little one.

Flagged by with.

Descriptive (A case of the Noun)

Suzanne takes you down to her place near the river.
The woman in blue is asking for revenge.
The hand of your beggar is burdened down with money.

Flagged by of, from, at, in, on, with, by, near, beside, before, after, along, up, down, around, across, under.

Topic

It is time we began to laugh about it all again.
Lets not talk of love or chains.
Sometimes I find I get to thinking of the past.

Flagged by of, about.

Cause

It rained so I got wet.

Flagged by so.

Purpose

You who build these alters to sacrifice these children, you must not do it any more.

You were traning with mirrors to make yourself perfect.

I lit a thin green candle to make you jealous of me.

Flagged by to.

Enable

He robbed the bank so he and his wife could have separate holidays.

I had to kick you down the stairs so I could savour unemployment once again.

We have to put her away so we can get back to the war.

Flagged by so.

Like

He sank beneath your wisdom like a stone.

Like a bird on a wire, like a drunk in a midnight choir, I have tried in my way to be free.

The sun pours down like honey on our lady of the morning.

Flagged by like.

With

The room just filled up with mosquitos.

He locked himself in a library shelf

with the details of our honey-moon.

Flagged by with.

SECTION 2.2: Determination of Case

With most of the cases above is listed the prepositions that flag each case. This is a gross over-simplification of the relationship between verbs, prepositions, and cases. A preposition which flags one case for one verb may very well flag another case for a different verb.

(2.1) I walked about the room.

(2.2) I talked about the room.

For verbs of movement like "walk", "about" flags the path case as in (2.1), but for verbs of communication like "talk", it usually flags the topic case as in (2.2). Martin notes this and proposes for each verb meaning to list all of the cases flagged by each preposition. Since there is a great deal of uniformity across verbs in this regard, however, it much more efficient to set up a master table for all verbs in the system, then for each verb, to list only the irregularities.

(2.3) The scandal was whispered about the room.

Sentence (2.3) illustrates that "about" <u>can</u> flag the path case for a communication verb, so we do not want to rule out the path case, we just want topic to be tried first. In (2.3) the topic slot will already be filled by "the scandal", so topic will be rejected, and the path case will be tried next.

This foregrounding of cases is specified by the person who

writes the dictionary. He may specify that for the verbs talk, laugh, whisper, etc., the occurrence of the preposition "about" should trigger the topic case before the path case.

There is another obvious way of determining case names for prepositional and noun phrases. Consider the sentences:

(2.4) Fred bought the car for Mary.

(2.5) Fred bought the car for one dollar.

The preposition "for" flags many cases. In (2.4) it flags the beneficiary case, and in (2.5) it flags the exchange case. Associated with each case is a test which a phrase must pass to be accepted as that case. In the beneficiary case the test is:

(MUST-BE ANIMATE)

indicating that the beneficiary has to be animate, while the test for the exchange case is:

(NOT (SHOULD-BE HUMAN ABSTRACT))

indicating that one usually dones not exchange something for a person or something abstract. These tests correctly sort out the cases in sentences (2.4) and (2.5).

In general tests on cases are very difficult to design adequately. What test would be appropriate for the topic case? What could not be talked, laughed, or cried about? Perhaps some complex verb and context dependant test could be concocted, but

one has not been designed for this system. The test for the topic case is therefore T. One must therefore be careful when invoking the topic case because it will always succeed.

The exchange case is almost as bad. Anything can be exchanged for something. The weak test

(NOT (SHOULD-BE HUMAN ABSTRACT))

is put in which will at first fail if a human or an abstract noun is the candidate, but will pass if nothing else seems to fit either. This simple test runs into problems with certain sentences.

(2.6) I paid a fortune for my mother's release.

(2.7) I paid a fortune for my mother.

(2.8) I paid a fortune for the prcsititute.

It will initially force the exchange case to reject "fcr my mother's release" in (2.6) because it is abstract, but later on it will accept it since all of the other cases flagged by "for" will also reject it. Sentence (2.7) is ambiguous, but "mcther" is almost certainly in the beneficiary role here, so again the test works correctly by rejecting the exchange case. Sentence (2.8) is also ambiguous, but our interpretation would usually be that "prostitute" is being used in the exchange case here. The system will, however, assign it the beneficiary case as it did in (2.7). Much work must be done on case tests if this paradigm is to be useful.

SECTION 2.3: Verb-specific Cases

The cases listed in SECTION 2.1 are general cases, almost all of which can be associated with any verb. Many verbs, however also have special cases.

The verb "to be" is a prime example. In this system "to be" has eight meanings. Let's look at a few of them.

The third meaning of "to be" is "to have the property ...", as in sentence (2.9).

(2.9) The house is red.

If you look at the entry for "to be" in the dictionary in APPENDIX IV you will see that this meaning is picked up by the form:

((ADJ-LIST T) ADJ OBL (AGENT T) AG OBL)

The function ADJ-LIST looks for an adjective phrase. So an adjective phrase is considered to be a special case of the verb "to be".

The sixth meaning of "to be" is "to be from . . .", as in sentence (2.10).

(2.10) The lady is from Ouagadougou.

This special case is looked for by the function CLAUSE, which takes as arguments a list of prepositions to look for, a test

that the prepositional phrase must pass, and a name for the case it is looking for. In this case the form is:

(CLAUSE '(FROM) '(MUST-BE COUNTRY CITY TOWN CONTINENT LOCATION) 'LOCATION) [1]

In other words: Look for a phrase starting with "from" which is a location of some kind.

(The third argument is needed for the back-up mechanism. It is possible to turn the MUST-BE tests off for only certain cases. This is useful when you are happy with the action of all of the tests but those associated with a certain case or two.)

My notion of case is rather general, in which an important function of a case is to distinguish betwen verb meanings. In the following chapter we will look carefully at just how cases are employed to do this.

---

[1] Note that (MUST-BE LOCATION) will suffice as the test here, since COUNTRY, CITY, etc., all have the attribute LOCATION. The other categories are put in since they occur most often in this structure, and will thus speed up the search. "He comes from the West.", would also be handled since "West" would also have the attribute LOCATION.

SECTION 2.4: The Structure Returned

APPENDIX III gives a formal description of the structures returned by the system. This section will give a more informal and intuitive description. APPENDIX I gives lots of examples of these structures.

The basic idea was to return a structure which could be EVALed by LISP for further processing. The top-level function was to be "<==>" which would be an FEXPR since it can take a variable number of arguments. The first argument to <==> is the agent or subject of the verb. The "or subject" is necessary in cases like sentence (2.11) because some verb primitives, HAVE-PROP in this case, allow <==> to take sentences as a first argument.

(2.11) That I came is silly.

The second argument to <==> is the tense of the verb.

The third argument is the predicate. This is always a list beginning with the verb symbol "<--". The idea again was that this would be a form to be EVALed. The first argument fo <-- is a verb-primitive. The second and third arguments are optional for some verbs, and not for others. The second is an adverbial phrase structure beginning with "<-ADV-". For instance,

. . . go quickly . . .

Would have the structure:

(<-- MOVE (<-ADV- QUICKLY))

If the verb takes a patient in the sentence, this patient follows the adverbial phrase if it is there, otherwise it follows the verb primitive.

The rest of the arguments to <==> are the cases of the verb, and any top-level adverbial phrases, like "well" in sentence (2.12).

(2.12) He played the piano well.

Their order is not considered important in the structure, so they just appear in a list at the same level as the verb symbol <--. The cases are lists beginning with an atom of the form

<-CASE-

were CASE is replaced by any of the cases but AGENT, PATIENT, or DESCRIPTIVE. These three cases are treated specially. As mentioned above, the agent is the first argument of <==>, and the patient is the last argument of <--. The DESCRIPTIVE case is a special case which is used to handle all of the prepositional phrases which modify nouns. This case, therefore, is embedded in the noun-phrase structure. This is made more explicit in CHAPTER 4.

CHAPTER 3

Thanks for the trouble
you took from her eyes
I thought it was there for good
so I never tried

-L. Cohen

THE DICTIONARY

Each word in the dictionary is set up as follows:

(WORD I1 P1 I2 P2 . . . In Pn)

where WORD is the word in question, and each pair (Ii Pi), i>0 is an indicator and a property which are to be put on the property list of WORD.[1]

For each word, one of the indicators should be a part of speech. The property under this indicator is information for the morphology routines. For a noun, for instance, the property under N indicates how the plural of the noun is formed; for verbs, the property under V indicates how the first-person singular and past tenses are formed; etc.

On the property list of nouns is a list of the noun's

---

[1] For a detailed description of just what properties go with what word types see APPENDIX IV.

properties under N-PROP, its superset under SUPERSET, and so on. For verbs one puts a procedural definition of the verb under V-MEAN. To understand just how the system works it is necessary to see how these procedural definitions work.

SECTION 3.1: Verb Definitions

The verb is treated as the focal point of the sentence. A verb can have many meanings. The system discovers which meaning is intended by looking at the rest of the sentence. In so doing, it builds a structure representing a parse of the sentence.

With each verb, then, is associated a set of cases of the verb, some obligatory, some optional, and some conditionally optional. These cases are embedded in a form on the property list of the verb. Ample examples of verbal definitions can be found in APPENDIX IV, but a careful look at one in detail is necessary for an understanding of the system. In CHAPTER 4 a long example is given showing just how these definitions work in parsing a sentence.

Consider the verb "to order." Its dictionary entry is as follows:

```
(ORDER V
       S-ED
       PREP-CASE ((WITH WITH))
       V-MEAN
       (IF ((AGENT (MUST-BE HUMAN))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (PATIENT (MUST-BE ANIMATE))
            PA
            OBL
            (TO-COMP (GETR PA))
            TOC
            OBL)
           (BUILDQ ("<==>" ? "+" ("<--" ORDER "+")) AG TNS TOC)
           ((AGENT (MUST-BE HUMAN))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (PATIENT (AND (MUST-BE THING) (NOT (MUST-BE HUMAN))))
            PA
            OBL)
           (BUILDQ ("<==>" ? "+" ("<--" ORDER ?)) AG TNS PA)))
```

Figure 2 - The Dictionary Entry for the Verb "to order"

Under the indicator V-MEAN you will see a form beginning

```
(IF ((AGENT . . .
```

IF is a function which takes an even, but otherwise variable, number of arguments, each pair representing a meaning of the verb. The first element of each pair is a set of cases to be looked for, and the second is the structure to be built if they are found.

It is in the first element of the pair that the complexity lies. Let us look at it more closely.

The list of cases is, in fact, a list of triples. The first element is a form to be EVALed. It is usually looking for

a case, but any form is admissible. The second element of the triple is an atom: a register name. If the first form EVALs to a non-NIL value, the value is put into this register. In our example, for instance, the first triple is:

(AGENT (MUST-BE HUMAN)) AG (OPT (GETR PASSIVE) 'SOMEONE))

The function of the first form is to find the agent of the sentence. If it succeeds, this agent is put into register AG.

The third element of the triple says what to do on failure. If it is the atom "OBL", then this indicates that the case was obligatory, so if it was not found, then IF should fail on this meaning of the verb. If the atom is "OPT", then the case is optional, so the register is left empty, and IF continues with this meaning. The third possibility is that this third element is a form, in which case it is EVALed. If it returns "OBL" or "OPT", then the result is as described above. If it returns anything else, then that is put into the register, and IF continues with this meaning of the verb.

Let us look at "to order" again. The third element of the first triple is

(OPT (GETR PASSIVE) 'SOMEONE)

OPT is a very simple function which, if its first argument is non-NIL, returns its second argument. Otherwise it returns "OBL".

(GETR PASSIVE)

is true if the sentence is in the passive voice.

So the first triple can be read as follows:

Look for an agent which must be human. If you find one, put it in register AG. Otherwise, if the sentence is passive, make SOMEONE the agent. Otherwise fail.

The second triple is more simple. It merely says: If you find an animate patient, then put it in register PA, else fail.

The third triple is equally simple, but it is not looking for a case, but a to-complement.[1] If these three elements are found in the sentence, then the system will look no further, but assume that it has found the correct meaning of the verb. It will EVAL the second form of the pair: in this case:

(BUILDQ ("<==>" ? + ("<--" ORDER +)) AG TNS TOC)

which builds the basic structure for the sentence.

BUILDQ takes a variable number of arguments. The first is a kind of template with slots in it. The rest of the arguments fill the slots.

The "+" denotes a slot which is filled by the contents of a register.

The "?" Is filled by the application of the function NOUN-PUT to the contents of a register.

---

[1] An example of a to-complement is: "Fred took the book to anger Mary."

Finally, the "#" indicates that a form is to be EVALed, and the result put into the slot. The slots are filled in order by the second, third, etc, arguments.

So in this case:

(NOUN-PUT (GETR AG)) is put in for the ?.

(GETR TNS) in place of the first +.

(GETR TOC) for the second +.

Where GETR returns the contents of a register.

CHAPTER 4

You who build the alters now
to sacrifice these children
you must not do it any more
A scheme is not a vision
and you never have been tempted
by a demon or a god

-L. Cohen

AN EXHAUSTIVE EXAMPLE

Programming details to not belong in a thesis of this kind. All of the code is in APPENDIX V for those interested. In the following example, then, function names and excessive details will be, on the whole, left out. A detailed account of the basic algorithm and control structure will be given. We will look at a simple sentence. More complex structures such as relative and subordinate clauses are treated in much the same way as their parent sentences.

Consider the sentence:

(4.1) The man beside the window played the piano for Mary.

The first step in the process is a partial parse using an augmented transition network (ATN) as described by Woods[15]. "Partial" is used because the parser is purely syntactic. The structural description usually associated with a parse is incomplete. That is, no decisions are made about what modifies

what, what meaning of the verb is being used, etc. The basic idea behind the ATN is to find the verb; but while it is doing this, it might as well chop the sentence up into its parts. There are problems with just how this chopping should be done, but with most sentences it is straight forward. A look at the sample parses in APPENDIX I will give more examples of this notion.

The ATN parse returned will look like this:

```
S
    NP NIL
        DET THE
        N MAN
            NUMBER SG
    PP NIL BESIDE
        NP NIL
            DET THE
            N WINDOW
                NUMBER SG
    VP NIL
        TNS
            PAST
            VOICE ACTIVE
        V PLAY
    NP NIL
        DET THE
        N PIANO
            NUMBER SG
    PP NIL FOR
        NP NIL
            NPR MARY
```

Figure 3 - An ATN Parse of Sentence (4.1)

ATNs have been found to be a useful formalism for procedural definitions of grammars, but have been criticized for their slowness. Because this ATN returns such a simple

structure, however, it is very fast, and consumes little of the total time to process a sentence. It is on this preliminary parse that the program works.

First, the main verb is found, and a function is invoked which controls the top-level back-up. This function EVALs the form on the property list of the verb under the indicator V-MEAN. This form for PLAY is a very long one, so rather than reproducing it here, you are refered to the entry for PLAY in the dictionary which is listed in APPENDIX IV.

The form in question is a call to IF, whose mechanism has been briefly described in SECTION 3.1. In this case IF has eight arguments, indicating that there are four meanings to the verb PLAY in the system.

The first meaning is "to play a musical instrument."

The first case looked for is the AGENT. This agent should be a musician, and must be human. This search is made by EVALing the first form in the first triple of the first argument to IF:

(AGENT (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)))

AGENT is fairly complex, but basically it looks for a component of the ATN parse (in future called the "p-parse", for partial-parse) which is in an appropriate position to be an agent, and which passes the test (the argument to AGENT.) By 'appropriate position' is meant, for instance, that if the sentence is in the active voice, the agent is <u>probably</u> the first noun phrase in the sentence.

In this case, AGENT immediately finds "the man" as the obvious candidate, and it applies the test

(AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN))

Now, unless something special has been put on the property list of MAN previously, the

(SHOULD-BE MUSICIAN)

part of the test will fail.

(There are two levels of tests in this system: SHOULD-BE tests and MUST-BE tests. This crude mechanism is very useful for forcing a verb like PLAY to look hard for a musician to play an instrument -- but to accept any human if it fails at first. This is especially powerful for resolving anaphoric references.)

So AGENT fails. This invokes the third element of the AGENT triple:

(OPT (GETR PASSIVE) 'SOMEONE)

This may be read as: AGENT is optional if the sentence is in the passive voice, in which case put SOMEONE in as the agent. Otherwise AGENT is obligatory. So, since the sentence is not passive, AGENT is obligatory. Since the AGENT case was not found, this first meaning of PLAY fails.

So IF goes on to the next pair of arguments. This pair is

designed to pick up the meaning of PLAY as in "tc play music."

Note that the test on AGENT is just like the previous cne, so it will also fail.

So we gc on the the third meaning of to PLAY: "tc play a sport." Here the test on AGENT is

(AND (SHOULD-BE SPORTS-MAN) (MUST-BE HUMAN))

Once again, providing MAN does not have SPORTS-MAN cn its property list, this meaning fails.

On to the fourth, and last, meaning. This is sort of a catch-all. It is just "to play" as in "tc entertain cneself." Here the test on AGENT is

(MUST-BE ANIMATE)

"The man" passes this test. (Note that in the dicticnary MAN has the property ANIMATE under N-PROP)

Since AGENT is the only case locked for, this meaninc is taken to be the correct one, and the fcllowing structure is built by the call to BUILDQ:

```
<==>
    N MAN
        NUMBER SG
        <-DEFINITE- THE
    PAST
    <-- DO
    <-CAUSE-
        <==>
            N MAN
                NUMBER SG
                <-DEFINITE- THE
            PRESENT
            <-- HAVE-PROP ENTERTAINED
```

Figure 4 - The Incorrect Initial Structure for Sentence (4.1)

So IF has completed its job. It has found what it takes to be the correct meaning of the verb. Now the rest of the sentence must be handled. The second element of every top-level list in the p-parse is a flag which is initially NIL, but which is turned on when that part of the sentence is considered to be correctly handled. In our example, so far only two parts are so flagged: the first noun phrase: "the man", and the verb phrase. The function which takes care of the rest of the sentence simply goes down the p-parse checking these flags. If it finds one which is NIL it works on that part of the sentence until it either succeeds, or fails -- causing back-up.

In our example, then, the first phrase it comes upon needing work is the prepositional phrase: "beside the window."

There is a structure in the system which associates each preposition with the cases it may flag. BESIDE flags the cases: LOCATION and DESCRIPTIVE.

All of the cases but DESCRIPTIVE are cases of the verb. DESCRIPTIVE is a special case which is used for preposition

phrases which modify nouns.

When the list of cases associated with a preposition is retrieved, there is a question as to which case to try first. For this there is a foregrounding routine. There are several criteria for foregrounding:

First of all, on the property list of each verb is kept a record of which prepositions flagged which cases in the previous sentences. The cases associated with the preposition in question (if there are any) are foregrounded, so they will be tried first (the most recent case first, etc.)

Secondly, if DESCRIPTIVE is one of the cases in the list of cases for this preposition, and if a noun phrase or a prepositional phrase immediately precedes the phrase in question, and if the noun in that noun phrase or prepositional phrase is not a proper noun, then DESCRIPTIVE is put at the front of the list, so it is tried first.

This seemingly obscure rule for foregrounding the DESCRIPTIVE case is just a heuristic. If the tests associated with each case are good enough, it makes no difference to the final outcome if the foregrounding is done or not. In some instances, however, if the DESCRIPTIVE case is not tried first, it will never be tried. In our example, for instance, it is the man who is beside the window (DESCRIPTIVE case); he did not play the piano beside the window (LOCATION case). But it is perfectly feasible for him to have played it beside the window (if we know nothing about the location of the piano.) So either of the cases will succeed. It is only the position of the prepositional phrase that indicates which case is correct.

Continuing with our example: the DESCRIPTIVE case is foregrounded, and so the descriptive case function, DESC, is invoked with the phrase "beside the window" as its argument. Since the descriptive case almost always involves a prepositional phrase modifying the noun phrase or prepositional phrase immediately before it, DESC first checks to see if "beside the window" is a possible descriptor of "the man."

Since we do not have a data base to check to see if there is a man beside a window, our check must be a general one. Most nouns have a size associated with them under the indicator OBJ-SIZE. This is a very crude breakdown of objects into eleven size categories. "The world" is size 10 and "a pin" is size 0. (As has been pointed out by an esteemed colleague, these sizes should be able to be changed by classifiers, adjectives, or modifying phrases. A toy elephant is probably not the same size as an elephant. This feature is currently not implemented.) The check for "beside" is merely used to rule out things like "the pin beside Canada." Abstract nouns have no size information, so sentences like "He had a thought beside the ocean" are not ambiguous. In any event, "beside the window" is found to be a likely modifier of "the man", so DESC succeeds.

Since "beside" is a locative type of preposition, DESC returns the structure:

(<-LOC- BESIDE (NP (N WINDOW (NUMBER SG) (<-DEFINATE- THE))))

A form is stacked which will put this structure into the main sentence structure if the rest of the sentence can be

handled. Just where it is placed is determined by DESC. Since the prepositional phrase modifies "the man", it will be put in as follows:

```
N MAN
    NUMBER SG
    <-DEFINITE- THE
    <-LOC- BESIDE
        N WINDOW
            NUMBER SG
            <-DEFINITE- THE
```

Figure 5 - The Structure for "the man beside the window"

So the prepostional phrase "beside the window" is flagged as completed, and the next unflagged phrase, "the piano", is picked up.

Here we run into problems. Where does "the piano" fit into the structure? What does it modify? What is its case? There are relatively few ways a noun phrase can be used at this point. It could be an example of the TIME case, as in "I came home this morning.", but "piano" fails the TIME-test. It could be a classifier, but the phrase following it would have to be a noun phrase for this to be the case. So we have a failure. Something has gone wrong. IF must have chosen the wrong meaning of the verb. So we back up.

All the parts of the sentence that are flagged as used are un-flagged, and we back up into IF again. Here it is found that there are no meanings of the verb left to try. One of the meanings that was rejected earlier must have been the correct one. So we fail out of IF and get back into the top-level back-

up mechanism.

There are two reasons we could have failed:

1) Either we did not look far enough back in our attempt to resolve an anaphoric reference.

or

2) Our tests were too severe. (ie: the SHOULD-BE tests made us fail when they should not have.)

The anaphoric part of the system has not been explained yet, that will come later, but there were no pronouns in the sentence, so the first reason can be ruled out. In order to weaken the tests, a flag is set to shut off the SHOULD-BE tests. That is, all SHOULD-BE tests will succeed in future.

So we enter IF once again.

We start as before, but this time the first invocation of AGENT will succeed, because the test

(AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN))

succeeds. So the structure it returns is put in the register AG.

Then IF takes us into the second triple of parameters. The form

(PATIENT (MUST-BE MUSICAL-INSTRUMENT))

is EVALed.

PATIENT is very similar to AGENT: it looks in the appropriate place in the sentence for the patient of the verb. It then applies its TEST to it. In an active sentence, such as our example, the candidate for PATIENT is the first noun phrase after the verb. "The piano" is found. Since it passes the test

(MUST-BE MUSICAL-INSTRUMENT)

PATIENT returns "the piano" as the patient of the sentence.

So again IF thinks it has found the correct meaning of the verb, and so it EVALs the BUILDQ associated with that meaning, and the following structure is built:

```
<==>
    N MAN
        NUMBER SG
        <-DEFINITE- THE
    PAST
    <-- DO
    <-CAUSE-
        <==>
            N PIANO
                NUMBER SG
                <-DEFINITE- THE
            PAST
            <-- EMIT
                NP
                    N SOUND
```

Figure 6 - The Correct Initial Structure for Sentence (4.1)

(An aside: I do not intend to defend the structures that my system returns. They are vaguely like the structures that Schank's system returns, as illustrated in Chapter 1, but are

more or less arbitrary. If the user does not like them they are easy to change.)

Once again we must try to clean up the unflagged parts of the sentence. The first one again is "beside the window", and exactly the same thing is done that was done before: it is decided that "beside the window" is a locative descriptor of "the man", and this decision is stacked for later action.

The only other part of the sentence to be handled is "for Mary." As with "beside", the cases associated with "for" are returned from the CASE-TABLE. They are: DURATION, BENEFICIARY, EXCHANGE, and IND-SUBJ. (IND-SUBJ has not been implemented yet.) Assuming that there have been no relevant previous sentences, the foregrounding of cases will have no effect on this ordering.

The DURATION case is tried first. DURATION is a particularly simple case. Basically it checks to see that the noun phrase in the prepositional phrase has the property TIME under the flag N-PROP. "Mary" fails this test, so DURATION is rejected as being the correct case.

The next case is BENEFICIARY. The only test for this case is that the noun phrase be animate. "Mary" passed this test since it has the SUPERSET WOMAN and WOMAN has the N-PROP ANIMATE. So BENEFICIARY succeeds and returns:

(<-BENEFICIARY- (NPR MARY))

Unlike "beside the window", this phrase is a case of the verb. All cases of the verb (but AGENT and PATIENT) are considered to

be essentially parallel with respect to the verb, so they are put into the structure at the same level: that of the verb symbol "<--", and their order is independent. So a form is stacked to put the above structure into the main sentence structure in the correct location.

Next the p-parse is checked for more unused phrases. None are found, so we are done.

The two forms are put into the structure, and it is returned as the "meaning" of the sentence:

```
<==>
    N MAN
        NUMBER SG
        <-DEFINITE- THE
        <-LOC- BESIDE
            N WINDOW
                NUMBER SG
                <-DEFINITE- THE
    PAST
    <-- DO
    <-CAUSE-
        <==>
            N PIANO
                NUMBER SG
                <-DEFINITE- THE
            PAST
            <-- EMIT
                NP
                    N SOUND
    <-BENEFICIARY-
        NPR MARY
```

Figure 7 - The Final Structure for Sentence (4.1)

CHAPTER 5

He gave you a german shepherd to walk
with a collar of leather and nails
and he never once made you explain or talk
about all of the little details

-L. Cohen

SOME DETAILS

SECTION 5.1 Anaphoric References

Perhaps it is not the function of a parser to resolve anaphoric references, but this system was constructed to be a front end to a larger system which would require references to be resolved, so a mechanism for resolving them was put in. As has been said before, this system is not designed with any particular domain in mind, so no data base is kept. The resolution of anaphoric references, therefore, must depend solely on the previous sentences. The anaphoric routines will have to be changed to work in a larger system which maintains some kind of global structure describing what has gone on so far and how it fits into the world knowledge that the system has. The anaphoric capabilities were put into this system because, if

a reference is unresolved the system often cannot tell what case it is. They were also put in to illustrate how well such a mechanism fits into the case paradigm.

Only the basic idea of how anaphoric references are resolved will be given here, along with examples illustrating the range of the system's capabilities.

When a pronoun is found in the sentence (in the case analysis, not the ATN parse), it triggers a call to the function ANAPHORIC. ANAPHORIC takes four arguments:

1. A list of cases to look for.
2. A test that the referent must pass.
3. A number indicating how far back in the history to look.
4. The pronoun referenced.

The search is breadth first, in that the program tries very hard to find the referent in the earliest possible sentence. The test is an arbitrary form. SHOULD-BE and MUST-BE elements of the test are shut off on failure as they are in the rest of the back-up procedure. (See Chapter 4)

Say, for instance, that the system is given the sentence:

(5.1) He played the piano.

The call to AGENT would be the form:

(AGENT (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)))

Since the obvious candidate for the agent is a pronoun,

ANAPHORIC would be invoked. Its TEST would be:

(AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN))

So ANAPHORIC would look back through the parses and p-parses of the recent sentences which are kept as global variables, looking for a noun phrase that will pass this test. As it becomes more and more desperate it will make the test less strict. Since "he" is the pronoun, ANAPHORIC is smart enough to insist that the referent be male and animate.

Most pronoun references within a sentence itself can also be resolved. For instance:

(5.2) Fred went to London so he could visit the queen.

(5.3) Jack took you up in his airplane.

References to events and places can also be handled:

(5.4) It was unfortunate that the children were killed.

(5.5) I went to France. Fred lives there.

The resolution of locational references ("here" and "there") is a hard problem. By treating "there" as a pronoun whose referent must be a location, "there" is handled fairly well by the system. "Here" is a much more difficult problem, since its resolution is highly context dependant, and is hardly addressed at all.

There is still one problem with references within a

sentence, and that is that if the referent of a pronoun in the main clause is in a subordinate clause, then it will not be resolved. For instance, in the sentence:

(5.6) While he walked home, Fred was mugged.

"Fred" is not even considered as a candidate for "he". It is a simple extension to put this in (ANAPHORIC just has to look into subordinate clauses), but it has just not been done yet.

More difficult problems are illustrated by sentence (5.7).

(5.7) Mary was aboard the Titanic when she sank.

This sentence is ambiguous: Mary could have sunk in a swimming pool while she was on the Titanic, but this is probably not the intended meaning. If "to sink" is defined with a test like

(SHOULD-BE BOAT)

then the system will pick up "Titanic" correctly. Its first choice as a candidate is "Mary", however, so if the test does not rule "Mary" out, the system will choose her as its initial guess.

This perhaps illustrates a fundamental problem with the system's anaphoric routines. The first candidate found which passes the test is chosen, rather than all of the candidates being looked at, and the most likely accepted.

SECTION 5.2: Particles

There are many verbs in English which, when used with a particle, take on a very different meaning.

(5.6) Pick a girl.

(5.7) Pick up a girl.

(5.8) Pick a girl up.

(5.9) Pick out a girl.

The policy with particles in this system is to create a new verb by tacking the particle onto the end of the infinitive form of the verb. So (5.7) and (5.8) both become

(5.10) Pick-up a girl.

This conjuction takes place in the ATN. Each verb that takes one or more particles has a list of particles on its property list under the indicator PARTICLES. The verb "to look", for instance has (FOR AT LIKE) on its property list. (Look can take other particles: up, into, etc., but they are not currently handled by the system.) In the dictionary explicit entries are made for the verbs LOOK-FOR, LOOK-AT, and LOOK-LIKE as well as LOOK. This usually works well, but it sometimes leads to problems. Consider the sentence

(5.11) John looked for Mary.

This sentence is ambiguous. "Mary" could be the patient of the

verb "look-for", or the patient could be unspecified, and "for Mary" be the beneficiary case, or even the exchange case. The problem is that the ATN is purely syntactic, and it is not part of the back-up mechanism, so the system will always choose the "look-for" meaning.

Let me re-iterate that the system usually works correctly on particles, but in a more competent system the decision about conjoining particles to verbs will have to be put off until the semantics are at work.

SECTION 5.3: Ergative Verbs

There are problems with many of the cases in this system. In particular, the agent case is not properly defined.

(5.12) The door opened.

(5.13) The key opened the door.

(5.14) He opened the door with the key.

(5.15) The key opened.

If AGENT were called on sentence (5.12) it would return "the door" as the agent, whereas the agent is actually unspecified, and "the door" is the patient. Similarily, it would choose "the key" as the agent in (5.13). There is a whole set of verbs that behave like open. They are described in SECTION 1.2. Simmons calls them ergative verbs. This system handles them specially.

Basically, a routine, ERGATIVE, was written which uses AGENT as a routine for returning the first noun phrase of the sentence. By checking to see if it is animate ERGATIVE can tell if it is the agent as in (5.14) or the patient or instrument. If it is the agent the sentence offers no problems in this regard. If it is not the agent, then the system can decide between the instrument case and the patient by checking to see if there is a patient following the verb. Thus the system would interpret "the key" in sentence (5.15) as the patient. For more details look at the dictionary entry for "to open" or "to break" in APPENDIX IV and the ERGATIVE routine in APPENDIX V.

SECTION 5.4: Some Obvious Extensions

The purpose of this thesis was tc implement a fairly large case system, and to explore the use of case and the interaction between case, syntax, and semantics. In the time allotted fcr this work, then, the main concentration was on cases. Thus some fairly large gaps were left in parts of the system. It can cnly handle declarative sentences, for instance. This is nct because it would be incapable of handling questions and imperatives, but just because there were other things that were more challenging. Adding the ability to parse these other types of sentences wculd basically just involve extending the ATN tc transfcrm them into declarative sentences with the appropriate flags indicating type, etc. There are lots of ATN's around which do this fairly well.

Harder questions which were not addressed are those of conjunction, ellipses, and negation. Again, there are some ATNs which do mediocre jobs in these areas, but I dc nct advocate pushing these difficult questions into that supposedly simple and fast portion of the system. It is possible that negation and conjuction could be put into the ATN, and still have a fast system if the ATN just kept them as separate entities in its initial parse. That is, if it did not attempt to say what was being negated or conjoined, but left these decisions to the routines using case and semantics.

CHAPTER 6

Some women wait for Jesus
some women wait for Cain
so I hang upon my alter
and hoist my axe again

-L. Cohen

CONCLUSION

A sentence parser is a very small part of what is needed to have machine intelligence. Among a vast number of other things, the structures returned by a parser must be incorporated into a global structure which represents at least the information conveyed in the previous utterances, and some relevant world knowledge.

This thesis was not an attempt to design or build such a global structure, but thoughts in this direction are certainly relevant to the extension of this type of system. Also relevant is the question of the usefulness of case in such a structure.

It is clear that we want more than a list of case parses in a global structure. Schank is almost certainly on the right track with his conceptual dependency networks. Much of what we convey in an utterance, or in a series of utterances is never actually stated. Much is implied or inferred by the listener, and if a computer is to have any competence with language it must be able to determine these implications and inferences, and to make them explicit in its global structure. Some attempt has

been made in this system to make explicit some of the unstated information in the sentential parses.

What has been learned by the work on this system? The most glaring message is probably that using even the most simple syntactic parser before using semantics leads to problems. If I were to write this system again, I would not use an ATN as a pre-parser. One of the problems that my use of the ATN causes is illustrated in SECTION 5.2, where the parser is unable to undo an erroneous decision to classify a word as a particle, but more generally, I do not believe that one can decide syntactically how to break a sentence up into the correct components for a semantic analysis. This becomes more and more obvious as one pushes this system into more complex sentences involving nested relative and subordinate clauses.

I am not sure that Schank's fourteen ACTs form a good set of verbal primitives: they were not intuitive enough for me when I started this project, but my experience with this system leads me to strongly support his intent, at least, of carefully laying down a small set of primitives, and being very hesitant to change the set later.

No semantic primitives were set down at the outset of this project. New ones were merely created as they were needed. The set of cases was extended in a similar, though more controlled manner. Future work with the current system will probably be hampered by this unstructured approach to these difficult problems. Ptolemaic astronomers have been scorned for their naivity when they added epicycles upon epicycles in their attempts to model the solar system. We should therefore be very

suspicious of a natural language system that does the same thing with semantic primitives and cases.

We have seen that there are some problems with this system, both linguistic and computational, but I feel that it has met all of its objectives. The system employs case to handle a wide range of English sentences. It has illustrated problems with a certain approach, and we have seen ways around these difficulties. Finally, this system lends support to those of us who believe that case will prove to be a useful tool in machine understanding of language.

It is your world . . .
It is your flesh that I wear

BIBLIOGRAPHY

[1] Bach, Emmon. "Noun and Noun Phrases" Universals in Linguistic Theory. Editted by Bach and Harms. New York: Holt, Rinehart, and Winston, 1968

[2] Bruce, Bertram. Case Systems for Natural Language. Rutgers, New Jersey, 1974

[3] Celce-Murcia, M. "Paradigms for Sentence Recognition", in System Developement Corp. Final Report No. ERT-15092/7907.

[4] Chafe, Wallace L. Meaning and the Structure of Language. Chicago: The University of Chicago Press, 1970

[5] Charniak, Eugene. Towards a Model of Children's Story Comprehension. Cambridge, Massachusetts: MIT 1972. PhD dissertation

[6] Fillmore, Charles. "The Case for Case" Universals in Liguisitic Theory. Editted by Bach and Harms. New York: Holt, Rinehart, and Winston, 1968

[7] Jesperson, Otto. The Philosophy of Grammar. New York: W. W. Norton Co., 1965. (original pub. 1924)

[8] McCawley, James D. "The Role of Semantics in a Grammar" Universals in Linguistic Theory. Editted by Bach and Harms. New York: Holt, Rinehart, and Winston, 1968

[9] Martin, William A. "Translation of English into MAPL Using Winograd's Syntax, State Transition Networks, and a Semantic Case Grammar" Automatic Programming Group Internal Memo 11, MIT Project MAC, 1973

[10] Rosenbaum, Peter S. The Grammar for English Predicate Complement Construction Cambridge, Massachusetts: MIT Press 1967

[11] Schank, Roger C. "Conceptual Dependancy: A Theory of Natural Language Understanding" Cognitive Psychology 3 (1972) 552-631

[12] Schank, Roger C. "Identification of Conceptualizations Underlying Natural Language" Computer Models of Thought and Language. Editted by Schank and Colby. San Fransisco: W. H. Freeman and Co. 1973

[13] Simmons, R. F. "Semantic Networks: Their Computation and Use for Understanding English Sentences" Computer Models of Thought and Language. Editted by Schank and Colby. San Fransisco: W. H. Freeman and Co. 1973

[14] Winograd, Terry "A Procedural Model of Language Understanding" Computer Models of Thought and Language. Editted by Schank and Colby. San Fransisco: W. H. Freeman and Co. 1973

[15] Woods, W. A. "Transition Network Grammars for Natural Language Analysis" Communications of the ACM Vol 13 (Oct. 1970) 591-606

APPENDIX I: Some Examples

The following is a sample run of the system. Each parse is in two parts. First comes the ATN partial parse. This is printed out only to illustrate the notion of the partial parse. After that comes the final parse of the sentence. It always begins with the symbol <==>.

```
 * (LETS-GO)
 >    HELLO. YOU ARE NOW IN CONVERSATIONAL MODE.
 >    ANY LISP FORMS WILL BE EVALED IN THIS MODE.
 >    YOU MAY RETURN TO LISP BY ENTERING A #.
 >    IF YOU ENTER A @ AS A WORD IN A SENTENCE, THAT SENTENCE
 >                   WILL BE ABORTED
 >
 >
 * THE MAN WITH THE WIFE WHO IS BIGGER THAN HE GOES
 * TO VIENNA WITH A WOMAN WHO IS SMALLER THAN HE.
 >    PARSE:
 >    S
 >        NP NIL
 >            DET THE
 >            N MAN
 >                NUMBER SG
 >        PP NIL WITH
 >            NP NIL
 >                DET THE
 >                N WIFE
 >                    NUMBER SG
 >        REL NIL
 >            RELPRO WHO
 >            VP NIL
 >                TNS
 >                    PRESENT
 >                    VOICE ACTIVE
 >                V BE
 >            <-ADJ- NIL BIG
 >                COMP-SUP COMPARATIVE
 >        THAN-PH NIL
 >            NP NIL
 >                PRO HE
 >        VP NIL
 >            TNS
 >                PRESENT
 >                VOICE ACTIVE
 >            V GO
 >        PP NIL TO
```

```
>            NP NIL
>                NPR VIENNA
>        PP NIL WITH
>            NP NIL
>                DET A
>                N WOMAN
>                    NUMBER SG
>        REL NIL
>            RELPRO WHO
>            VP NIL
>                TNS
>                    PRESENT
>                    VOICE ACTIVE
>                V BE
>            <-ADJ- NIL SMALL
>                COMP-SUP COMPARATIVE
>        THAN-PH NIL
>            NP NIL
>                PRO HE
>    BY HE I ASSUME YOU MEAN THE MAN
>    IS THAT CORRECT?
* Y
>    <==>
>        N MAN
>            NUMBER SG
>            <-DEFINITE- THE
>            <-DESC- WITH
>                N WIFE
>                    NUMBER SG
>                    <-DEFINITE- THE
>                    <==>
>                        N WIFE
>                            NUMBER SG
>                            <-DEFINITE- THE
>                        PRESENT
>                        <-- HAVE-PROP
>                            BIG
>                                <-COMPARED-TO-
>                                    N MAN
>                                        NUMBER SG
>                                        <-DEFINITE- THE
>        PRESENT
>        <-- MOVE
>        <-SOURCE- SOMEPLACE
>        <-DESTINATION-
>            NPR VIENNA
>        <-CO-AGENT-
>            N WOMAN
>                NUMBER SG
>                <-INDEFINITE- A
>                <==>
>                    N WOMAN
>                        NUMBER SG
>                        <-INDEFINITE- A
>                    PRESENT
>                    <-- HAVE-PROP
```

```
>                          SMALL
>                              <-COMPARED-TO-
>                                  N MAN
>                                      NUMBER SG
>                                      <-DEFINITE- THE
>
>
* FRED LOVED THE OLD WOMAN BEFORE HE CAME TO CANADA.
>    PARSE:
>    S
>         NP NIL
>             NPR FRED
>         VP NIL
>             TNS
>                 PAST
>                 VOICE ACTIVE
>             V LOVE
>         NP NIL
>             DET THE
>             N WOMAN
>                 NUMBER SG
>                 <-ADJ- OLD
>         BEFORE NIL
>             NP NIL
>                 PRO HE
>             VP NIL
>                 TNS
>                     PAST
>                     VOICE ACTIVE
>                 V COME
>         PP NIL TO
>             NP NIL
>                 NPR CANADA
>    BY HE I ASSUME YOU MEAN FRED
>    IS THAT CORRECT?
* Y
>    <==>
>         NPR FRED
>         PAST
>         <-- HAVE-AFFECTION-FOR
>             <-ADV- MUCH
>             N WOMAN
>                 NUMBER SG
>                 <-ADJ- OLD
>                 <-DEFINITE- THE
>         <-TIME- BEFORE
>             <==>
>                 NPR FRED
>                 PAST
>                 <-- MOVE
>                 <-SOURCE- SOMEPLACE
>                 <-DESTINATION-
>                     NPR CANADA
>
>
* THERE IS A SMALL PEN IN THAT BOX.
```

```
>     PARSE:
>     S
>         <-ADV- NIL THERE
>         VP NIL
>             TNS
>                 PRESENT
>                 VOICE ACTIVE
>             V BE
>         NP NIL
>             DET A
>             N PEN
>                 NUMBER SG
>                 <-ADJ- SMALL
>         PP NIL IN
>             NP NIL
>                 DET THAT
>                 N BOX
>                     NUMBER SG
>     <==>
>         N PEN
>             NUMBER SG
>             <-ADJ- SMALL
>             <-INDEFINITE- A
>         PRESENT
>         <-- HAVE-LOC
>             IN
>                 N BOX
>                     NUMBER SG
>                     <-DEFINITE- THAT
>
>
* IT IS IDIOTIC THAT FRED WENT TO INDIA TO PLAY FOOTBALL.
>     PARSE:
>     S
>         NP NIL
>             PRO IT
>                 SUBJ
>                 OBJ
>                 NUMBER SG
>         VP NIL
>             TNS
>                 PRESENT
>                 VOICE ACTIVE
>             V BE
>         <-ADJ- NIL IDIOTIC
>         THAT-COMP NIL
>             <==>
>                 NPR FRED
>                 PAST
>                 <-- MOVE
>                 <-SOURCE- SOMEPLACE
>                 <-DESTINATION-
>                     NPR INDIA
>                 <-PURPOSE-
>                     <==>
>                         NPR FRED
```

```
>                           PRESENT
>                           <-- PLAY
>                               N FOOTBALL
>                                   NUMBER SG
>     <==>
>         <==>
>             NPR FRED
>             PAST
>             <-- MOVE
>             <-SOURCE- SOMEPLACE
>             <-DESTINATION-
>                 NPR INDIA
>             <-PURPOSE-
>                 <==>
>                     NPR FRED
>                     PRESENT
>                     <-- PLAY
>                         N FOOTBALL
>                             NUMBER SG
>         PRESENT
>         <-- HAVE-PROP
>             IDIOTIC
>
>
* THE PIANO WAS PLAYED IN THE EMPTY HALL.
>     PARSE:
>     S
>         NP NIL
>             DET THE
>             N PIANO
>                 NUMBER SG
>         VP NIL
>             TNS
>                 PAST
>                 VOICE PASSIVE
>             V PLAY
>         PP NIL IN
>             NP NIL
>                 DET THE
>                 N HALL
>                     NUMBER SG
>                     <-ADJ- EMPTY
>     <==> SOMEONE
>         PAST
>         <-- DO
>         <-CAUSE-
>             <==>
>                 N PIANO
>                     NUMBER SG
>                     <-DEFINITE- THE
>                 PAST
>                 <-- EMIT
>                     NP
>                         N SOUND
>         <-LOC- IN
>             N HALL
```

```
>                NUMBER SG
>                <-ADJ- EMPTY
>                <-DEFINITE- THE
>
>
* THE MAN WITH THE BIG HANDS PLAYED THE PIANO
* LIKE BEETHOVEN FOR MARY.
>    PARSE:
>    S
>         NP NIL
>             DET THE
>             N MAN
>                 NUMBER SG
>         PP NIL WITH
>             NP NIL
>                 DET THE
>                 N HAND
>                     NUMBER PL
>                     <-ADJ- BIG
>         VP NIL
>             TNS
>                 PAST
>                 VOICE ACTIVE
>             V PLAY
>         NP NIL
>             DET THE
>             N PIANO
>                 NUMBER SG
>         PP NIL LIKE
>             NP NIL
>                 NPR BEETHOVEN
>         PP NIL FOR
>             NP NIL
>                 NPR MARY
>    <==>
>         N MAN
>             NUMBER SG
>             <-DEFINITE- THE
>             <-DESC- WITH
>                 N HAND
>                     NUMBER PL
>                     <-ADJ- BIG
>                     <-DEFINITE- THE
>         PAST
>         <-- DO
>         <-CAUSE-
>             <==>
>                 N PIANO
>                     NUMBER SG
>                     <-DEFINITE- THE
>                 PAST
>                 <-- EMIT
>                     NP
>                         N SOUND
>         <-BENEFICIARY-
>             NPR MARY
```

```
>          <-LIKE-
>              NPR BEETHOVEN
>
>
* THE MAN IN THE HOUSE HAS HANDS.
>     PARSE:
>     S
>          NP NIL
>              DET THE
>              N MAN
>                  NUMBER SG
>          PP NIL IN
>              NP NIL
>                  DET THE
>                  N HOUSE
>                      NUMBER SG
>          VP NIL
>              TNS
>                  PRESENT
>                  VOICE ACTIVE
>              V HAVE
>          NP NIL
>              N HAND
>                  NUMBER PL
>     <==>
>          N MAN
>              NUMBER SG
>              <-DEFINITE- THE
>              <-LOC- IN
>                  N HOUSE
>                      NUMBER SG
>                      <-DEFINITE- THE
>          PRESENT
>          <-- POSSESS
>              N HAND
>                  NUMBER PL
>
>
* FRED WENT TO INDIA WITH JACK.
>     PARSE:
>     S
>          NP NIL
>              NPR FRED
>          VP NIL
>              TNS
>                  PAST
>                  VOICE ACTIVE
>              V GO
>          PP NIL TO
>              NP NIL
>                  NPR INDIA
>          PP NIL WITH
>              NP NIL
>                  NPR JACK
>     <==>
>          NPR FRED
```

```
>         PAST
>         <-- MOVE
>         <-SOURCE- SOMEPLACE
>         <-DESTINATION-
>             NPR INDIA
>         <-CO-AGENT-
>             NPR JACK
>
>
* FRED PLAYED JACK FOOTBALL.
>     PARSE:
>     S
>         NP NIL
>             NPR FRED
>         VP NIL
>             TNS
>                 PAST
>                 VOICE ACTIVE
>             V PLAY
>         NP NIL
>             NPR JACK
>         NP NIL
>             N FOOTBALL
>                 NUMBER SG
>     <==>
>         NPR FRED
>         PAST
>         <-- PLAY
>             N FOOTBALL
>                 NUMBER SG
>         <-CO-AGENT-
>             NPR JACK
>
>
* FRED PLAYED FOOTBALL WITH JACK.
>     PARSE:
>     S
>         NP NIL
>             NPR FRED
>         VP NIL
>             TNS
>                 PAST
>                 VOICE ACTIVE
>             V PLAY
>         NP NIL
>             N FOOTBALL
>                 NUMBER SG
>         PP NIL WITH
>             NP NIL
>                 NPR JACK
>     <==>
>         NPR FRED
>         PAST
>         <-- PLAY
>             N FOOTBALL
>                 NUMBER SG
```

```
>          <-CO-AGENT-
>              NPR JACK
>
>
* THE HOUSE WITH THE PIANO IN IT WAS GIVEN TO FRED BY HIS WIFE.
>     PARSE:
>     S
>          NP NIL
>              DET THE
>              N HOUSE
>                  NUMBER SG
>          PP NIL WITH
>              NP NIL
>                  DET THE
>                  N PIANO
>                      NUMBER SG
>          PP NIL IN
>              NP NIL
>                  PRO IT
>                      SUBJ
>                      OBJ
>                      NUMBER SG
>          VP NIL
>              TNS
>                  PAST
>                  VOICE PASSIVE
>              V GIVE
>          PP NIL TO
>              NP NIL
>                  NPR FRED
>          PP NIL BY
>              NP NIL
>                  DET
>                      POSSPRO HIS
>                  N WIFE
>                      NUMBER SG
>     BY HIS I ASSUME YOU MEAN FRED
>     IS THAT CORRECT?
* Y
>     <==>
>          N WIFE
>              NUMBER SG
>              <-POSS-BY- FRED
>          PAST
>          <-- TRANSFER
>              N HOUSE
>                  NUMBER SG
>                  <-DEFINITE- THE
>                  <==>
>                      N HOUSE
>                          NUMBER SG
>                          <-DEFINITE- THE
>                      PRESENT
>                          <-- CONTAIN
>                              N PIANO
>                                  NUMBER SG
```

```
>                                        <-DEFINITE- THE
>          <-RECIPIENT-
>              NPR FRED
>          <-SOURCE-
>              N WIFE
>                  NUMBER SG
>                  <-POSS-BY- FRED
>
>
* BEETHOVEN WROTE A PIANO SONATA IN VIENNA FOR FRED.
>     PARSE:
>     S
>          NP NIL
>              NPR BEETHOVEN
>          VP NIL
>              TNS
>                  PAST
>                  VOICE ACTIVE
>              V WRITE
>          NP NIL
>              DET
>                  DET A
>              N SONATA
>                  NUMBER SG
>                  <-CLASS-
>                      N PIANO
>                          NUMBER SG
>          PP NIL IN
>              NP NIL
>                  NPR VIENNA
>          PP NIL FOR
>              NP NIL
>                  NPR FRED
>     <==>
>          NPR BEETHOVEN
>          PAST
>          <-- COMPOSE
>              N SONATA
>                  NUMBER SG
>                  <-CLASS-
>                      N PIANO
>                          NUMBER SG
>          <-BENEFICIARY-
>              NPR FRED
>          <-LOC- IN
>              NPR VIENNA
>
>
* HE WROTE A SYMPHONY FOR HIM IN MUNICH.
>     PARSE:
>     S
>          NP NIL
>              PRO HE
>          VP NIL
>              TNS
>                  PAST
```

```
>                   VOICE ACTIVE
>               V WRITE
>           NP NIL
>               DET A
>               N SYMPHONY
>                   NUMBER SG
>           PP NIL FOR
>               NP NIL
>                   PRO HIM
>           PP NIL IN
>               NP NIL
>                   NPR MUNICH
>       BY HE I ASSUME YOU MEAN BEETHOVEN
>       IS THAT CORRECT?
* Y
>       BY HIM I ASSUME YOU MEAN FRED
>       IS THAT CORRECT?
* Y
>       <==>
>           NPR BEETHOVEN
>           PAST
>           <-- COMPOSE
>               N SYMPHONY
>                   NUMBER SG
>                   <-INDEFINITE- A
>           <-LOC- IN
>               NPR MUNICH
>           <-BENEFICIARY-
>               NPR FRED
>
>
* THE BIG MAN IN THE RED HOUSE PLAYS THE PIANO WELL.
>       PARSE:
>       S
>           NP NIL
>               DET THE
>               N MAN
>                   NUMBER SG
>                   <-ADJ- BIG
>           PP NIL IN
>               NP NIL
>                   DET THE
>                   N HOUSE
>                       NUMBER SG
>                       <-ADJ- RED
>           VP NIL
>               TNS
>                   PRESENT
>                   VOICE ACTIVE
>               V PLAY
>           NP NIL
>               DET THE
>               N PIANO
>                   NUMBER SG
>           <-ADV- NIL WELL
>       <==>
```

```
>         N MAN
>             NUMBER SG
>             <-ADJ- BIG
>             <-DEFINITE- THE
>             <-LOC- IN
>                 N HOUSE
>                     NUMBER SG
>                     <-ADJ- RED
>                     <-DEFINITE- THE
>         PRESENT
>         <-- DO
>         <-CAUSE-
>             <==>
>                 N PIANO
>                     NUMBER SG
>                     <-DEFINITE- THE
>                 PRESENT
>                 <-- EMIT
>                     NP
>                         N SOUND
>
>
>         <-ADV- WELL
* THE MUSIC PLAYED LOUDLY FROM THE SMALL ROOM.
>    PARSE:
>    S
>         NP NIL
>             DET THE
>             N MUSIC
>                 NUMBER SG
>         VP NIL
>             TNS
>                 PAST
>                 VOICE ACTIVE
>             V PLAY
>         <-ADV- NIL LOUD
>         PP NIL FROM
>             NP NIL
>                 DET THE
>                 N ROOM
>                     NUMBER SG
>                     <-ADJ- SMALL
>    <==> SOMEONE
>         PAST
>         <-- PLAY
>             N MUSIC
>                 NUMBER SG
>                 <-DEFINITE- THE
>         <-SOURCE-
>             N ROOM
>                 NUMBER SG
>                 <-ADJ- SMALL
>                 <-DEFINITE- THE
>         <-ADV- LOUD
>
>
```

```
* THE HAMMER BROKE THE WINDOW.
>    PARSE:
>    S
>         NP NIL
>              DET THE
>              N HAMMER
>                   NUMBER SG
>         VP NIL
>              TNS
>                   PAST
>                   VOICE ACTIVE
>              V BREAK
>         NP NIL
>              DET THE
>              N WINDOW
>                   NUMBER SG
>    <==> SOMEONE
>         PAST
>         <-- BREAK
>              N WINDOW
>                   NUMBER SG
>                   <-DEFINITE- THE
>         <-INSTRUMENT
>              N HAMMER
>                   NUMBER SG
>                   <-DEFINITE- THE
>
>
* THE DOOR OPENED.
>    PARSE:
>    S
>         NP NIL
>              DET THE
>              N DOOR
>                   NUMBER SG
>         VP NIL
>              TNS
>                   PAST
>                   VOICE ACTIVE
>              V OPEN
>    <==> SOMEONE
>         PAST
>         <-- OPEN
>              N DOOR
>                   NUMBER SG
>                   <-DEFINITE- THE
>
>
* THE KEY OPENED THE DOOR WITH EASE.
>    PARSE:
>    S
>         NP NIL
>              DET THE
>              N KEY
>                   NUMBER SG
>         VP NIL
```

```
>            TNS
>                PAST
>                VOICE ACTIVE
>            V OPEN
>        NP NIL
>            DET THE
>            N DOOR
>                NUMBER SG
>        PP NIL WITH
>            NP NIL
>                N EASE
>                    NUMBER SG
>    <==> SOMEONE
>        PAST
>        <-- OPEN
>            N DOOR
>                NUMBER SG
>                <-DEFINITE- THE
>        <-INSTRUMENT
>            N KEY
>                NUMBER SG
>                <-DEFINITE- THE
>        <-MANNER- WITH
>            N EASE
>                NUMBER SG
>
>
* I BROKE THE WINDOW WITH VENGENCE WITH A HAMMER.
>    PARSE:
>    S
>        NP NIL
>            PRO I
>                SUBJ
>                NUMBER SG
>        VP NIL
>            TNS
>                PAST
>                VOICE ACTIVE
>            V BREAK
>        NP NIL
>            DET THE
>            N WINDOW
>                NUMBER SG
>        PP NIL WITH
>            NP NIL
>                N VENGENCE
>                    NUMBER SG
>        PP NIL WITH
>            NP NIL
>                DET A
>                N HAMMER
>                    NUMBER SG
>    WHAT IS YOUR NAME?
* BROCK
>    <==>
>        NPR BROCK
```

```
>          PAST
>          <-- BREAK
>              N WINDOW
>                  NUMBER SG
>                  <-DEFINITE- THE
>          <-INSTRUMENT-
>              N HAMMER
>                  NUMBER SG
>                  <-INDEFINITE- A
>          <-MANNER- WITH
>              N VENGENCE
>                  NUMBER SG
```

APPENDIX II: Some Non-examples

In CHAPTER 2 many examples are given of phrases which fit into my set of cases. There are, however, many which do not. Here are a few of them.

I know from her smile that tonight will be fine.
The METHOD case expects a sentence after "from".

I listen all night for your step on the stair.
LISTEN-FOR is not defined in the system, although it could be.

Looking at the late late show through a semi-precious stone.
There is no DIRECTION or MEDIUM case.

I walked through the morning.
This is not really the PATH case, although that is what the system would call it.

. . . Between the ocean and your open vein.

. . . Between the hour and the age.

. . . Rusted from the elbow to the finger.
The type of structure in these three examples is not defined to the system.

I met a woman long ago.
The TIME case will not accept this.

I am turning into gold.
TURN-INTO is not defined.

You held onto me like I was a crucifix.
The LIKE case does not expect a sentence.

He looked once behind his shoulder.
This is another example of the non-existant DIRECTION case.

She died without a whisper.
The system does not know about "without". It is a difficult word.

APPENDIX III: A Formal Description of What the System Returns.

Symbols in upper-case are terminal symbols. Symbols in lower case are non-terminal symbols. A vertical bar, "|", means either, but not both, can be taken. A symbol in parentheses, "(. . .)", is optional. A symbol in square brackets with an asterisk following it, "[. . . .]*", may be repeated one or more times.

```
        s ::= <==> subj tns pred cases (adjp)
     subj ::= s | np | npc
       np ::= nh noun nmods
       nh ::= N | NPR
     noun ::= MAN | WOMAN | FRED | DOG | TREE | . . .
    nmods ::= (NUMBER nu) (det) (s) (adjp) (class)
              (quant) (desc) (location) (poss)
       nu ::= SG | PL
      det ::= def | indef
      def ::= <-DEFINATE- defdet
   defdet ::= THE | THAT | THIS | . . . .
    indef ::= <-INDEFINATE- indefdet
 indefdet ::= A | AN | ANY | . . . .
     adjp ::= [<-ADJ- adj (advp) (comp-sup) ]*
      adj ::= BIG | SMALL | RED | HAPPY | . . . .
     advp ::= <-ADV- adv (advp)
      adv ::= VERY | SUPRISINGLY | . . . .
 comp-sup ::= COMP-SUP cs
       cs ::= COMPARATIVE | SUPERLATIVE
    class ::= <-CLASS- cnp
      cnp ::= nh noun
    quant ::= QUANT qp
       qp ::= (intp) (ordp)
     intp ::= INTEGER int
      int ::= ONE | TWO | THREE | . . .
     ordp ::= ORDINAL ord (qp)
      ord ::= FIRST | SECOND | THIRD | . . . .
     desc ::= <-DESC- prep np
     prep ::= WITH | OF | BESIDE | ON | IN | UPON | . . .
     poss ::= <-POSS-BY- np
      npc ::= SOMEPLACE | SOMEONE | . . . .
      tns ::= PAST | PRESENT | FUTURE | . . .
     pred ::= <-- vp
       vp ::= vtp | vip | vcp | vl
      vtp ::= vt (advp) np
       vt ::= TRANSFER | PLAY | HEAR | INHABIT |
              SEARCH-FOR | EXAMINE | ORDER | . . .
      vip ::= vi (advp).
       vi ::= MOVE | DO | . . .
      vcp ::= vc adjmod
       vc ::= HAVE-PROP | APPEAR | . . .
   adjmod ::= adjp | adjmod1 | adj
  adjmod1 ::= adj (comp-sup) (than-ph)
  than-ph ::= <-COMPARED-TO- np
```

```
        vl ::= HAVE-LOC lprep np
     lprep ::= IN | ON| BY | NEAR | BESIDE | BEFORE | UNDER
     cases ::= location | time | duration | manner |
               destination | co-agent | instrument |
               exchange | beneficiary | recipient | topic |
               purpose | path | method | quantity |
               source | expected-ef | cause | enable |
               like | with
  location ::= <-LOC- lprep np
      time ::= <-TIME- tt
        tt ::= prep-time | noun-time | stime
 prep-time ::= tprep np
     tprep ::= IN | AT | BY | ABOUT | NEAR | BEFORE | AFTER |
               DURING
 noun-time ::= YESTERDAY | TODAY | TOMORROW | . . .
     stime ::= wp s
        wp ::= WHEN | WHILE | BEFORE | AFTER
  duration ::= <-DURATION- dprep np
     dprep ::= IN | FOR
    manner ::= <-MANNER- mprep np
     mprep ::= IN | WITH
destination ::= <-DESTINATION dtprep np
    dtprep ::= TO | INTO | ONTO | TOWARD | TOWARDS
       dsn ::= np | SOMEPLACE
  co-agent ::= CO-AGENT- np
instrument ::= <-INSTRUMENT- np
  exchange ::= <-EXCHANGE- np
beneficiary ::= <-BENEFICIARY- np
 recipient ::= <-RECIPIENT- np
     topic ::= <-TOPIC- np
   purpose ::= <-PURPOSE- s
      path ::= <-PATH- pathprep np
  pathprep ::= ACROSS | AROUND | DOWN | UP | ALONG | ABOUT | BY
    method ::= <-METHOD- s
  quantity ::= <-QUANTITY- BY np
    source ::= <-SOURCE- (FROM) dsn
expected-ef ::= <-EXPECTED-EFFECT- s
     cause ::= <-CAUSE- s
    enable ::= <-ENABLE- s
      like ::= <-LIKE- np
      with ::= <-WITH- np
```

APPENDIX IV: The Dictionary

Below is a copy of the dictionary currently used by the system. Chapters 3 and 4 briefly refer to certain parts of word definitions, but it is appropriate to give a fuller description of what appears in the dictionry.

Each dictionary entry is of the form

(WORD I1 P1 I2 P2 . . . Ii Pi)

where Pi is a property which is to be put on the property list of WORD under the indicator Ii. Different word types have different indicators on their property lists. If a word falls into more than one syntactic category it may have the properties of all of the categories. For instance "close" is a verb and an adjective, thus it would have the indicator V and the indicator ADJ. The only types of word whose dictionary entries will be described in full here are the two complex ones: nouns and verbs. If these are understood, the lesser word types will be easy to follow.

Nouns have the following properties:

N: This indicates that the WORD is a noun. The property under N indicates how plurals are formed. For instance:

(BOY N S)

(CLASS N ES)

```
(OX N IRR)
(OXEN N (OX (NUMBER PL)))
(FRED NPR NONE)
(WATER N MASS)
```

The entry for "ox" indicates that it forms its plural irregularily. The plural form must be in the dictionary as a separate word. NPR is a special case of N indicating that the noun is a proper noun. NONE indicates that there is no plural form of the noun. Mass nouns have the property MASS under N.

N-PROP: Under this indicator is a list of properties of the noun.

```
(BOY N S N-PROP (YOUNG))
```

indicates that a boy has the property YOUNG.

SUPERSET: The noun's super-sets are put in a list under this indicator:

```
(BOY N S N-PROP (YOUNG) SUPERSET (MAN))
(MAN N IRR N-PROP (MALE) SUPERSET (HUMAN))
(HUMAN N S N-PROP (ANIMATE) SUPERSET (ANIMAL))
```

the N-PROP and SUPERSET properties form a network whereby BOY, for instance, will be found to be animate through the

above chain of SUPERSETS and N-PROPS. To avoid a lot of searching, redundant information is often put in:

(BOY N S N-PROP (HUMAN MALE YOUNG)
SUPERSET (MAN HUMAN))

OBJ-SIZE: is the indicator whose property is a number from 0 - 10 giving a rough estimate of the size of an object. Some nouns such as JOY will not have this property.

NUM: This is used if a noun can have more than one number. For instance, the noun "paper" is defined as:

(PAPER N S N-PROP (INFLAMABLE LITERATURE)
SUPERSET (OBJECT) OBJ-SIZE 1 NUM (MASS))

The property under NUM indicates that paper can be a mass noun, as in "the room is full of paper."
"Football" has

NUM (MASS (SPORT GAME))

on its property list indicating that if football appears as a mass noun, it only has the N-PROPs SPORT and GAME from the whole set on its property list:

N-PROP (SPORT GAME OBJECT)

This is needed because the system only uses one set of properties to cover all of its meanings.

Verbs have the following indicators:

V: This is the verb indicator. Its property indicates how the first-person singular and the past tense are formed.

```
(KICK V S-ED)
(GIVE V IRR)
(GAVE V (GIVE (TNS PAST)))
(GIVING V (GIVE (PART PREPART)))
(GIVES V (GIVE (TNS PRESENT)))
```

Irregular verbs have IRR as the property under V. All of the tenses of an irregular verb must be listed separately in the dictionary, giving the infinitive form of the verb and the tense or whatever of the form in question.

PREP-CASE: This is for the foregrounding routines. It is a list of lists. Each sub-list starts with a prepostion, which is followed by a variable number of cases. These cases are always tried first and in the order in which they appear when the prepostion is encountered in the sentence with that verb.

V-MEAN: The procedural definition of the verb is the property of

V-MEAN. CHAPTER 3 gives all of the details.

TO-COMP: If the verb can take a to-complement, T is under this indicator.

THAT-COMP: If the verb can take a that-complement, T is under this indicator.

PARTICLES: If the verb takes one or more particles, they are listed here.

```
(A DET *)
(ABOUT PREP *)
(ACROSS PREP *)
(ADAGE N S OBJ-SIZE 3 NUM (MASS) N-PROP (MACHINE))
(AFRICA NPR NONE N-PROP (LOCATION) SUPERSET (CONTINENT) OBJ-SIZE 9)
(AFTER PREP *)
(AIRPLANE N S OBJ-SIZE 4 NUM (MASS) N-PROP (CONVEYANCE MACHINE))
(ALE N S N-PROP (LIQUID ALCOHOLIC) SUPERSET (BEVERAGE) NUM (MASS))
(ALL QUANTIFIER *)
(ALONG PREP *)
(AM V (BE (TNS PRESENT) (PNCODE "1SG")))
(AN DET *)
(AND CONJ *)
(ANGER N NONE N-PROP (ABSTRACT))
(ANIMAL N S SUPERSET (THING) OBJ-SIZE 3)
(ANNOY V
       S-ED
        PREP-CASE ((WITH INSTRUMENT))
       V-MEAN
       (IF ((AGENT (MUST-BE ANIMATE))
             AG
             (OPT (GETR PASSIVE) *SOMEONE)
             (PATIENT (MUST-BE ANIMATE))
             PA
             OBL)
           (BUILDQ ("<==>" ? "+" ("<--" ANNOY ?)) AG TNS PA)))
(ANYONE GENPRO * N-PROP (HUMAN))
(APPLE N S N-PROP (EATABLE) OBJ-SIZE 1 SUPERSET (FRUIT))
(ARE V (BE (TNS PRESENT)))
(ARM N S OBJ-SIZE 2)
(AROUND PREP *)
(AT PREP (AT ADJ) PARTICLE T)
(AT PREP *)
(ATE V (EAT (TNS PAST)))
(AUSTRALIA NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
(AUSTRIA NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
(AUTOMOBILE N S OBJ-SIZE 3 NUM (MASS) N-PROP (CONVEYANCE MACHINE))
(BACH NPR
      *
      SUPERSET
      (COMPOSER MAN)
      N-PROP
      (COMPOSER HUMAN MALE)
      DEFAULT
      ((PATIENT) (WRITE COMPOSE) (N MUSIC ("<-NUMBER-" SG)))
      OBJ-SIZE
      3)
(BAD ADJ IRR N NONE N-PROP (ABSTRACT))
(BADLY ADV *)
(BE V
    IRR
     PREP-CASE ((WITH WITH))
    THAT-COMP
    T
    V-MEAN
    (IF ((THERE-IS) TH OBL (AGENT T) AG OBL (BE-LOC1) LOC OBL)
        (BUILDQ ("<==>" ? "+" ("<--" HAVE-LOC "+")) AG TNS LOC)
        ((THERE-IS) TH OBL (AGENT T) AG OBL)
        (BUILDQ ("<==>" ? "+" ("<--" EXIST)) AG TNS)
        ((ADJ-LIST T) ADJ OBL (AGENT T) AG OBL)
        (BUILDQ ("<==>" ? "+" ("<--" HAVE-PROP "+")) AG TNS ADJ)
        ((PATIENT (MUST-BE THING)) PA OBL (AGENT (NOT (EQ (CAR TESTEE) 'THAT-COMP))) AG OBL)
        (BUILDQ ("<==>" ? "+" ("<--" # ?)) AG TNS (BE1? (GETR AG)) PA)
        ((COPULAR-ADV? (NOT (EQ (CAR TESTEE) 'THAT-COMP))) ADV OBL (AGENT T) AG OBL)
        (BUILDQ ("<==>" ? "+" ("<--" HAVE-LOC "+")) AG TNS ADV)
        ((CLAUSE '(FROM) '(MUST-BE COUNTRY CITY TOWN CONTINENT LOCATION) 'LOCATION)
          LOC
          OBL
          (AGENT (MUST-BE HUMAN))
          AG
          OBL)
        (BUILDQ ("<==>" ? PAST ("<--" INHABIT ?)) AG LOC)
        ((BE-LOC) LOC OBL (AGENT (NOT (EQ (CAR TESTEE) 'THAT-COMP))) AG OBL)
        (BUILDQ ("<==>" ? "+" ("<--" HAVE-LOC "+")) AG TNS LOC)
        ((AGENT (NOT (EQ (CAR TESTEE) 'THAT-COMP))) AG OBL)
        (BUILDQ ("<==>" ? "+" ("<--" EXIST)) AG TNS)))
(BEATRICE NPR * N-PROP (HUMAN FEMALE) SUPERSET (WOMAN) OBJ-SIZE 3)
(BEAUTY N NONE N-PROP (ABSTRACT))
(BEEF N MASS N-PROP (FOOD ANIMAL) OBJ-SIZE 1 SUPERSET (FOOD))
(BEER N S N-PROP (LIQUID ALCOHOLIC) SUPERSET (BEVERAGE) NUM (MASS))
(BEETHOVEN NPR
           *
           SUPERSET
           (COMPOSER MAN)
           N-PROP
           (COMPOSER HUMAN MALE)
           DEFAULT
           ((PATIENT) (WRITE COMPOSE) (N MUSIC ("<-NUMBER-" SG)))
           OBJ-SIZE
           3)
(BEFORE PREP (BEFORE ADV ADJ))
(BEHIND PREP (BEHIND ADV ADJ))
(BERKLEY NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
(BESIDE PREP *)
```

```
(BEST ADJ (GOOD (SUPERLATIVE)) ADV (WELL (SUPERLATIVE)))
(BETTER ADJ (GOOD (COMPARATIVE)) ADV (WELL (COMPARATIVE)))
(BIG ADJ ER-EST)
(BLUE ADJ R-ST)
(BOAT N S OBJ-SIZE 3 NUM (MASS) N-PROP (CONVEYANCE MACHINE))
(BOOK N S N-PROP (LITERATURE) SUPERSET (OBJECT) OBJ-SIZE 1)
(BOX N ES N-PROP (CONTAINER) OBJ-SIZE 2 SUPERSET (OBJECT))
(BRAHMS NPR
        *
        SUPERSET
        (COMPOSER MAN)
        N-PROP
        (COMPOSER HUMAN MALE)
        DEFAULT
        ((PATIENT) (WRITE COMPOSE) (N MUSIC ("<-NUMBER-" SG)))
        OBJ-SIZE
        3)
(BREAD N S N-PROP (EATABLE) SUPERSET (FOOD) OBJ-SIZE 1)
(BREAK V IRR PREP-CASE ((WITH INSTRUMENT)) TO-COMP T V-MEAN (ERGATIVE 'BREAK))
(BREAKING V (BREAK (PART PRESPART)))
(BREAKS V (BREAK (TNS PRESENT)))
(BROCK NPR * N-PROP (HUMAN MALE) SUPERSET (MAN) OBJ-SIZE 3)
(BROKE V (BREAK (TNS PAST)))
(BROKEN V (BREAK (PART PASTPART)))
(BOUGHT V (BUY (TNS PAST)(PART PASTPART)))
(BULL N S N-PROP (MALE BOVINE ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(BUSH N ES N-PROP (VEGETATION) OBJ-SIZE 3 SUPERSET (OBJECT))
(BUT CONJ *)
(BUY V
    IRR
     PREP-CASE ((WITH EXCHANGE))
    TO-COMP
    T
    V-MEAN
    (IF ((AGENT (MUST-BE HUMAN))
         AG
         (OPT '(GETR PASSIVE) 'SOMEONE)
         (PATIENT (NOT (SHOULD-BE HUMAN)))
         PA
         SOMETHING
         (PREP-CASE-LOOK '(EXCHANGE) '((FOR)) '(SHOULD-BE MONEY))
         EX
         MONEY
         (OR (PREP-CASE-LOOK '(EXCHANGE) '((FOR WITH)) '(SHOULD-BE HUMAN))
             ((IND-OBJ (SHOULD-BE HUMAN ORGANIZATION))))
         REC
         (GETR AG)
         (PREP-CASE-LOOK '(SOURCE) '((FROM)) '(SHOULD-BE HUMAN ORGANIZATION))
         SOU
         SOMEONE)
        (BUILDQ ("<==>" ?
                        "+"
                        ("<--" TRANSFER ?)
                        ("<-RECIPIENT-" ?)
                        ("<-SOURCE-" ?)
                        ("<-EXCHANGE-" ?))
                AG
                TNS
                PA
                REC
                SOU
                EX)))
(BUYING V (BUY (PART PRESPART)))
(BUYS V (BUY (TNS PRESENT)))
(BY PREP (BY ADV ADJ))
(CABBAGE N S N-PROP (EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
(CAKE N S N-PROP (FOOD) OBJ-SIZE 1 SUPERSET (FOOD))
(CAME V (COME (TNS PAST)))
(CAN MODAL *)
(CANADA NPR * N-PROP (PLACE COUNTRY LOCATION) OBJ-SIZE 8)
(CANDOR N NONE N-PROP (ABSTRACT))
(CANDY N ES N-PROP (FOOD SWEET) OBJ-SIZE 0 SUPERSET (FOOD))
(CAR N S OBJ-SIZE 3 NUM (MASS) N-PROP (CONVEYANCE MACHINE))
(CARROT N S N-PROP (ORANGE EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
(CAT N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(CHICKEN N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(CHOCOLATE N S N-PROP (FOOD SWEET) OBJ-SIZE 0 SUPERSET (FOOD))
(CIDER N S N-PROP (LIQUID ALCOHOLIC) SUPERSET (BEVERAGE) NUM (MASS))
(CLOSE ADJ R-ST V S-ED TO-COMP T V-MEAN (ERGATIVE 'CLOSE))
(COFFEE N S N-PROP (LIQUID) SUPERSET (BEVERAGE) NUM (MASS))
(COLOUR N S NUM (MASS))
(COME V
      IRR
      PREP-CASE
      ((TO DESTINATION PURPOSE) (WITH WITH) (AT TIME RATE))
      V-MEAN
      (IF ((AGENT (MUST-BE ANIMATE))
           AG
           OBL
           (PREP-CASE-LOOK '(DESTINATION) '((TO INTO ONTO TOWARDS TOWARD)) T)
           DEST
           HERE
           (PREP-CASE-LOOK '(SOURCE) '((FROM)) T)
           SRC
           (COND ((WHERE-IS (NOUN-LIST-GET (GETR AG))))
                 (T 'SOMEPLACE)))
          (BUILDQ ("<==>" ? "+" ("<--" MOVE) ("<-SOURCE-" ?) ("<-DESTINATION-" ?))
                  AG
                  TNS
                  SRC
```

```
                    DEST)))
(COMES V (COME (TNS PRESENT) (PNCODE "3SG")))
(COMPOSER N S SUPERSET (MAN MUSIC-MAKER) N-PROP (HUMAN MUSIC-MAKER) OBJ-SIZE 3)
(COMPUTER N S OBJ-SIZE 4 NUM (MASS) N-PROP (MACHINE))
(CONDUCTOR N S SUPERSET (MAN MUSIC-MAKER) N-PROP (HUMAN MUSIC-MAKER) OBJ-SIZE 3)
(CONTAIN V S-ED)
(CONTENTMENT N NONE N-PROP (ABSTRACT))
(CONTINENT N ES N-PROP (LOCATION PLACE) OBJ-SIZE 9)
(COUNTRY N ES N-PROP (LOCATION PLACE) OBJ-SIZE 8)
(COULD MODAL *)
(COW N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(CUP N S N-PROP (CONTAINER VESSEL) SUPERSET (UTENSIL) OBJ-SIZE 1)
(CYDNEY NPR * N-PROP (HUMAN FEMALE) SUPERSET (WOMAN) OBJ-SIZE 3)
(DAY N S N-PROP (TIME) NUM (MASS))
(DENMARK NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
(DESPITE PREP *)
(DID V (DO (TNS PAST)))
(DISCOMFORT N NONE N-PROP (ABSTRACT))
(DISCONTENTMENT N NONE N-PROP (ABSTRACT))
(DISLIKE TO-COMP
         T
         V
         S-D
         PREP-CASE
         ((FOR CAUSE))
         V-MEAN
         (IF ((AGENT (MUST-BE HUMAN)) AG OBL (OR (TO-COMP (GETR AG)) (PATIENT T)) PA OBL)
             (BUILDQ ("<==>" ? "+" ("<--" DISLIKE ?)) AG TNS PA)))
(DISSATISFACTION N NONE N-PROP (ABSTRACT))
(DO V IRR)
(DOG N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(DOING V (DO (TNS PRESENT) (PART PRESPART)))
(DONE V (DO (TNS PAST) (PART PASTPART)))
(DOOR N S SUPERSET (OBJECT) OBJ-SIZE 3)
(DOWN PART * PREP *)
(DRANK V (DRINK (TNS PAST)))
(DRINK V
       IRR
        PREP-CASE ((WITH INSTRUMENT))
       V-MEAN
       (IF ((AGENT (MUST-BE ANIMATE))
            AG
            (OPT '(GETR PASSIVE) 'SOMEONE)
            (PATIENT (MUST-BE LIQUID))
            PA
            (OPT '(NOT (GETR PASSIVE)) 'LIQUID)
            (PREP-CASE-LOOK '(SOURCE) '((FROM)) T)
            SOU
            GLASS)
           (BUILDQ ("<==>" ? "+" ("<--" TRANSFER ?) ("<-SOURCE-" ?) ("<-DESTINATION-" INTO ?))
                   AG
                   TNS
                   PA
                   SOU
                   AG)))
(DRINKING V (DRINK (PART PRESPART)))
(DRINKS V (DRINK (TNS PRESENT)))
(DRUNK V (DRINK (PART PASTPART)) N S N-PROP (HUMAN) SUPERSET (MAN WOMAN))
(DURING PREP *)
(EACH QUANTIFIER *)
(EARLY ADV * ADV-PROP (TIME))
(EAT V
     IRR
      PREP-CASE ((WITH INSTRUMENT))
     V-MEAN
     (IF ((AGENT (MUST-BE ANIMATE))
          AG
          (OPT '(GETR PASSIVE) 'SOMEONE)
          (PATIENT (SHOULD-BE FOOD))
          PA
          (OPT '(NOT (GETR PASSIVE)) 'FOOD))
         (BUILDQ ("<==>" ?
                         "+"
                         ("<--" TRANSFER ?)
                         ("<-SOURCE-" SOMEPLACE)
                         ("<-DESTINATION-" INTO ?))
                 AG
                 TNS
                 PA
                 AG)))
(EATEN V (EAT (PART PASTPART)))
(EATING V (EAT (PART PRESPART)))
(EATS V (EAT (TNS PRESENT) (PNCODE "3SG")))
(EIGHT NUMBER *)
(EIGHTH ORDINAL EIGHT ADV *)
(ELECTRIC ADJ *)
(EMPTY ADJ
       ER-EST
        PREP-CASE ((WITH INSTRUMENT))
       V
       ES-ED
       V-MEAN
       (IF ((AGENT (MUST-BE ANIMATE))
            AG
            (OPT (GETR PASSIVE) 'SOME-ONE)
            (PATIENT (MUST-BE CONTAINER PLACE LOCATION))
            PA
            OBL
            (SEC-SUBJ (OR (GREATERP (SIZE (GETR PA)) (SIZE TESTEE))))
            SEC
```

```
301                (BUILDQ (CONTENTS-OF "+") PA))
302               (BUILDQ ("<==>" ? "+" ("<--" TRANSFER #) ("<-SOURCE-" ?) ("<-DESTINATION-" #))
303                       AG
304                       TNS
305                       (COND ((EQ 'CONTENTS-OF (CAR (GETR SEC))) (GETR SEC))
306                             (T (NOUN-PUT SEC)))
307                       PA
308                       (BUILDQ (OUT-OF ?) PA))))
309     (EMPTY V ES-ED ADJ *)
310     (ENGLAND NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
311     (EVERY QUANTIFIER *)
312     (FAR ADV *)
313     (FAST ADV ER-EST)
314     (FAT ADJ ER-EST)
315     (FIFTH ORDINAL FIVE ADV *)
316     (FIRST ORDINAL ONE ADV *)
317     (FISH N ES N-PROP (ANIMATE) SUPERSET (ANIMAL) NUM (MASS) OBJ-SIZE 2)
318     (FIVE NUMBER *)
319     (FOOD N MASS N-PROP (EATABLE) SUPERSET (THING) OBJ-SIZE 1)
320     (FOOTBALL N S N-PROP (GAME SPORT OBJECT) OBJ-SIZE 1 NUM (MASS (SPORT GAME)))
321     (FOR PREP (FOR ADJ) PARTICLE T)
322     (FORK N S SUPERSET (UTENSIL) OBJ-SIZE 1)
323     (FOUR NUMBER *)
324     (FOURTH ORDINAL FOUR ADV *)
325     (FRANCE NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
326     (FRED NPR
327           *
328           N-PROP
329           (HUMAN MALE SPORTS-MAN)
330           DEFAULT
331           ((PATIENT) (PLAY) (N SPORT ("<-NUMBER-" SG)))
332           SUPERSET
333           (MAN)
334           OBJ-SIZE
335           3)
336     (FROM PREP *)
337     (FRUIT N S N-PROP (VEGETATION FOOD) SUPERSET (FOOD) OBJ-SIZE 1)
338     (FUNNY ADJ ER-EST)
339     (FURTHER ADV (FAR (COMPARATIVE)))
340     (FURTHEST ADV (FAR (SUPERLATIVE)))
341     (GAME N S N-PROP (SPORT))
342     (GAVE V (GIVE (TNS PAST)))
343     (GERMANY NPR * N-PROP (PLACE COUNTRY LOCATION) OBJ-SIZE 8)
344     (GERTRUDE NPR * N-PROP (HUMAN FEMALE) SUPERSET (WOMAN) OBJ-SIZE 3)
345     (GIVE V
346           IRR
347           FEATURES
348           (TRANS INDOBJ)
349           PARTICLES
350           (UP IN OUT)
351           PREP-CASE
352           ((TO RECIPIENT) (FROM SOURCE))
353           V-MEAN
354           (COND ((AND (EQ 'GIVE (CADR (START-LIST 'V SENTENCE)))
355                       (CASE ((AGENT (MUST-BE ANIMATE))
356                              AG
357                              (OPT (GETR PASSIVE) 'SOMEONE)
358                              (OR (PREP-CASE-LOOK '(RECIPIENT) '((TO)) '(MUST-BE ANIMATE))
359                                  (IND-OBJ (MUST-BE ANIMATE)))
360                              REC
361                              'SOMEONE
362                              (PATIENT (NOT (SHOULD-BE ANIMATE)))
363                              PA
364                              OBL))
365                       (SETQ STRUCT (BUILDQ ("<==>" ?
366                                                    "+"
367                                                    ("<--" TRANSFER ?)
368                                                    ("<-RECIPIENT-" ?)
369                                                    ("<-SOURCE-" ?))
370                                            AG
371                                            TNS
372                                            PA
373                                            REC
374                                            AG))
375                       (!AIL-ENDS))
376                  STRUCT)))
377     (GIVE-UP V
378              IRR
379              PREP-CASE
380              ((TO RECIPIENT) (FROM SOURCE))
381              V-MEAN
382                        (IF ((AGENT (MUST-BE ANIMATE))
383                             AG
384                             OBL
385                             (PATIENT (NOT (MUST-BE ABSTRACT)))
386                             PA
387                             OBL
388                             (PREP-CASE-LOOK '(RECIPIENT) '((TO)) '(MUST-BE ANIMATE))
389                             REC
390                             'SOMEONE)
391                            (BUILDQ ("<==>" ?
392                                                "+"
393                                                ("<--" TRANSFER ?)
394                                                ("<-MANNER-" UNWILLINGLY)
395                                                ("<-SOURCE-" ?)
396                                                ("<-RECIPIENT-" ?))
397                                        AG
398                                        TNS
399                                        PA
400                                        AG
```

```
401                                        REC)
402                              ((AGENT (MUST-BE ANIMATE)) AG OBL)
403                             (BUILDQ ("<==>" ? "+" ("<--" SURRENDER)) AG TNS)))
404     (GIVEN V (GIVE (PART PASTPART)))
405     (GIVES V (GIVE (TNS PRESENT) (PNCODE "3SG")))
406     (GLASS N ES N-PROP (CONTAINER VESSEL) SUPERSET (UTENSIL) OBJ-SIZE 1)
407     (GLEE N NONE N-PROP (ABSTRACT))
408     (GO V
409         IRR
410         PREP-CASE
411         ((TO DESTINATION PURPOSE) (AT RATE TIME)(WITH WITH))
412         V-MEAN
413         (IF ((AGENT (MUST-BE ANIMATE))
414               AG
415               OBL
416               (PREP-CASE-LOOK '(DESTINATION) '((TO INTO ONTO TOWARDS TOWARD)) T)
417               DEST
418               SOMEPLACE
419               (PREP-CASE-LOOK '(SOURCE) '((FROM)) T)
420               SRC
421               (COND ((WHERE-IS (NOUN-LIST-GET (GETR AG))))
422                     (T 'SOMEPLACE)))
423             (BUILDQ ("<==>" ? "+" ("<--" MOVE) ("<-SOURCE-" ?) ("<-DESTINATION-" ?))
424                     AG
425                     TNS
426                     SRC
427                     DEST)))
428     (GO PART * V IRR FEATURES (INTRANS))
429     (GOES V (GO (TNS PRESENT) (PNCODE "3SG")))
430     (GOING V (GO (PART PRESPART)))
431     (GONE V (GO PASTPART))
432     (GOOD ADJ IRR N S N-PROP (ABSTRACT THING) NUM (MASS (ABSTRACT)))
433     (GRAPE N S N-PROP (EATABLE) OBJ-SIZE 0 SUPERSET (FRUIT))
434     (GRAPEFRUIT N S N-PROP (EATABLE) OBJ-SIZE 1 SUPERSET (FRUIT))
435     (GREEN ADJ ER-EST)
436     (GUITAR N S N-PROP (MUSICAL-INSTRUMENT) OBJ-SIZE 2 NUM (MASS))
437     (HAD V (HAVE (TNS PAST) (PART PASTPART)))
438     (HALL N S N-PROP (BUILDING PLACE LOCATION CONTAINER) SUPERSET (BUILDING THING) OBJ-SIZE 4)
439     (HAMMER N S N-PROP (TOOL INSTRUMENT) SUPERSET (TOOL OBJECT))
440     (HAND N S OBJ-SIZE 1)
441     (HAPPINESS N NONE N-PROP (ABSTRACT))
442     (HAPPY ADJ ER-EST)
443     (HAS V (HAVE (TNS PRESENT)))
444     (HATE TO-COMP
445           T
446           V
447           S-D
448           PREP-CASE
449           ((FOR CAUSE))
450           V-MEAN
451           (IF ((AGENT (MUST-BE HUMAN)) AG OBL (OR (TO-COMP (GETR AG)) (PATIENT T)) PA OBL)
452               (BUILDQ ("<==>" ? "+" ("<--" DISLIKE ("<-ADV-" MUCH) ?)) AG TNS PA)))
453     (HAVE V
454           IRR
455           V-MEAN
456           (IF ((AGENT (AND (NOT (MUST-BE ABSTRACT)) (SHOULD-BE ANIMATE)))
457                 AG
458                 OBL
459                 (PATIENT T)
460                 PA
461                 OBL)
462               (BUILDQ ("<==>" ? "+" ("<--" POSSESS ?)) AG TNS PA)))
463     (HAVING V (HAVE (PART PRESPART)))
464     (HE PRO * PERSONAL T)
465     (HEAR V
466           IRR
467            PREP-CASE ((WITH INSTRUMENT))
468           THAT-COMP
469           T
470           PREP-CASE
471           ((OF TOPIC))
472           V-MEAN
473           (IF ((PATIENT (MUST-BE MUSIC MUSIC-MAKER))
474                 PA
475                 OBL
476                 (AGENT (MUST-BE ANIMATE))
477                 AG
478                 (OPT (GETR PASSIVE) 'SOMEONE))
479               (BUILDQ ("<==>" ? "+" ("<--" HEAR ?)) AG TNS PA)))
480     (HEARD V (HEAR (TNS PAST) (PART PASTPART)))
481     (HELEN NPR * N-PROP (HUMAN FEMALE) SUPERSET (WOMAN) OBJ-SIZE 3)
482     (HELGA NPR * N-PROP (HUMAN FEMALE) SUPERSET (WOMAN) OBJ-SIZE 3)
483     (HER DET (HER (POSSPRO)) PRO (HER (POSS) (OBJ) (NUMBER SG)) PERSONAL T)
484     (HERE ADV * ADV-PROP (LOCATION))
485     (HIM PRO * PERSONAL T)
486     (HIS DET (HIS (POSSPRO)) PERSONAL T)
487     (HIXON NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
488     (HOLLAND NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
489     (HORSE N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
490     (HOUR N S N-PROP (UNIT TIME))
491     (HOUSE N S N-PROP (PLACE LOCATION CONTAINER) SUPERSET (BUILDING THING) OBJ-SIZE 4)
492     (HOW QDET *)
493     (HUMAN N S SUPERSET (ANIMAL) N-PROP (ANIMATE) OBJ-SIZE 3)
494     (HUSBAND N S OBJ-SIZE 3 N-PROP (HUMAN MALE) SUPERSET (MAN))
495     (I PRO (I (SUBJ) (NUMBER SG)) PERSONAL T)
496     (IDIOT N S N-PROP (ANIMATE) OBJ-SIZE 3)
497     (IDIOTIC ADJ *)
498     (IN PREP (IN ADV ADJ) PARTICLE T)
499     (INDIA NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
500     (INNSBROOK NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
```

```
(INTO PREP (INTO ADV))
(IS V (BE (TNS PRESENT) (PNCODE "3SG")))
(IT PRO (IT (SUBJ) (OBJ) (NUMBER SG)))
(IVES NPR
      *
      SUPERSET
      (COMPOSER MAN)
      N-PROP
      (COMPOSER HUMAN MALE)
      DEFAULT
      ((PATIENT) (WRITE COMPOSE) (N MUSIC ("<-NUMBER-" SG)))
      OBJ-SIZE
      3)
(JACK NPR
      *
      N-PROP
      (HUMAN MALE MUSICIAN)
      SUPERSET
      (MAN)
      DEFAULT
      ((PATIENT (PLAY) (N MUSICAL-INSTRUMENT)))
      OBJ-SIZE
      3)
(JAGUAR N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(JOAN NPR * N-PROP (HUMAN FEMALE) SUPERSET (WOMAN) OBJ-SIZE 3)
(JOHN NPR * N-PROP (HUMAN MALE))
(JOY N NONE N-PROP (ABSTRACT))
(JUICE N S N-PROP (LIQUID) SUPERSET (BEVERAGE) NUM (MASS))
(KEY N S N-PROP (INSTRUMENT) SUPERSET (OBJECT) OBJ-SIZE 1)
(KNIFE N IRR OBJ-SIZE 1 SUPERSET (UTENSIL))
(LAKE N S N-PROP (CONVEYOR WATER) SUPERSET (THING) OBJ-SIZE 6 NUM (MASS))
(LARGE ADJ R-ST)
(LAST ORDINAL LAST ADV *)
(LATE ADV * ADV-PROP (TIME))
(LEAST * *)
(LEAVE V
       IRR
       PREP-CASE
       ((TO PURPOSE) (AT RATE TIME) (IN MANNER) (BY METHOD) (FOR DURATION) (WITH WITH))
       V-MEAN
       (IF ((AGENT (MUST-BE ANIMATE))
            AG
            (OPT (GETR PASSIVE) 'SOME-ONE)
            (PATIENT (MUST-BE PLACE BUILDING LOCATION))
            PA
            (COND ((WHERE-IS (GETR AG)))
                  (T 'SOMEPLACE)))
           (BUILDQ ("<==>" ? "+" ("<--" MOVE) ("<-SOURCE-" #) ("<-DESTINATION-" #))
                   AG
                   TNS
                   (LEAVE-SOU (GETR PA))
                   (LEAVE-DEST (GETR PA)))))
(LEAVES V (LEAVE (TNS PRESENT)))
(LEAVING V (LEAVE (PART PRESPART)))
(LEFT V (LEAVE (TNS PAST) (PART PASTPART)))
(LEMON N S N-PROP (EATABLE) OBJ-SIZE 1 SUPERSET (FRUIT))
(LEOPARD N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(LETTER N S N-PROP (LITERATURE) OBJ-SIZE 1)
(LETTUCE N S N-PROP (EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
(LIKE PREP
      *
PARTICLE T
       TO-COMP
       T
       V
       S-D
       PREP-CASE
       ((FOR CAUSE))
       V-MEAN
       (IF ((AGENT (MUST-BE HUMAN)) AG OBL (OR (TO-COMP (GETR AG)) (PATIENT T)) PA OBL)
           (BUILDQ ("<==>" ? "+" ("<--" HAVE-AFFECTION-FOR ?)) AG TNS PA)))
(LION N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(LISTEN V
         S-ED
         PREP-CASE
         ((TO TOPIC))
         V-MEAN
         (IF ((AGENT (MUST-BE ANIMATE))
              AG
              (OPT (GETR PASSIVE) 'SOMEONE)
              (TOPIC-LOOK (MUST-BE NOISE HUMAN MUSIC-MAKER THING))
              TOPIC
              (OR (ANAPHORIC '(TOPIC PATIENT)
                             '(MUST-BE NOISE HUMAN MUSIC-MAKER THING)
                             (COND ((EQ ANAPHOR-DEPTH 0) (SETQ ANAPHOR-DEPTH 1))
                                   (T ANAPHOR-DEPTH))
                             NIL)
                  'CPT))
             (BUILDQ ("<==>" ? "+" ("<--" LISTEN ("<-TOPIC-" #)))
                     AG
                     TNS
                     (LISTEN-TOPIC? (GETR TOPIC)))))
(LIVE V
       S-D
       PREP-CASE ((WITH WITH))
       V-MEAN
       (IF ((AGENT (MUST-BE ANIMATE)) AG OBL (PREP-CASE-LOOK '(LOCATION) '((IN AT)) T) L OPT)
           (BUILDQ ("<==>" ? "+" ("<--" INHABIT ?)) AG TNS L)
           ((AGENT (MUST-BE ANIMATE))
            AG
```

```
601                OBL
602                (PREP-CASE-LOOK '(METHOD) '((ON)) '(SHOULD-BE FOOD))
603                PA
604                OBL)
605              (BUILDQ ("<==>" ?
606                             "+"
607                             ("<--" EXIST)
608                             ("<-METHOD-" ("<==>" ?
609                                                  "+"
610                                                  ("<--" TRANSFER ?)
611                                                  ("<-SOURCE-" SOMEPLACE)
612                                                  ("<-DESTINATION-" INTO ?))))
613                      AG
614                      TNS
615                      AG
616                      TNS
617                      PA
618                      AG)
619              ((AGENT (MUST-BE ANIMATE)) AG OBL)
620              (BUILDQ ("<==>" ? "+" ("<--" EXIST)) AG TNS)))
621     (LONDON NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
622     (LOOK V
623           S-ED
624     PARTICLES (FOR AT LIKE)
625           PREP-CASE ((WITH INSTRUMENT))
626           V-MEAN
627           (IF ((AGENT (MUST-BE ANIMATE)) AG OBL (ADJ-LIST T) ADJ OBL)
628               (BUILDQ ("<==>" ? "+" ("<--" APPEAR "+")) AG TNS ADJ)
629               ((AGENT (MUST-BE ANIMATE))
630                 AG
631                 OBL
632                 (PREP-CASE-LOOK '(DIRECTION)
633                                 '((ACROSS INTO OUT UP ONTO TOWARD TOWARDS ALONG DOWN AROUND UNDER))
634                                 T)
635                 DIR
636                 OPT)
637               (BUILDQ (a ("<==>") (?) ("+") (("<--" LOOK)) #)
638                      AG
639                      TNS
640                      (COND ((NULL (GETR DIR)) NIL)
641                            (T (GETR DIR))))))
642     (LOOK-AT V S-ED PREP-CASE ((WITH INSTRUMENT))
643               V-MEAN (IF ((AGENT (MUST-BE ANIMATE))
644                           AG (OPT (GETR PASSIVE) 'SOMEONE)
645                           (PATIENT T) PA OBL)
646                          (BUILDQ ("<==>" ? + ("<--" EXAMINE ?)
647                                              ("<-INSTRUMENT-" (N EYE (NUMBER PL)("<-POSS-BY-" ?))))
648                                  AG TNS PA AG)))
649     (LOOK-FOR V S-ED PREP-CASE ((WITH INSTRUMENT))
650               V-MEAN (IF ((AGENT (MUST-BE ANIMATE))
651                           AG (OPT (GETR PASSIVE) 'SOMEONE)
652                           (PATIENT T) PA OBL)
653                          (BUILDQ ("<==>" ? + ("<--" SEARCH-FOR ?)) AG TNS PA)))
654     (LOOK-LIKE V S-ED
655               V-MEAN (IF ((AGENT (MUST-BE ANIMATE))
656                           AG OBL
657                           (PATIENT T) PA OBL)
658                          (BUILDQ ("<==>" ? + ("<--" HAVE-APPEARANCE-OF ?)) AG TNS PA)))
659     (LOUD ADJ ER-EST)
660     (LOVE TO-COMP
661           T
662           V
663           S-D
664           PREP-CASE
665           ((FOR CAUSE))
666           V-MEAN
667           (IF ((AGENT (MUST-BE HUMAN)) AG OBL (OR (TO-COMP (GETR AG)) (PATIENT T)) PA OBL)
668               (BUILDQ ("<==>" ? "+" ("<--" HAVE-AFFECTION-FOR ("<-ADV-" MUCH) ?)) AG TNS PA)))
669     (MAN N IRR SUPERSET (ANIMAL) N-PROP (ANIMATE HUMAN MALE) OBJ-SIZE 3)
670     (MARY NPR * N-PROP (HUMAN FEMALE) SUPERSET (WOMAN) OBJ-SIZE 3)
671     (MEAT N MASS N-PROP (FOOD ANIMAL) OBJ-SIZE 1 SUPERSET (FOOD))
672     (MELBOURNE NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
673     (MEN N (MAN (NUMBER PL)))
674     (MEXICO NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
675     (MONEY N MASS OBJ-SIZE 0)
676     (MORE ADV * COMP *)
677     (MOST * *)
678     (MOST ADV *)
679     (MUG N S N-PROP (CONTAINER VESSEL) SUPERSET (UTENSIL) OBJ-SIZE 1)
680     (MUNIC NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
681     (MUNICH NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
682     (MUSHROOM N S N-PROP (EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
683     (MUSIC N NONE SUPERSET (SOUND) N-PROP (MUSIC SOUND))
684     (MY DET (MY (POSSPRC)) PERSONAL T)
685     (NEAR PREP *)
686     (NEW ADJ ER-EST)
687     (NEWPAPER N S N-PROP (LITERATURE) SUPERSET (PAPER) OBJ-SIZE 1)
688     (NEXT ORDINAL NEXT ADV *)
689     (NIGHT N S N-PROP (TIME) NUM (MASS))
690     (NINE NUMBER *)
691     (NINETH ORDINAL NINE ADV *)
692     (NO NEG *)
693     (NOISE N S)
694     (NONE NEG *)
695     (NOT NEG *)
696     (NOTE N S N-PROP (LITERATURE) OBJ-SIZE 1)
697     (NOVEL N S N-PROP (LITERATURE) SUPERSET (BOOK) OBJ-SIZE 1)
698     (OBJECT N S)
699     (OCEAN N S N-PROP (CONVEYOR WATER) SUPERSET (THING) OBJ-SIZE 9 NUM (MASS))
700     (OF PREP (OF ADJ))
```

```
(OLD ADJ ER-EST)
(ON PREP (ON ADV ADJ))
(ONE NUMBER * PRO *)
(ONION N S N-PROP (EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
(ONTO PREP (ONTO ADV))
(OPEN V S-ED TO-COMP T V-MEAN (ERGATIVE 'OPEN))
(OR CONJ *)
(ORANGE N S N-PROP (EATABLE) OBJ-SIZE 1 SUPERSET (FRUIT))
(ORCHESTRA N S SUPERSET (MUSIC-MAKER OBJ-SIZE 4) N-PROP (MUSIC-MAKER))
(ORDER V
       S-ED
      PREP-CASE ((WITH WITH))
       V-MEAN
       (IF ((AGENT (MUST-BE HUMAN))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (PATIENT (MUST-BE ANIMATE))
            PA
            OBL
            (TO-COMP (GETR PA))
            TOC
            OBL)
           (BUILDQ ("<==>" ? "+" ("<--" ORDER "+")) AG TNS TOC)
           ((AGENT (MUST-BE HUMAN))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (PATIENT (AND (MUST-BE THING) (NOT (MUST-BE HUMAN))))
            PA
            OBL)
           (BUILDQ ("<==>" ? "+" ("<--" ORDER ?)) AG TNS PA)))
(OTTAWA NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
(OWN V
     S-ED
     V-MEAN
     (IF ((AGENT (AND (NOT (MUST-BE ABSTRACT)) (SHOULD-BE ANIMATE)))
          AG
          OBL
          (PATIENT T)
          PA
          OBL)
         (BUILDQ ("<==>" ? "+" ("<--" POSSESS ?)) AG TNS PA)))
(PAPER N S N-PROP (THIN FLAMABLE LIGHT LITERATURE) SUPERSET (OBJECT) OBJ-SIZE 1 NUM (MASS))
(PEACH N ES N-PROP (EATABLE) OBJ-SIZE 1 SUPERSET (FRUIT))
(PEANUT N S OBJ-SIZE 0)
(PEAR N S N-PROP (EATABLE) OBJ-SIZE 1 SUPERSET (FRUIT))
(PEN N S N-PROP (WRITING-INSTRUMENT) SUPERSET (OBJECT) OBJ-SIZE 1)
(PENCIL N S N-PROP (WRITING-INSTRUMENT) SUPERSET (OBJECT) OBJ-SIZE 1)
(PIANO N S N-PROP (MUSICAL-INSTRUMENT) OBJ-SIZE 3 NUM (MASS))
(PIE N S N-PROP (FOOD) OBJ-SIZE 1 SUPERSET (FOOD))
(PIG N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(PLANE N S OBJ-SIZE 4 NUM (MASS) N-PROP (CONVEYANCE MACHINE))
(PLAY V
     S-ED
     V-MEAN
     (IF ((AGENT (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)))
          AG
          (OPT (GETR PASSIVE) 'SOMEONE)
          (PATIENT (MUST-BE MUSICAL-INSTRUMENT))
          PA
          (COND ((AND (NOT FAIL-TEST) (DEFAULT 'PATIENT (NOUN-GET (GETR AG))))
                 (T 'OBL)))
          (BUILDQ ("<==>" ? "+" ("<--" DO) ("<-CAUSE-" ("<==>" ? + ("<--" EMIT (NP (N SOUND))))))
                  AG
                  TNS
                  PA
                  TNS)
          ((AGENT (AND (SHOULD-BE MUSICIAN) (MUST-BE HUMAN)))
           AG
           (OPT (GETR PASSIVE) 'SOMEONE)
           (PATIENT (MUST-BE MUSIC))
           PA
           OBL)
          (BUILDQ ("<==>" ? "+" ("<--" PLAY ?)) AG TNS PA)
          ((AGENT (AND (MUST-BE HUMAN) (SHOULD-BE SPORTS-MAN)))
           AG
           (OPT (GETR PASSIVE) 'SOMEONE)
           (IND-OBJ (MUST-BE HUMAN))
           CO-A
           OPT
           (PATIENT (MUST-BE SPORT))
           PA
           (COND ((NOT FAIL-TEST) (DEFAULT 'PATIENT (NOUN-GET (GETR AG))))
                 (T 'OBL)))
          (BUILDQ (@ ("<==>") (?) ("+") (("<--" PLAY ?)) #)
                  AG
                  TNS
                  PA
                  (PROG (TEMP)
                        (RETURN (COND ((SETQ TEMP (GETR CO-A))
                                       (LIST (LIST '"<-CO-AGENT-" (SOFT-NOUN-LIST-GET (NP-BUILD TEMP))))))))))
          ((AGENT (MUST-BE ANIMATE)) AG OBL)
          (BUILDQ ("<==>" ?
                          "+"
                          ("<--" DO)
                          ("<-CAUSE-" ("<==>" ? "+" ("<--" HAVE-PROP ENTERTAINED))))
                  AG
                  TNS
                  AG
                  TNS)
          ((AGENT (MUST-BE MUSIC)) PA OBL)
```

```
            (BUILDQ ("<=*>" SOMEONE + ("<--" PLAY ?)) TNS PA)))
(PLAY N S N-PROP (LITERATURE) SUPERSET (BOOK) OBJ-SIZE 1)
(PLUM N S N-PROP (EATABLE) OBJ-SIZE 1 SUPERSET (FRUIT))
(POP N S N-PROP (LIQUID) SUPERSET (BEVERAGE) NUM (MASS))
(PORTUGAL NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
(POSSESS V
         ES-ED
         V-MEAN
         (IF ((AGENT (AND (NOT (MUST-BE ABSTRACT)) (SHOULD-BE ANIMATE)))
              AG
              OBL
              (PATIENT T)
              PA
              OBL)
             (BUILDQ ("<==>" ? "+" ("<--" POSSESS ?)) AG TNS PA)
             ((PATIENT (AND (SHOULD-BE HUMAN) (MUST-BE ANIMATE))) PA OBL (AGENT T) AG DEVIL)
             (BUILDQ ("<==>" ? "+" ("<--" CONTROL (N MIND ("<-POSS-BY-" ?)))) AG TNS PA)))
(POTATO N S N-PROP (EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
(POUND N S OBJ-SIZE 0 N-PROP (QUANTITY))
(PRAGUE NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
(QUIET ADJ ER-EST)
(QUILL N S N-PROP (ANIMAL-PART WRITING-INSTRUMENT) SUPERSET (OBJECT) OBJ-SIZE 1)
(RADISH N ES N-PROP (EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
(RAN V (RUN (TNS PAST) (PART PASTPART)))
(RAY NPR * N-PROP (HUMAN MALE) SUPERSET (MAN) OBJ-SIZE 3)
(RED ADJ ER-EST)
(RICHARD NPR * N-PROP (HUMAN MALE) SUPERSET (MAN) OBJ-SIZE 3)
(RIVER N S N-PROP (CONVEYOR WATER) SUPERSET (THING) OBJ-SIZE 5 NUM (MASS))
(ROAD N S N-PROP (CONVEYOR) SUPERSET (THING) OBJ-SIZE 5 NUM (MASS))
(ROOF N S OBJ-SIZE 3)
(ROOM N S N-PROP (PLACE LOCATION CONTAINER) SUPERSET (BUILDING THING) OBJ-SIZE 4)
(ROOSTER N S N-PROP (MALE ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
(RUN V
     IRR
      PREP-CASE ((WITH WITH))
     FEATURES
     (TRANS INTRANS)
     PREP-CASE
     ((BY PATH METHOD))
     V-MEAN
     (IF ((AGENT (MUST-BE ANIMATE))
          AG
          OBL
          (PREP-CASE-LOOK '(DESTINATION) '((TO INTO ONTO TOWARD TOWARDS)) T)
          DEST
          SOMEPLACE
          (PREP-CASE-LOOK '(SOURCE) '((FROM)) T)
          SRC
          (RUN-SOURCE?))
         (BUILDQ ("<==>" ?
                         "+"
                         ("<--" MOVE ("<-ADV-" QUICKLY))
                         ("<-MANNER-" BY (N FOOT (NUMBER PL)))
                         ("<-SOURCE-" ?)
                         ("<-DESTINATION-" ?))
                 AG
                 TNS
                 SOU
                 DEST)
         ((AGENT (MUST-BE MACHINE)) AG OBL)
         (BUILDQ ("<==>" SOMEONE
                         "+"
                         ("<--" DO)
                         ("<-CAUSE-" ("<==>" ? "+" (@ ("<--" FUNCTION) #))))
                 TNS
                 AG
                 TNS
                 (START-LIST '"<-ADV-" (START-LIST 'VP SENTENCE)))))
(RUNNING V (RUN (PART PRESPART)))
(RUNS V (RUN (TNS PRESENT) (PNCODE "3SG")))
(RUSSIA NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
(SADNESS N NONE N-PROP (ABSTRACT))
(SALLY NPR * N-PROP (HUMAN FEMALE) SUPERSET (WOMAN) OBJ-SIZE 3)
(SALZBURG NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
(SATISFACTION N NONE N-PROP (ABSTRACT))
(SCHOOL N S N-PROP (PLACE LOCATION SCHOOL) OBJ-SIZE 5 SUPERSET (THING))
(SECOND ORDINAL TWO ADV *)
(SEE V
     IRR
      PREP-CASE ((WITH INSTRUMENT))
     V-MEAN
     (IF ((AGENT (MUST-BE ANIMATE)) AG (OPT '(GETR PASSIVE) 'SOMEONE) (PATIENT T) PA OBL)
         (BUILDQ ("<==>" # "+" ("<--" SEE #)) (NOUN-PUT AG) TNS (NOUN-PUT PA))))
(SELL V
      IRR
      TO-COMP
      T
      V-MEAN
      (IF ((AGENT (MUST-BE HUMAN))
           AG
           (OPT '(GETR PASSIVE) 'SOMEONE)
           (PATIENT (NOT (SHOULD-BE HUMAN)))
           PA
           SOMETHING
           (PREP-CASE-LOOK '(EXCHANGE) '((FOR)) '(SHOULD-BE-MONEY))
           EX
           MONEY
           (OR (IND-OBJ (SHOULD-BE HUMAN ORGANIZATION))
               (PREP-CASE-LOOK '(RECIPIENT) '((TO)) '(SHOULD-BE HUMAN ORGANIZATION)))
           REC
```

```
              SOMEONE
              (PREP-CASE-LOOK '(BENEFICIARY) '((FOR)) '(SHOULD-BE HUMAN ORGANIZATION))
              BEN
              (GETR AG))
            (BUILDQ ("<==>" ?
                            "+"
                            ("<--" TRANSFER ?)
                            ("<-SOURCE-" ?)
                            ("<-RECIPIENT-" ?)
                            ("<-EXCHANGE-" ?))
                    AG
                    TNS
                    PA
                    BEN
                    REC
                    EX)))
(SELLING V (SELL (PART PRESPART)))
(SELLS V (SELL (TNS PRESENT)))
(SEVEN NUMBER *)
(SEVENTH ORDINAL SEVEN ADV *)
(SHALL MODAL *)
(SHRUB N S N-PROP (VEGETATION) OBJ-SIZE 3 SUPERSET (OBJECT))
(SIDE N S NUM (MASS))
(SILENT ADJ ER-EST)
(SIX NUMBER *)
(SIXTH ORDINAL SIX ADV *)
(SLOW ADV ER-EST)
(SMALL ADJ ER-EST)
(SO PREP *)
(SOFT ADJ ER-EST)
(SOLD V (SELL (TNS PAST) (PART PASTPART)))
(SOME QUANTIFIER *)
(SOMEONE GENPRO * N-PROP (HUMAN))
(SONATA N S SUPERSET (MUSIC) N-PROP (MUSIC))
(SOON ADV * ADV-PROP (TIME))
(SOUND N S SUPERSET (NOISE) N-PROP (NOISE))
(SPAIN NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
(SPOON N S SUPERSET (UTENSIL) OBJ-SIZE 1)
(STACK N S OBJ-SIZE 3)
(STEAL V
       IRR
        PREP-CASE ((WITH WITH))
       TO-COMP
       T
       V-MEAN
       (IF ((AGENT (SHOULD-BE ANIMATE)) AG OBL
            (PREP-CASE-LOOK '(DESTINATION PATH) '((INTO TO ONTO TOWARD TOWARDS)
                                                  (BY ABOUT BESIDE ALONG UP DOWN AROUND ACROSS))
                            '(AND (NOT (MUST-BE HUMAN)) (SHOULD-BE PLACE LOCATION)))
           DEST OBL)
           (BUILDQ ("<==>" ? "+" ("<--" MOVE ("<-ADV-" STEALTHILY)) +) AG TNS DEST)
           ((AGENT (MUST-BE HUMAN))
             AG
             (OPT '(GETR PASSIVE) 'SOMEONE)
             (PATIENT (NOT (SHOULD-BE HUMAN ABSTRACT)))
             PA
             SOMETHING
             (OR (IND-OBJ (SHOULD-BE HUMAN))
                 (PREP-CASE-LOOK '(RECIPIENT) '((FOR)) '(SHOULD-BE HUMAN ORGANIZATION)))
             REC
             (GETR AG)
             (PREP-CASE-LOOK '(SOURCE) '((FROM)) '(SHOULD-BE HUMAN ORGANIZATION PLACE))
             SOU
             SOMEPLACE)
           (BUILDQ ("<==>" ?
                           "+"
                           ("<--" TRANSFER ?)
                           ("<-SOURCE-" ?)
                           ("<-RECIPIENT-" ?)
                           ("<-EXCHANGE-" NOTHING))
                   AG
                   TNS
                   PA
                   SOU
                   REC)))
(STEALING V (STEAL (PART PRESPART)))
(STEALS V (STEAL (TNS PRESENT)))
(STOLE V (STEAL (TNS PAST)))
(STOLEN V (STEAL (PART PASTPART)))
(STORY N ES N-PROP (LITERATURE) SUPERSET (BOOK) OBJ-SIZE 1)
(STRANGE ADJ ER-EST)
(STUPID ADJ ER-EST)
(SUPPORT N S OBJ-SIZE 3)
(SUPRISINGLY ADV *)
(SYMPHONY N S SUPERSET (MUSIC) N-PROP (MUSIC))
(TABLE N S OBJ-SIZE 3)
(TAKE V
       IRR
       FEATURES
       (TRANS)
       PREP-CASE
       ((TO RECIPIENT) (FROM SOURCE) (BY MANNER) (WITH INSTRUMENT) (INTO DESTINATION))
       V-MEAN
       (IF ((AGENT (MUST-BE ANIMATE))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (PATIENT T)
            PA
            OBL
            (PREP-CASE-LOOK '(RECIPIENT) '((TO)) '(MUST-BE HUMAN))
```

```
                REC
                OBL
                (PREP-CASE-LOOK '(SOURCE) '((FROM)) '(MUST-BE PLACE LOCATION HUMAN CONTAINER))
                SOU
                (COND ((WHERE-IS (GETR PA)))
                      (T 'SOMEPLACE)))
              (BUILDQ ("<==>" ? "+" ("<--" TRANSFER ?) ("<-SOURCE-" ?) ("<-RECIPIENT-" ?))
                      AG
                      TNS
                      PA
                      SOU
                      REC)
              ((AGENT (MUST-BE ANIMATE))
                AG
                OBL
                (PATIENT T)
                PA
                OBL
                (PREP-CASE-LOOK '(SOURCE) '((FROM)) '(MUST-BE HUMAN CONTAINER PLACE LOCATION))
                SOU
                (COND ((WHERE-IS (GETR PA)))
                      (T 'SOMEPLACE))
                (NOT (PREP-CASE-LOOK '(RECIPIENT) '((TO)) '(MUST-BE HUMAN)))
                REC
                OBL)
              (BUILDQ ("<==>" ? "+" ("<--" TRANSFER ?) ("<-SOURCE-" ?) ("<-RECIPIENT-" ?))
                      AG
                      TNS
                      PA
                      SCU
                      AG)
              ((AGENT (MUST-BE ANIMATE))
                AG
                (OPT (GETR PASSIVE) 'SOMEONE)
                (PATIENT T)
                PA
                OBL
                (PREP-CASE-LOOK '(DESTINATION)
                                '((TO TOWARD TOWARDS NEAR BESIDE INTO ONTO))
                                '(MUST-BE PLACE LOCATION))
                DEST
                OBL
                (PREP-CASE-LOOK '(SOURCE) '((FROM)) T)
                SOU
                (COND ((WHERE-IS (GETR PA)))
                      (T 'SOMEPLACE)))
              (BUILDQ ("<==>" ? "+" ("<--" TRANSFER ?) ("<-SOURCE-" ?) ("<-DESTINATION-" ?))
                      AG
                      TNS
                      PA
                      SCU
                      DEST)))
    (TAKEN V (TAKE (PART PASTPART)))
    (TAKES V (TAKE (TNS PRESENT) (PNCODE "3SG")))
    (TAKING V (TAKE (PART PRESPART)))
    (TEA N S N-PROP (LIQUID) SUPERSET (BEVERAGE) NUM (MASS))
    (TEN NUMBER *)
    (TENTH ORDINAL TEN ADV *)
    (TERRY NPR * N-PROP (HUMAN MALE) SUPERSET (MAN) OBJ-SIZE 3)
    (THAN * *)
    (THAT DET * PRO (THAT (SUBJ) (OBJ) (NUMBER SG)))
    (THE DET *)
    (THEM PRO (THAT (SUBJ) (OBJ) (NUMBER PL)))
    (THERE ADV * ADV-PROP (LOCATION))
    (THESE PRO (THIS (SUBJ) (OBJ) (NUMBER PL)))
    (THING N S)
    (THIRD ORDINAL THREE ADV *)
    (THIS PRO (THIS (SUBJ) (OBJ) (NUMBER SG)))
    (THOSE PRO (THAT (SUBJ) (OBJ) (NUMBER PL)))
    (THREE NUMBER *)
    (TIGER N S N-PROP (ANIMATE) SUPERSET (ANIMAL) OBJ-SIZE 3)
    (TO PREP (TO ADV))
    (TO PREP *)
    (TODAY ADV * ADV-PROP (TIME))
    (TOM NPR * N-PROP (HUMAN MALE) SUPERSET (MAN) OBJ-SIZE 3)
    (TOMATO N ES N-PROP (EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
    (TOMORROW ADV * ADV-PROP (TIME))
    (TOOK V (TAKE (TNS PAST)))
    (TOOL N S SUPERSET (OBJECT) OBJ-SIZE 1)
    (TOP N S NUM (MASS))
    (TORONTO NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
    (TOWARD PREP *)
    (TOWARDS PREP *)
    (TRAIN N S OBJ-SIZE 4 NUM (MASS) N-PROP (CONVEYANCE MACHINE))
    (TREE N S N-PROP (VEGETATION) OBJ-SIZE 4 SUPERSET (OBJECT))
    (TROMBONE N S N-PROP (MUSICAL-INSTRUMENT) OBJ-SIZE 2 NUM (MASS))
    (TURNIP N S N-PROP (EATABLE) SUPERSET (VEGETABLE) OBJ-SIZE 1)
    (TWO NUMBER *)
    (UBC N NONE N-PROP (UNIVERSITY LOCATION) OBJ-SIZE 5 SUPERSET (UNIVERSITY))
    (UGLINESS N NONE N-PROP (ABSTRACT))
    (UNCOVER V S-ED)
    (UNDER PREP (UNDER ADV ADJ))
    (UNDER PREP *)
    (UNEASINESS N NONE N-PROP (ABSTRACT))
    (UNHAPPINESS N NONE N-PROP (ABSTRACT))
    (UNIVERSITY N ES N-PROP (PLACE LOCATION SCHOOL) OBJ-SIZE 5 SUPERSET (SCHOOL))
    (UP PART *)
    (UP PREP * PARTICLE T)
    (UPON PREP *)
```

```
(USA NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
(USSR NPR NONE N-PROP (LOCATION) SUPERSET (COUNTRY) OBJ-SIZE 8)
(UTENSIL N S SUPERSET (TOOL) OBJ-SIZE 1)
(VANCOUVER NPR NONE N-PROP (LOCATION) SUPERSET (CITY) OBJ-SIZE 6)
(VENGENCE N NONE N-PROP (ABSTRACT))
(VERY ADV *)
(VIENNA NPR * N-PROP (PLACE COUNTRY LOCATION) OBJ-SIZE 6)
(VINCE NPR * N-PROP (HUMAN MALE) SUPERSET (MAN) OBJ-SIZE 3)
(VINE N S N-PROP (VEGETATION) OBJ-SIZE 3 SUPERSET (OBJECT))
(WANT TO-COMP
      T
      THAT-COMP
      T
      V
      S-ED
      V-MEAN
      (IF ((AGENT (MUST-BE ANIMATE))
           AG
           (OPT (GETR PASSIVE) 'SOMEONE)
           (PATIENT (MUST-BE THING))
           PA
           OBL)
          (BUILDQ ("<==>" SOMEONE
                          FUTURE
                          ("<--" TRANSFER ?)
                          ("<-SOURCE-" #)
                          (RECIPIENT ?)
                          ("<-CAUSE-" ("<==>" ? FUTURE ("<--" HAVE-PROP PLEASED))))
                  PA
                  (COND ((NOUN-LIST-GET (WHERE-IS (GETR PA))))
                        (T 'SOMEPLACE))
                  AG
                  AG)
          ((AGENT (MUST-BE ANIMATE))
           AG
           (OPT (GETR PASSIVE) 'SOMEONE)
           (TO-COMP (GETR AG))
           PA
           OBL)
          (BUILDQ ("<==>" "+" ("<-CAUSE-" ("<==>" ? FUTURE ("<--" HAVE-PROP PLEASED)))) PA AG)
          ((AGENT (MUST-BE ANIMATE))
           AG
           (OPT (GETR PASSIVE) 'SOMEONE)
           (PATIENT (MUST-BE THING))
           PA
           OBL
           (TO-COMP (GETR PA))
           TOC
           OBL)
          (BUILDQ ("<==>" "+" ("<-CAUSE-" ("<==>" ? FUTURE ("<--" HAVE-PROP PLEASED))))
                  TOC
                  AG)))
(WAS V (BE (TNS PAST)))
(WELL ADV IRR)
(WENT V (GO (TNS PAST)))
(WHAT QDET *)
(WHERE QDET * ADJ *)
(WHICH QDET * RELPRO T)
(WHILE PREP *)
(WHO QDET * RELPRO T)
(WHY QDET *)
(WIFE N IRR OBJ-SIZE 3 N-PROP (FEMALE HUMAN) SUPERSET (WOMAN))
(WILL MODAL *)
(WINDOW N S N-PROP (PLACE LOCATION) SUPERSET (OBJECT) OBJ-SIZE 2)
(WINE N S N-PROP (LIQUID ALCOHOLIC) SUPERSET (BEVERAGE) NUM (MASS))
(WIT N S)
(WITH PREP (WITH ADV ADJ))
(WOMAN N IRR SUPERSET (ANIMAL) N-PROP (ANIMATE HUMAN FEMALE) OBJ-SIZE 3)
(WOMEN N (WOMAN (NUMBER PL)))
(WORLD N S N-PROP (PLACE LOCATION EARTH) SUPERSET (PLANET OBJECT) OBJ-SIZE 10)
(WORSE ADJ (BAD (COMPARATIVE)))
(WORST ADJ (BAD (SUPERLATIVE)))
(WOULD MODAL *)
(WRITE THAT-COMP
       T
       TO-COMP
       T
       PREP-CASE
       ((OF TOPIC)(WITH INSTRUMENT))
       V
       IRR
       V-MEAN
       (IF ((AGENT (AND (MUST-BE HUMAN) (SHOULD-BE COMPOSER)))
            AG
            (OPT (GETR PASSIVE) 'SOMEONE)
            (PATIENT (MUST-BE MUSIC LITERATURE))
            PA
            (OR (DEFAULT 'PATIENT (NOUN-GET (GETR AG)))
                '(NP NIL (DET A) (N LETTER (NUMBER SG))))
            (IND-OBJ (MUST-BE HUMAN))
            IND
            OPT)
           (BUILDQ (@ ("<==>") (?) ("+") (("<--" # ?)) #)
                   AG
                   INS
                   (WRITE? (GETR PA))
                   PA
                   (WRITE-IND-OBJ (GETR IND) (GETR PA)))))
(WRITES V (WRITE (TNS PRESENT)))
(WRITING V (WRITE (TNS PRESENT) (PART PRESPART)))
```

```
200     (WRITTEN V (WRITE (PART PASTPART)))
201     (WROTE V (WRITE (TNS PAST)))
202     (YEAR N S N-PROP (TIME) SUPERSET (TIME))
203     (YELLOW ADJ ER-EST)
204     (YESTERDAY ADV * ADV-PROP (TIME))
205     (YOU PRO (YOU (SUBJ) (OBJ) (NUMBER SG-PL)))
206     (YOUNG ADJ ER-EST)
207     (YOUR PRO (YOU (NUMBER SG-PL) (POSSESIVE)) PERSONAL T)
208     (YOUTH N S N-PROP (HUMAN TIME) SUPERSET (TIME MAN WOMAN) OBJ-SIZE 3 NUM (MASS))
209     ("1805" N NONE N-PREP (YEAR TIME))
210     ("1974" N NONE N-PROP (YEAR TIME))
211     ("3270" N S OBJ-SIZE 3 NUM (MASS) N-PROP (MACHINE))
212     NIL
```

APPENDIX V: The Code

The code for this system is presented here in four parts. The first is the main part of the system: the case-analysis code. The routines are in alphabetic order.

The next part is the ATN grammar. This starts on page 129.

The third part is on page 137, and is just a few auxiliary routines.

Finaly is listed the ATN parser which was written by Dr. Raymond Reiter at UBC.

```
(DEFUN ADJ-LIST NEXPR (TEST)
;  THIS IS A CASE OF THE VERB "TO BE". IT HANDLES:
;  "HE IS BIG." AND "HE IS SMALLER THAN FRED."
  (PROG (TESTEE K-S FT RET)
        (SETQ K-S KEEP-SENT)
      LOOP
        (COND ((NULL K-S)
               (COND (FT (SETQ FAIL-TEST T FAIL-TEST-LIST (APPEND (LIST 'ADJ-LIST) FAIL-TEST-LIST))))
               (RETURN NIL))
              ((AND (EQ (CAAR K-S) '"<-ADJ-")
                    (NOT (CADAR K-S))
                    (SETQ TESTEE (CAR K-S))
                    (SETQ FT T)
                    (EVAL TEST))
               (FLAG-ON TESTEE KEEP-SENT T)
               (COND ((AND (EQ (CAAR (SETQ K-S (CDR K-S))) 'THAN-PH)
                           (EQ 'COMPARATIVE (CADR (START-LIST 'COMP-SUP TESTEE))))
                      (DELETE '(COMP-SUP COMPARATIVE) TESTEE)
                      (RPLACD (END-LIST TESTEE)
                              (LIST (LIST '"<-COMPARED-TO-"
                                          (NOUN-LIST-GET (COPY (SUPER-NP-BUILD (COPY (CAR K-S))))))))
                      (FLAG-ON (CAR K-S) KEEP-SENT T)))
               (RETURN (CDDR TESTEE)))
              (T (SETQ K-S (CDR K-S)) (GO LOOP)))))


(DEFUN ADJTYPE NEXPR (Z)
  (EQ (GET NEW-WORD 'ADJ) Z))


(DEFUN ADV-MOD? (PART TEST KASE)
;  THIS WORKS OUT ADVERBIAL MODIFICATIONS:
;  "HE CAME YESTERDAY." "HE WENT THERE."
  (PROG (TEMP)
        (COND ((AND (MEMBER 'TIME (SETQ TEMP (GET (CADDR PART) 'ADV-PROP))) (EVAL TEST))
               (SETQ FN '(RPLACD (END-LIST (FIND-LIST STRUCT '"<==>")) (LIST STRCT))
                     MOD (PHRASE-NUM 'VP KEEP-SENT))
               (LIST '"<-TIME-" (CADDR PART)))
              ((MEMBER 'LOCATION TEMP) (ADV-LOC PART TEST KASE))
              (T (SETQ FN '(RPLACD (END-LIST (FIND-LIST STRUCT '"<==>")) (LIST STRCT))
                       MOD (PHRASE-NUM 'VP KEEP-SENT))
                 (CONS (CAR (COPY PART)) (CDDR PART))))))


(DEFUN ADV-LOC (PART TEST KASE)
;  THIS ROUTINE TRIES TO RESOLVE REFERENCES INVOLVING "HERE" AND "THERE".
  (PROG (TEMP AD RET)
        (COND ((NOT (MEMBER (CADDR PART) '(THERE HERE))) (RETURN NIL)))
        (COND ((ZEROP ANAPHOR-DEPTH) (SETQ ANAPHOR-DEPTH 1)))
        (SETQ AD ANAPHOR-DEPTH)
      LOOP
        (COND ((AND (NOT (NULL KASE)) (SETQ TEMP (CASE-ANAPHOR KASE PART TEST)))
               (FLAG-ON PART KEEP-SENT (SETQ RET (LIST (MAKE-ARROW KASE) (NOUN-LIST-GET TEMP))))
               (SET-UP-FN KASE NIL)
               (RETURN RET))
              ((SETQ TEMP (ANAPHORIC 'LOCATION TEST ANAPHOR-DEPTH (CDR PART)))
               (FLAG-ON PART
                        KEEP-SENT
                        (SETQ RET (COND ((GET (CAR TEMP) 'PREP)
                                         (LIST '"<-LOC-" (CAR TEMP) (NOUN-LIST-GET TEMP)))
                                        (T (LIST '"<-LOC-" TEMP)))))
               (SETQ ANAPHOR-DEPTH AD
                     FN '(RPLACD (END-LIST (FIND-LIST STRUCT '"<==>")) (LIST STRCT))
                     NUM (PHRASE-NUM 'VP KEEP-SENT))
               (RETURN RET))
              ((LESSP ANAPHOR-DEPTH 4) (SETQ ANAPHOR-DEPTH (ADD1 ANAPHOR-DEPTH)) (GO LOOP))
              (T (PRIN1 '"I CANNOT RESOLVE YOUR REFERENCE TO")
                 (PRIN1 (CADDR PART))
                 (TERPRI)
                 (PRINT '"I SHALL CONTINUE ANYWAY.")
                 (FLAG-ON PART KEEP-SENT (SETQ RET (LIST '"<-LOC-" 'SOMEPLACE)))
                 (SETQ ANAPHOR-DEPTH AD
                       FN '(RPLACD (END-LIST (FIND-LIST STRUCT '"<==>")) (LIST STRCT))
                       MOD (PHRASE-NUM 'VP KEEP-SENT))
                 (RETURN RET))))))
```

```
(DEFUN AFTER-AND-MOD? (PREP NN)
  (LESSP (PHRASE-NUM 'VP KEEP-SENT) NN))


(DEFUN AGENT NEXPR (TEST)
;  AGENT FINDS THE AGENT CASE.
;  IF THE SENTENCE IS PASSIVE IT CALLS PASSIVE-AGENT WITH A LIST OF ALL OF
;  PHRASES STARTING WITH "BY".
;  IF THE SENTENCE IS NOT PASSIVE IT TAKES THE FIRST NOUN PHRASE OF
;  THE SENTENCE TO BE THE AGENT.  THIS IS A MAJOR ERROR AND SHOULD
;  BE CORRECTED TO MAKE AGENT MORE INTELLEGENT.
;  IF THE SENTECE BEGINS WITH A THAT-COMPLEMENT OR A TO-COMPLEMENT,
;  IT IS THE AGENT.
  (PROG (TESTEE TEMP PRN LST NUMS VNUM NUM)
        (COND ((GETR PASSIVE)
               (PROG (K-S)
                     (SETQ K-S KEEP-SENT NUM 1)
                   LOOP
                     (COND ((NULL K-S) (RETURN (PASSIVE-AGENT LST TEST NUMS VNUM)))
                           ((EQ (CAAR K-S) 'VP) (SETQ VNUM NUM))
                           ((AND (EQ (CAAR K-S) 'PP)
                                 (EQ (CADDAR K-S) 'BY)
                                 (SETQ LST (APPEND LST (LIST (CAR K-S))) NUMS (APPEND NUMS (LIST NUM))))))
                     (SETQ K-S (CDR K-S) NUM (ADD1 NUM))
                     (GO LOOP)))
              ((SETQ TESTEE (COND ((AND (EQ 'NP (CAAR L)) (NOT (CADAR L))) (CAR L))
                                  ((AND (ATOM (CAR L)) (EQ 'NP (CAADR L)) (NOT (CADADR L))) (CADR L))))
               (COND ((NULL (SETQ PRN (START-LIST 'PRO TESTEE)))
                      (COND ((OR (TEST-OFF? 'AGENT) (EVAL TEST))
                             (FLAG-ON TESTEE KEEP-SENT T)
                             (RETURN TESTEE))
                            (T (SETQ FAIL-TEST T FAIL-TEST-LIST (APPEND (LIST 'AGENT) FAIL-TEST-LIST))
                               NIL)))
                     (T (COND ((NOT (NULL (SETQ TEMP (ANAPHORIC '(AGENT BENEFICIARY
                                                                         RECIPIENT
                                                                         CO-AGENT
                                                                         PATIENT
                                                                         TOPIC)
                                                                TEST
                                                                ANAPHOR-DEPTH
                                                                PRN))))
                              (FLAG-ON TESTEE KEEP-SENT T)
                              (RETURN TEMP))))))
              ((AND (GET VERB 'THAT-COMP)
                    (EQ 'THAT-COMP (CAR (SETQ TESTEE (CAR KEEP-SENT))))
                    (FLAG-ON TESTEE KEEP-SENT T)
                    TESTEE))
              ((AND (GET VERB 'TO-COMP)
                    (EQ 'TO-COMP (CAR (SETQ TESTEE (CAR KEEP-SENT))))
                    (SETQ TESTEE (EVAL (CONS 'S
                                             (LIST (APPEND (LIST '(NP T (N SOMEONE)))
                                                           (REL-MODS-JOIN (CDDR TESTEE) 1))))))
                    (FLAG-ON (CAR KEEP-SENT) KEEP-SENT T)
                    TESTEE)))))


(DEFUN ALL-FLAG-OFF (K-S)
  (MAPC '(LAMBDA (X) (FLAG-OFF X K-S)) K-S)
  K-S)


(DEFUN ALL-HIST-OFF (S-H)
  (COND ((NULL S-H))
        (T (ALL-FLAG-OFF (CAR S-H)) (ALL-HIST-OFF (CDR S-H)))))


(DEFUN ALL-PREPS (NN K-S)
;  THIS ROUTINE RETURNS T IF ALL OF THE FIRST NN SENTENCE PARTS
;  OF K-S ARE PREPOSITION PHRASES.
  (COND ((NOT (EQ 'PP (CAAR K-S))) NIL)
        ((EQ NN 0))
        (T (ALL-PREPS (SUB1 NN) (CDR K-S)))))


(DEFUN ANAPHOR? (TESTEE KASE HIST TEST PRN)
  (AND (OR (TEST-OFF? KASE) (EVAL TEST))
       (OR (NULL PRN)
           (AND (OR (NOT (EQ 'PRO (CAR PRN))) (PRN-MATCH? PRN TESTEE)) (ANAPHOR-QUERY PRN TESTEE)))
       TESTEE))


(DEFUN ANAPHOR-CASE-LOOK (KASE TEMP)
  (PROG (TEMP K-S)
        (SETQ K-S KEEP-SENT)
      LOOP1
        (COND ((NULL KASE) (RETURN NIL)))
      LOOP2
        (COND ((NULL K-S) (SETQ KASE (CDR KASE) K-S KEEP-SENT) (GO LOOP1))
              ((OR (CADAR K-S) (EQ 'PP (CAAR K-S))) (SETQ K-S (CDR K-S)) (GO LOOP2))
              ((AND (EQ '"<-ADV-" (CAAR K-S))
                    (MEMBER 'LOCATION (GET (CADAR K-S) 'ADV-PROP))
                    (SETQ TEMP (ADV-MOD? (CAR K-S) TEST NIL))
                    (FLAG-ON (CAR K-S) KEEP-SENT T))
               (RETURN TEMP))
              (T (SETQ K-S (CDR K-S)) (GO LOOP2)))))
```

```
(DEFUN ANAPHORIC (KASE TEST DEPTH PRN)
;  ANAPHORIC IS THE MAIN ANAPHORIC REFERENCE ROUTINE.
;  IT TAKES A LIST OF CASES TO LOOK FOR, A TEST THE REFERENT
;  MUST PASS, A NUMBER INDICATING WHAT SENTENCE IN THE HISTORY
;  TO LOOK AT, AND THE PRONOUN BEING RESOLVED.
  (PROG (HIST TESTEE D K H-S)
        (COND ((MEMBER (CADR PRN) '(I ME MY MINE)) (RETURN (SPEAKER (CADR PRN))))
              ((AND THAT-CMP
;  THIS HANDLES SENTENCES LIKE: "IT IS UNFORTUNATE THAT YOU CAME."
                    (GET VERB 'THAT-COMP)
                    (MEMBER 'AGENT KASE)
                    (EQ (CADR PRN) 'IT)
                    (SETQ TESTEE (START-LIST 'THAT-COMP KEEP-SENT))
                    (NOT (CADR TESTEE)))
               (FLAG-ON TESTEE KEEP-SENT T)
               (RETURN TESTEE))
              ((AND (GET VERB 'TO-COMP)
;  THIS HANDLES SENTENCES LIKE: "IT IS NECESSARY TO GO HOME."
                    (MEMBER 'AGENT KASE)
                    (EQ (CADR PRN) 'IT)
                    (SETQ TESTEE (START-LIST 'TO-COMP KEEP-SENT))
                    (NOT (CADR TESTEE)))
               (FLAG-ON TESTEE KEEP-SENT T)
               (RETURN (TO-COMP '(NP T (N SOMEONE)))))
;  THIS RETURNS INTERNAL REFERENCES TO THE AGENT AS IN:
;  "FRED MARRIED THE WOMAN THAT HE LOVED."
              ((AND AG-ANAPHOR (ANAPHOR? AG-ANAPHOR KASE HIST TEST PRN)) (RETURN AG-ANAPHOR))
;  THIS PICKS UP INTERNAL REFERENCES TO THE PATIENT AS IN:
;  "FRED TOOK MARY TO HER HOUSE."
              ((AND PA-ANAPHOR (ANAPHOR? PA-ANAPHOR KASE HIST TEST PRN)) (RETURN PA-ANAPHOR))
              ((AND AG-ANAPHOR
;  THIS RESOLVES OTHER INTERNAL REFERENCES LIKE:
;  "FRED PLAYED THE MUSIC FOR MARY SO SHE WOULD BE HAPPY."
                    (SETQ TESTEE (FIND-ANAPHOR SUPER-SENT 'CHECK TEST PRN SUPER-SENT))
                    (RETURN TESTEE))))
        (SETQ HIST HISTORY
;  THIS PART DOES THE REFERENCES TO PREVIOUS SENTENCES.
              D 1
              H-S SENT-HIST
              K (COND ((AND (ATOM KASE) (NOT (NULL KASE))) (LIST KASE))
                      (T KASE)))
        (COND ((GREATERP D DEPTH) (RETURN NIL)))
      LOOP
        (COND ((NULL HIST) (RETURN NIL))
              ((LESSP D DEPTH) (SETQ D (ADD1 D) H-S (CDR H-S) HIST (CDR HIST)) (GO LOOP))
              ((FIND-ANAPHOR (CAR HIST) K TEST PRN (CAR H-S))))))


(DEFUN ANAPHOR-QUERY (PRN TESTEE)
;  ONCE A POTENTIAL REFERENT HAS BEEN FOUND FOR A PRONOUN
;  THE USER IS ASKED IF IT CORRECT.  THE FIRST PART MAKES SURE
;  IT HAS NOT ALREADY BEEN ASKED.
  (PROG (RESP TEMP1 TEMP2)
        (COND ((EQUAL (CAR (SETQ TEMP1 (TOP-LEVEL-START-LIST PRN ANAPHOR-RESP))) PRN)
               (COND ((EQ (MAIN-WORD-GET TESTEE) (MAIN-WORD-GET (CADR TEMP1)))
                      (RETURN (SELECTQ (CADADR TEMP1) (YES T) (NO NIL) NIL)))
                     ((EQ (CADADR TEMP1) 'YES) (RETURN NIL))))
              ((DEEP-MEMBER1 (LIST TESTEE 'YES) ANAPHOR-RESP) (RETURN NIL)))
        (PRIN1 'BY)
        (PRIN1 (CADR PRN))
        (PRIN1 '"I ASSUME YOU MEAN")
        (PRIN1 (SETQ TEMP2 (MAIN-WORD-GET TESTEE)))
        (TERPRI)
        (PRINT '"IS THAT CORRECT?")
      LOOP
        (COND ((EQ 'Y (CAR (EXPLODE (SETQ RESP (MYEVAL (READ))))))
               (SETQ ANAPHOR-RESP (APPEND (LIST (LIST PRN (LIST TESTEE 'YES))) ANAPHOR-RESP)))
              ((EQ 'N (CAR (EXPLODE RESP)))
               (SETQ ANAPHOR-RESP (APPEND (LIST (LIST PRN (LIST TESTEE 'NO))) ANAPHOR-RESP))
               NIL)
              (T (PRINT '"PLEASE ANSWER YES OR NO") (GO LOOP)))))


(DEFUN BE1? (NLIST)
  (COND ((EQ 'PL (CADR (START-LIST 'NUMBER NLIST))) 'SUBSET-OF-SET)
        (T (BE1-DET NLIST))))


(DEFUN BE1-DET (NL)
  (PROG (TEMP)
        (COND ((NULL (SETQ TEMP (START-LIST 'DET NL))) 'ELEMENT-OF-SET)
              ((DEFINITE-DET? (CADR TEMP)) 'ELEMENT-OF-SET)
              (T 'SUBSET-OF-SET))))
```

```
(DEFUN BEFORE-AFTER-DESC? (TESTEE PREP NN)
;  THIS ROUTINE IS FOR PHRASES LIKE:  "THE HOUSE AFTER THIS ONE. . ."
  (COND ((NOT (EQ 'NP (CAT TESTEE))) NIL)
        (T (PROG (NU PRO DET NOUN)
                 (COND ((OR (AND (MEMBER (CADR (SETQ DET (START-LIST 'DET TESTEE)))
                                         '(THIS THAT THESE THOSE))
                                 (EQ (CADR (SETQ NOUN (START-LIST 'N TESTEE))) 'ONE)
                                 (SETQ NU (CADR (START-LIST 'NUMBER NOUN))))
                            (AND (MEMBER (CADR (SETQ PRO (START-LIST 'PRO TESTEE)))
                                         '(THIS THAT THESE THOSE))
                                 (SETQ NU (CADR (START-LIST 'NUMBER PRO)))))
                        (BEFORE-AFTER-DESC1? TESTEE PREP NN NU)))))))


(DEFUN BEFORE-AFTER-DESC1? (TESTEE PREP NN NU)
  (PROG (BFORE)
        (SETQ BFORE (NOUN-LIST-GET (ELMT (SUB1 NN KEEP-SENT))))
        (COND ((EQ NU (CADR (START-LIST 'NUMBER BFORE))) (NP-BUILD BFORE)))))


(DEFUN BE-LOC ()
;  THIS IS THE LOCATIONAL CASE OF "TO BE":
;  "FRED IS IN THE ATTIC."
;  "FRED IS WITH MARY."
  (PROG (PREPS PREP TESTEE RET)
        (SETQ PREPS '(IN ON UNDER WITH AT BY NEAR BESIDE BEFORE UP DOWN AROUND ACROSS UPON))
        (RETURN (COND ((SETQ TESTEE (ELMT (ADD1 (PHRASE-NUM 'VP KEEP-SENT)) KEEP-SENT))
                       (COND ((AND (EQ (CAR TESTEE) 'PP)
                                   (NOT (CADR TESTEE))
                                   (OR (TEST-OFF? 'LOCATION) (NOT (MUST-BE ABSTRACT)))
                                   (SETQ PREP (CAR (MEMBER (CADDR TESTEE) PREPS))))
                              (FLAG-ON TESTEE
                                       KEEP-SENT
                                       (SETQ RET (COND ((EQ PREP 'WITH)
                                                        (LIST '"<-LOC-OF-"
                                                              (NOUN-LIST-GET (NP-BUILD TESTEE))))
                                                       (T (LIST PREP
                                                                (NOUN-LIST-GET (NP-BUILD TESTEE)))))))
                              RET)))))))


(DEFUN BE-LOC1 ()
  (PROG (PREPS PREP TESTEE RET NNN)
        (SETQ PREPS '(IN ON UNDER WITH AT BY NEAR BESIDE BEFORE UP DOWN AROUND ACROSS UPON))
        (SETQ NNN (ADD 1 (PHRASE-NUM 'VP KEEP-SENT)))
      LOOP
        (COND ((SETQ TESTEE (ELMT NNN KEEP-SENT))
               (COND ((AND (EQ (CAR TESTEE) 'PP)
                           (NOT (CADR TESTEE))
                           (OR (TEST-OFF? 'LOCATION) (NOT (MUST-BE ABSTRACT)))
                           (SETQ PREP (CAR (MEMBER (CADDR TESTEE) PREPS))))
                      (FLAG-ON TESTEE
                               KEEP-SENT
                               (SETQ RET (COND ((EQ PREP 'WITH)
                                                (LIST '"<-LOC-OF-" (NOUN-LIST-GET (NP-BUILD TESTEE))))
                                               (T (LIST PREP (NOUN-LIST-GET (NP-BUILD TESTEE)))))))
                      (RETURN RET))
                     (T (SETQ NNN (ADD1 NNN)) (GO LOOP))))))


(DEFUN BENEFICIARY (TESTEE PREP)
;  THIS FINDS THE BENEFICIARY CASE: "FRED PLAYS THE PIANO FOR MARY."
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PROG (TEMP)
                             (COND ((SETQ TEMP (PRO-ANAPHOR? '(AGENT BENEFICIARY
                                                               CO-AGENT
                                                               TOPIC
                                                               PATIENT)
                                                             '(MUST-BE ANIMATE)
                                                             PRN
                                                             TESTEE))
                                    (SET-UP-FN 'BENEFICIARY PREP)
                                    (RETURN (LIST '"<-BENEFICIARY-" TEMP))))))
                      ((MUST-BE ANIMATE)
                       (SET-UP-FN 'BENEFICIARY PREP)
                       (LIST '"<-BENEFICIARY-" (NOUN-LIST-GET (NP-BUILD TESTEE))))))))


(DEFUN BUILD1 (FRAG)
;  THIS IS PART OF THE BUILDQ FUNCTION FOR BUILDING FINAL STRUCTURES.
  (PROG (A)
        (RETURN (COND ((NULL FRAG) NIL)
                      ((EQ FRAG '"*") (GETRV (CAR REGLST)))
                      ((EQ FRAG '?) (NOUN-PUT (CAR REGLST)))
                      ((EQ FRAG '#) (EVAL (CAR REGLST)))
                      ((EQ FRAG '*) *)
                      ((ATOM FRAG) FRAG)
                      ((EQ (CAR FRAG) '"*")
                       (SETQ A (UNCONS REGLST REGLST))
                       (CONS (GETRV A) (BUILD1 (CDR FRAG))))
                      ((EQ (CAR FRAG) '?)
                       (SETQ A (UNCONS REGLST REGLST))
                       (CONS (NOUN-PUT A) (BUILD1 (CDR FRAG))))
                      ((EQ (CAR FRAG) '#)
                       (SETQ A (UNCONS REGLST REGLST))
                       (CONS (EVAL A) (BUILD1 (CDR FRAG))))
                      ((EQ (CAR FRAG) '*) (CONS * (BUILD1 (CDR FRAG))))
                      ((EQ (CAR FRAG) '@) (GLUE (BUILD1 (CDR FRAG))))
                      (T (CONS (BUILD1 (CAR FRAG)) (BUILD1 (CDR FRAG))))))))
```

```
(DEFUN CAN-CLASS? (NP TMP)
;  THIS ROUTINE ASKS THE USER IF A NOUN PHRASE IS A CLASSIFIER
;  OF ANOTHER NOUN PHRASE.
  (COND ((AND (EQ (CAR TMP) 'NP) (NULL (START-LIST 'DET TMP)) (NOUN-GET NP))
         (PRINI 'IS)
         (PRINI (NOUN-GET NP))
         (PRINI '"A CLASSIFIER OF")
         (PRINI (NOUN-GET TMP))
         (TERPRI)
         (COND ((EQ 'Y (CAR (EXPLODE (READ))))
                (SETQ FN '(RPLACD (END-LIST PLACE) (LIST STRCT)) PLACE (NOUN-LIST-GET TMP))
                (RETURN (LIST '"<-CLASS-" (NOUN-LIST-GET (NP-BUILD PART)))))))))


(DEFUN CAN-MOD-DESC? (TESTEE PART PREP)
  (APPLY1 (IMPLODE (LIST 'DESC '"-" PREP)) TESTEE PART))


(DEFUN CASE NEXPR (CASES)
;  THIS ROUTINE GOES THROUGH A LIST OF CASE FRAMES FOR A VERB TO
;  FIND THE CORRECT ONE.
  (PROG (OB-OPT FAIL-TEST)
      LOOP
        (COND ((NULL CASES))
              ((PROGN (SETQ FAIL-TEST NIL) (EV-SETR (CADR CASES) (CAR CASES)))
               (SETQ CASES (CDDDR CASES))
               (GO LOOP))
              ((EQ 'OBL (SETQ OB-OPT (MYEVAL (CADDR CASES)))) (ALL-FLAG-OFF KEEP-SENT) NIL)
              ((EQ 'OPT OB-OPT) (SETQ CASES (CDDDR CASES)) (GO LOOP))
              (T (EV-SETR (CADR CASES) OB-OPT) (SETQ CASES (CDDDR CASES)) (GO LOOP)))))


(DEFUN CASE-ANAPHOR (KASE PART TEST)
;  THIS ROUTINE IS INTENDED TO BE EXTENDED.
;  CURRENTLY IT ONLY WORKS WITH THE SOURCE CASE.  IF THE SOURCE
;  CASE IS LOOKED FOR BUT NOT FOUND IN A SENTENCE, THIS
;  ROUTINE TRIES TO FIND THE SOURCE BY LOOKING FOR WHERE THE
;  THING OR PERSON WAS BEFORE THE MOVEMENT DESCRIBED IN THE SENTENCE.
  (PROG (TESTEE)
        (SELECTQ KASE
                 (SOURCE (AND (SETQ TESTEE (WHERE-IS (NOUN-LIST-GET (GETR AG))))
                              (OR (TEST-OFF? KASE) (EVAL TEST))
                              TESTEE))
                 NIL)))


(DEFUN CASE-ATOM-ASSOC (KASE)
  (SELECTQ KASE (LOCATION 'LOC) KASE))


(DEFUN CASES? (PART NN)
;  THIS ROUTINE IS GIVEN A SENTENCE PART AND ASKED TO FIND WHAT CASE IT IS.
;  IF IT STARTS WITH A PREPOSTITION OR IS A COMPLEMENT, THEN
;  CASE-CALLER IS CALLED.  OTHERWISE MODIFY? IS INVOKED.
  (PROG (NUM FAIL-LIST PLACE PREP&CASE STRCT FN TEMP)
        (COND ((AND (PREP-START PART) (CASE-CALLER (CADDR PART) VERB (CDDDR PART) NN)))
              ((AND (SETQ TEMP (COMPLEMENT PART)) (CASE-CALLER TEMP VERB (CDDR PART) NN)))
              ((MODIFY? (COND ((ATOM (CAR PART)) PART) (T (CAR PART))) NN)))))


(DEFUN CASE-CALLER (PREP VERB NP NN)
;  CASE-CALLER CALLS THE FOREGROUNDING ROUTINE, THEN INVOKES THE
;  ROUTINE FOREGROUNDED.  IF THAT ROUTINE FAILS, OR THE SYSTEM
;  FAILS LATER (VIA CONTINUE) THEN THE NEXT ROUTINE IS INVOKED.
;  THIS IS THEREFORE THE ROUTINE WHICH CONTROLS THE DECISION POINTS
;  FOR THE BACK-UP.
  (PROG (TEMP)
        (SETQ TEMP (CASE-FORE-GROUND VERB PREP (PREP-LIST-GET PREP CASE-TABLE) NN))
      LOOP
        (COND ((NULL TEMP) NIL)
              ((AND (EQ (LENGTH NP) 2)
                    (NOT (MEMBER (CAR TEMP) '(METHOD CAUSE TIME PURPOSE EXPECTED-EFFECT))))
               (SETQ TEMP (CDR TEMP))
               (GO LOOP))
              ((AND (SETQ STRCT (APPLY1 (CAR TEMP) NP PREP)) (CONTINUE)))
              (T (SETQ TEMP (CDR TEMP)) (GO LOOP)))))
```

```
(DEFUN CASE-CLAUSE-BUILD (KASE PREP TESTEE)
  (LIST (IMPLODE (LIST '"<-" (CASE-ATOM-ASSOC KASE) '"-")) PREP (NOUN-LIST-GET (NP-BUILD TESTEE))))


(DEFUN CASE-DELETE (VERB PREP L)
;  THIS ALLOW CERTAIN CASES TO BE RULED OUT FOR CERTAIN VERBS AND PREPOSTIONS.
  (PROG (TEMP)
        (RETURN (COND ((NULL (SETQ TEMP (PREP-LIST-GET VERB ANTI-CASE-TABLE))) L)
                      ((OR (NULL (CADDR TEMP)) (MEMBER PREP (CADDR TEMP)))
                        (SOFT-DELETE (CADR TEMP) T L))
                      (T L)))))


(DEFUN CASE-FIND (X KASE)
;  THIS IS USED FOR FINDING A CASE IN A PREVIOUS FINAL STRUCTURE.
;  IT IS USED BY THE ANAPHORIC ROUTINES.
  (PROG (TEMP)
        (RETURN (COND ((SETQ TEMP (CDR (START-LIST (SELECTQ KASE
                                                            (AGENT '"<==>")
                                                            (PATIENT '"<--")
                                                            (LOCATION '"<-LOC-")
                                                            (RECIPIENT '"<-RECIPIENT-")
                                                            (TIME '"<-TIME-")
                                                            (DURATION '"<-DURATION-")
                                                            (DESC '"<-DESC-")
                                                            (TOPIC '"<-TOPIC-")
                                                            (BENEFICIARY '"<-BENEFICIARY-")
                                                            (PATH '"<-PATH-")
                                                            (EXCHANGE '"<-EXCHANGE-")
                                                            (CO-AGENT '"<-CO-AGENT-")
                                                            (DESTINATION '"<-DESTINATION-")
                                                            (SOURCE '"<-SOURCE-")
                                                            (LIKE '"<-LIKE-")
                                                            (WITH '"<-WITH-")
                                                            NIL)
                                                  X)))
                       (COND ((EQ KASE 'PATIENT) (CADR TEMP))
                             ((EQ KASE 'AGENT) (CAR TEMP))
                             ((NOT (ATOM (CAR TEMP))) (CAR TEMP))
                             (T TEMP)))
                      ((SETQ TEMP (START-LIST '"<==>" (CDR X))) (CASE-FIND TEMP KASE)))))


(DEFUN CASE-FORE-GROUND (VERB PREP KASES NN)
;  THIS IS THE TOP-LEVEL FOREGROUNDING ROUTINE.  AFTER V-FORE-GRND IS
;  CALLED IT CHECKS TO SEE IF DESC IS ONE OF THE CASES TO BE CALLED.
;  IF SO, IF THE PREVIOUS PHRASE IS A NOUN- OR A PREP-PHRASE, IT
;  FOREGROUNDS DESC.
  (PROG (TEMP BEF)
        (SETQ TEMP (CASE-DELETE VERB
                                PREP
                                (APPEND (SETQ TEMP (V-FORE-GRND VERB
                                                                PREP
                                                                (CDR (PREP-LIST-GET PREP
                                                                                    (GET VERB
                                                                                         'PREP-CASE)))))
                                        (COMP-INTERSECTION (CDR KASES) TEMP))))
        (COND ((AND (MEMBER 'DESC TEMP)
                    (MEMBER (CAR (SETQ BEF (ELMT (SUB1 NN) KEEP-SENT))) '(PP NP))
                    (NOT (START-LIST 'NPR BEF)))
               (RETURN (APPEND (LIST 'DESC) (SOFT-DELETE 'DESC T TEMP))))
              (T TEMP)))))


(DEFUN CASE-PREPS (VERB K)
  (PROG (TEMP L CSS)
        (SETQ TEMP (GET VERB 'PREP-CASE) CSS (COND ((ATOM K) (LIST K)) (T K)))
      LOOP
        (COND ((NULL TEMP) (RETURN L))
              ((INT-LIST CSS (CDAR TEMP)) (SETQ L (APPEND (LIST (CAAR TEMP)) L))))
        (SETQ TEMP (CDR TEMP))
        (GO LOOP)))


(DEFUN CASE-RETURN (ARROW PREP PH)
  (APPEND (LIST (MAKE-ARROW ARROW))
          (COND ((NOT (NULL PREP)) (LIST PREP)))
          (LIST (LIST (NOUN-LIST-GET (NP-BUILD (CAR PH)))))))
```

```
(DEFUN CAUSE (TESTEE PREP)
;  CAUSE IS INVOKED FOR SO-COMPLEMENTS.  IT REJECTS "SO. . .WOULD"
;  CONSTRUCTS, THEN IT SORTS OUT THE ENABLE AND CAUSE CASES.
;  "SO. . .CAN" CONSTRUCTS ARE THE ENABLE CASE.  SINCE SO-COMPLEMENTS
;  ARE SENTENCES, S IS CALLED.  AG-ANAPHOR AND PA-ANAPHOR
;  ARE SET FOR INTERNAL REFERENCES.
  (SELECTQ PREP
           ('SO
             (COND ((NOT (EQ 'WOULD (CAR TESTEE)))
                    (PROG (TEMP)
                          (SETQ AG-ANAPHOR (GETR AG) PA-ANAPHOR (GETR PA))
                          (COND ((NOT (ATOM (SETQ TEMP (EVAL (COND ((NCT (EQ 'S (CAAR TESTEE)))
                                                                    (CADR TESTEE))
                                                                   (T (CAR TESTEE))))))))
                                (SET-UP-FN 'CAUSE NIL)
                                (SETQ AG-ANAPHOR NIL PA-ANAPHOR NIL)
                                (RETURN (BUILDQ (# #)
                                                (COND ((MEMBER (CAR TESTEE) '(COULD CAN))
                                                       '"<-ENABLE-")
                                                      (T '"<-CAUSE-"))
                                                TEMP)))))))
           NIL))

(DEFUN CHARACTERISTIC (A D)
;  THIS CASE IS NOT IMPLEMENTED YET.
  NIL)

(DEFUN CHOOSE-ELMT (PART NN F-L)
;  THIS ROUTINE IS USED TO CHOOSE A CANDIDATE WHEN TRYING TO FIGURE
;  WHAT A CERTAIN SENTENCE PART MODIFIES.
;  F-L IS THE LIST OF CANDIDATES ALREADY FAILED.  IT BEGINS BY
;  LOOKING AT THE PART BEFORE THE PART IN QUESTION, THEN THE
;  ONE BEFORE THAT, ETC.  IF THEY ALL FAIL, IT LOOKS AT THE ONE
;  AFTER THE PART IN QUESTION, THEN THE ONE AFTER THAT, ETC.
  (PROG (P)
        (COND ((AND (NOT (MEMBER (SETQ NUM (SUB1 NN)) F-L)) (LESSP 0 NUM))
               (RETURN (ELMT NUM KEEP-SENT)))
              ((AND (NOT (MEMBER (SETQ NUM (ADD1 NN)) F-L)) (NOT (GREATERP NUM (LENGTH KEEP-SENT))))
               (RETURN (ELMT NUM KEEP-SENT)))
              (T (SETQ NUM (SUB 2 NN))
                 (RETURN (PROG NIL
                               LOOP
                               (COND ((GREATERP 0 NUM)
                                      (COND ((NOT (MEMBER NUM F-L)) (RETURN (ELMT NUM KEEP-SENT)))
                                            (T (SETQ NUM (SUB1 NUM)) (GO LOOP))))
                                     (T (SETQ NUM (ADD 2 NN))
                                        (RETURN (PROG NIL
                                                      LOOP1
                                                      (COND ((AND (NOT (MEMBER NUM F-L))
                                                                  (SETQ P (ELMT NUM KEEP-SENT)))
                                                             (RETURN P))
                                                            (P (SETQ NUM (ADD1 NUM)) (GO LOOP1)))))))))))))

(DEFUN CHOOSE-ELMT-PREP (NP NN F-L)
;  THIS ROUTINE IS USED BY THE LOCATIVE CASE.  IF THE SENTENCE
;  PART IN QUESTION IS THE FIRST IN THE SENTENCE OR FOLLOWS
;  ONLY PREP-PHRASES, THEN IT RETURNS THE VERB-PHRASE AS
;  THE LIKELY CANDIDATE.
;  THAT IS, THE PHRASE IS CONSIDERED TO BE THE LOCATIVE CASE, NOT THE DESCRIPTIVE CASE.
  (COND ((OR (EQ NN 1) (ALL-PREPS NN KEEP-SENT)) (ELMT (SETQ NUM (PHRASE-NUM 'VP KEEP-SENT)) KEEP-SENT))
        (T (CHOOSE-ELMT NP NN F-L))))

(DEFUN CLASS? (NP)
  (PROG (TEMP) (RETURN (COND ((SETQ TEMP (ELMT (SETQ NUM (ADD1 N)) KEEP-SENT)) (CAN-CLASS? NP TEMP))))))
```

```
(DEFUN CLAUSE (PREP TEST KASE)
;  THIS ROUTINE LOOKS FOR A PREPOSITION PHRASE BEGINNING WITH ONE OF THE
;  PREPOSITIONS IN PREP.
 (PROG (TESTEE K-S TEMP PRN PR RET FG)
       (SETQ K-S KEEP-SENT)
    LOOP
      (COND ((NULL K-S) NIL)
            ((AND (EQ 'PP (CAR (SETQ TESTEE (CAR K-S))))
                  (NOT FG)
                  (NOT (CADR TESTEE))
                  (SETQ PR (CAR (MEMBER (CADDR TESTEE) PREP))))
              (COND ((SETQ PRN (PRO? TESTEE)) (PRO-ANAPHOR? KASE TEST PRN TESTEE))
                    ((AND (EQ '"<-ADV-" (CAAR (CDDDR TESTEE)))
                          (SETQ TEMP (ADV-MOD? (CADDDR TESTEE) TEST KASE)))
                     (FLAG-ON TESTEE K-S TEMP)
                     (RETURN TEMP))
                    ((AND (OR (TEST-OFF? KASE) (EVAL TEST))
                          (PROGN (SETQ FAIL-TEST NIL)
                                 (FLAG-ON TESTEE K-S (SETQ RET (CASE-CLAUSE-BUILD KASE PR TESTEE)))
                                 RET)))
                    (T (SETQ FAIL-TEST T K-S (CDR K-S)) (GO LOOP))))
            (T (SETQ K-S (CDR K-S))))
;  THIS PART IS AN ATTEMPT NOT TO BREAK UP RELATIVE CLAUSES.
;  IF FG IS T THE ROUTINE THINKS IT IS IN A RELATIVE CLAUSE
;  AND WILL NOT ACCEPT THE PHRASE.
      (COND ((MEMBER (CAR TESTEE) '(REL OBJ-REL RED-REL OBJ-RED-REL)) (SETQ FG T))
            ((EQ (CAR TESTEE) 'VP) (SETQ FG NIL)))
      (GO LOOP)))))


(DEFUN CO-AGENT (TESTEE PREP)
;  THIS IS THE CO-AGENT CASE.
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PROG (TEMP)
                             (COND ((SETQ TEMP (PRO-ANAPHOR? '(AGENT RECIPIENT
                                                               CO-AGENT
                                                               PATIENT
                                                               BENEFICIARY)
                                                             '(MUST-BE ANIMATE)
                                                             PRN
                                                             TESTEE))
                                    (SET-UP-FN 'CO-AGENT PREP)
                                    (RETURN (LIST '"<-CO-AGENT-" TEMP))))))
                      ((MUST-BE ANIMATE)
                       (SET-UP-FN 'CO-AGENT PREP)
                       (LIST '"<-CO-AGENT-" (NOUN-LIST-GET (NP-BUILD TESTEE))))))))


(DEFUN COMP-INTERSECTION (A B)
  (MAPC '(LAMBDA (X) (COND ((MEMBER X B) (SETQ A (SOFT-DELETE X T A))))) A)
  A)


(DEFUN COMPLEMENT (L)
  (SELECTQ (CAR L)
           ('TO-COMP 'TO)
           ('BY-COMP 'BY)
           ('SO-COMP 'SO)
           ('WHEN 'WHEN)
           ('WHILE 'WHILE)
           ('BEFORE 'BEFORE)
           ('AFTER 'AFTER)
           NIL))


(DEFUN CONTEXT (A S)
;  THIS CASE IS NOT IMPLEMENTED YET.
  NIL)


(DEFUN CONTINUE ()
;  THIS IS THE ROUTINE THAT CONTROLS THE STACK OF CASES.  FN IS THE FUNCTION
;  WHICH WILL ADD A STRUCTURE TO THE CURRENT STRUCTURE IN STRUCT.
;  TAIL-ENDS SUCCEEDS IF THE REST OF THE SENTENCE CAN BE ANALYZED IN THE
;  CURRENT CASE FRAME.
  (SETQ FAIL-LIST (APPEND FAIL-LIST (LIST NUM)))
  (FLAG-ON PART KEEP-SENT STRCT)
  (COND ((TAIL-ENDS) (EVAL FN) (PUT-IN-LIST VERB 'CASE-HIST PREP&CASE))))
```

```
(DEFUN COPULAR-ADV? NEXPR (TEST)
; THIS IS A CASE OF THE VERB "TO BE".  IT LOOKS AFTER SENTENCE LIKE:
;  "I AM WELL."
  (PROG (TESTEE K-S FT NN TEMP RET)
        (SETQ K-S KEEP-SENT NN (PHRASE-NUM 'VP KEEP-SENT) TESTEE (ELMT (SETQ NN (ADD1 NN)) KEEP-SENT))
      LOOP
        (COND ((NULL TESTEE)
               (COND (FT (SETQ FAIL-TEST T FAIL-TEST-LIST (APPEND (LIST 'COPULAR-ADV) FAIL-TEST-LIST))))
               (RETURN NIL))
              ((AND (EQ '"<-ADV-" (CAR TESTEE)) (NOT (CADR TESTEE)) (SETQ FT T) (EVAL TEST))
               (SETQ TEMP (ADV-LOC TESTEE T 'COPULAR-ADV))
               (FLAG-ON TESTEE KEEP-SENT (SETQ RET (CDDR TEMP)))
               (RETURN RET))
              (T (SETQ TESTEE (ELMT (SETQ NN (ADD1 NN)) KEEP-SENT)) (GO LOOP)))))


(DEFUN DATE (NP)
  (MEMBER 'YEAR (GET (NOUN-GET NP) 'N-PROP)))


(DEFUN DBLQ (Z)
  (LIST 'QUOTE (LIST 'QUOTE Z)))


(DEFUN DEEP-MEMBER (L1 L2)
;  THIS ROUTINE SUCCEEDS IF L2 IS A DEEP MEMBER OF L1.
  (PROG NIL
      LOOP
        (COND ((NULL L1) (RETURN NIL))
              ((DEEP-MEMBER1 (CAR L1) L2) (RETURN (CAR L1)))
              (T (SETQ L1 (CDR L1)) (GO LOOP)))))


(DEFUN DEEP-MEMBER1 (L1 L2)
  (COND ((NULL L2) NIL)
        ((EQUAL L1 L2))
        ((ATOM L2) NIL)
        (T (OR (DEEP-MEMBER1 L1 (CAR L2)) (DEEP-MEMBER1 L1 (CDR L2))))))


(DEFUN DEEP-PHRASE-NUM (A L)
  (PROG (NN)
        (SETQ NN 0)
      LOOP
        (COND ((NULL L) 0)
              ((DEEP-MEMBER1 A (CAR L)) (ADD1 NN))
              (T (SETQ NN (ADD1 NN) L (CDR L)) (GO LOOP)))))


(DEFUN DEFAULT (A B)
;  THIS ROUTINE ALLOWS THE DICTIONARY WRITER TO LET CASES DEFAULT IF NOT
;  SPECIFIED IN THE SENTENCE.  IE: "THE PIANIST PLAYED ALL DAY." --
;  THE PATIENT CASE COULD DEFAULT TO "THE PIANO" BY USING THIS ROUTINE
;  IN THE DEFINITION OF "TO PLAY".
  (PROG (TEMP)
        (SETQ TEMP (GET B 'DEFAULT))
      LOOP
        (COND ((NULL TEMP) 'OBL)
              ((AND (MEMBER VERB (CADR TEMP)) (MEMBER A (CAR TEMP)))
                (RETURN (LIST 'QUOTE (CADDR TEMP))))
              (T (SETQ TEMP (CDDDR TEMP)) (GO LOOP)))))


(DEFUN DEFINITE-DET? (D)
  (MEMBER D '(THE ALL THAT THOSE THIS THESE)))
```

```
(DEFUN DESC (TESTEE PREP)
;  THIS IS THE DESCRIPTIVE CASE:  THE CASE OF THE NOUN.
;  THE PHRASE PRECEEDING IS THE ONLY ONE CONSIDERED AS THE
;  ONE BEING MODIFIED.  THIS COULD BE CHANGED BY INVOKING
;  CHOOSE-ELMT.  THE ELEMENT CHOOSEN MUST BE A PREP-PHRASE
;  OR A NOUN PHRASE.
  (PROG (PRN TEMP)
        (COND ((AND (MEMBER PREP '(BEFORE AFTER)) (SETQ TEMP (BEFORE-AFTER-DESC? TESTEE PREP NN)))
               (SETQ TESTEE TEMP))
              ((SETQ PRN (PRO? TESTEE))
               (SETQ TESTEE (PRO-ANAPHOR? '(PATIENT DESC TOPIC AGENT) T PRN TESTEE))))
        (COND ((NULL TESTEE) (RETURN NIL))
              ((OR (EQ NN 1) (NOT (MEMBER (CAR (SETQ TEMP (ELMT (SUB1 NN) KEEP-SENT))) '(PP NP))))
               (RETURN NIL)))
        (COND ((AND (OR (START-LIST 'DET TESTEE) (NOT (EQ (CADR (START-LIST 'NUMBER TESTEE)) 'SG)))
                    (DESC-MOD? TESTEE TEMP PREP)))
              (T (RETURN NIL)))))


(DEFUN DESC-ACROSS (A B)
  (DESC-TRAJ 'ACROSS A B))


(DEFUN DESC-AROUND (A B)
  (DESC-TRAJ 'AROUND A B))


(DEFUN DESC-AT (TESTEE M-PH)
  (NOT (MUST-BE HUMAN)))


(DEFUN DESC-BESIDE (A B)
  (DESC-BY-NEAR-BESIDE 'BESIDE A B))


(DEFUN DESC-BY (A B)
  (DESC-BY-NEAR-BESIDE 'BY A B))


(DEFUN DESC-BY-NEAR-BESIDE (KASE D-PH M-PH)
;  THIS HANDLES THE DESCRIPTIVE CASE FOR PHRASES BEGINNING WITH
;  BY, NEAR, OR BESIDE.  IT INSISTS THAT THE NOUNS IN QUESTION EITHER
;  BOTH BE SMALL OR ELSE BE ABOUT THE SAME SIZE, OR ELSE THE SECOND
;  ONE MUST BE A BODY OF WATER: "I WALKED BESIDE THE OCEAN."
  (PROG (S1 S2)
        (SETQ S2 (GET (NOUN-GET M-PH) 'OBJ-SIZE))
        (RETURN (COND ((AND (OR (START-LIST 'DET D-PH)
                                (AND (HABITUAL?) (NOT (EQ 'SG (CADR (START-LIST 'NUMBER D-PH))))))
                            (OR (AND (LESSP (SETQ S1 (GET (NOUN-GET D-PH) 'OBJ-SIZE)) 6) (LESSP S2 6))
                                (LESSP (ABS (SUB S1 S2)) 2)
                                (MEMBER (NOUN-GET D-PH) '(OCEAN SEA LAKE))))
                      (CASE-RETURN 'LOC KASE D-PH))))))


(DEFUN DESC-FROM (D-PH M-PH)
;  THIS HANDLES THE STRUCTURE: "THE MAN FROM PARIS. . ."
  (PROG (TESTEE)
        (SETQ TESTEE M-PH)
        (COND ((GREATERP (GET (NOUN-GET D-PH) 'OBJ-SIZE) 3)
               (COND ((AND (MUST-BE ANIMATE) (SETQ TESTEE D-PH) (MUST-BE LOCATION))
                      (BUILDQ ("<==>" # PAST ("<--" INHABIT #))
                              (NOUN-LIST-GET (COPY M-PH))
                              (NOUN-LIST-GET (NP-BUILD D-PH))))
                     (T (CASE-RETURN 'DESC 'FROM D-PH)))))))


(DEFUN DESC-IN (DESC-PH MOD?-PH)
;  THIS HANDLES PHRASES LIKE: "THE MAN IN THE HOUSE."
  (PROG (S1 S2)
        (SETQ S1 (GET (NOUN-GET DESC-PH) 'OBJ-SIZE) S2 (GET (NOUN-GET MOD?-PH) 'OBJ-SIZE))
        (RETURN (COND ((OR (NULL S1) (NULL S2)) NIL)
                      ((AND TEST-OFF (GREATERP (SUB S1 2) S2)) (CASE-RETURN 'LOC 'IN DESC-PH))
                      ((AND SHOULD-TEST-OFF (GREATERP (SUB1 S1) S2)) (CASE-RETURN 'LOC 'IN DESC-PH))
                      ((GREATERP S1 S2) (CASE-RETURN 'LOC 'IN DESC-PH))))))


(DEFUN DESC-MOD? (TESTEE PART PREP)
  (PROG (TEMP)
        (COND ((AND (OR (START-LIST 'DET PART) (NOT (EQ (CADR (START-LIST 'NUMBER PART)) 'SG)))
                    (SETQ TEMP (CAN-MOD-DESC? TESTEE PART PREP)))
               (SETQ FN '(REPLACD (END-LIST PLACE) (LIST STRCT))
                     PLACE (NOUN-LIST-GET PART)
                     PREP&CASE (LIST PREP 'DESC)
                     NUM (PHRASE-NUM2 PART KEEP-SENT))
               TEMP))))


(DEFUN DESC-NEAR (A B)
  (DESC-BY-NEAR-BESIDE 'NEAR A B))
```

```
(DEFUN DESC-OF (TESTEE PART)
;  THIS HANDLES "THE BOOKS OF JOHN. . ." AND "THE BOX OF BOOKS. . ."
  (PROG (TEMP)
        (SETQ TEMP TESTEE)
        (COND ((MUST-BE HUMAN)
               (LIST '"<==>"
                     (NOUN-LIST-GET (NP-BUILD TESTEE))
                     (LIST '"<--" (POSSESSION TESTEE PART) (NOUN-LIST-GET (COPY PART)))))
              ((AND (SETQ TESTEE PART) (MUST-BE CONTAINER))
               (LIST '"<==>" (NOUN-GET PART) (LIST '"<--" 'CONTAIN (NOUN-LIST-GET (NP-BUILD TEMP)))))
              (T (LIST '"<-DESC-" PREP (NOUN-LIST-GET (NP-BUILD TESTEE)))))))


(DEFUN DESC-ON (A B)
  (DESC-ON-UNDER 'ON A B))


(DEFUN DESC-ON-UNDER (KASE D-PH M-PH)
;  THIS HANDLES: "THE MAN ON THE TABLE" AND "THE MAN UNDER THE CAR."
;  IT HAS THE OBVIOUS SIZE RESTRAINTS.
  (PROG (S1 S2)
        (SETQ S1 (GET (NOUN-GET D-PH) 'OBJ-SIZE) S2 (GET (NOUN-GET M-PH) 'OBJ-SIZE))
        (RETURN (COND ((OR (NULL S1) (NULL S2)) NIL)
                      ((OR (AND TEST-OFF (LESSP S1 10))
                           (AND SHOULD-TEST-OFF (LESSP S1 7))
                           (AND (LESSP S1 7) (OR (GREATERP S1 0) (LESSP S2 2))))
                       (CASE-RETURN 'LOC KASE D-PH))))))


(DEFUN DESC-TRAJ (KASE D-PH M-PH)
;  THIS HANDLES "THE MAN ACROSS THE ROOM." ETC
  (PROG (S1 S2)
        (SETQ S1 (GET (NOUN-GET D-PH) 'OBJ-SIZE) S2 (GET (NOUN-GET M-PH) 'OBJ-SIZE))
        (RETURN (COND ((OR (NULL S1) (NULL S2)) NIL)
                      ((OR (AND TEST-OFF (GREATERP S1 0))
                           (AND SHOULD-TEST-OFF (GREATERP S1 1))
                           (GREATERP S1 2))
                       (CASE-RETURN 'LOC KASE D-PH))))))


(DEFUN DESC-UNDER (A B)
  (DESC-ON-UNDER 'UNDER A B))


(DEFUN DESC-WITH (D-PH M-PH)
;  THIS HANDLES "THE BOX WITH THE PENCILS IN IT. . ."
;  "THE GIRL WITH THE RED HAIR. . .", ETC
  (PROG (NEXT PRN)
        (RETURN (COND ((AND (NOT SHOULD-TEST-OFF) (START-LIST 'NPR M-PH)) NIL)
                      ((NULL (SETQ NEXT (ELMT (ADD1 (PHRASE-NUM2 (CONS 'PP (CONS NIL (CONS 'WITH D-PH)))
                                                                  KEEP-SENT))
                                              KEEP-SENT)))
                       (CASE-RETURN 'DESC 'WITH D-PH))
                      ((AND (EQ (CAR NEXT) 'PP) (SETQ PRN (PRO? (CADDDR NEXT))) (PRN-MATCH? PRN M-PH))
                       (FLAG-ON NEXT KEEP-SENT T)
                       (PREP-VERB-ASSQ-COMPLX-RETURN (COPY M-PH) D-PH NEXT))
                      (T (CASE-RETURN 'DESC 'WITH D-PH))))))


(DEFUN DESTINATION (TESTEE PREP)
  (SOURCE-DEST TESTEE PREP 'DESTINATION))


(DEFUN DROP (N LETS)
  (COND ((ZEROP N) LETS)
        (T (DROP (SUB1 N) (CDR LETS)))))


(DEFUN DROP&GO (LST LETS)
  (SETQ LETS (DROP (LENGTH (CAR LST)) LETS))
  (EVAL (CADR LST)))


(DEFUN DURATION (NP PREP)
;  THIS IS THE DURATION CASE.
  (COND ((DURATION? NP)
         (SET-UP-FN 'DURATION PREP)
         (BUILDQ ("<-DURATION-" #) (NOUN-LIST-GET (NP-BUILD NP))))))


(DEFUN DURATION? (NP)
;  THIS IS THE TEST FOR THE DURATION CASE.
;  IT SUCCEEDS ON "THEY PLAYED FOR AN HOUR."
;  "THEY PLAYED IN AN HOUR."
  (OR (AND (IS-IT NP 'TIME) (EQ (CAR (GETR TNS)) 'PAST) (NOT (DATE NP)))
      (AND (IS-IT NP 'TIME) (EQ (CAR (GETR TNS)) 'PRESENT) (HABITUAL?))))
```

```
(DEFUN ELMT (N K-S)
;  THIS RETURNS THE NTH ELEMENT OF THE LIST K-S.
  (PROG NIL
      LOOP
        (COND ((OR (NULL K-S) (LESSP N 1)) NIL)
              ((LESSP 1 N) (SETQ N (SUB1 N) K-S (CDR K-S)) (GO LOOP))
              (T (RETURN (CAR K-S))))))


(DEFUN END-LIST (TEMP)
;  THIS RETURNS THE LAST ELEMENT OF A LIST.
  (PROG NIL LOOP (COND ((NULL (CDR TEMP)) (RETURN TEMP)) (T (SETQ TEMP (CDR TEMP)) (GO LOOP)))))


(DEFUN ERGATIVE (VERB)
;  THIS  IS THE DICTIONARY ENTRY FOR ERGATIVE VERBS.
  (IF ((AGENT (MUST-BE ANIMATE))
       AG
       SOMEONE
       (COND ((AND (EQ 'SOMEONE (GETR AG)) (AGENT (NOT (SHOULD-BE HUMAN))))))
       INST
       OPT
       (PATIENT (NOT (SHOULD-BE HUMAN)))
       PA
       (PROGN (SETR PA (GETR INST)) (SETR INST NIL) 'OPT))
      (BUILDQ (@ ("<==>") (?) ("+") (("<--" # ?)) #)
              AG
              TNS
              VERB
              PA
              (COND ((NULL (GETR INST)) NIL)
                    (T (LIST (LIST '"<-INSTRUMENT" (NOUN-PUT 'INST))))))))


(DEFUN EV-SETR FEXPR (X)
;  THIS ROUTINE EVALS (CAR X) AND SETR'S IT TO (CADR X).
  (PROG (TEMP)
        (SETQ REGS (CONS (CONS (EVAL (CAR X))
                               (SETQ TEMP (COND ((ATOM (SETQ TEMP (EVAL (CADR X)))) TEMP)
                                                (T (MYEVAL (EVAL (CADR X)))))))
                         REGS))
        (RETURN TEMP)))


(DEFUN EXCHANGE (TESTEE PREP)
;  THIS IS THE EXCHANGE CASE.
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PRO-ANAPHOR? '(PATIENT EXCHANGE BENEFICIARY AGENT)
                                     '(NOT (SHOULD-BE HUMAN ABSTRACT))
                                     PRN
                                     TESTEE))
                      ((NOT (SHOULD-BE HUMAN ABSTRACT))
                       (SET-UP-FN 'EXCHANGE PREP)
                       (LIST '"<-EXCHANGE-" (NOUN-LIST-GET (NP-BUILD TESTEE))))))))


(DEFUN EXPECTED-EFFECT (TESTEE PREP)
;  THIS IS THE EXPECTED-EFFECT CASE.
;  "FRED PLAYED THE PIANO SO WE WOULD LEAVE."
  (PROG (SUPER-SENT)
        (SETQ SUPER-SENT (ALL-FLAG-OFF (COPY KEEP-SENT)))
        (COND ((AND (EQ 'SO PREP) (EQ (CAR TESTEE) 'WOULD))
               (PROG (TEMP)
                     (SETQ AG-ANAPHOR (GETR AG) PA-ANAPHOR (GETR PA))
                     (COND ((NOT (ATOM (SETQ TEMP (EVAL (CADR TESTEE)))))
                            (SET-UP-FN 'EXPECTED-EFFECT PREP)
                            (SETQ AG-ANAPHOR NIL PA-ANAPHOR NIL)
                            (RETURN (LIST '"<-EXPECTED-EFFECT-" TEMP))))))))


(DEFUN FIND-ANAPHOR (HIST K TEST PRN HIST-SNT)
  (PROG (TP TEMP)
      LOOP
        (COND ((NULL HIST-SNT) (RETURN NIL))
              ((AND (OR (AND (NULL K) (NOT (CADAR HIST-SNT))) (EQ K 'CHECK))
                    (SETQ TEMP (ANAPHOR? (CAR HIST-SNT) NIL HIST-SNT TEST PRN))
                    (PROGN (FLAG-ON (CAR HIST-SNT) (CAR SENT-HIST) T) T)
                    TEMP))
              ((AND (NOT (NULL K))
                    (NOT (EQ 'CHECK K))
                    (SETQ TEMP (CASE-FIND HIST (CAR K)))
                    (SETQ TP (ANAPHOR? TEMP (CAR K) HIST TEST PRN))
                    TP))
              ((AND (NOT (NULL K))
                    (NOT (EQ 'CHECK K))
                    (SETQ TEMP (NOUN-LIST-GET (SECOND-CHANCE (CAR K) PRN HIST-SNT HIST)))))
              ((AND (NOT (NULL K)) (NOT (EQ 'CHECK K)) (THIRD-CHANCE (CAR K) PRN HIST)))
              ((NULL K) (SETQ HIST-SNT (CDR HIST-SNT)) (GO LOOP))
              ((EQ K 'CHECK)
                (COND ((NOT (MEMBER '(SO-COMP THAT-COMP TO-COMP BY-COMP)
                                    (CAAR (SETQ HIST-SNT (CDR HIST-SNT)))))
                       (GO LOOP))
                      (T (RETURN NIL))))
              ((NOT (NULL (SETQ K (CDR K)))) (GO LOOP)))))
```

```
(DEFUN FIND-LIST (L A)
  (COND ((OR (NULL L) (ATOM L)) NIL)
        ((EQ A (CAR L)) L)
        ((FIND-LIST (CAR L) A))
        (T (FIND-LIST (CDR L) A))))


(DEFUN FLAG-OFF (X K-S)
;  THIS TURNS X'S FLAG OFF.
  (PROG NIL
      LOOP
        (COND ((OR (NULL K-S) (EQ 'PARTICLE (CAAR K-S))) X)
              ((EQUAL X (CAR K-S))
                (RPLACA (CDAR K-S) NIL)
                (COND ((MEMBER (CAR X) '(REL OBJ-REL RED-REL OBJ-RED-REL))
                       (ALL-FLAG-OFF (START-REL (CAR K-S)))))
                X)
              (T (SETQ K-S (CDR K-S)) (GO LOOP)))))


(DEFUN FLAG-OFF-HIST (A B)
  T)



(DEFUN FLAG-ON (P K-S FLAG)
;  THIS TURNS P'S FLAG ON (SETS IT TO FLAG).
  (PROG NIL
      LOOP
        (COND ((NULL K-S) NIL)
              ((EQUAL P (CAR K-S)) (RPLACA (CDAR K-S) FLAG) P)
              (T (SETQ K-S (CDR K-S)) (GO LOOP)))))


(DEFUN GET-UNFLGD-NP-AFTR-VB ()
;  THIS RETURNS THE FIRST UNFLAGGED NP AFTER THE VERB.
  (PROG (K)
        (SETQ K KEEP-SENT)
      LOOP
        (COND ((NULL K) (RETURN NIL))
              ((EQ (CAAR K) 'VP))
              (T (SETQ K (CDR K)) (GO LOOP)))
        (COND ((AND (NULL (CADAR (SETQ K (CADR K)))) (EQ 'NP (CAR K))) (RETURN K)))))


(DEFUN HABITUAL? ()
  (OR HABITUAL (DEEP-MEMBER '(USUALLY ALWAYS SOMETIMES OFTEN) INPUT#)))


(DEFUN IF FEXPR (X)
;  IF IS THE MAIN FUNCTION USED IN DEFINING CASE FRAMES.
;  IT TAKES AN EVEN NUMBER OF ARGUMENTS AND APPLIES CASE TO
;  THE FIRST OF THE PAIRS, BUILDS THE STRUCTURE ASSOCIATED WITH THAT
;  CASE FRAME, CALLS TAIL-ENDS, AND RETURNS THE STRUCTURE THUS BUILT.
;  IF ANY OF THESE FAILS, IT GOES ON TO THE NEXT PAIR.
  (PROG NIL
      LOOP
        (COND ((NULL X) NIL)
              ((AND (APPLY1 'CASE (CAR X)) (SETQ STRUCT (EVAL (CADR X))) (TAIL-ENDS)) (RETURN STRUCT))
              (T (SETQ X (CDDR X)) (GO LOOP)))))


(DEFUN IMPLODE (L)
  (COND ((EQ (LENGTH L) 1) (CAR L))
        (T (IMPLODE (APPEND (LIST (MKATOM (CAR L) (CADR L))) (CDDR L))))))


(DEFUN INDEFINITE-DET (D)
  (NOT (DEFINITE-DET D)))


(DEFUN IND-SUBJ (A F)
;  THIS CASE IS NOT IMPLEMENTED YET.
  NIL)


(DEFUN IND-OBJ NEXPR (TEST)
;  THIS FINDS AN INDIRECT OBJECT WHEN IT IS A NOUN PHRASE:
;  "FRED PLAYED JACK TENNIS."
  (PROG (TESTEE PRN TEMP)
        (COND ((NOT (NULL (SETQ TESTEE (GET-UNFLGD-NP-AFTR-VB))))
               (COND ((NULL (SETQ PRN (START-LIST 'PRO TESTEE)))
                      (COND ((OR (TEST-OFF? 'IND-OBJ) (EVAL TEST))
                             (FLAG-ON TESTEE KEEP-SENT T)
                             (RETURN TESTEE))
                            (T (SETQ FAIL-TEST T) NIL)))
                     (T (PRO-ANAPHOR? '(AGENT BENEFICIARY RECIPIENT CO-AGENT PATIENT) TEST PRN TESTEE)))))))
```

```
(DEFUN INFER? (A)
  A)

(DEFUN INSTRUMENT (TESTEE PREP)
;  THIS IS THE INSTRUMENT CASE.
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PROG (TEMP)
                             (COND ((SETQ TEMP (PRO-ANAPHOR? '(PATIENT INSTRUMENT)
                                                             '(INST-TEST TESTEE VERB PREP)
                                                             PRN
                                                             TESTEE))
                                    (SET-UP-FN 'INSTRUMENT PREP)
                                    (RETURN (LIST '"<-INSTRUMENT-" TEMP))))))
                      ((INST-TEST TESTEE VERB PREP)
                       (SET-UP-FN 'INSTRUMENT PREP)
                       (LIST '"<-INSTRUMENT-" (NOUN-LIST-GET (NP-BUILD TESTEE))))))))


(DEFUN INST-TEST (TESTEE VERB PREP)
  (AND (NOT (MUST-BE HUMAN)) (EQ (CAAR TESTEE) 'NP) (MUST-BE OBJECT)))


(DEFUN INTERSECT (A B)
;  THIS RETURNS THE INTERSECTION OF LISTS A & B.
  (PROG (TEMP)
        (COND ((NULL A) (RETURN NIL)))
    LOOP
        (COND ((NULL B) (RETURN NIL))
              ((MYOR (MAPCAR '(LAMBDA (X) (MEMBER X A))
                             (SETQ TEMP (APPEND (PROP-LIST B 'SUPERSET) (PROP-LIST B 'N-PROP) B))))
              (T (SETQ B (SUPERSET B)) (GO LOOP)))))


(DEFUN INT-LIST (A B)
  (COND ((OR (NULL A) (NULL B)) NIL)
        ((MEMBER (CAR A) B))
        (T (INT-LIST (CDR A) B))))


(DEFUN IS-IT (PH PROP)
  (MEMBER PROP (MYEVAL (GET (NOUN-GET PH) 'N-PROP))))


(DEFUN LEAVE-DEST (A)
;  THIS IS A STUB.  IN "FRED LEFT THE HOUSE."  THE DESTINATION IS
;  UNSPECIFIED.  A ROUTINE COULD BE WRITTEN TO TRY TO FIGURE OUT WHERE HE
;  WENT. (OUTSIDE?)
  'SOMEPLACE)


(DEFUN LEAVE-SOU (TESTEE)
  (COND ((MUST-BE ANIMATE) (LIST "<-LOC-OF-" (NOUN-PUT TESTEE)))
        (T (NOUN-PUT TESTEE))))
```

```
(DEFUN LETS-GO ()
;  THIS IS THE VERY TOP-LEVEL ROUTINE WHICH READS IN SENTENCES
;  AS STRINGS OF INDIVIDUAL ATOMS.  IT CONSIDERS A SENTENCE ENDED IF IT
;  FINDS A . ? OR ! AT THE END OF A WORD.  THESE ARE THE ONLY
;  PUNCTUATION MARKS ALLOWED.  IT CHECKS FOR A @ TO ABORT A SENTENCE,
;  OR A # TO LEAVE THIS MODE.  ANY LISP FORM WILL BE EVALED IN THIS MODE.
;  IF A ROUTINE IS NOT LOADED (VIA LOADER) IT WILL NOT BE EVALED.
  (PROG (HISTORY INPUT## THAT-CMP WORD SUPER-SENT LEVEL W1 TM ANAPHOR-DEPTH FAIL-TEST-LIST TEST-OFF
         SPEAKER ANAPHOR-RESP REL-LEVEL SHOULD-TEST-OFF SENT-HIST TNS-FIX? AG-ANAPHOR PA-ANAPHOR)
        (PRINT '"HELLO. YOU ARE NOW IN CONVERSATIONAL MODE.")
        (PRINT '"ANY LISP FORMS WILL BE EVALED IN THIS MODE.")
        (PRINT '"YOU MAY RETURN TO LISP BY ENTERING A #.")
        (PRINT '"IF YOU ENTER A @ AS A WORD IN A SENTENCE, THAT SENTENCE")
        (PRINT '"                  WILL BE ABORTED")
        (VERB-CLEAN)
        (SETQ SPEAKER NIL)
      LOOP1
        (SETQ INPUT## NIL)
      LOOP2
        (SETQ WORD (READ))
        (COND ((NOT (ATOM WORD)) (PRINT (MYEVAL WORD)) (GO LOOP2))
              ((EQ WORD '#) (RETURN '"BYE BYE"))
              ((EQ WORD '@) (PRINT '"SENTENCE ABORTED") (GO LOOP1))
              ((NOT (MEMBER (CAR (SETQ W1 (REVERSE (EXPLODE WORD)))) '(. ? !)))
                (SETQ INPUT## (APPEND INPUT## (LIST WORD)))
                (GO LOOP2)))
        (SETQ THAT-CMP NIL
              INPUT## (APPEND INPUT## (LIST (IMPLODE (REVERSE (CDR W1)))))
              LEVEL 0
              ANAPHOR-DEPTH 0
              TEST-OFF NIL
              FAIL-TEST-LIST NIL
              TNS-FIX? NIL
              ANAPHOR-RESP NIL
              SHOULD-TEST-OFF NIL
              REL-LEVEL 0
              HISTORY (APPEND (COND ((ATOM (CAR (SETQ TM (LIST (INFER? (PRTY-PRINT (EVAL (MYPARSE INPUT##
                                                                                                  S))))))))
                                    NIL)
                                   (T TM))
                              HISTORY))
        (ALL-HIST-OFF SENT-HIST)
        (GO LOOP1)))


(DEFUN LIKE (TESTEE PREP)
;  THIS IS THE LIKE CASE.
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PROG (TEMP)
                             (COND ((SETQ TEMP (PRO-ANAPHOR? '(LIKE PATIENT TOPIC AGENT BENEFICIARY)
                                                             T
                                                             PRN
                                                             TESTEE))
                                    (SET-UP-FN 'LIKE PREP)
                                    (RETURN (LIST '"<-LIKE-" TEMP))))))
                      (T (SET-UP-FN 'LIKE PREP) (LIST '"<-LIKE-" (NOUN-LIST-GET (NP-BUILD TESTEE))))))))



(DEFUN LISTEN-TOPIC? (TESTEE)
;  THIS BUILDS THE TOPIC CASE FOR LISTEN.  "LISTEN TO BEETHOVEN"
;  MEANS LISTEN TO THE MUSIC COMPOSED BY BEETHOVEN. ETC.
  (COND ((NULL TESTEE) '(N (SOMETHING)))
        ((MUST-BE MUSIC) (NOUN-LIST-GET TESTEE))
        ((MUST-BE COMPOSER)
          (LIST 'MUSIC (LIST '"<==" (LIST '"<==>" (NOUN-LIST-GET TESTEE) 'COMPOSE 'MUSIC))))
        ((MUST-BE MUSICIAN)
          (LIST 'MUSIC (LIST '"<==" (LIST '"<==>" (NOUN-LIST-GET TESTEE) 'PLAY 'MUSIC))))
        (T (LIST 'SOUND (LIST '"<==" (LIST '"<==>" (NOUN-LIST-GET TESTEE) 'CAUSE 'SOUND))))))


(DEFUN LIST-IT (A)
  (COND ((NULL A) NIL)
        (T (LIST A))))


(DEFUN LIST-IT-1 (A)
  (COND ((NULL A) NIL)
        (T (COND ((EQ (LENGTH A) 2) (SETQ A (APPEND (CAR A) (CADR A))))
                 (T (SETQ A (CAR A))))
           (LIST (MAPCAR 'LIST-IT-2 A)))))


(DEFUN LIST-IT-2 (A)
  (COND ((OR (ATOM A) (NULL A)) A)
        ((AND (NOT (NULL (CADR A))) (NOT (EQ T (CADR A)))) A)
        (T (CONS (CAR A) (CDDR A)))))
```

```
(DEFUN LOCATION (TESTEE PRLP)
;  THIS IS THE LOCATION CASE.
  (PROG NIL
        (COND ((NOT (LOC-TEST PREP)) NIL)
              (T (COND ((AFTER-AND-MOD? PREP NN)
                        (SET-UP-FN 'LOCATION PREP)
                        (LIST '"<-LOC-" PREP (NOUN-LIST-GET (NP-BUILD TESTEE))))
                       (T (PROG (TEMP)
                                LOOP
                                  (SETQ TEMP (CHOOSE-ELMT-PREP TESTEE NN FAIL-LIST))
                                  (COND ((NULL TEMP) NIL)
                                        ((LOC-MOD? TESTEE TEMP PREP))
                                        (T (SETQ FAIL-LIST (APPEND FAIL-LIST (LIST NUM))) (GO LOOP))))))))))


(DEFUN LOC-MOD? (NP PHRASE PREP)
  (PROG (TEMP)
        (RETURN (COND ((EQ 'VP (CAR PHRASE))
                       (SET-UP-FN 'LOCATION PREP)
                       (LIST '"<-LOC-" PREP (NOUN-LIST-GET (NP-BUILD NP))))
                      ((OR (START-LIST 'N PHRASE) (START-LIST 'NPR PHRASE))
                       (SETQ PREP&CASE (LIST PREP 'LOCATION)
                             PLACE (NOUN-LIST-GET PHRASE)
                             FN '(RPLACD (END-LIST PLACE) (LIST STRCT))
                             NUM (PHRASE-NUM2 PHRASE KEEP-SENT))
                       (LIST '"<-LOC-" PREP (NOUN-LIST-GET (NP-BUILD NP))))))))


(DEFUN LOC-TEST (PREP)
  (OR (AND (EQ PREP 'IN) (SHOULD-BE CONTAINER BUILDING COUNTRY CITY CONTINENT STATE PROVINCE))
      (AND (EQ PREP 'AT) (NOT (MUST-BE HUMAN ABSTRACT)))
      (MUST-BE THING PLACE LOCATION)))


(DEFUN LOOK-DOWN (TESTEE PRN TEST)
;  THIS ROUTINE IS CALLED BY THIRD-CHANCE.  IT LOOKS BACK THROUGH
;  THE OLD PARSES TO RESOLVE ANAPHORIC REFERENCES.
  (PROG (TEMP)
        (COND ((OR (NULL TESTEE) (NULL TEST) (ATOM TESTEE)) NIL)
              ((EVAL TEST))
              ((AND (SETQ TEMP (LOOK-DOWN (CAR TESTEE) PRN TEST))
                    (ANAPHOR-QUERY PRN (NOUN-LIST-GET TEMP))
                    TEMP))
              ((AND (SETQ TEMP (LOOK-DOWN (CDR TESTEE) PRN TEST))
                    (ANAPHOR-QUERY PRN (NOUN-LIST-GET TEMP))
                    TEMP)))))


(DEFUN MAIN-WORD-GET (TESTEE)
;  THIS RETURNS THE MAIN WORD OF A PHRASE.
  (PROG (TEMP)
        (RETURN (COND ((SETQ TEMP (NOUN-GET TESTEE)) TEMP)
                      ((SETQ TEMP (START-LIST 'V TESTEE)) (CADR TEMP))
                      ((SETQ TEMP (START-LIST 'PRO TESTEE)) (CADR TEMP))))))


(DEFUN MAKE-ARROW (KASE)
  (IMPLODE (LIST '"<-" KASE '"-")))


(DEFUN MANNER (TESTEE PREP)
;  THIS IS THE MANNER CASE.
  (COND ((SETQ PRN (PRO? TESTEE)) NIL)
        ((MANNER-TEST TESTEE VERB PREP)
          (SET-UP-FN 'MANNER PREP)
          (LIST '"<-MANNER-" PREP (NOUN-LIST-GET (NP-BUILD TESTEE))))))


(DEFUN MANNER-TEST (TESTEE VERB PREP)
;  THIS IS THE TEST FOR THE MANNER CASE.
  (OR (AND (EQ 'GO VERB) (MUST-BE CONVEYANCE)) (MUST-BE ABSTRACT)))


(DEFUN METHOD (TESTEE PREP)
;  THIS IS THE METHOD CASE.
;  "THEY WENT BY CAR."
;  "FRED HURT MARY BY HITTING HER."
  (PROG (TEMP)
        (COND ((AND (EQ (CAAR TESTEE) 'NP) (MUST-BE CONVEYOR CONVEYANCE))
               (SET-UP-FN 'METHOD PREP)
               (LIST (MAKE-ARROW 'METHOD) PREP (NOUN-LIST-GET (NP-BUILD TESTEE))))
              ((AND (EQ 'BY PREP) (SETQ AG-ANAPHOR (GETR AG)))
               (SETQ PA-ANAPHOR (GETR PA))
               (COND ((NOT (ATOM (SETQ TEMP (EVAL (CONS 'S
                                                        (LIST (APPEND (LIST (COPY AG-ANAPHOR))
                                                                      (REL-MODS-JOIN TESTEE NN))))))))
                      (SET-UP-FN 'METHOD PREP)
                      (SETQ AG-ANAPHOR NIL PA-ANAPHOR NIL)
                      (LIST (MAKE-ARROW 'METHOD) TEMP)))))))
```

```
(DEFUN MODIFY? (PART N)
;  THIS IS CALLED IF SOMETHING OTHER THAN A PREPOSITIONAL PHRASE
;  OR A COMPLEMENT IS FOUND UN-FLAGGED IN THE TAIL-ENDS PROCESS.
;  ADVERB PHRASES, RELATIVE CLAUSES, AND CLASSIFIERS ARE HANDLED HERE.
;  THE SYSTEM DOES NOT HANDLE THINGS LIKE: "I WENT HOME LAST NIGHT."
;  SINCE THE NOUN-PHRASE "LAST NIGHT" IS NOT HANDLED IN THIS ROUTINE.
;  IT WOULD BE SIMPLE TO CHANGE.
  (PROG (TEMP NUM)
        (RETURN (COND ((AND (EQ (CAR PART) '*<-ADV-*) (SETQ STRCT (ADV-MOD? PART T NIL)) (CONTINUE)))
                      ((AND (EQ 'REL (CAR PART))
                            (SETQ STRCT (REL-CLAUSE (REL-MODS-JOIN PART N) N))
                            (CONTINUE)))
                      ((AND (EQ 'RED-REL (CAR PART))
                            (SETQ STRCT (RED-REL-CLAUSE (REL-MODS-JOIN PART N) N))
                            (CONTINUE)))
                      ((AND (EQ 'OBJ-REL (CAR PART))
                            (SETQ STRCT (OBJ-REL-CLAUSE (REL-MODS-JOIN PART N) N))
                            (CONTINUE)))
                      ((AND (EQ 'OBJ-RED-REL (CAR PART))
                            (SETQ STRCT (ORJ-RED-REL-CLAUSE (REL-MODS-JOIN PART N) N))
                            (CONTINUE)))
                      ((AND (EQ 'NP (CAR PART)) (SETQ STRCT (CLASS? PART)) (CONTINUE)))
                      (T (PRINL '*MODIFY?) (PRINL PART) (TERPRI))))))


(DEFUN MORPH (WORD)
;  THIS IS THE MORPHOLOGY ROUTINE.
  (COND ((GET WORD 'DICT))
        (T (SETQ LETTERS (REVERSE (EXPLODE WORD)))
           (COND ((MORPHIT MORPH-TABLE LETTERS) T)
                 (T (QUERY WORD))))))


(DEFUN MORPHIT (M-T LETS)
  (COND ((NULL M-T) NIL)
        ((MYEQ (CAAR M-T) LETS) (COND ((NOTNIL (DROP&GO (CAR M-T) LETS))) (T (MORPHIT (CDR M-T) LETS))))
        (T (MORPHIT (CDR M-T) LETS))))


(DEFUN MUST-BE FEXPR (A)
  (PROG (TEMP) (COND ((INTERSECT A (SETQ TEMP (NOUN-MEAN (NOUN-GET TESTEE)))) (RETURN TEMP)))))


(DEFUN MYEQ (LST LETS)
  (COND ((NULL LST))
        ((NOT (EQ (CAR LST) (CAR LETS))) NIL)
        (T (MYEQ (CDR LST) (CDR LETS)))))


(DEFUN MYEVAL (Z)
;  THIS TAKES A FORM AND CALLS SOFT-EVAL ON IT IF IT IS A LIST
;  WITH A FUNCTION NAME STARTING IT.
;  OTHERWISE IT JUST RETURNS THE FORM.
  (PROG (Q FN TEMP)
        (COND ((ATOM Z) Z)
              ((OR (SETQ FN (GET (SETQ Q (CAR Z)) (SETQ TEMP 'EXPR)))
                   (SETQ FN (GET Q (SETQ TEMP 'SUBR)))
                   (SETQ FN (GET Q (SETQ TEMP 'FSUBR)))
                   (SETQ FN (GET Q (SETQ TEMP 'NSUBR))))
                (SOFT-EVAL FN Z TEMP))
              (T Z))))


(DEFUN MYGET (W L)
  (COND ((ATOM L) (GET W L))
        (T (GET W (CAR L)))))


(DEFUN MYGETR FEXPR (LST)
  (COND ((NULL (CDR LST)) (GETRV (EVAL (CAR LST))))
        ((NUMBERP (CADR LST))
          (COND ((ZEROP (CADR LST)) (GETRV (EVAL (CAR LST))))
                (T (CDR (ASSQ (CAR LST) (CADAR (NTH STACK (CADR LST))))))))
        (T (CDR (ASSQ (CAR LST) (CADR (ASSQ (CADR LST) STACK)))))))


(DEFUN MYOR (L)
  (COND ((NULL L) NIL)
        ((CAR L))
        (T (MYOR (CDR L)))))


(DEFUN MYPARSE FEXPR (L)
  (PROG (A :LEVEL :TAB ST)
        (SETQ :LEVEL 0)
        (COND ((NULL (CDR L)) (PERROR SECOND ARGUMENT TO PARSE IS UNSPECIFIED) (RETURN NIL))
              (T (SETQ ST (CADR L))))
        (SETQ A (#ERRSET (PARSE1 (EVAL (CAR L)))))
        (RETURN (COND ((ATOM A) NIL)
                      (T (COND (TEST-OFF (PRINT "THAT SEEMS STRANGE, BUT ANYWAY:"))) (CAR A))))))
```

```
(DEFUN MYPUT (L)
  (PUT (CAR L) 'GRAMMAR (CDR L)))


(DEFUN MYRETURN (WRD FLAG)
  (MYEVAL (GET WRD FLAG)))


(DEFUN NEXT (CLASS)
;  THIS RETURNS T IF THE NEXT WORD IN THE INPUT STREAM IS OF TYPE CLASS.
  (PROG NIL
        (COND ((ATOM CLASS) (SETQ CLASS (LIST CLASS))))
      LOOP
        (COND ((NULL CLASS) NIL)
              ((GET LEX (CAR CLASS)))
              (T (SETQ CLASS (CDR CLASS)) (GO LOOP)))))


(DEFUN NOTNIL (L)
  (COND ((NULL L) NIL)
        ((NOT (NULL (CAR L))))
        (T (NOTNIL (CDR L)))))


(DEFUN NOUN-GET (PH)
;  THIS ROUTINE AND THE NEXT TWO RETURN DIFFERENT LEVELS IN THE NOUN-PHRASE
;  STRUCTURE.
  (CADR (NOUN-LIST-GET PH)))


(DEFUN NOUN-LIST-GET (PH)
  (COND ((ATOM PH) PH)
        ((MEMBER (CAR PH) '(THAT-COMP TO-COMP)) (CADDR PH))
        (T (NOUN-LIST-GET-FULL PH))))


(DEFUN NOUN-LIST-GET-FULL (PH)
  (COND ((START-LIST 'N PH))
        ((START-LIST 'NPR PH))))


(DEFUN NOUN-MEAN (NOUN)
;  THIS ALLOWS THE USER TO PUT A FORM ON THE PROPERTY LIST OF A NOUN
;  WHICH WILL BE EVALED AND RETURNED AS THE MEANING OF THE NOUN
;  INSTEAD OF THE NOUN ITSELF.  THIS FEATURE IS NOT USED CURRENTLY.
  (PROG (TEMP) (COND ((SETQ TEMP (GET NOUN 'N-MEAN)) (EVAL TEMP)) (T (LIST NOUN)))))


(DEFUN NOUN-PUT (ZZ)
  (PROG (Z) (COND ((EQ '"<==>" (CAR (SETQ Z (GETRV ZZ)))) Z) (T (SOFT-NOUN-LIST-GET (NP-BUILD Z))))))


(DEFUN NP-BUILD (NP)
;  THIS TAKES A NOUN PHRASE AND BUILDS THE APPROPRIATE STRUCTURE
;  FOR IT.  ITS MAIN CONCERN IS WITH THE DETERMINER STRUCTURE.
  (PROG (DET W WWP)
        (SETQ WWP (START-LIST 'NP NP))
        (COND ((NULL WWP) (RETURN NP))
              (T (SETQ NP WWP)))
        (COND ((NULL (SETQ DET (CADR (START-LIST 'DET NP)))) NP)
              ((START-LIST '"<-DEFINITE-" NP) NP)
              ((START-LIST '"<-INDEFINITE-" NP) NP)
              ((START-LIST '"<-POSS-BY-" NP) NP)
              ((DEFINITE-DET? DET) (SETQ W (BUILDQ ("<-DEFINITE-" #) DET)))
              ((MEMBER DET '(A AN SOME ANY)) (SETQ W (BUILDQ ("<-INDEFINITE-" #) DET)))
              ((EQ 'POSSPRO (CAR DET))
                (SETQ W (BUILDQ ("<-POSS-BY-" #) (NOUN-GET (NP-BUILD (POSS-ANAPHOR DET NP)))))))
        (RETURN (COND ((NULL W) NP) (T (RPLACD (END-LIST (CADDDR NP)) (LIST W)) NP)))))


(DEFUN NTYPE NEXPR (Z)
  (EQ (GET NEW-WORD 'N) Z))


(DEFUN NUMAGREE (DET NOUN N-NPR NUM)
;  THIS CHECKS NUMBER AGREEMENT BETWEEN A DETERMINER OR PRONOUN
;  AND A NOUN.
  (COND ((NUMAGREE1 DET NOUN N-NPR NUM))
        ((OR (GET NOUN 'NUM) (EQ 'NONE (GET NOUN 'N))))))


(DEFUN NUMAGREE1 (DET NOUN N-NPR NUM)
  (COND ((EQ N-NPR 'NPR) (NULL DET))
        ((EQ NUM 'SG) (AND (NOT (NULL DET)) (NOT (MEMBER DET '(THESE THOSE)))))
        ((EQ NUM 'PL) (NOT (MEMBER DET '(A THAT THIS))))
        (T T)))
```

```
(DEFUN NUMBER? ()
  (PROG (TEMP) (COND ((SETQ TEMP (GETF NUMBER)) (LIST '"<-NUMBER-" TEMP)))))


(DEFUN OBJ-RED-REL-CLAUSE (REL POS)
;  THIS HANDLES OBJECTIVE REDUCED RELATIVE CLAUSES: "THE BALL JOHN HOLDS."
  (PROG (TEMP)
      LOOP
        (COND ((AND (SETQ TEMP (START-LIST 'NP (ELMT (SUB1 POS) KEEP-SENT)))
                    (OBJ-REL-EVAL (CDDR REL) TEMP)))
              (T (SETQ POS (SUB1 POS)) (GO LOOP)))))


(DEFUN OBJ-REL-CLAUSE (REL POS)
;  THIS HANDLES OBJECTIVE RELATIVE CLAUSES.: "THE BALL WHICH JOHN HOLDS."
  (PROG (TESTEE PRO)
        (SETQ PRO (CADR (CADDR REL)))
      LOOP
        (COND ((NULL (SETQ TESTEE (START-LIST 'NP (ELMT (SUB1 POS) KEEP-SENT)))) (RETURN NIL))
              ((OR (AND (MEMBER PRO '(WHICH THAT)) (NOT (MUST-BE HUMAN)))
                   (AND (MEMBER PRO '(WHOM WHOSE)) (MUST-BE HUMAN)))
               (OBJ-REL-EVAL (CDDDR REL) TESTEE))
              (T (SETQ POS (SUB1 POS)) (GO LOOP)))))


(DEFUN OBJ-REL-EVAL (REL OBJ)
;  THIS TAKES A SURFACE OBJECTIVE RELATIVE CLAUSE, BUILDS IT UP
;  INTO A STRUCTURE WHICH LOOKS LIKE A SENTENCE, PUTS THE OBJECT IN IT CORRERT
;  PLACE, AND CALL S ON IT.
;  "THE BALL WHICH JOHN HOLDS."
;  "THE LAKE INTO WHICH I JUMPED."
  (PROG (TEMP)
        (SETQ REL-LEVEL (ADD1 REL-LEVEL)
              AG-ANAPHOR (GETR AG)
              PA-ANAPHOR (GETR PA)
              TEMP (EVAL (APPEND (LIST 'S)
                                 (COND ((EQ 'PREP (CAAR REL))
                                        (LIST (APPEND (CDR REL)
                                                      (LIST (CONS 'PP
                                                                  (CONS NIL
                                                                        (CONS (CADAR REL)
                                                                              (LIST (COPY OBJ)))))))))
                                       (T (LIST (APPEND REL (LIST (COPY OBJ))))))))
              REL-LEVEL (SUB1 REL-LEVEL)
              AG-ANAPHOR NIL
              PA-ANAPHOR NIL)
        (COND ((NOT (NULL TEMP))
               (SETQ FN '(RPLACD (END-LIST (CAR (END-LIST PLACE))) (LIST STRCT))
                     PLACE (START-LIST 'NP OBJ))
               (RETURN TEMP)))))


(DEFUN OPT (TEST SUBST)
  (COND (TEST SUBST)
        (T 'OBL)))


(DEFUN PASSIVE ()
  (PROG (TEMP)
        (COND ((EQ 'BE (GETR V))
               (COND ((AND (SETQ TEMP (GETF PART)) (EQ TEMP 'PASTPART))
                      (PROGN (COND ((NOT (MEMBER 'PROGRESSIVE (GETR ASPECT))) (LIFTR HABITUAL T S)))
                             T)))))))


(DEFUN PASSIVE-AGENT (LST TEST NUMS VNUM)
;  THIS IS PASSED A LIST OF ALL OF THE PREP-PHRASES STARTING
;  WITH "BY" BY AGENT IF THE SENTENCE IS PASSIVE.  IT GOES THROUGH
;  THEM UNTIL IT FINDS ONE THAT PASSES TEST.  IT RETURNS NIL ON FAILURE.
  (PROG (TESTEE LL SH-F)
        (SETQ SH-F SHOULD-TEST-OFF)
      LOOP1
        (SETQ LL (RE-ORDER LST NUMS VNUM))
      LOOP2
        (COND ((AND (NULL LL) SHOULD-TEST-OFF) (RETURN NIL))
              ((NULL LL) (SETQ SHOULD-TEST-OFF T) (GO LOOP1))
              ((AND (SETQ TESTEE (CADR (CDDAR LL))) (OR (TEST-OFF? 'AGENT) (EVAL TEST)))
               (SETQ SHOULD-TEST-OFF SH-F)
               (FLAG-ON (CAR LL) KEEP-SENT T)
               (RETURN TESTEE))
              (T (SETQ LL (CDR LL)) (GO LOOP2)))))
```

```
(DEFUN PATH (TESTEE PREP)
;  THIS IS THE PATH CASE. IT IS ONLY APPLICABLE TO VERBS OF MOVEMENT.
  (PROG (PRN)
        (RETURN (COND ((NOT (EQUAL 'MOVE (CADR (START-LIST '"<--" STRUCT)))) NIL)
                      ((SETQ PRN (PRO? TESTEE))
                        (PRO-ANAPHOR? '(PATIENT TRAJECORY TOPIC)
                                      '(AND (NOT (MUST-BE HUMAN)) (SHOULD-BE CONVEYOR))
                                      PRN
                                      TESTEE))
                      ((AND (NOT (MUST-BE ANIMATE)) (SHOULD-BE CONVEYOR))
                        (SET-UP-FN 'PATH PREP)
                        (LIST '"<-PATH-" PREP (NOUN-LIST-GET (NP-BUILD TESTEE)))))))


(DEFUN PATIENT NEXPR (TEST)
;  THIS IS THE PATIENT CASE.  IT FINDS THE FIRST UNFLAGGED NP
;  AFTER THE VERB IN ACTIVE SENTENCES, OR THE FIRST IN THE SENTENCE
;  IN PASSIVE SENTENCES.
  (PROG (TESTEE LL TEMP PRN)
        (SETQ LL L)
        (COND ((GETR PASSIVE) (SETQ LL (REVERSE LL))))
        (COND ((ATOM (CAR LL)) (GO LOOP2)))
      LOOP1
        (COND ((AND (CDR LL) (NOT (ATOM (CADR LL)))) (SETQ LL (CDR LL)) (GO LOOP1)))
        (SETQ LL (CDR LL))
      LOOP2
        (COND ((AND (NOT (NULL (CDR LL))) (CADAR LL)) (SETQ LL (CDR LL)) (GO LOOP2)))
        (COND ((AND (EQ 'NP (CAAR LL)) (NOT (CADAR LL)))
               (COND ((NULL (SETQ PRN (START-LIST 'PRO
                                                  (SETQ TESTEE (COND ((AND (GETR PASSIVE)
                                                                           (EQ 'NP (CAADR LL)))
                                                                      (CADR LL))
                                                                     (T (CAR LL)))))))
                      (COND ((OR (TEST-OFF? 'PATIENT) (EVAL TEST))
                             (FLAG-ON TESTEE KEEP-SENT T)
                             (RETURN TESTEE))
                            (T (SETQ FAIL-TEST T
                                     FAIL-TEST-LIST (APPEND (LIST 'PATIENT) FAIL-TEST-LIST))
                               NIL)))
                     (T (COND ((NOT (NULL (SETQ TEMP (ANAPHORIC 'PATIENT TEST ANAPHOR-DEPTH PRN))))
                               (FLAG-ON TESTEE KEEP-SENT TEMP)
                               (RETURN TEMP))))))
              ((AND (GET VERB 'THAT-COMP) (SETQ TESTEE (TOP-LEVEL-START-LIST 'THAT-COMP KEEP-SENT)))
                (FLAG-ON TESTEE KEEP-SENT T)
                TESTEE))))


(DEFUN PERSONAL-PRN (PRN TESTEE)
  (COND ((GET PRN 'PERSONAL) (MUST-BE ANIMATE))
        (T (NOT (MUST-BE ANIMATE)))))


(DEFUN PHRASE-NUM (A L)
;  THIS RETURNS 0 IF A DOES NOT START A TOP-LEVEL LIST IN L.
;  OTHERWISE IT RETURNS THE NUMBER OF THE LIST THAT A STARTS.
  (PROG (NUM)
        (SETQ NUM 1)
      LOOP
        (COND ((NULL L) 0)
              ((EQ (CAAR L) A) NUM)
              (T (SETQ L (CDR L) NUM (ADD1 NUM)) (GO LOOP)))))


(DEFUN PHRASE-NUM2 (A L)
;  THIS RETURNS 0 IF A IS NOT A TOP-LEVEL ELEMENT OF L,
;  OTHERWISE IT RETURNS THE NUMBER IN THE LIST THAT A IS.
  (PROG (NUM)
        (SETQ NUM 1)
      LOOP
        (COND ((NULL L) 0)
              ((EQUAL A (CAR L)) NUM)
              (T (SETQ NUM (ADD1 NUM) L (CDR L)) (GO LOOP)))))
```

```
(DEFUN POSS-ANAPHOR (PRN NP)
;  THIS ROUTINE SYSTEMATICALLY GOES THROUGH A SENTENCE TRYING TO
;  RESOLVE A POSSESIVE PRONOUN REFERENCE.
  (PROG (TESTEE REG NN NUMP)
        (SETQ REG '(AG PA REC SOU BEN))
      LOOP1
        (COND ((NULL REG) (GO LOOP2))
              ((AND (SETQ TESTEE (GETRV (CAR REG))) (PRN-MATCH? PRN TESTEE) (ANAPHOR-QUERY PRN TESTEE))
                (RETURN TESTEE))
              (T (SETQ REG (CDR REG)) (GO LOOP1)))
      LOOP2
        (SETQ NN (PHRASE-NUM2 NP KEEP-SENT) NUMP 0)
        (COND ((ZEROP NN) (SETQ NN (DEEP-PHRASE-NUM NP KEEP-SENT))))
      LOOP3
        (SETQ NUMP (ADD1 NUMP))
        (COND ((NOT (LESSP NUMP NN)) (GO LOOP4))
              ((AND (SETQ TESTEE (ELMT NUMP KEEP-SENT))
                    (PRN-MATCH? PRN TESTEE)
                    (ANAPHOR-QUERY PRN TESTEE))
                (RETURN TESTEE))
              (T (GO LOOP3)))
      LOOP4
        (RETURN (ANAPHORIC NIL T ANAPHOR-DEPTH PRN))))


(DEFUN POSSESSION (AG PA)
;  THIS IS A START ON THE PROBLEM OF POSSESSION.  IN
;  "BEETHOVEN'S MUSIC" WE DON'T WANT TO SAY THAT BEETHOVEN OWNS
;  THE MUSIC, BUT THAT HE WROTE IT.
  (PROG (TESTEE)
        (SETQ TESTEE AG)
        (COND ((AND (MUST-BE COMPOSER) (SETQ TESTEE PA) (MUST-BE MUSIC)) 'COMPOSE)
              (T 'OWN))))


(DEFUN PREP-CASE-LOOK (KASE PREP TEST)
;  THIS ROUTINE TAKES A LIST OF CASES, A LIST OF PREPOSITONS, AND A TEST.
;  IT LOOKS THROUGH THE CURRENT SENTENCE FOR A PREP-PHRASE STARTING
;  WITH ONE OF THE PREPOSITIONS WHICH PASSES THE TEST.
;  IF NONE IS FOUND, IT LOOKS BACK INTO THE PREVIOUS SENTENCE.
  (PROG (KK)
        (SETQ KK KASE)
      LOOP
        (COND ((OR (NULL KASE) (NULL PREP)) (ANAPHOR-CASE-LOOK KK TEST))
              ((CLAUSE (CAR PREP) TEST (CAR KASE)))
              (T (SETQ KASE (CDR KASE) PREP (CDR PREP)) (GO LOOP)))))


(DEFUN PREP-CLAUSE (PREPS SNT)
;  THIS RETURNS A LIST OF ALL OF THE PHRASES IN A SENTENCE BEGINNING
;  WITH ANY OF THE LIST OF PREPS IN PREP.
  (PROG (LST)
      LOOP
        (COND ((NULL SNT) LST)
              ((AND (EQ 'PP (CAAR SNT)) (MEMBER (CADDAR SNT) PREPS))
                (SETQ LST (APPEND LST (LIST (CAR SNT))) SNT (CDR SNT))
                (GO LOOP))
              (T (SETQ SNT (CDR SNT)) (GO LOOP)))))


(DEFUN PREP-LIST-GET (PREP L)
  (PROG (TEMP)
      LOOP
        (COND ((NULL L) NIL)
              ((OR (AND (ATOM (SETQ TEMP (CAAR L))) (EQ TEMP PREP)) (MEMBER PREP TEMP)) (CAR L))
              (T (SETQ L (CDR L)) (GO LOOP)))))


(DEFUN PREP-REL-CLAUSE (REL PREP PRO POS)
;  THIS HANDLES "THE MAN IN WHOSE HOUSE. . ."
  (PROG (TESTEE)
      LOOP
        (COND ((NULL (SETQ TESTEE (START-LIST 'NP (ELMT (SUB1 POS) KEEP-SENT)))) (RETURN NIL))
              ((OR (AND (EQ 'WHICH PRO) (NOT (MUST-BE HUMAN)))
                   (AND (MEMBER PRO '(WHOM WHOSE)) (MUST-BE HUMAN)))
                (PREP-REL-EVAL REL PREP TESTEE))
              (T (SETQ POS (SUB1 POS)) (GO LOOP)))))


(DEFUN PREP-REL-EVAL (REL PREP OBJ)
  (PROG (TEMP)
        (SETQ REL-LEVEL (ADD1 REL-LEVEL)
              TEMP (EVAL (APPEND (LIST 'S) (LIST (APPEND REL (LIST (CONS 'PP (CONS PREP (COPY OBJ))))))))
              REL-LEVEL (SUB1 REL-LEVEL))
        (COND ((NOT (NULL TEMP))
                (SETQ FN '(RPLACD (END-LIST (CAR (END-LIST PLACE))) (LIST STRCT)) PLACE OBJ)
                (RETURN TEMP)))))
```

```
(DEFUN PREP-START (PART)
 (COND ((ATOM PART) NIL)
       ((EQ (CAR PART) 'PP))))


(DEFUN PREP-VERB-ASSO-COMPLX-RETURN (PH1 PH2 PH3)
  THIS CONVERTS "THE BOX WITH THE PENS IN IT" INTO
  "THE BOX, THE BOX CONTAINS THE PENS."
 (LIST '"<==>"
       (NOUN-LIST-GET (NP-BUILD PH1))
       'PRESENT
       (LIST '"<--"
             (SELECTQ (CADDR PH3) ('ON 'SUPPORT) ('IN 'CONTAIN) 'RELATION)
             (NOUN-LIST-GET (NP-BUILD PH2)))))


(DEFUN PRO? (TESTEE)
 (START-LIST 'PRO TESTEE))


(DEFUN PRO-ANAPHOR? (CSS TEST PRN TESTEE)
  THIS FLAGS A PRONOUN AS BEING HANDLED IF ANAPHORIC CAN RESOLVE IT.
 (PROG (TEMP)
       (RETURN (COND ((NOT (NULL (SETQ TEMP (ANAPHORIC CSS TEST ANAPHOR-DEPTH PRN))))
                      (FLAG-ON TESTEE KEEP-SENT TEMP)
                      TEMP)))))


(DEFUN PROP-LIST (L FLG)
  THIS RETURNS A LIST OF WHAT IS UNDER FLG ON THE PROPERTY
  LISTS OF ALL OF THE ELEMENTS IN L.
 (PROG (TEMP)
     LOOP
       (COND ((NULL L) TEMP)
             (T (SETQ TEMP (APPEND TEMP (GET (CAR L) FLG)) L (COR L)) (GO LOOP)))))


(DEFUN PRN-GEN (PRN)
;  THIS ROUTINE RETURNS A PRONOUN'S GENDER.
 (COND ((MEMBER PRN '(HE HIM HIS)) 'MASCULINE)
       ((MEMBER PRN '(SHE HER)) 'FEMININE)
       ((MEMBER PRN '(YOUR I THEY THEM WE MY)) 'BOTH)
       ((MEMBER PRN '(IT THAT THOSE)) 'NEITHER)
       (T (PRN-GEN-QUERY PRN))))


(DEFUN PRN-GEN-QUERY (PRN)
;  IF A PRONOUN GENDER IS NOT KNOWN, THE USER IS ASKED FOR IT, AND
;  THE ROUTINE PRN-GEN IS CHANGED.
 (PROG (TEMP)
       (PRINI 'IS)
       (PRINI PRN)
       (PRINI '"MASCULINE OR FEMININE?")
       (TERPRI)
     LOOP
       (PRINT '"ENTER MASCULINE, FEMININE, BOTH, OR NEITHER")
       (PRINT '"ENTER # TO ABORT")
       (COND ((NOT (MEMBER (SETQ TEMP (READ)) '(MASCULINE FEMININE BOTH NEITHER #))) (GO LOOP))
             ((EQ TEMP '#) (RETURN NIL)))
       (PRN-RPLACD 'PRN-GEN PRN (SELECTQ TEMP (MASCULINE 1) (FEMININE 2) (BOTH 3) (NEITHER 4)))
       (RETURN TEMP)))


(DEFUN PRN-MATCH? (PRN TESTEE)
;  THIS ROUTINE CHECKS TO SEE IF THE PRONOUN PRN MATCHES TESTEE.
 (AND (PRN-NUMGEN? (PRN-NUM (CADR PRN)) (PRN-GEN (CADR PRN)) TESTEE) (PERSONAL-PRN (CADR PRN) TESTEE)))


(DEFUN PRN-NUM (PRN)
;  THIS ROUTINE RETURNS A PRONOUN'S NUMBER.
 (COND ((MEMBER PRN '(IT HE SHE I HIM HER THAT HIS MY)) '(SG MASS))
       ((MEMBER PRN '(THEY THEM THOSE WE)) 'PL)
       ((MEMBER PRN '(YOU)) '(SGPL MASS))
       (T (PRN-NUM-QUERY PRN))))


(DEFUN PRN-NUMGEN? (NUM GEN TESTEE)
 (AND (SOFT-MEM NUM (CADR (START-LIST '"<-NUMBER-" TESTEE)))
      (SOFT-MEM GEN (CADR (START-LIST '"<-GENDER-" TESTEE)))
      (OR (AND (EQ GEN 'MASCULINE) (MUST-BE MALE))
          (AND (EQ GEN 'FEMININE) (MUST-BE FEMALE))
          (NOT (MEMBER GEN '(MASCULINE FEMININE))))))
```

```
(DEFUN PRN-NUM-QUERY (PRN)
  IF A PRONOUN'S NUMBER IS NOT KNOWN, THIS ROUTINE ASKS THE USER FOR IT.
 (PROG (TEMP)
       (PRIN1 "IS")
       (PRIN1 PRN)
       (PRIN1 "SINGULAR OR PLURAL")
       (TERPRI)
     LOOP
       (PRINT "ENTER SG, PL, SGPL")
       (PRINT "ENTER # TO ABORT")
       (COND ((NOT (MEMBER (SETQ TEMP (READ)) '(SG PL SGPL MASS #))) (GO LOOP))
             ((EQ TEMP '#) (RETURN NIL)))
       (PRN-RPLACD 'PRN-NUM PRN (SELECTQ TEMP (SG 1) (PL 2) (SGPL 3)))
       (RETURN TEMP)))


(DEFUN PRN-RPLACD (FCN PRN NUM)
  THIS ROUTINE ACTUALLY INSETS ADDED INFORMATION ABOUT NUMBER AND GENDER
  INTO FCN.
 (SELECTQ NUM
          (1 (RPLACD (END-LIST (CADADR (CDAADR (CADDR (GET FCN 'EXPR))))) (LIST PRN)))
          (2 (RPLACD (END-LIST (CADADR (CDAADR (CADDDR (GET FCN 'EXPR))))) (LIST PRN)))
          (3 (RPLACD (END-LIST (CADADR (CDAADR (CADDR (CDDR (GET FCN 'EXPR)))))) (LIST PRN)))
          (4 (RPLACD (END-LIST (CADADR (CDAADR (CADDR (CDDDR (GET FCN 'EXPR)))))) (LIST PRN)))))


(DEFUN PRTY-PRINT (LST)
  (COND ((ATOM LST) (PRINT LST))
        ((NOT (NULL LST)) (PRINTPARSE LST 0))
        (T (PRINT "THAT DID NOT PARSE")))
  LST)


(DEFUN PURPOSE (SENT PREP)
;  THIS IS THE PURPOSE CASE.
;  "I RAN HOME TO GET MY BOOK."
  (SELECTQ PREP
           ('TO
             (PROG (TEMP)
                   (RETURN (COND ((SETQ AG-ANAPHOR (GETR AG))
                                  (SETQ PA-ANAPHOR (GETR PA))
                                  (COND ((NOT (ATOM (SETQ TEMP (EVAL (CONS 'S
                                                                           (LIST (APPEND (LIST (COPY AG-ANAPHOR
                                                                                               (REL-MODS-JOIN SENT
                                                                                                              NN)))))

                                                (SET-UP-FN 'PURPOSE PREP)
                                                (SETQ AG-ANAPHOR NIL PA-ANAPHOR NIL)
                                                (LIST (MAKE-ARROW 'PURPOSE) TEMP)))))))))

           NIL))


(DEFUN PUT-IN-LIST (AT IND WHAT)
;  THIS ADDS WHAT TO THE LIST ON THE PROPERTY LIST OF AT UNDER IND.
  (PROG (TEMP)
        (COND ((NULL (SETQ TEMP (GET AT IND))) (PUT AT IND (LIST WHAT)))
              (T (PUT AT IND (APPEND (LIST WHAT) TEMP))))))


(DEFUN PUTON (W L)
  (PUT W 'DICT 'T)
  (PUT W
       (COND ((ATOM (CADR L)) (CADR L))
             (T (CADR (CADR L))))
       (COND ((EQUAL (CADDR (CDR L)) '*) '*)
             (T (APPEND (LIST NEW-WORD) (CDDDR L))))))


(DEFUN PUTTHEM (W L)
;  THIS IS USED IN READING IN THE DICTIONARY.  IT PUTS PAIRS ON THE PROPERTY LIST.
  (COND ((NULL L) (PUT W 'DICT 'T))
        (T (PUT W (CAR L) (CADR L)) (PUTTHEM W (CDDR L)))))


(DEFUN QUANTITY (TESTEE PREP)
;  THIS IS THE QUANTITY CASE: "WHAT COMES BY THE QUART?"
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PROG (TEMP)
                             (COND ((SETQ TEMP (PRO-ANAPHOR? '(QUANTITY)
                                                             '(MUST-BE QUANTITY)
                                                             PRN
                                                             TESTEE))
                                    (SET-UP-FN 'QUANTITY PREP)
                                    (RETURN (LIST '"<-QUANTITY-" PREP TEMP))))))
                      ((MUST-BE QUANTITY)
                        (SET-UP-FN 'QUANTITY PREP)
                        (LIST '"<-QUANTITY-" PREP (NOUN-LIST-GET (NP-BUILD TESTEE))))))))
```

```
(DEFUN QUERY (WORD)
;  THIS IS THE ROUTINE THAT QUERIES THE USER ABOUT A WORD THAT IT DOES
;  NOT KNOW.
  (PROG (L)
        (PRINI '"I DO NOT KNOW WHAT")
        (PRINI WORD)
        (PRINI 'MEANS)
        (TERPRI)
        (PRINT '"PLEASE ENTER A DICTIONARY ENTRY FOR IT'S ROOT WORD")
        (PRINT '"OR ANOTHER WORD")
        (PRINT '"ENTER @ TO ABORT THIS SENTENCE.")
        (SETQ L (READ))
        (COND ((EQ L '@) (PRINT '"SENTENCE ABORTED") NIL)
              ((ATOM L) (MORPH L) L)
              (T (PUTTHEM (CAR L) (CDR L)) (MORPH WORD)))))


(DEFUN RATE (A S))
;  THIS CASE HAS NOT BEEN IMPLEMENTED YET.


(DEFUN READIN ()
;  THIS READS IN THE DICTIONARY FROM BROC:DICT .
  (PROG (L)
        (OPEN (BUF 255 BROC:DICT))
      LOOP
        (SETQ L (READ BUF))
        (COND ((NOT (NULL L)) (PUTTHEM (CAR L) (CDR L)) (GO LOOP)))))


(DEFUN RECIPIENT (TESTEE PREP)
;  THIS IS THE RECIPIENT CASE.
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PROG (TEMP)
                             (COND ((SETQ TEMP (PRO-ANAPHOR? '(AGENT BENEFICIARY
                                                               CC-AGENT
                                                               RECIPIENT
                                                               TOPIC
                                                               PATIENT)
                                                             '(MUST-BE ANIMATE)
                                                             PRN
                                                             TESTEE))
                                    (SET-UP-FN 'RECIPIENT PREP)
                                    (RETURN (LIST '"<-RECIPIENT-" TEMP))))))
                      ((MUST-BE ANIMATE)
                       (SET-UP-FN 'RECIPIENT PREP)
                       (LIST '"<-RECIPIENT-" (NOUN-LIST-GET (NP-BUILD TESTEE))))))))


(DEFUN RED-REL-CLAUSE (REL POS)
;  THIS ROUTINE SETS UP THE CALL TO REL-EVAL FOR REDUCED RELATIVE CLAUSES.
  (PROG (TEMP T1 T2)
      LOOP
        (COND ((AND (SETQ TEMP (START-LIST 'NP (ELMT (SUB1 POS) KEEP-SENT)))
                    (COND ((NULL (SETQ T1 (CADR TEMP))))
                          (T (FLAG-OFF TEMP KEEP-SENT)))
                    (SETQ T2 (REL-EVAL (CDDR REL) TEMP))
                    (COND (T1 (FLAG-ON TEMP KEEP-SENT T1))
                          (T (FLAG-OFF TEMP KEEP-SENT)))
                    T2))
              ((LESSP 1 POS) (SETQ POS (SUB1 POS)) (GO LOOP)))))


(DEFUN REL-CLAUSE (REL POS)
;  THIS SORTS OUT RELATIVE CLAUSES FROM PREPOSITIONAL RELATIVE CLAUSES.
  (PROG (RELPRO)
        (RETURN (COND ((EQ (CAR (SETQ RELPRO (CADDR REL))) 'RELPRO)
                       (REL-CLAUSE-MOD? (CDDDR REL) (CADR RELPRO) POS))
                      ((EQ (CAR RELPRO) 'PREP)
                       (PREP-REL-CLAUSE (CDDDDR REL) (CADR RELPRO) (CADR (CADDDR REL)) POS))))))
```

```
(DEFUN REL-CLAUSE-MOD? (REL PRO POS)
;  THIS SETS UP THE CALL TO REL-EVAL FOR RELATIVE CLAUSES.
  (PROG (TESTEE T1 T2)
      LOOP
        (COND ((AND (SETQ TESTEE (START-LIST 'NP (ELMT (SUB1 POS) KEEP-SENT)))
                    (COND ((NULL (SETQ T1 (CADR TESTEE))))
                          (T (FLAG-OFF TESTEE KEEP-SENT)))
                    (SETQ T2 (COND ((OR (AND (EQ 'WHO PRO) (MUST-BE HUMAN))
                                        (AND (EQ 'WHICH PRO) (NOT (MUST-BE HUMAN))))
                                    (REL-EVAL REL TESTEE))))
                    (COND (T1 (FLAG-ON TESTEE KEEP-SENT T1))
                          (T (FLAG-OFF TESTEE KEEP-SENT)))
                    T2))
              ((LESSP 1 POS) (SETQ POS (SUB1 POS)) (GO LOOP)))))


(DEFUN REL-EVAL (REL SUBJ)
;  THIS ROUTINE BUILDS UP A SENTENCE PARSE OUT OF THE RELATIVE CLAUSE, AND EVALS IT.
  (PROG (TEMP)
        (SETQ SUPER-SENT KEEP-SENT
              REL-LEVEL (ADD1 REL-LEVEL)
              AG-ANAPHOR (GETR AG)
              PA-ANAPHOR (GETR PA)
              TEMP (EVAL (APPEND (LIST 'S) (LIST (APPEND (LIST (COPY (START-LIST 'NP SUBJ))) REL))))
              REL-LEVEL (SUB1 REL-LEVEL)
              AG-ANAPHOR NIL
              PA-ANAPHOR NIL
              SUPER-SENT NIL)
        (COND ((NOT (NULL TEMP))
               (SETQ FN '(RPLACD (END-LIST (CAR (END-LIST PLACE))) (LIST STRCT))
                     PLACE (START-LIST 'NP SUBJ))
               (RETURN TEMP)))))


(DEFUN REL-MODS-JOIN (PART NN)
;  THIS ROUTINE TAKES A SENTENCE PART AND ITS ELEMENT NUMBER, AND
;  APPENDS TO IT ALL OF THE FOLLOWING PARTS WHICH ARE NP, PP, ETC.
;  IN THE PARTIAL PARSE RETURNED BY THE ATN, A RELATIVE CLAUSE
;  IS OFTEN BROKEN INTO SEVERAL PARTS.  THIS PUTS IT BACK TOGETHER AGAIN.
;  LATER, IF TOO MUCH IS ADDED ON IN THIS ROUTINE, THE END ONES ARE DROPPED
;  ONE AT A TIME.
  (PROG (TEMP)
      LOOP
        (COND ((AND (SETQ TEMP (ELMT (ADD1 NN) KEEP-SENT))
                    (MEMBER (CAR TEMP)
                            '(PP NP
                                 REL
                                 RED-REL
                                 OBJ-REL
                                 OBJ-RED-REL
                                 THAT-COMP
                                 TO-COMP
                                 SO-COMP
                                 BY-COMP
                                 THAN-PH
                                 "<-ADV-"
                                 "<-ADJ-"))
                    (NOT (CADR TEMP)))
               (SETQ PART (APPEND PART (LIST TEMP)) NN (ADD1 NN))
               (GO LOOP))
              (T (RETURN PART)))))


(DEFUN RE-ORDER (LST NUMS VNUMS)
  (COND ((NULL NUMS) LST)
        ((LESSP (CAR NUMS) VNUMS) (RE-ORDER (APPEND (CDR LST) (LIST (CAR LST))) (CDR NUMS) VNUMS))
        (T LST)))


(DEFUN S NEXPR (SENTENCE)
;  THIS ROUTINE IS THE VERY FIRST FUNCTION CALLED AFTER THE ATN RETURNS.
;  IT TAKES AN ATN PARSE AND INVOKES THE CASE ANALYSIS ON IT.
;  IT SETS UP A FEW GLOBAL VARIABLES, AND BUILDS A LIST L OF THE
;  NOUN PHRASES AND THE VERB.  THIS IS USED BY THE AGENT AND PATIENT ROUTINES.
  (PROG (N L SENT VERB KEEP-SENT STRUCT VP)
        (SETQ LEVEL (ADD1 LEVEL) N 0 SENT SENTENCE KEEP-SENT SENTENCE)
        (SETR PASSIVE
              (COND ((EQ 'PASSIVE
                         (CADR (START-LIST 'VOICE (SETQ VP (TOP-LEVEL-START-LIST 'VP KEEP-SENT))))))))
        (SETQ VERB (CADR (START-LIST 'V 'VP)))
        (SETR TNS (CADR (START-LIST 'TNS VP)))
        (COND (TNS-FIX? (TENSE-FIX (GETR TNS))))
      LOOP
        (COND ((NULL SENT)
               (SETQ TMP (WHICH-MEANING L SENTENCE N VERB))
               (COND ((EQ LEVEL 1) (SETQ SENT-HIST (APPEND (LIST (ALL-FLAG-OFF KEEP-SENT)) SENT-HIST))))
               (SETQ LEVEL (SUB1 LEVEL))
               TMP)
              ((EQ (CAAR SENT) 'NP)
               (SETQ L (APPEND L (LIST (CAR SENT))) SENT (CDR SENT) N (ADD1 N))
               (GO LOOP))
              ((EQ (CAAR SENT) 'VP)
               (SETQ L (APPEND L (LIST (SETQ VERB (CADR (START-LIST 'V (CAR SENT)))))) SENT (CDR SENT))
               (GO LOOP))
              (T (SETQ SENT (CDR SENT)) (GO LOOP))))
```

```
(DEFUN SECOND-CHANCE (KASE PRN H-S HIST)
;  IF ANAPHORIC FAILS TO RESOLVE A LOCATIVE REFERENCE, THIS ROUTINE
;  IS CALLED WHICH TRIES EVERY PHRASE IN THE SENTENCE.
  (SELECTQ KASE (LOCATION (ANAPHORIC NIL '(MUST-BE LOCATION) ANAPHOR-DEPTH PRN)) NIL))


(DEFUN SET-UP ()
;  SET-UP SETS UP THREE TABLES.  MORPH-TABLE IS THE SET OF MORPHOLOGY RULES.
;  CASE-TABLE ASSOCIATES WITH EACH PREPOSITION THE CASES IT MIGHT FLAG.
;  ANTI-CASE-TABLE IS A LIST OF VERBS WITH A SET OF CASES WHICH THEY WILL
;  NOT TAKE.
  (PROGN (SETQ MORPH-TABLE '(((S)
                              (MAPCAR 'TRY
                                      '((NIL N (NTYPE S) (NUMBER PL))
                                        (NIL V
                                             (OR (VBTYPE S-ED) (VBTYPE S-D))
                                             (TNS PRESENT)
                                             (PNCODE "3SG")))))
                             ((S E I)
                              (MAPCAR 'TRY
                                      '(((Y) N (NTYPE ES) (NUMBER PL))
                                        ((Y) V (VBTYPE ES-ED) (TNS PRESENT) (PNCODE "3SG")))))
                             ((S E)
                              (MAPCAR 'TRY
                                      '((NIL N (NTYPE ES) (NUMBER PL))
                                        (NIL V (VBTYPE ES-ED) (TNS PRESENT) (PNCODE "3SG")))))
                             ((S E V)
                              (MAPCAR 'TRY
                                      '(((E F) N (NTYPE IRR) (NUMBER PL))
                                        ((F) N (NTYPE IRR) (NUMBER PL)))))
                             ((S "'")
                              (MAPCAR 'TRY
                                      '((NIL N T (POSSESIVE T) (NUMBER SG))
                                        (NIL NPR T (POSSESIVE T) (NUMBER SG)))))
                             (("'" S)
                              (MAPCAR 'TRY
                                      '((NIL N T (POSSESIVE T) (NUMBER PL))
                                        (NIL NPR T (POSSESIVE T) (NUMBER PL)))))
                             ((Y L) (MAPCAR 'TRY '((NIL (ADJ ADV) 'T ER-EST) ((E L) (ADJ ADV) 'T *))))
                             ((Y L I) (TRY ((Y) (ADJ ADV) 'T *)))
                             ((Y L G N I) (TRY ((E) (V ADV) (VBTYPE S-D) *)))
                             ((G N I)
                              (MAPCAR 'EVAL
                                      '((TRY (NIL V 'T (PART PRESPART)))
                                        (TRY ((E) V (VBTYPE S-D) (PART PRESPART)))
                                        (COND ((EQ (CAR LETS) (CADR LETS))
                                               (DROP&GO (LIST (LIST (CAR LETS))
                                                              '(TRY (NIL V 'T (PART PRESPART))))
                                                        LETS))))))
                             ((D) (TRY (NIL V (VBTYPE S-D) (PART PASTPART) (TNS PAST))))
                             ((D E)
                              (MAPCAR 'EVAL
                                      '((TRY (NIL V
                                                  (OR (VBTYPE ES-ED) (VBTYPE S-ED))
                                                  (PART PASTPART)
                                                  (TNS PAST)))
                                        (COND ((EQ (CAR LETS) (CADR LETS))
                                               (DROP&GO (LIST (LIST (CAR LETS))
                                                              '(TRY (NIL V
                                                                         (OR (VBTYPE ES-ED)
                                                                             (VBTYPE S-ED))
                                                                         (PART PASTPART)
                                                                         (TNS PAST))))
                                                        LETS))))))
                             ((D E I) (TRY ((Y) V (VBTYPE ES-ED) (TNS PAST) (PART PASTPART))))
                             ((D) (TRY (NIL V (VBTYPE S-D) (TNS PAST) (PART PASTPART))))
                             ((R E)
                              (MAPCAR 'EVAL
                                      '((TRY (NIL ADJ (ADJTYPE ER-EST) (COMP-SUP COMPARATIVE)))
```

```
                                        (TRY (NIL ADV (ADVTYPE ER-EST) (COMP-SUP COMPARATIVE)))
                                        (COND ((EQ (CAR LETS) (CADR LETS))
                                               (DROP&GO (LIST (LIST (CAR LETS))
                                                              '(TRY (NIL ADJ
                                                                         (ADJTYPE ER-EST)
                                                                         (COMP-SUP COMPARATIVE))))
                                                        LETS))))))
                                ((R E I) (TRY ((Y) ADJ (ADJTYPE ER-EST) (COMP-SUP COMPARATIVE))))
                                ((R) (TRY (NIL ADJ (ADJTYPE R-ST) (COMP-SUP COMPARATIVE))))
                                ((T S E I) (TRY ((Y) ADJ (ADJTYPE ER-EST) (COMP-SUP SUPERLATIVE))))
                                ((T S) (TRY (NIL ADJ (ADJTYPE R-ST) (COMP-SUP SUPERLATIVE))))
                                ((T S E)
                                  (MAPCAR 'EVAL
                                          '((TRY (NIL ADJ (ADJTYPE ER-EST) (COMP-SUP SUPERLATIVE)))
                                            (TRY (NIL ADV (ADVTYPE ER-EST) (COMP-SUP SUPERLATIVE)))
                                            (COND ((EQ (CAR LETS) (CADR LETS))
                                                   (DROP&GO (LIST (LIST (CAR LETS))
                                                                  '(TRY (NIL ADJ
                                                                             'T
                                                                             (COMP-SUP SUPERLATIVE))))
                                                            LETS)))))))))
        (SETQ CASE-TABLE '((IN LOCATION
                               TIME
                               DESC
                               DURATION
                               MANNER
                               DESTINATION
                               CONTEXT
                               CHARACTERISTIC
                               WITH-RESP-TO)
                           (WITH CO-AGENT
                                 INSTRUMENT
                                 WITH
                                 MANNER
                                 DESC
                                 DESTINATION
                                 EXCHANGE
                                 IND-SUBJ
                                 PREREQUISITE-USED)
                           (FOR DURATION BENEFICIARY EXCHANGE IND-SUBJ)
                           (ON LOCATION DESC CHARACTERISTIC DESTINATION IND-SUBJ)
                           (TO RECIPIENT DESTINATION TOPIC PURPOSE IND-SUBJ)
                           (AT LOCATION DESC TIME RATE DESTINATION)
                           (OF DESC TOPIC SEC-OBJECT WITH-RESP-TO)
                           (BY LOCATION DESC TIME PATH METHOD QUANTITY)
                           (ABOUT TIME TOPIC PATH)
                           (INTO DESTINATION)
                           (ONTO DESTINATION)
                           (TOWARD DESTINATION)
                           (TOWARDS DESTINATION)
                           (NEAR LOCATION DESC TIME)
                           (BESIDE LOCATION DESC)
                           (DURING TIME)
                           (BEFORE TIME LOCATION DESC)
                           (AFTER TIME DESC)
                           (ALONG PATH)
                           (UP PATH DESC)
                           (FROM SOURCE DESC)
                           (DOWN PATH DESC)
                           (AROUND PATH DESC)
                           (ACROSS PATH DESC)
                           (SO EXPECTED-EFFECT CAUSE)
                           (UNDER LOCATION CONTEXT DESC)
                           (LIKE LIKE)
                           (DESPITE DESPITE)
                           (WHEN TIME)
                           (WHILE TIME)))
        (SETQ ANTI-CASE-TABLE '((GO (LOCATION)) (COME (LOCATION))))
        (GRAMMAR-IN)
        (READIN)))

(SET-UP)

(DEFUN SET-UP-FN (CS PRN)
;  WHEN A CASE HAS BEEN DETERMINED FOR A PREPOSIONAL PHRASE, IF THE CASE
;  IS OF THE STANDARD TYPE, SET-UP-FN IS INVOKED TO BUILD THE USUAL
;  FUNCTION FN TO PUT THE STRUCTURE STRCT INTO STRUCT CO-ORDINATE WITH
;  THE VERB.
  (PROG (TEMP)
        (SETQ PREP&CASE (LIST PRN CS)
              FN (COND ((AND (SETQ TEMP (START-LIST (MAKE-ARROW (CASE-ATOM-ASSOC CS)) STRUCT))
                             (MEMBER (CADR TEMP) '(SOMEONE SOMPLACE HERE)))
                       (PROGN (SETQ PLACE TEMP) '(RPLACD PLACE (LIST (CDR STRCT)))))
                       (T '(RPLACD (END-LIST (FIND-LIST STRUCT '"<==>")) (LIST STRCT))))
              NUM (PHRASE-NUM 'VP KEEP-SENT))))
```

```
(DEFUN SHOULD-BE FEXPR (A)
  (OR SHOULD-TEST-OFF (INTERSECT A (NOUN-MEAN (NOUN-GET TESTEE)))))

(DEFUN SIZE (NP)
  (COND ((GET (NOUN-GET NP) 'OBJ-SIZE))
        (T 0)))

(DEFUN SOFT-DELETE (ATS TEST L)
;  THIS ROUTINE REMOVES ALL ELEMENTS OF ATS IN L (OR ELEMENTS WHICH START
;  WITH ATS) WITHOUT SCREWING UP THE LIST THE WAY DELETE DOES.
  (COND ((AND (ATOM ATS) (NOT (NULL ATS))) (SETQ ATS (LIST ATS))))
  (COND ((OR (NULL L) (NULL ATS)) NIL)
        ((AND (MEMBER (CAR L) ATS) (EVAL TEST)) (SOFT-DELETE ATS TEST (CDR L)))
        ((AND (NOT (ATOM (CAR L))) (MEMBER (CAAR L) ATS) (EVAL TEST)) (SOFT-DELETE ATS TEST (CDR L)))
        (T (APPEND (LIST (CAR L)) (SOFT-DELETE ATS TEST (CDR L))))))

(DEFUN SOFT-EQ (A B)
  (OR (EQ A B) (NULL A) (NULL B)))

(DEFUN SOFT-EVAL (FN LST TP)
;  SOFT-EVAL TAKES A LIST STARTING WITH A FUNCTION NAME, AND WHERE
;  POSSIBLE CHECKS TO SEE THAT THE NUMBER OF ARGUMENTS IS CORRECT.
;  IF SO, IT EVALS IT, ELSE IT JUST RETURNS IT.
  (COND ((MEMBER TP '(SUBR FSUBR NSUBR)) (EVAL LST))
        ((AND (EQ TP 'EXPR) (OR (EQ (LENGTH (CADR FN)) (SUB1 (LENGTH LST))) (EQ 'FLAMBDA (CAR FN))))
          (EVAL LST))
        (T LST)))

(DEFUN SOFT-MEM (B A)
  (COND ((ATOM B) (SOFT-EQ A B))
        (T (OR (MEMBER A B) (NULL A) (NULL B)))))

(DEFUN SOFT-NOUN-LIST-GET (A)
  (COND ((NOUN-LIST-GET A))
        (T A)))

(DEFUN SOFT-NOUN-PUT (ZZ)
  (SOFT-NOUN-LIST-GET (NP-BUILD (GETR ZZ)) NIL))

(DEFUN SOU-DEST-PRO? (PRN TESTEE PREP VERB KASE)
  (PROG (TEMP)
        (COND ((SETQ TEMP (PRO-ANAPHOR? '(AGENT CO-AGENT BENEFICIARY) '(MUST-BE HUMAN) PRN TESTEE))
               (SET-UP-FN 'DESTINATION PREP)
               (RETURN (LIST (MAKE-ARROW KASE) (LIST 'LOCATION 'OF TEMP))))
              ((SETQ TEMP (PRO-ANAPHOR? NIL '(MUST-BE PLACE LOCATION BUILING) PRN TESTEE))
               (SET-UP-FN KASE PREP)
               (RETURN (LIST (MAKE-ARROW KASE) TEMP))))))

(DEFUN SOURCE (TESTEE PREP)
  (SOURCE-DEST TESTEE PREP 'SOURCE))

(DEFUN SOURCE-DEST (TESTEE PREP KASE)
;  THIS IS THE SOURCE AND DESTINATION CASE.
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE)) (SOU-DEST-PRO? PRN TESTEE PREP VERB KASE))
                      ((MUST-BE LOCATION PLACE BUILDING CONTAINER)
                       (SET-UP-FN KASE PREP)
                       (LIST (MAKE-ARROW KASE) (NOUN-LIST-GET (NP-BUILD TESTEE))))))))
```

```
(DEFUN SPEAKER (PRN)
;  IF A FIRST PERSONAL SINGULAR PRONOUN IS USED, THE SPEAKER IS ASKED
;  FOR HIS NAME.  IF HE IS NOT KNOWN TO THE SYSTEM, HE IS ASKED FOR HIS SEX.
  (COND ((NULL SPEAKER)
         (PROG (NAME N-PROP SEX SUPER RESP)
             LOOP1
               (PRINT '"WHAT IS YOUR NAME?")
               (SETQ NAME (READ))
               (COND ((NOT (ATOM NAME)) (PRINT '"DONT BE DUMB") (GO LOOP1)))
             LOOP2
               (COND ((NOT (MEMBER (CAR (GET NAME 'SUPERSET)) '(MAN WOMAN)))
                      (PRINT '"ARE YOU MALE OR FEMALE?")
                      (SETQ SEX (READ))
                      (COND ((MEMBER SEX '(MALE MASCULINE M MAN BOY GENTLEMAN))
                             (SETQ N-PROP '(HUMAN MALE) SUPERSET '(MAN)))
                            ((MEMBER SEX '(FEMALE F FEMININE WOMAN GIRL CHICK))
                             (SETQ N-PROP '(HUMAN FEMALE) SUPERSET '(WOMAN)))
                            ((MEMBER SEX '(BOTH NEITHER BI BISEXUAL))
                             (PRINT '"IN THAT CASE . . . ")
                             (STOP))
                            (T (PRINT '"WOULD YOU GIVE ME A STRAIGHT ANSWER") (GO LOOP2)))
                      (PUT NAME 'N-PROP N-PROP)
                      (PUT NAME 'SUPERSET SUPER)
                      (PUT NAME 'OBJ-SIZE '"3")
                      (PUT NAME 'DICT T)
                      (PUT NAME 'NPR '*)
                      (RETURN (SETQ SPEAKER (LIST 'NP NIL (LIST 'NPR NAME)))))
                     (T (RETURN (SETQ SPEAKER (LIST 'NP NIL (LIST 'NPR NAME))))))))
        (T SPEAKER)))


(DEFUN START-LIST (ZZ L)
;  THIS ROUTINE RETURNS THE FIRST SUB-LIST OF L BEGINNING WITH ZZ.
  (COND ((NULL L) NIL)
        ((EQ (CAR L) ZZ) L)
        ((ATOM L) NIL)
        ((START-LIST ZZ (CAR L)))
        (T (START-LIST ZZ (CDR L)))))


(DEFUN START-REL (L)
  (SETQ L (CDDR L))
  (COND ((EQ (CAAR L) 'RELPRO) (SETQ L (CDR L))))
  (COND ((EQ (CAAR L) 'PREP) (SETQ L (CDR L))))
  L)


(DEFUN SUPERSET (X)
;  THIS RETURNS A LIST OF ALL OF THE SUPERSETS OF THE ELEMENTS OF X.
  (PROG (L)
        (SETQ L NIL)
      LOOP
        (COND ((NULL X) L)
              ((ATOM X) (SUPERSET (LIST X)))
              (T (SETQ L (APPEND L (GET (CAR X) 'SUPERSET)) X (CDR X)) (GO LOOP)))))


(DEFUN SUPER-NP-BUILD (NP)
;  THIS ROUTINE CALLS NP-BUILD, BUT IF NP IS A PRONOUN IT RESOLVES THE REFERENCE FIRST.
  (PROG (TEMP)
        (COND ((SETQ TEMP (PRO? NP))
               (SETQ AG-ANAPHOR (GETR AG) PA-ANAPHOR (GETR PA))
               (SETQ TEMP (ANAPHORIC NIL T ANAPHOR-DEPTH TEMP))
               (SETQ AG-ANAPHOR NIL PA-ANAPHOR NIL)
               (RETURN (NP-BUILD TEMP)))
              (T (NP-BUILD NP)))))


(DEFUN SUPER-START-LIST (ZZ L TEST)
;  THIS IS THE SAME AS START-LIST, BUT TEST MUST BE TRUE FOR IT TO SUCCEED ON
;  A PART OF L.
  (COND ((NULL L) NIL)
        ((AND (EQ (CAR L) ZZ) (EVAL TEST)) L)
        ((ATOM L) NIL)
        ((SUPER-START-LIST ZZ (CAR L) TEST))
        ((SUPER-START-LIST ZZ (CDR L) TEST))))
```

```
(DEFUN TAIL-ENDS ()
;  TAIL-ENDS IS THE ROUTINE WHICH IS INVOKED WHEN IF THINKS IT HAS
;  FOUND THE CORRECT CASE FRAME FOR THE VERB.  IT LOOKS DOWN THE PARTIAL
;  PARSE FOR ELEMENTS WHICH HAVE NOT BEEN HANDLED YET.  CASES? IS INVOKED
;  FOR THEM.
  (PROG (SNR NN)
        (SETQ SNR KEEP-SENT NN 1)
      LOOP
        (COND ((NULL SNR))
              ((OR (CADAR SNR) (EQ 'VP (CAAR SNR)) (CASES? (CAR SNR) NN))
                (SETQ NN (ADD1 NN) SNR (CDR SNR))
                (GO LOOP))
              ((EQ 0 REL-LEVEL) (ALL-HIST-OFF SENT-HIST) NIL)
              (T T))))


(DEFUN TENSE-FIX (TNS)
  (PROG (TEMP)
      LOOP
        (COND ((NULL (SETQ TEMP (SUPER-START-LIST 'TNS KEEP-SENT '(NULL (CADR L))))) (RETURN NIL))
              (T (RPLACA (CDR TEMP) TNS) (GO LOOP)))))


(DEFUN TEST-OFF? (K)
  (AND TEST-OFF (NOT (GETR THAT-COMP)) (MEMBER K FAIL-TEST-LIST)))


(DEFUN THERE-IS ()
;  THIS IS A CASE OF "TO BE": "THERE IS A MAN IN THE HOUSE."
  (COND ((AND (EQ '"<-ADV-" (CAAR KEEP-SENT)) (EQ 'THERE (CADDAR KEEP-SENT)))
         (FLAG-ON (CAR KEEP-SENT) KEEP-SENT T)
         T)))


(DEFUN THIRD-CHANCE (KASE PRN HIST)
;  IF A LOCATIVE REFERENCE CANNOT BE RESOLVED, THIRD CHANCE LOOKS
;  BACK OVER HIST, THE PREVIOUS PARSES.
  (LOOK-DOWN HIST PRN (COND ((MEMBER KASE '(SOURCE DESTINATION LOCATION)) '(MUST-BE LOCATION)))))


(DEFUN TIME (NP PREP)
;  THIS IS THE TIME CASE.  W-W-TIME IS CALLED FOR THE POSSIBLY COMPLEX CONSTRUCTS.
  (COND ((AND (EQ PREP 'IN) (DURATION? NP)) NIL)
        ((MEMBER PREP '(BEFORE AFTER WHEN WHILE)) (W-W-TIME NP PREP))
        ((IS-IT NP 'TIME)
          (SETQ FN '(RPLACD (END-LIST (FIND-LIST STRUCT '"<==>")) (LIST STRCT)))
          (SETQ PREPCASE (LIST PREP 'TIME))
          (SETQ MOD (PHRASE-NUM 'VP KEEP-SENT))
          (TIME-BUILD NP PREP))))


(DEFUN TIME-BUILD (NP PREP)
  (BUILDQ ('"<-TIME-" # #) PREP (CADR (START-LIST 'N NP))))


(DEFUN TO-COMP (SUBJ)
;  THIS ROUTINE TAKES A TO-COMPLEMENT, ADDS SUBJ BEFORE IT, AND ANY
;  POSSIBLE PHRASES ON THE END OF IT, THUS BUILDING UP A SENTENCE PARSE,
;  AND INVOKES S ON IT.
  (PROG (K-S TEMP NN)
        (COND ((NULL SUBJ) (RETURN NIL)))
        (SETQ K-S KEEP-SENT NN 1)
      LOOP
        (COND ((NULL K-S) (RETURN NIL))
              ((AND (EQ 'TO-COMP (CAAR K-S))
                    (SETQ TEMP (EVAL (APPEND (LIST 'S)
                                             (LIST (APPEND (LIST SUBJ) (REL-MODS-JOIN (CDDAR K-S) NN)))))))
                    (FLAG-ON (CAR K-S) KEEP-SENT T)
                    TEMP))
              (T (SETQ NN (ADD1 NN) K-S (CDR K-S)) (GO LOOP)))))


(DEFUN TOPIC (TESTEE PREP)
;  THIS IS THE TOPIC CASE. THERE IS NO TEST ON THIS CASE.
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PRO-ANAPHOR? '(PATIENT AGENT TOPIC BENEFICIARY CO-AGENT) T PRN TESTEE))
                      (T (SET-UP-FN 'TOPIC PREP) (LIST '"<-TOPIC-" (NOUN-LIST-GET (NP-BUILD TESTEE))))))))
```

```
(DEFUN TOPIC-LOOK NEXPR (TEST)
;  THIS ROUTINE LOOKS FOR THE TOPIC CASE IF IT IS A NOUN-PHRASE.
  (PROG (TEMP TESTEE CLSS PRN)
        (SETQ CLSS (PREP-CLAUSE (CASE-PREPS VERB 'TOPIC) KEEP-SENT))
      LOOP
        (COND ((NULL CLSS) (RETURN NIL))
              ((NULL (SETQ PRN (START-LIST 'PRO (SETQ TESTEE (CAR CLSS)))))
                (COND ((OR (TEST-OFF? 'TOPIC) (EVAL TEST)) (RETURN (DBLC (NP-BUILD TESTEE))))
                      (T (SETQ CLSS (CDR CLSS)) (GO LOOP))))
              (T (COND ((NOT (NULL (SETQ TEMP (ANAPHORIC '(TOPIC PATIENT) TEST ANAPHOR-DEPTH PRN))))
                        (FLAG-ON TESTEE KEEP-SENT TEMP)
                        (DBLQ (RETURN TEMP)))
                       (T (SETQ CLSS (CDR CLSS)) (GO LOOP)))))))


(DEFUN TOP-LEVEL-START-LIST (AT L)
;  THIS IS THE SAME AS START-LIST, BUT IT LOOKS ONLY FOR AT IN THE
;  TOP-LEVEL OF L.
  (PROG NIL
      LOOP
        (COND ((NULL L) NIL)
              ((ATOM (CAR L)) (SETQ L (CDR L)) (GO LOOP))
              ((EQUAL (CAAR L) AT) (CAR L))
              (T (SETQ L (CDR L)) (GO LOOP)))))


(DEFUN TRY NEXPR (L)
;  THIS IS PART  OF THE MORPHOLOGY ROUTINES.  IT ADDS A LIST OF LETTERS ONTO
;  A ROOT WORD, AND CHECKS TO SEE IF IT IS A KNOWN WORD.
  (PROG (NEW-WORD)
        (SETQ NEW-WORD (IMPLODE (REVERSE (APPEND (CAR L) LETS))))
        (COND ((AND (GET NEW-WORD 'DICT) (MYGET NEW-WORD (CADR L)) (EVAL (CADDR L))) (PUTON WORD.L)))))


(DEFUN VBTYPE NEXPR (Z)
  (EQ (GET NEW-WORD 'V) Z))


(DEFUN VERB-CLEAN ()
  (MAPOB '(LAMBDA (X) (REM X 'CASE-HIST))))


(DEFUN VERB-PREP? (VERB PREP)
  (PROG (TEMP)
        (SETQ TEMP (GET VERB 'PREP-CASE))
      LOOP
        (COND ((NULL TEMP) NIL)
              ((EQ PREP (CAAR TEMP)))
              (T (SETQ TEMP (CDR TEMP)) (GO LOOP)))))


(DEFUN V-FORE-GRND (VERB PREP LST)
;  THIS ROUTINE TAKES A LIST OF CASES, AND FOREGROUNDS THOSE CASES
;  USED PREVIOUSLY WITH VERB AND FLAGGED BY PREP.  IT DOES NOT
;  TOUCH THE DESC CASE.
  (PROG (TEMP)
        (SETQ TEMP (GET VERB 'CASE-HIST))
      LOOP
        (COND ((NULL TEMP) (RETURN LST))
              ((AND (EQ PREP (CAAR TEMP)) (NOT (EQ 'DESCRIPTIVE (CADAR TEMP))))
                (RETURN (APPEND (LIST (CADAR TEMP)) (SOFT-DELETE (CADAR TEMP) T LST))))
              (T (SETQ TEMP (CDR TEMP)) (GO LOOP)))))


(DEFUN WEVAL (STRING1 STATE1 STACK1 REGS1 HOLD1 PATH1)
;  THIS IS PART OF RAY REITER'S ATN PARSER.  I MADE A SMALL CHANGE TO IT.
  (PROG (A BODY-S ARCTYPE * LEX)
        (PTRACE ENTERING STATE (STATE1) : STRING = (STRING1))
        (PTREGS)
        (SETQ * (COND ((NULL STRING1) NIL)
                      ((SETQ MORPHTEMP (MORPH (CAR STRING1)))
                        (COND ((EQ MORPHTEMP T) (CAR STRING1))
                              ((EQ MORPHTEMP '#) (RETURN NIL))
                              (T (RPLACA STRING1 MORPHTEMP))))
                      (T (ERROR (DICTERR)))))
              LEX *)
        (COND ((NOT (GET STATE1 'GRAMMAR)) (ERROR (ERROR2))))
        (SETQ BODY-S (GET STATE1 'GRAMMAR))
      TAG
        (COND ((NULL BODY-S) (BLOCKED) (RETURN NIL))
              (T (SETQ ARCTYPE (CAAR BODY-S))))
        (COND ((SETQ A (EVAL (CAR BODY-S))) (RETURN A))
              (T (SETQ BODY-S (CDR BODY-S)) (GO TAG))))
```

```
(DEFUN WHERE-IS (WHO)
;  THIS ROUTINE TRIES TO FIND OUT WHERE WHO IS.
  (PROG (HIST TEMP)
        (SETQ HIST HISTORY)
      LOOP
        (COND ((NULL HIST) (RETURN 'SOMEPLACE))
              ((OR (EQUAL WHO (CASE-FIND (CAR HIST) 'AGENT))
                   (EQUAL WHO (CASE-FIND (CAR HIST) 'CO-AGENT))
                   (EQUAL WHO (CASE-FIND (CAR HIST) 'BENEFICIARY)))
               (COND ((SETQ TEMP (WHERE-IS-1 WHO (CAR HIST)))
                      (RETURN (COND ((EQ (LENGTH TEMP) 1) (LIST 'QUOTE TEMP)) (T (DBLQ TEMP)))))
                     (T (SETQ HIST (CDR HIST)) (GO LOOP))))
              (T (SETQ HIST (CDR HIST)) (GO LOOP)))))


(DEFUN WHERE-IS-1 (WHO SENT)
  (OR (CASE-FIND SENT 'DESTINATION)
      (CASE-FIND SENT 'LOCATION)
      (AND (EQ 'INHABIT (CADR (START-LIST '"<--" SENT))) (CASE-FIND SENT 'PATIENT))))


(DEFUN WHICH-MEANING (L SENT N VERB)
;  THIS FUNCTION CONTROLS THE TOP-LEVEL BACK-UP.
;  THE FIRST LINE INVOKES THE DEFINITION OF THE VERB.
;  THE NEXT PART CHANGES ANAPHOR-DEPTH TO ALLOW ANAPHORIC TO LOOK
;  DEEPER INTO THE HISTORY OF THE CONVERSTATION TO RESOLVE
;  REFERENCES.  IT CANNOT GO BACK FURTHER THAN 2.
;  THE NEXT THING IT TRIES IS TO SHUT OFF THE SHOULD-BE TESTS.
;  AS A LAST ATTEMPT IT SHUTS OFF THE MUST-BE TESTS.
  (COND ((EVAL (GET VERB 'V-MEAN)))
        ((EQ 0 REL-LEVEL)
          (COND ((GREATERP (MIN 2 (LENGTH HISTORY)) ANAPHOR-DEPTH 0)
                 (SETQ ANAPHOR-DEPTH (ADD1 ANAPHOR-DEPTH))
                 (ALL-FLAG-OFF KEEP-SENT)
                 (WHICH-MEANING L SENT N VERB))
                ((NOT SHOULD-TEST-OFF)
                 (SETQ SHOULD-TEST-OFF T)
                 (ALL-FLAG-OFF KEEP-SENT)
                 (COND ((GREATERP ANAPHOR-DEPTH 0) (SETQ ANAPHOR-DEPTH 1)))
                 (WHICH-MEANING L SENT N VERB))
                ((NOT TEST-OFF)
                 (SETQ TEST-OFF T)
                 (ALL-FLAG-OFF KEEP-SENT)
                 (COND ((GREATERP ANAPHOR-DEPTH 0) (SETQ ANAPHOR-DEPTH 1)))
                 (WHICH-MEANING L SENT N VERB))
                (T '"I DONT GET IT")))))


(DEFUN WITH (TESTEE PREP)
;  THIS IS THE WITH CASE.
  (PROG (PRN)
        (RETURN (COND ((SETQ PRN (PRO? TESTEE))
                       (PROG (TEMP)
                             (COND ((SETQ TEMP (PRO-ANAPHOR? '(WITH PATIENT TOPIC AGENT)
                                                             '(NOT (MUST-BE ABSTRACT HUMAN))
                                                             PRN
                                                             TESTEE))
                                    (SET-UP-FN 'WITH PREP)
                                    (RETURN (LIST "<-WITH-" TEMP))))))
                      ((NOT (MUST-BE ABSTRACT HUMAN))
                       (SET-UP-FN 'WITH PREP)
                       (LIST '"<-WITH-" (NOUN-LIST-GET (NP-BUILD TESTEE))))))))


(DEFUN WITH-RESP-TO (A S)
;  THIS CASE IS NOT IMPLEMENTED YET.
  NIL)


(DEFUN WRITE? (A)
  (COND ((IS-IT A 'MUSIC) 'COMPOSE)
        (T 'WRITE)))


(DEFUN WRITE-IND-OBJ (IND PA)
;  THIS FINDS THE CASE OF AN INDIRECT OBJECT FOR THE VERB "TO WRITE".
;  IF MUSIC IS WRITTEN, THE INDIRECT OBJECT IS THE BENEFICIARY CASE, ELSE IT IS THE
;  RECIPIENT.
  (COND ((NULL IND) NIL)
        ((IS-IT PA 'MUSIC) (LIST "<-BENEFICIARY-" (NOUN-LIST-GET (NP-BUILD PA))))
        (T (LIST '"<-RECIPIENT-" (NOUN-LIST-GET (NP-BUILD PA))))))
```

```
(DEFUN W-W-TIME (NP PREP)
;  THIS HANDLES COMPLEX TIME CASES.
;  "BEFORE JOHN CAME. . ."
;  "WHILE I WALKED HOME. . "
  (PROG (PART RET)
        (SETQ PART (REL-MODS-JOIN NP (PHRASE-NUM2 (CONS PREP (CONS NIL NP)) KEEP-SENT)))
        (COND ((PROGN (SETQ AG-ANAPHOR (GETR AG) PA-ANAPHOR (GETR PA))
                      (SETQ RET (EVAL (CONS 'S (LIST PART))))
                      (SETQ AG-ANAPHOR NIL PA-ANAPHOR NIL)
                      (NOT (ATOM RET)))
               (SET-UP-FN 'TIME PREP)
               (RETURN (CONS '"<-TIME-" (CONS PREP (LIST RET))))))))
```

```
(FUN GRAMMAR-IN ()
(MAPC 'MYPUT
     '((S (CAT PARTICLE
               (AND (NOT (GETR PARTICLE))
                    (GETR VP)
                    (MEMBER * (GET (CAR (SETQ VV (CDR (START-LIST 'V (GETR VP))))) 'PARTICLES)))
               (ADDR SENT (BUILDQ (PARTICLE T *)))
               (RPLACA VV (MKATOM (CAR VV) '"-" *))
               (SETR PARTICLE T)
               (TO S))
          (WRD THAN T (TO THAN-PH))
          (WRD SO T (TO SO-COMP))
          (MEM (WHEN WHILE AFTER BEFORE) T (SETR WH-WH *) (TO WHEN-WHILE))
          (WRD IN T (TO IN-IN))
          (WRD BY T (TO BY-COMP))
          (WRD THAT (OR (NOT (GETR SENT)) (GETR VP)) (TO THAT-COMP))
          (WRD TO (OR (NOT (GETR SENT)) (GETR VP)) (TO TO-COMP))
          (PUSH PP (NEXT 'PREP) (ADDR SENT *) (TO S))
          (PUSH VP
                (NOT (GETR VP))
                (SENDR V (GETR V))
                (SENDR TENSE (GETR TENSE))
                (SETR VP *)
                (ADDR SENT *)
                (TO S))
          (PUSH REL T (ADDR SENT *) (TO S))
          (PUSH ADJP
                (OR (NEXT 'ADJ) (NEXT 'ADV) (NEXT 'V))
                (PROG (TEMP)
                      (SETQ TEMP *)
                    LOOP
                      (COND ((NULL TEMP))
                            (T (ADDR SENT (CAR TEMP)) (SETQ TEMP (CDR TEMP)) (GO LOOP))))
                (TO S))
          (PUSH ADV (NEXT 'ADV) (ADDR SENT *) (TO S))
          (PUSH NP T (SENDR NP T) (ADDR SENT *) (TO S))
          (JUMP S-YESNO (NEXT 'V))
          (POP (BUILDQ (S "+") SENT) (PROGN (SETQ HABITUAL (GETR HABITUAL)) T)))
     (WHEN-WHILE (PUSH NP T (SETR W-W (LIST (GETR WH-WH) NIL *)) (TO W-W-NP)))
     (W-W-NP (PUSH PP (NEXT 'PREP) (ADDR W-W *) (TO W-W-NP))
             (PUSH VP (NEXT 'V) (ADDR W-W *) (ADDR SENT (GETR W-W)) (TO S))
             (PUSH REL T (ADDR W-W *) (TO W-W-NP)))
     (THAN-PH (PUSH NP T (ADDR SENT (BUILDQ (THAN-PH NIL *))) (TO S)))
     (SO-COMP (WRD THAT T (SETR MODAL-FIND T) (TO SO-COMP))
              (PUSH S
                    T
                    (SENDR SO-COMP T)
                    (COND ((AND (GETR MODAL-FIND) (NOT (GETR MODAL))) NIL)
                          ((GETR MODAL) (ADDR SENT (BUILDQ (SO-COMP NIL "+" *) MODAL)))
                          ((AND (NOT (GETR MODAL-FIND)) (NOT (GETR MODAL)))
                            (ADDR SENT (BUILDQ (SO-COMP NIL *)))))
                    (TO S)))
     (THAT-COMP (PUSH S
                      T
                      (SENDR THAT-COMP T)
                      (SENDR TENSE (GETR TENSE))
                      (COND ((NOT (NULL *))
                             (PROG (THAT-COMP)
                                   (SETR THAT-COMP T)
                                   (ADDR SENT (BUILDQ (THAT-COMP NIL #) (EVAL *))))
                             (SETQ THAT-CMP T)))
                      (TO S)))
     (TO-COMP (PUSH VP
                    (NEXT 'V)
                    (SENDR TENSE (GETR TENSE))
                    (SENDR TO-COMP T)
                    (SETR TO-VP *)
                    (TO TO-COMP-VP)))
     (TO-COMP-VP (PUSH PP
                       (NEXT 'PREP)
                       (SETR TO-COMP T)
                       (ADDR SENT (BUILDQ (TO-COMP NIL "+" *) TO-VP))
                       (TO S))
                 (PUSH NP
                       T
                       (SETR TO-COMP T)
                       (ADDR SENT (BUILDQ (TO-COMP NIL "+" *) TO-VP))
                       (TO S))
                 (JUMP S T (ADDR SENT (BUILDQ (TO-COMP NIL "+") TO-VP))))
     (IN-IN (WRD ORDER T (TO IN-ORDER)))
     (IN-ORDER (WRD TO T (TO TO-COMP)))
     (BY-COMP (CAT V
                   (EQ 'PRESPART (GETF PART))
                   (SETR S-V (GETR V))
                   (COND ((GETR V))
                         (T (SETQ TNS-FIX? T)))
                   (SETR TENSE (GETR TENSE))
                   (SETR V *)
                   (SETR BY-V (BUILD-VP))
                   (SETR V (GETR S-V))
```

```
                (TO BY-COMP-V)))
(BY-COMP-V (PUSH NP
                T
                (COND ((NOT (NULL *)) (ADDR SENT (BUILDQ (BY-COMP NIL "+" *) BY-V))))
                (TO S)))
(S-PPU-POP (POP (BUILD-PPU) T))
(S-PP (JUMP S-YESNO (AND (GETR WH-PHRASE) (AUXV LEX))) (JUMP S T))
(S-DCL (PUSH NP T (SETR SUBJ *) (SETR S-SUBJ (GETR HEADNOUN)) (TO S-NP)))
(S-IMP (CAT V
            (GETF UNTENSED)
            (SETR SUBJ '(NP (DET NIL) (PRO YOU) (NU SG-PL)))
            (SETR V *)
            (SETR TENSE '(UNTENSED))
            (SETR S-SUBJ '(DUMMY YOU NIL))
            (SETR S-VERB (BUILDQ (@ (V *) (#)) FEATURES))
            (TO V-AUX)))
(S-NPU (PUSH NP T (SETR SUBJ *) (TO S-NPU-POP)))
(S-YESNO (CAT V
              (AUX*)
              (SETR V (BUILDQ (V *)))
              (SETR TYPE 'YES-NO)
              (SETR TENSE (LIST (GETF TNS)))
              (SETR YESNO T)
              (TO S-YESNO-NEG)))
(S-YESNO-NEG (WRD NOT T (SETR NEG '(NEGATIVE)) (TO S)) (JUMP S T))
(S-WH (CAT QADV
           (COND ((NULL (CDR STRING)) NIL)
                 ((AUXV (CADR STRING))))
           (ADDR ADVERBS (BUILDQ (QADV *)))
           (SETR TYPE '(QADV))
           (TO S-YESNO))
      (PUSH QP
           (AND (GET LEX 'QPRO) (NOT (ASSQ 'SUBJ (CDR (GET LEX 'QPRO)))))
           (COND ((NULL STRING) (ABORT))
                 ((NOT (AUXV LEX)) (ABORT)))
           (HOLD (APPEND * (GETR PREPPHRASES)))
           (SETR TYPE '(QPRO))
           (SETR QOBJ (GETR HEADQPRO))
           (SETR S-DO (GETR QOBJ))
           (SETR PREPPHRASES NIL)
           (TO S-YESNO))
      (PUSH QP
           (AND (GET LEX 'QPRO) (NOT (ASSQ 'OBJ (CDR (GET LEX 'QPRO)))))
           (COND ((NULL STRING) (ABORT))
                 ((NOT (GET LEX 'V)) (ABORT)))
           (SETR SUBJ (APPEND * (GETR PREPPHRASES)))
           (SETR S-SUBJ (GETR HEADQPRO))
           (SETR TYPE '(QPRO))
           (SETR PREPPHRASES NIL)
           (TO S-NP))
      (CAT QDET T (SETR QDET *) (TO S-QDET)))
(S-QDET (PUSH NP-DET
              T
              (SENDR DET (GETR QDET))
              (SENDR SEM-DET (BUILDQ (QDET "+") QDET))
              (SETR WH-NP (APPEND * (GETR PREPPHRASES)))
              (SETR TYPE '(QDET))
              (SETR PREPPHRASES NIL)
              (TO S-WH-NP)))
(S-WH-NP (JUMP S-YESNO
               (AUXV LEX)
               (SETR S-DO (GETR HEADNOUN))
               (HOLD (GETR WH-NP))
               (SETR QOBJ (GETR HEADNOUN)))
         (JUMP S-NP (GET LEX 'V) (SETR SUBJ (GETR WH-NP)) (SETR S-SUBJ (GETR HEADNOUN))))
(REL (MEM (WHICH WHO WHOM THAT WHOSE) T (SETR RELPRO (BUILDQ (RELPRO *))) (TO REL-PRO))
     (CAT PREP (SETR PREP (BUILDQ (PREP *))) (TO REL-PREP))
     (JUMP RED-REL (NEXT '(N NPR DET PRO V))))
(REL-PREP (MEM (WHICH WHOM WHOSE) (SETR RELPRO (BUILDQ (RELPRO *))) (TO REL-PRO)))
(RED-REL (CAT V
              (EQ 'PRESPART (GETF PART))
              (SETR V *)
              (SETR TENSE '(PRESENT))
              (SETR VP (BUILD-VP))
              (SETR REDUCED T)
              (TO REL-VP))
         (CAT V
              (EQ 'PASTPART (GETF PART))
              (SETR V *)
              (SETR VOICE '(VOICE PASSIVE))
              (SETR TENSE '(PAST))
              (SETR VP (BUILD-VP))
              (SETR REDUCED T)
              (TO REL-VP)))
(REL-PRO (PUSH NP T (SETR OBJ-REL T) (SETR NPL *) (TO REL-NP1))
         (PUSH VP
              (AND (NOT (GETR PREP)) (MEMBER (CADR (GETR RELPRO)) '(WHICH WHO THAT)))
              (SETR VP *)
              (TO REL-VP)))
```

```
(REL-NP1 (PUSH PP (NEXT *PREP) (ADDR MOD-NP1 *) (TO REL-NP1))
         (PUSH VP T (SETR VP *) (TO REL-NP1-VP))
         (PUSH REL T (SETR MOD-NP1 *) (TO REL-NP1)))
(REL-NP1-VP (POP (BUILDQ (@ # (NIL) # # ("+") # ("+"))
                         (LIST (COND ((AND (GETR REDUCED) (GETR OBJ-REL)) *OBJ-RED-REL)
                                     ((GETR REDUCED) *RED-REL)
                                     ((GETR OBJ-REL) *OBJ-REL)
                                     (T *REL)))
                         (LIST-IT (GETR RELPRO))
                         (LIST-IT (GETR PREP))
                         NP1
                         (LIST-IT (GETR MOD-NP1))
                         VP)
                 T))
(REL-VP (PUSH ADJP
              (AND (OR (NEXT *ADJ) (NEXT *ADV)) (EQ *BE (CADR (START-LIST *V (GETR VP)))))
              (MAPC *(LAMBDA (X) (ADDR ADJ X)) *)
              (TO REL-VP-NP2))
        (JUMP REL-VP-NP2 T))
(REL-VP-NP2 (POP (BUILDQ (@ # (NIL) # # # ("+") "+" #)
                         (LIST (COND ((AND (GETR REDUCED) (GETR OBJ-REL)) *OBJ-RED-REL)
                                     ((GETR REDUCED) *RED-REL)
                                     ((GETR OBJ-REL) *OBJ-REL)
                                     (T *REL)))
                         (LIST-IT (GETR RELPRO))
                         (LIST-IT (GETR PREP))
                         (LIST-IT (GETR NP1))
                         VP
                         ADJ
                         (LIST-IT (GETR NP2)))
                 T))
(VP (JUMP V-AUX (GETR V))
    (CAT V
         (OR (GETF TNS) (AND (GETR TOFLAG) (GETF UNTENSED)))
         (COND ((NOT (GETR TENSE)) (SETR TENSE (LIST (GETF TNS)))))
         (COND ((AND (NOT (GETR TO-COMP))
                     (NOT (GETR THAT-COMP))
                     (NOT (GETR BY-COMP))
                     (NOT (GETR SO-COMP)))
                (LIFTR TENSE (GETR TENSE) S)))
         (SETR S-VERB (BUILDQ (@ (V *) (#)) FEATURES))
         (SETR V *)
         (TO V-AUX))
    (CAT MODAL
         T
         (SETR MODAL *)
         (LIFTR MODAL * SO-COMP)
         (COND ((MEMBER * '(WOULD WILL SHALL)) (SETR TENSE '(FUTURE))))
         (TO VP))
    (JUMP VP-VP (GETR V)))
(S-NPU-POP (POP (BUILD-NPU) T))
(V-AUX (WRD NOT (VFEATURE (GETR V) *AUX) (SETR NEG '(NEGATIVE)) (TO V-AUX))
       (CAT V
            (PERFECT)
            (SETR V *)
            (SETR ASPECT '(PERFECT))
            (SETR S-VERB (BUILDQ (@ (V *) (#)) FEATURES))
            (TO V-AUX))
       (CAT V
            (PROGRESSIVE)
            (SETR V *)
            (SETR ASPECT (APPEND (GETR ASPECT) '(PROGRESSIVE)))
            (SETR S-VERB (BUILDQ (@ (V *) (#)) FEATURES))
            (TO V-AUX))
       (CAT V
            (FUTURE)
            (SETR V *)
            (SETR TENSE '(FUTURE))
            (SETR S-VERB (BUILDQ (@ (V *) (#)) FEATURES))
            (TO V-AUX))
       (CAT V
            (PASSIVE)
            (SETR V *)
            (SETR S-VERB (BUILDQ (@ (V *) (#)) FEATURES))
            (SETR VOICE '(VOICE PASSIVE))
            (TO VP-VP))
       (CAT V
            (DO-AUX)
            (SETR V *)
            (SETR S-VERB (BUILDQ (@ (V *) (#)) FEATURES))
            (COND ((AND (NOT (GETR YESNO)) (NOT (GETR NEG))) (SETR MODAL '(MODAL DO))))
            (TO VP-VP))
       (CAT V
            (MODAL)
            (SETR MODAL (BUILDQ (MODAL "+") V))
            (SETR V *)
            (SETR S-VERB (BUILDQ (@ (V *) (#)) FEATURES))
            (TO V-AUX))
       (CAT PREP
            (MEMBER * (GET (GETR V) 'PARTICLES))
            (SETR V (MKATOM (GETR V) '"-" *))
```

```
            (TO VP+VP))
        (JUMP VP-VP T))
(VP-VP (POP (BUILD-VP) T))
(S-V (VIR NP T (COND ((NOT (V-DO-AGREEMENT)) (ABORT))) (SETR OBJ *) (TO S-V-TOCOMP))
     (PUSH NP
           (OR (VFEATURE (GETR V) 'TRANS) (VFEATURE (GETR V) 'COPULA))
           (SENDR MOTIONFLAG T)
           (SETR S-DO (GETR HEADNOUN))
           (COND ((NOT (V-DO-AGREEMENT)) (ABORT)))
           (SETR OBJ *)
           (TO S-V-TOCOMP))
     (PUSH ADJP
           (AND (VFEATURE (GETR V) 'COPULA) (OR (GET LEX 'ADV) (GET LEX 'ADJ)))
           (SETR V *)
           (COND ((NOT (PREDADJ-SEMANTICS (LIST (GETR HEADADJ)) (GETR S-SUBJ))) (ABORT)))
           (SETR S-VERB *)
           (TO S-PREDADJ))
     (CAT PREP
          (AND (NOT (GETR PARTICLE)) (MEMBER * (GET (GETR V) 'PARTICLES)))
          (SETR V (MKATOM (GETR V) '"-" *))
          (SETR S-VERB (BUILDQ (V "+") V))
          (SETR PARTICLE T)
          (TO S-V))
     (WRD THAT (VFEATURE (GETR V) 'THAT) (TO S-V-WRD=THAT))
     (WRD TO T (TO S-V-WRD=TO))
     (PUSH ADV (GET LEX 'ADV) (ADDR ADVERBS *) (TO S-V))
     (JUMP S-V-WRD=THAT
           (AND (VFEATURE (GETR V) 'THAT)
                (NOT (MEMQ LEX '(TO THAT)))
                (NOT (GETR PASSIVEFLAG))))
     (JUMP S-V-PP T))
(S-PREDADJ (WRD THAN (COMPVERB (GETR V)) (TO S-PREDADJ-COMP)) (JUMP S-MAINCLAUSE T))
(S-PREDADJ-COMP (PUSH NP T (SETR OBJ *) (TO S-MAINCLAUSE)))
(S-V-NP (VIR NP
             (AND (VFEATURE (GETR V) 'INDOBJ) (MEMQ (CAR (GETR TYPE)) '(QPRO QDET)))
             (ADDR PREPPHRASES (BUILDQ (PP (PREP TO) "+") OBJ))
             (SETR S-INDO (GETR S-DO))
             (SETR S-DO (GETR QOBJ))
             (COND ((OR (NOT (V-DO-SEMANTICS (GETR S-VERB) (GETR S-DO)))
                        (NOT (V-INDO-SEMANTICS (GETR S-VERB) (GETR S-INDO))))
                    (ABORT)))
             (SETR OBJ *)
             (TO S-V-PP))
        (PUSH NP
              (VFEATURE (GETR V) 'INDOBJ)
              (SENDR AGFLAG (GETR AGFLAG))
              (COND ((OR (AND (GETR WHOBJ) (NOT (GETR PASSIVEFLAG))) (GETR QOBJ))
                         (SETR S-INDO (GETR HEADNOUN))
                         (ADDR PREPPHRASES (BUILDQ (PP (PREP TO) *))))
                    (T (ADDR PREPPHRASES (BUILDQ (PP (PREP TO) "+") OBJ))
                       (SETR S-INDO (GETR S-DO))
                       (SETR S-DO (GETR HEADNOUN))
                       (COND ((OR (NOT (V-DO-SEMANTICS (GETR S-VERB) (GETR S-DO)))
                                  (NOT (V-INDO-SEMANTICS (GETR S-VERB) (GETR S-INDO))))
                              (ABORT)))
                       (SETR OBJ *)))
              (TO S-V-PP))
        (PUSH ADV (GET LEX 'ADV) (ADDR ADVERBS *) (TO S-V-NP))
        (JUMP S-V-PP T))
(S-V-PREP=BY (PUSH NP
                   T
                   (SENDR AGFLAG (GETR AGFLAG))
                   (SENDR MOTIONFLAG T)
                   (COND ((NOT (S-V-SEMANTICS (GETR HEADNOUN) (GETR S-VERB)))
                          (ADDR PREPPHRASES (BUILDQ (PP (PREP BY) *))))
                         (T (SETR SUBJ *) (SETR S-SUBJ (GETR HEADNOUN)) (SETR AGFLAG NIL)))
                   (TO S-V-PP)))
(S-V-PP (WRD BY (GETR AGFLAG) (TO S-V-PREP=BY))
        (PUSH PP
              (GET LEX 'PREP)
              (SENDR MOTIONFLAG (COND ((NOT (ASSQ 'MOTION (CDR (GET LEX 'PREP))))))
              (SENDR AGFLAG (GETR AGFLAG))
              (ADDR PREPPHRASES *)
              (TO S-V-PP))
        (PUSH ADV (GET LEX 'ADV) (ADDR ADVERBS *) (TO S-V-PP))
        (JUMP S-MAINCLAUSE T))
(S-V-WRD=THAT (PUSH S-DCL
                    T
                    (SENDR TYPE '(DCL))
                    (ADDR COMPL (BUILDQ (COMPL (CTYPE THAT) *)))
                    (AND (VFEATURE (GETR V) 'TRANS) (SETR S-DO (CAR (GETR COMPL))))
                    (TO S-MAINCLAUSE)))
(S-V-TOCOMP (CAT PREP
                 (AND (NOT (GETR PARTICLE)) (MEMBER * (GET (GETR V) 'PARTICLES)))
                 (SETR V (MKATOM (GETR V) '"-" *))
                 (SETR S-VERB (BUILDQ (V "+") V))
                 (SETR PARTICLE T)
                 (TO S-V-TOCOMP))
            (WRD TO T (TO S-V-WRD=TO))
            (JUMP S-V-NP T))
```

```
(S-V-WRD=TO (PUSH S-NP
                  T
                  (SENDR TOFLAG T)
                  (SENDR TYPE '(QCL))
                  (SENDR SUBJ (COND ((GETR OBJ)) ((GETR SUBJ))))
                  (SENDR TENSE (GETR TENSE))
                  (SENDR S-SUBJ (COND ((GETR OBJ) (GETR S-DO)) ((GETR SUBJ) (GETR S-SUBJ))))
                  (ADDR COMPL (BUILDQ (COMP (CTYPE TO) *)))
                  (AND (VFEATURE (GETR V) 'TRANS) (SETR S-DO (CAR (GETR COMPL))))
                  (TO S-MAINCLAUSE)))
(S-MAINCLAUSE (CAT CONJ T (SETR CONJ *) (TO S-CONJ)) (JUMP S-S T))
(S-CONJ (CAT V
             (AND (NOT (NEXT V))
                  (NOT (GETR PASSIVEFLAG))
                  (NOT (EQUAL (GETR TYPE) '(IMP)))
                  (NOT (GETR YESNO)))
             (SETR CONJ-V *)
             (TO S-CONJ-V-ACTIVE))
        (CAT V
             (AND (NOT (NEXT V))
                  (OR (GETR PASSIVEFLAG) (GETR YESNO))
                  (NOT (EQUAL (GETR TYPE) '(IMP))))
             (AND (CDR STRING) (GETR OBJ) (HOLD (GETR OBJ)))
             (SETR CONJ-V *)
             (TO S-CONJ-V-PASSIVE))
        (PUSH S T (SETR S-CONJ *) (TO S-S)))
(S-CONJ-V-ACTIVE (PUSH S-V-SEMANTICS
                       (NOT (NULL STRING))
                       (SENDR SUBJ (GETR SUBJ))
                       (SENDR S-SUBJ (GETR S-SUBJ))
                       (SENDR TYPE (GETR TYPE))
                       (SENDR NEG (GETR NEG))
                       (SENDR TENSE (GETR TENSE))
                       (SENDR ASPECT (GETR ASPECT))
                       (SENDR V (GETR CONJ-V))
                       (SENDR S-VERB (BUILDQ (V "+") CONJ-V))
                       (SETR S-CONJ *)
                       (TO S-S))
                 (JUMP S-CONJ-BUILD (NULL STRING)))
(S-CONJ-V-PASSIVE (PUSH S-V-SEMANTICS
                        (NOT (NULL STRING))
                        (SENDR SUBJ (GETR SUBJ))
                        (SENDR S-SUBJ (GETR S-SUBJ))
                        (SENDR TYPE (GETR TYPE))
                        (SENDR NEG (GETR NEG))
                        (SENDR VOICE (GETR VOICE))
                        (SENDR TENSE (GETR TENSE))
                        (SENDR ASPECT (GETR ASPECT))
                        (SENDR V (GETR CONJ-V))
                        (SENDR S-VERB (BUILDQ (V "+") CONJ-V))
                        (SENDR PASSIVEFLAG (GETR PASSIVEFLAG))
                        (SENDR AGFLAG (GETR PASSIVEFLAG))
                        (SENDR S-DO (GETR S-DO))
                        (SENDR YESNO (GETR YESNO))
                        (SENDR ADVERBS (COND ((EQUAL (GETR TYPE) '(QADV)) (GETR ADVERBS))))
                        (SETR S-CONJ *)
                        (TO S-S))
                  (JUMP S-CONJ-BUILD (NULL STRING) (SETR CONJ-PASSIVE T)))
(S-CONJ-BUILD (JUMP S-S
                    (S-SEMANTICS (GETR S-SUBJ) (BUILDQ (V "+") CONJ-V) (GETR S-DO) NIL)
                    (SETR S-CONJ (BUILD-S-CONJ))))
(S-S (POP (BUILD-S) (S-SEMANTICS (GETR S-SUBJ) (GETR S-VERB) (GETR S-DO) (GETR S-INDO))))
(NP (CAT GENPRO
         T
         (SETR PROPER (BUILDQ (GENPRO *)))
         (SETR NU 'SG)
         (SETR SEM-NOUN (BUILDQ (GENPRO * ((NUMBER SG)))))
         (LIFTR HEADNOUN (GETR SEM-NOUN))
         (TO NP-GENPRO))
    (CAT DET
         (NOT (GETF POSSPRO))
         (SETR DET *)
         (SETR SEM-DET (LIST 'DET * FEATURES))
         (SETR QFLAG (GETF QUANT))
         (TO NP-DET))
    (CAT DET
         (GETF POSSPRO)
         (SETR SEM-DET (BUILDQ (POSS *)))
         (SETR DET (BUILDQ (POSSPRO *)))
         (TO NP-DET))
    (CAT NPR
         (GETF POSSESSIVE)
         (SETR POSSFLAG T)
         (SETR SEM-DET (BUILDQ (POSS *)))
         (ADDR MODS (BUILDQ (POSS (NPR *)))
         (TO NP-DET))
    (CAT PRO
         (NOT (GETF POSSPRO))
         (SETR PROPER (BUILDQ (@ (PRO *) #) FEATURES))
         (SETQ ANAPHOR-DEPTH 1)
```

```
        (SETR SEM-NOUN (BUILDQ (@ (PRO *) (#)) FEATURES))
        (LIFTR HEADNOUN (GETR SEM-NOUN))
        (SETR NU (GETF NUMBER))
        (SETR PRO T)
        (TO NP-N))
    (CAT NPR
        (NOT (GETF POSSESSIVE))
        (SETR PROPER (BUILDQ (@ (NPR *) #) FEATURES))
        (SETR SEM-NOUN (BUILDQ (@ (NPR *) (#)) FEATURES))
        (LIFTR HEADNOUN (GETR SEM-NOUN))
        (SETR NU 'SG)
        (TO NP-N))
    (CAT QDET
        (NOT (EQ * 'WHO))
        (SETR DET *)
        (SETR SEM-DET (BUILDQ (QDET *)))
        (LIFTR WH-PHRASE T S)
        (MAPC '(LAMBDA (X) (APPLY 'LIFTR (LIST 'TYPE ''(QDET) X)))
              '(S S-DCL S-V S-V-NP S-PREDADJ-COMP S-V-PREP=BY S-V-PP))
        (SETR QFLAG (GETF QUANT))
        (TO NP-ORD))
    (CAT PRO
        (GETF POSSPRO)
        (SETR SEM-DET (BUILDQ (POSS *)))
        (SETR DET (BUILDQ (POSSPRO *)))
        (SETR N 'ONES)
        (SETR NU 'SG-PL)
        (SETR SEM-NOUN '(DUMMY ONES NIL))
        (LIFTR HEADNOUN (GETR SEM-NOUN))
        (TO NP-MAINPHRASE))
    (JUMP NP-DET T))
(NP-GENPRO (PUSH ADJP
                 (OR (GET LEX 'ADJ) (GET LEX 'ADV) (GET LEX 'V))
                 (ADDR MODS *)
                 (TO NP-N))
           (JUMP NP-N T))
(NP-DET (CAT ORDINAL T (ADDR MODS (BUILDQ (("<-ORD-" NIL *)))) (SETR ORD T) (TO NP-ORD))
        (CAT ADJ
             (GETF SUPERLATIVE)
             (ADDR MODS (BUILDQ (ORD (@ ("<-ADJ-" *) #)) FEATURES))
             (SETR ORD T)
             (TO NP-ORD))
        (JUMP NP-ORD T))
(NP-ORD (PUSH QUANTP T (SETR QUANT T) (ADDR MODS *) (SETR SEM-DET NIL) (TO NP-QUANT))
        (JUMP NP-QUANT T)
        (JUMP NP-MAINPHRASE
              (OR (AND (GETR ORD) (GETR DET)) (GETR QFLAG) (GETR POSSFLAG))
              (SETR N 'ONES)
              (SETR NU 'SG-PL)
              (SETR SEM-NOUN '(DUMMY ONES NIL))
              (LIFTR HEADNOUN (GETR SEM-NOUN))
              (SETR NU 'SG-PL)))
(NP-QUANT (WRD OF
               (OR (AND (GETR ORD) (GETR DET)) (GETR QFLAG) (GETR QUANT))
               (TO NP-QUANT-PREP=OF))
          (JUMP NP-QUANT-PREP=OF
                (AND (NOT (GETR ORD))
                     (NOT (GETR QUANT))
                     (GETR QFLAG)
                     (MEMQ (GETR DET) '(ALL BOTH))))
          (JUMP NP-PART (NOT (EQ * 'OF)))
          (JUMP NP-MAINPHRASE
                (OR (GETR QUANT) (GETR QFLAG) (GETR POSSFLAG) (AND (GETR ORD) (GETR DET)))
                (SETR N 'ONES)
                (SETR NU 'SG-PL)
                (SETR SEM-NOUN '(DUMMY ONES NIL))
                (LIFTR HEADNOUN (GETR SEM-NOUN))
                (SETR NU 'SG-PL)))
(NP-QUANT-PREP=OF (PUSH NP
                        (
                        (ADDR PP (BUILDQ (PP (PREP OF) *)))
                        (SETR N 'ONES)
                        (SETR NU 'SG-PL)
                        (SETR SEM-NOUN '(DUMMY ONES NIL))
                        (LIFTR HEADNOUN (GETR SEM-NOUN))
                        (SETR NU 'SG-PL)
                        (TO NP-MAINPHRASE)))
(NP-PART (CAT N
              (GETF POSSESSIVE)
              (ADDR SMODS
                    ("<-POSS-BY-" (@ (*) (#) #))
                    FEATURES
                    (COND ((GETR CLASS) (LIST-IT-1 (GETR MODS)))))
              (COND ((GETR CLASS) (SETR MODS NIL) (SETR CLASS NIL)))
              (SETR MODS (GETR SMODS))
              (ADDR MODS (BUILDQ (@ (POSS (N *) #)) FEATURES))
              (SETR QFLAG NIL)
              (SETR ORD NIL)
              (SETR POSSFLAG T)
```

```
                (SETR QUANT NIL)
                (SETR SEM-DET NIL)
                (SETR SEM-ADJS NIL)
                (TO NP-DET))
          (CAT N
                (AND (NOT (GETF POSSESSIVE)) (NUMAGREE (GETR DET) * 'N (GETF NUMBER)))
                (SETR N *)
                (SETR NFEAT FEATURES)
                (SETR NU (GETF NUMBER))
                (SETR SEM-NOUN (BUILDQ (@ (N *) (#)) FEATURES))
                (TO NP-N))
          (CAT N
                (AND (NOT (GETF POSSESSIVE)) (NOT (GETR CLASS)))
                (SETR N *)
                (SETR NFEAT FEATURES)
                (SETR NU (GETF NUMBER))
                (SETR SMODS (GETR MODS))
                (SETR MODS NIL)
                (SETR CLASS (BUILDQ ("<-CLASS-" #) (NP-BUILD (NOUN-LIST-GET (BUILD-NP)))))
                (AAAA)
                (SETR MODS (GETR SMODS))
                (ADDR MODS (GETR CLASS))
                (AAAA)
                (SETR N NIL)
                (TO NP-PART))
          (CAT NPR
                (NOT (GETF POSSESSIVE))
                (ADDR MODS (BUILDQ (ADJ (@ (N "+") "+")) N NFEAT))
                (SETR NFEAT FEATURES)
                (SETR NPR *)
                (SETR NU 'SG)
                (SETR SEM-NOUN (BUILDQ (@ (NPR *) (#)) FEATURES))
                (TO NP-N))
          (PUSH ADJP
                (OR (GET LEX 'ADJ) (GET LEX 'ADV) (GET LEX 'V))
                (SENDR NP T)
                (ADDR MODS *)
                (ADDR SEM-ADJS (GETR HEADADJ))
                (TO NP-PART)))
(NP-ADJ (CAT N
                (GETF POSSESSIVE)
                (ADDR MODS (BUILDQ (ADJ (@ (N "+") "+")) N NFEAT))
                (ADDR SMODS
                      ("<-POSS-BY-" (@ (*) (#) #))
                      FEATURES
                      (COND ((GETR CLASS) (LIST-IT-1 (GETR MODS)))))
                (COND ((GETR CLASS) (SETR MODS NIL) (SETR CLASS NIL)))
                (SETR MODS (GETR SMODS))
                (ADDR MODS (BUILDQ (POSS (@ (N *) #)) FEATURES))
                (SETR QFLAG NIL)
                (SETR ORD NIL)
                (SETR POSSFLAG T)
                (SETR QUANT NIL)
                (SETR SEM-DET NIL)
                (SETR SEM-ADJS NIL)
                (TO NP-DET))
          (CAT N
                (AND (NOT (GETF POSSESSIVE)) (NUMAGREE (GETR DET) * 'N (GETF NUMBER)))
                (ADDR MODS (BUILDQ (ADJ (@ (N "+") "+")) N NFEAT))
                (SETR N *)
                (SETR NFEAT FEATURES)
                (SETR NU (GETF NUMBER))
                (SETR SEM-NOUN (BUILDQ (@ (N *) (#)) FEATURES))
                (TO NP-ADJ))
          (CAT N
                (AND (NOT (GETF POSSESSIVE)) (NOT (GETR CLASS)))
                (SETR N *)
                (SETR NFEAT FEATURES)
                (SETR NU (GETF NUMBER))
                (SETR SMODS (GETR MODS))
                (SETR CLASS (BUILDQ ("<-CLASS-" #) (BUILD-NP)))
                (SETR MODS (GETR SMODS))
                (ADDR MODS (GETR CLASS))
                (SETR N NIL)
                (TO NP-PART))
          (CAT NPR
                (NOT (GETF POSSESSIVE))
                (ADDR MODS (BUILDQ (ADJ (@ (N "+") "+")) N NFEAT))
                (SETR NFEAT FEATURES)
                (SETR NPR *)
                (SETR NU 'SG)
                (SETR SEM-NOUN (BUILDQ (@ (NPR *) (#)) FEATURES))
                (TO NP-ADJ))
          (JUMP NP-N T))
(NP-N (POP (BUILD-NP) T))
(NP-MAINPHRASE (CAT CONJ T (SETR CONJ *) (TO NP-CONJ)) (JUMP NP-NP T))
(NP-CONJ (PUSH NP
                T
                (SENDR CONJFLAG T)
                (SETR NP-CONJ *)
                (COND ((ATOM (CADR (GETR HEADNOUN)))
```

```
                    (LIFTR HEADNOUN
                           (BUILDQ (N ("+" "+") ((NUMBER #)))
                                   SEM-NOUN
                                   HEADNOUN
                                   (COND ((EQ (GETR CONJ) 'AND) 'PL)
                                         ('SG-PL)))))
                    (T (LIFTR HEADNOUN
                              (LIST (CAR (GETR HEADNOUN))
                                    (APPEND1 (CADR (GETR HEADNOUN)) (GETR SEM-NOUN))
                                    (CADDR (GETR HEADNOUN))))))
               (TO NP-NP)))
(NP-NP (POP (BUILD-NP) (NP-SEMANTICS (GETR SEM-DET) (GETR SEM-ADJS) (GETR SEM-NOUN))))
(ADJ (MEM (MORE MOST) T (SETR FEAT (BUILDQ ((#)) (COMP-SUPER))) (TO ADJ-FEAT))
     (CAT V
          (AND (PARTICIPLE) (GETR NP))
          (SETR ADJ (BUILDQ (# *) (PARTICIPLE)))
          (LIFTR HEADADJ (BUILDQ (V * (# *)) (PARTICIPLE)) NP-PART)
          (LIFTR HEADADJ (BUILDQ (V * (# *)) (PARTICIPLE)) S-V)
          (TO ADJ-ADJ))
     (JUMP ADJ-FEAT T))
(ADJ-FEAT (CAT ADJ
               T
               (SETR ADJ *)
               (OR (GETR FEAT) (SETR FEAT FEATURES))
               (LIFTR HEADADJ (BUILDQ (ADJ * ("+")) FEAT) NP-PART)
               (LIFTR HEADADJ (BUILDQ (ADJ * ("+")) FEAT) S-V)
               (TO ADJ-ADJ)))
(ADJ-ADJ (POP (BUILDQ (@ ("<-ADJ-" NIL "+") "+") ADJ FEAT) T))
(ADV (MEM (MORE MOST) T (SETR FEAT (BUILDQ ((#)) (COMP-SUPER))) (TO ADV-FEAT))
     (JUMP ADV-FEAT T))
(ADV-FEAT (CAT ADV
               T
               (SETR ADV *)
               (OR (GETR FEAT) (SETR FEAT FEATURES))
               (COND ((MEMBER * '(THERE HERE)) (SETQ ANAPHOR-DEPTH 1)))
               (TO ADV-ADV)))
(ADV-ADV (POP (BUILDQ (@ ("<-ADV-" NIL "+") "+") ADV FEAT) T))
(ADJP (PUSH ADJ
            (OR (NEXT '(ADJ ADV V)))
            (SENDR NP (GETR NP))
            (COND ((NULL (GETR ADV)) (ADDR ADJ *))
                  (T (ADDR ADJ (APPEND * (LIST (LIST-IT-2 (GETR ADV))))) (SETR ADV NIL)))
            (TO ADJP))
      (PUSH ADV
            (OR (NEXT 'ADV) (NEXT 'V))
            (COND ((NULL (GETR ADV)) (SETR ADV *))
                  (T (SETR ADV (APPEND * (LIST (LIST-IT-2 (GETR ADV)))))))
            (TO ADJP))
      (JUMP ADJP-ADJP T))
(ADJP-ADJP (POP (GETR ADJ) (NOT (NULL (GETR ADJ)))))
(PP (CAT PREP (SETR PREP *) (TO PP-PREP)))
(PP-PREP (MEM (THERE HERE) (SETR NP (BUILDQ ("<-ADV-" NIL *))) (TO PP-PP))
         (PUSH NP T (SETR NP *) (TO PP-PP))
         (PUSH QP
               (GET LEX 'QPRO)
               (SETR NP *)
               (LIFTR TYPE 'QPRO S)
               (LIFTR WH-PHRASE T S)
               (TO PP-PP)))
(PP-PP (POP (BUILDQ (PP NIL "+" "+") PREP NP) T))
(QP (CAT QPRO T (SETR QPRO *) (TO QP-QPRO)))
(QP-QPRO (PUSH PP (GET LEX 'PREP) (SETR PP *) (TO QP-QP)) (JUMP QP-QP T))
(QP-QP (POP (BUILD-QP) T))
(QUANTP (CAT NUMBER T (SETR Q (BUILDQ (INTEGER *))) (TO QUANTP-Q))
        (CAT QUANT T (SETR Q *) (TO QUANTP-Q))
        (JUMP QUANTP-Q T))
(QUANTP-Q (WRD AT T (TO QUANTP-PREP=AT))
          (CAT COMP T (SETR COMP (BUILDQ (ADV *))) (TO QUANTP-COMP))
          (JUMP QUANTP-QUANTP (GETR Q) (SETR QWORD T)))
(QUANTP-PREP=AT (CAT SUPER T (SETR SUPER (BUILDQ (ADV *))) (TO QUANTP-MODS)))
(QUANTP-COMP (WRD THAN T (TO QUANTP-MODS))
             (JUMP QUANTP-QUANTP
                   (COND ((AND (GETR COMP)
                               (EQ (CADR (GETR COMP)) 'MORE)
                               (GET LEX 'ADJ)
                               (EQ (GET LEX 'ADJ) '*))
                          NIL)
                         (T))
                   (SETR QWORD (NOT (AND (GETR COMP) (GETR Q))))))
(QUANTP-MODS (CAT DET (NOT (GETR DET)) (SETR DET *) (TO QUANTP-MODS))
             (PUSH QUANTP T (SETR QUANT *) (TO QUANTP-QUANTP))
             (JUMP QUANTP-QUANTP (AND (GETR Q) (NOT (GETR COMP)))))
(QUANTP-QUANTP (POP (BUILD-QUANTP) T)))))
```

```
; THESE ARE SOME AUXILIARY ROUTINES
(DEFUN BUILD-VP ()
  (SETR TYPE (BUILDQ (@ (MODO) "+" "+") NEG TYPE))
  (COND ((GETR VOICE))
        (T (SETR VOICE '(VOICE ACTIVE))))
  (LIFTR TYPE (GETR TYPE))
  (LIFTR VOICE (GETR VOICE))
  (SETR AUX (BUILDQ (TNS (@ "+" "+") "+") TENSE ASPECT VOICE))
  (SETR V (BUILDQ (V "+") V))
  (COND ((NULL (GETR ADVERBS)))
        (T (ADDL ADVERVS "<-ADV-")))
  (BUILDQ (@ (VP) (NIL) ("+") ("+") "+") AUX V ADVERBS))


(DEFUN BUILD-NP ()
  (SETR DET (COND ((NULL (GETR DET)) NIL) (T (BUILDQ ((DET "+")) DET))))
  (SETR POSS (COND ((GETR POSS) (BUILDQ ((POSS "+")) POSS))))
  (SETR NFEAT (BUILDQ (@ # #) (LIST-IT (GETR NFEAT)) (LIST-IT-I (GETR MODS))))
  (SETR N
        (COND ((GETR N) (BUILDQ (@ (N "+") "+") N NFEAT))
              ((GETR NPR) (BUILDQ (@ (NPR "+") "+") NPR NFEAT))
              ((GETR PROPER))))
  (SETR NP (BUILDQ (@ (NP) (NIL) "+" "+" ("+")) DET POSS N))
  (COND ((GETR CONJ) (BUILDQ (NP ("+" "+" "+")) CONJ NP NP-CONJ))
        (T (GETR NP))))


(DEFUN BUILD-QP ()
  (SETR PP (COND ((GETR PP) (LIST (GETR PP)))))
  (BUILDQ (@ (NP) ((DET NIL)) ((QPRO "+")) ((NU SG-PL)) "+") QPRO PP))

(DEFUN BUILD-QUANTP ()
  (COND ((AND (GETR QWORD) (GETR Q)) (BUILDQ (QUANT "+") Q))
        ((AND (GETR QWORD) (GETR COMP)) (BUILDQ (QUANT "+") COMP))
        (T (SETR Q (COND ((GETR Q) (BUILDQ ((QUANT "+")) Q))))
           (SETR COMP (COND ((GETR COMP) (BUILDQ ((COMP "+")) COMP))))
           (SETR SUPER (COND ((GETR SUPER) (BUILDQ ((SUPER "+")) SUPER))))
           (SETR DET (COND ((GETR DET) (BUILDQ ((DET "+")) DET))))
           (SETR QUANT (COND ((GETR QUANT) (LIST (GETR QUANT)))))
           (BUILDQ (@ (QUANTP) "+" "+" "+" "+" "+") Q COMP SUPER DET QUANT))))


(DEFUN BUILD-ADJP ()
  (GETR ADJ))


(DEFUN PERFECT ()
  (AND (EQ (PARTICIPLE) 'PASTPART) (EQ (GETR V) 'HAVE)))

(DEFUN PROGRESSIVE ()
  (AND (EQ (PARTICIPLE) 'PRESPART) (EQ (GETR V) 'BE)))


(DEFUN FUTURE ()
  (AND (GETF UNTENSED) (MEMQ (GETR V) '(WILL SHALL))))


(DEFUN PARTICIPLE ()
  (COND ((EQ 'PRESPART (GETF PART)) 'PRESPART)
        ((EQ 'PASTPART (GETF PART)) 'PASTPART)))

(DEFUN COMP-SUPER ()
  (MAKE-ARROW (CADR (ASSQ * '((MORE COMPARATIVE) (MOST SUPERLATIVE))))))
```

```
(DEFPROP SETR EXPR
  (FLAMBDA (X)
           (SETQ REGS (CONS (CONS (CAR X) (EVAL (CADR X))) REGS))))

(DEFPROP GETR EXPR
  (FLAMBDA (LST)
           (COND ((NULL (CDR LST)) (GETRV (CAR LST)))
                 ((NUMBERP (CADR LST))
                  (COND ((ZEROP (CADR LST)) (GETRV (CAR LST)))
                        (T (CDR (ASSQ (CAR LST) (CADAR (NTH STACK (CADR LST))))))))
                 (T (CDR (ASSQ (CAR LST) (CADR (ASSQ (CADR LST) STACK))))))))

(DEFPROP GETRV EXPR
  (LAMBDA (REG)
          (PROG (B)
                (RETURN
                  (COND ((EQ REG (QUOTE *)) *)
                        ((SETQ B (ASSQ REG REGS)) (CDR B)))))))

(DEFPROP BUILDQ EXPR
  (FLAMBDA (L)
           (BUILD (CAR L) (CDR L))))

(DEFPROP BUILD EXPR
  (LAMBDA (FRAG REG*)
          (PROG (REGLST)
                (SETQ REGLST REG*)
                (RETURN (BUILD1 FRAG)))))

(DEFPROP BUILD1 EXPR
  (LAMBDA (FRAG)
          (PROG (A)
                (RETURN
                  (COND ((NULL FRAG) NIL)
                        ((EQ FRAG (QUOTE +)) (GETRV (CAR REGLST)))
                        ((EQ FRAG (QUOTE #)) (EVAL (CAR REGLST)))
                        ((EQ FRAG (QUOTE *)) *)
                        ((ATOM FRAG) FRAG)
                        ((EQ (CAR FRAG) (QUOTE +))
                         (SETQ A (UNCONS REGLST (QUOTE REGLST)))
                         (CONS (GETRV A) (BUILD1 (CDR FRAG))))
                        ((EQ (CAR FRAG) (QUOTE #))
                         (SETQ A (UNCONS REGLST (QUOTE REGLST)))
                         (CONS (EVAL A) (BUILD1 (CDR FRAG))))
                        ((EQ (CAR FRAG) (QUOTE *)) (CONS * (BUILD1 (CDR FRAG))))
                        ((EQ (CAR FRAG) (QUOTE @)) (GLUE (BUILD1 (CDR FRAG))))
                        (T (CONS (BUILD1 (CAR FRAG)) (BUILD1 (CDR FRAG))))))))))

(DEFPROP PARSE1 EXPR
  (LAMBDA (SENTENCE)
          (PROG (A MORPHTEMP)
                (INIT-TRACE)
                (SETQ REGS NIL)
                (COND
                  ((SETQ A (NEVAL SENTENCE ST NIL NIL NIL (QUOTE (NIL . 0))))
                   (AND *PRINTPARSE
                        (PROGN (PRIN1 (QUOTE PARSE:)) (TERPRI) (PRINTPARSE A 0)))
                   (RETURN A))))))

(DEFPROP NEVAL EXPR
  (LAMBDA (STRING1 STATE1 STACK1 REGS1 HOLD1 PATH1)
          (PROG (A BODY-S ARCTYPE * LEX)
                (PTRACE ENTERING STATE (STATE1) & STRING = (STRING1))
                (PTREGS)
                (SETQ *
                      (COND ((NULL STRING1) NIL)
                            ((SETQ MORPHTEMP (MORPH (CAR STRING1)))
                             (COND ((EQ MORPHTEMP T) (CAR STRING1))
                                   (T (RPLACA STRING1 MORPHTEMP))))
                            (T (ERROR (DICTERR)))))
                LEX
                *)
                (COND ((NOT (GET STATE1 (QUOTE GRAMMAR))) (ERROR (ERROR2))))
                (SETQ BODY-S (GET STATE1 (QUOTE GRAMMAR)))
            TAG (COND ((NULL BODY-S) (BLOCKED) (RETURN NIL))
                      (T (SETQ ARCTYPE (CAAR BODY-S))))
                (COND ((SETQ A (EVAL (CAR BODY-S))) (RETURN A))
                      (T (SETQ BODY-S (CDR BODY-S)) (GO TAG))))))

(DEFPROP PUSH EXPR
  (FLAMBDA (L)
           (PROG (STRING STATE STACK REGS HOLD PUSHREGS)
                 (INITIALIZE)
                 (RETURN
                   (COND ((NULL STRING) NIL)
                         ((NOT (EVAL (CADR L))) NIL)
                         (T (PRINTPUSH)
                            (NEVAL STRING
                                   (CAR L)
                                   (PUSHSTK (LIST STATE REGS (FINDACTIONS (CDDR L))))
                                   (COSENDRS (CDDR L))
                                   HOLD
```

```
                                        (CONS STATE (SUB1 :LEVEL)))))))))

  (DEFPROP WRD EXPR
    (FLAMBDA (L)
            (PROG (STRING STATE STACK REGS HOLD)
                  (INITIALIZE)
                  (RETURN
                    (COND ((NULL STRING) NIL)
                          ((AND (EQ * (CAR L)) (EVAL (CADR L)))
                           (PRINTARC ARCTYPE)
                           (APPLY (QUOTE PROGNX) (CDDR L))))))))

  (DEFPROP PROGNX EXPR
    (FLAMBDA (L)
            (COND ((NULL L) (ERROR (ERROR1)))
                  ((NULL (CDR L))
                   (COND ((NOT (EQ (CAAR L) (QUOTE TO))) (ERROR (ERROR1)))
                         (T (EVAL (CAR L)))))
                  (T (EVAL (CAR L)) (APPLY (QUOTE PROGNX) (CDR L))))))

  (DEFPROP MEM EXPR
    (FLAMBDA (L)
            (PROG (STRING STATE STACK REGS HOLD)
                  (INITIALIZE)
                  (RETURN
                    (COND ((NULL STRING) NIL)
                          ((AND (MEMQ * (CAR L)) (EVAL (CADR L)))
                           (PRINTARC ARCTYPE)
                           (APPLY (QUOTE PROGNX) (CDDR L))))))))

  (DEFPROP POP EXPR
    (FLAMBDA (L)
            (PROG (STRING STATE STACK REGS HOLD PS LEV)
                  (INITIALIZE)
                  (COND ((NULL (CDR L)) (ERROR (ERROR3)))
                        ((NOT (EVAL (CADR L))) (RETURN NIL))
                        (T
                          (RETURN
                            (COND
                              ((AND (NULL STRING) (NULL STACK) (NULL HOLD))
                               (PTRACE ABOUT TO POP : SUCCESS)
                               (EVAL (CAR L)))
                              ((NULL STACK) NIL)
                              ((NOT (HOLDLEVEL)) NIL)
                              (T (SETQ PS (CAR STACK))
                                 (POPSTK)
                                 (SETQ * (EVAL (CAR L)) REGS (CADR PS) LEV :LEVEL)
                                 (PRINTPOP)
                                 (APPLY (QUOTE PUSHPROGN) (CADDR PS))))))))))

  (DEFPROP DOSENDRS EXPR
    (LAMBDA (L)
            (COND ((OR (NULL L) (NOT (EQ (CAAR L) (QUOTE SENDR)))) NIL)
                  (T (EVAL (CAR L)) (DOSENDRS (CDR L)) PUSHREGS))))

  (DEFPROP SENDR EXPR
    (FLAMBDA (L)
            (COND ((NOT (EQ ARCTYPE (QUOTE PUSH))) (ERROR (ERROR6))))
            (SETQ PUSHREGS
                  (CONS (CONS (CAR L) (EVAL (CADR L))) PUSHREGS))))

  (DEFPROP FINDACTIONS EXPR
    (LAMBDA (L)
            (COND ((NULL L) NIL)
                  ((NOT (EQ (CAAR L) (QUOTE SENDR))) L)
                  (T (FINDACTIONS (CDR L))))))

  (DEFPROP PUSHSTK EXPR
    (LAMBDA (S)
            (CONS S STACK)))

  (DEFPROP POPSTK EXPR
    (LAMBDA NIL
            (SETQ STACK (CDR STACK))))

  (DEFPROP TO EXPR
    (FLAMBDA (S)
            (COND ((NULL STRING) NIL)
                  (T
                    (WEVAL (CDR STRING)
                           (CAR S)
                           STACK
                           REGS
                           HOLD
                           (CONS STATE :LEVEL))))))

  (DEFPROP JUMP EXPR
    (FLAMBDA (L)
            (PROG (STRING STATE STACK REGS HOLD)
```

```
201                     (INITIALIZE)
202                     (RETURN
203                       (COND ((NOT (EVAL (CADR L))) NIL)
204                             (T (MAPC (QUOTE EVAL) (CDDR L))
205                                (PTRACE JUMP TO ((CAR L)))
206                                (MEVAL STRING (CAR L) STACK REGS HOLD (CONS STATE :LEVEL)))))))))
207
208
209     (DEFPROP CAT EXPR
210       (FLAMBDA (L)
211               (PROG (STRING STATE STACK REGS HOLD FEATURES)
212                     (INITIALIZE)
213                     (RETURN
214                       (COND ((NULL STRING) NIL)
215                             ((AND (CATEGORY (CAR L)) (EVAL (CADR L)))
216                              (PRINTARC ARCTYPE)
217                              (APPLY (QUOTE PROGN) (CDDR L)))))))))
218
219
220     (DEFPROP PUSHPROGN EXPR
221       (FLAMBDA (L)
222               (COND ((NULL L) (ERROR (ERROR1)))
223                     ((NULL (CDR L))
224                      (COND ((NOT (EQ (CAAR L) (QUOTE TO))) (ERROR (ERROR1)))
225                            (T (MEVAL STRING (CADAR L) STACK REGS HOLD (CONS STATE LEV)))))
226                     (T (EVAL (CAR L)) (APPLY (QUOTE PUSHPROGN) (CDR L))))))
227
228
229     (DEFPROP TRACEPARSE EXPR
230       (LAMBDA NIL
231               (SETQ :TRACE T)))
232
233
234     (DEFPROP UNTRACEPARSE EXPR
235       (LAMBDA NIL
236               (SETQ :TRACE NIL)))
237
238
239     (DEFPROP INIT-TRACE EXPR
240       (LAMBDA NIL
241               (COND
242                 ((:TRACE (SETQ :TAB 0)
243                          (PRINT (QUOTE SENTENCE:))
244                          (PRINT SENTENCE)
245                          (TERPRI))))))
246
247
248     (DEFPROP PRINTARC EXPR
249       (LAMBDA (ARCTYPE)
250               (PTRACE TAKING (ARCTYPE (CAR L)) ARC)))
251
252
253     (DEFPROP PRINTPUSH EXPR
254       (LAMBDA NIL
255               (SETQ :LEVEL (ADD1 :LEVEL))
256               (AND (PTRACE ABOUT TO PUSH) (SETQ :TAB (ADD :TAB 2)))))
257
258
259     (DEFPROP PRINTPOP EXPR
260       (LAMBDA NIL
261               (SETQ :LEVEL (SUB1 :LEVEL))
262               (AND (PTRACE ABOUT TO POP * = (*))
263                    (SETQ :TAB (SUB :TAB 2)))))
264
265
266     (DEFPROP PARSE EXPR
267       (FLAMBDA (L)
268               (PROG (A :LEVEL :TAB ST)
269                     (SETQ :LEVEL 0)
270                     (COND
271                       ((NULL (CDR L))
272                        (PERROR SECOND ARGUMENT TO PARSE IS UNSPECIFIED)
273                        (RETURN NIL))
274                       (T (SETQ ST (CADR L))))
275                     (SETQ A (ERRSET (PARSE (CAR L))))
276                     (RETURN (COND ((ATOM A) NIL) (T (CAR A)))))))
277
278
279     (DEFPROP ERROR1 EXPR
280       (LAMBDA NIL
281               (PERROR NO TERMINATING NEXT STATE ACTION ON CURRENT ARC)))
282
283
284     (DEFPROP ERROR2 EXPR
285       (LAMBDA NIL
286               (PERROR STATE (STATE1) UNDEFINED)))
287
288
289     (DEFPROP HOLDLEVEL EXPR
290       (LAMBDA NIL
291               (COND ((NULL HOLD) T)
292                     ((EQ (CAAR HOLD) :LEVEL) NIL)
293                     (T)))))
294
295
296     (DEFPROP VIR EXPR
297       (FLAMBDA (L)
298               (PROG (STRING STATE STACK REGS HOLD LEV)
299                     (INITIALIZE)
```

```
300                     (RETURN
301                       (COND ((NULL HOLD) NIL)
302                             ((AND (SETQ * (CDAR HOLD))
303                                   (VIRCATEGORY (CAR L))
304                                   (EVAL (CADR L)))
305                              (PRINTARC ARCTYPE)
306                              (POPHOLD)
307                              (SETQ LEV :LEVEL)
308                              (APPLY (QUOTE PUSHPROGN) (CDDR L)))))))
309
310
311     (DEFPROP POPHOLD EXPR
312       (LAMBDA NIL
313               (SETQ HOLD (CDR HOLD))
314               (PTRACE POPPING
315                       HOLD
316                       LIST
317                       *
318                       =
319                       (*)
320                       CURRENT
321                       HOLD
322                       IS
323                       ((COND (HOLD (CDAR HOLD))))))))
324
325
326     (DEFPROP ERROR3 EXPR
327       (LAMBDA NIL
328               (PERROR NO TEST ON POP ARC OF STATE (STATE))))
329
330
331     (DEFPROP BLOCKED EXPR
332       (LAMBDA NIL
333               (PTRACE BLOCKED)
334               (SETQ :LEVEL (CDR PATH1) :TAB (ADD :LEVEL :LEVEL))
335               (COND ((CAR PATH1) (PTRACE BACKED UP TO STATE ((CAR PATH1))))))))
336
337
338     (DEFPROP HOLD EXPR
339       (FLAMBDA (L)
340                (PROG (A)
341                      (SETQ A (EVAL (CAR L)))
342                      (PTRACE HOLDING (A))
343                      (SETQ HOLD (CONS (CONS :LEVEL A) HOLD)))))
344
345
346     (DEFPROP CATEGORY EXPR
347       (LAMBDA (CAT)
348               (COND
349                 ((GET LEX CAT)
350                  (SETQ FEATURES (GETCATFEATURES). * (GETROOT)))
351                 T))))
352
353
354     (DEFPROP GETCATFEATURES EXPR
355       (LAMBDA NIL
356               (PROG (PROP)
357                     (SETQ PROP (GET LEX CAT))
358                     (COND
359                       ((ATOM PROP)
360                        (RETURN
361                          (SELECT CAT
362                                  ((QUOTE N)
363                                   (COND ((EQ PROP (QUOTE MASS)) (QUOTE ((NUMBER MASS))))
364                                         (T (QUOTE ((NUMBER SG))))))
365                                  ((QUOTE V)
366                                   (LIST (QUOTE (TNS PRESENT))
367                                         (QUOTE (PNCODE X3SG))
368                                         (QUOTE (UNTENSED))))
369                                  NIL))))
370                     (SETQ PROP (CDR PROP))
371                     (RETURN
372                       (COND
373                         ((AND (EQ CAT (QUOTE V))
374                               (ASSQ (QUOTE TNS) PROP)
375                               (NOT (ASSQ (QUOTE PNCODE) PROP)))
376                          (CONS (QUOTE (PNCODE ANY)) PROP))
377                         ((AND (EQ CAT (QUOTE PRO))
378                               (NOT (ASSQ (QUOTE SUBJ) PROP))
379                               (NOT (ASSQ (QUOTE OBJ) PROP)))
380                          (CONS (QUOTE (SUBJ)) PROP))
381                         (T PROP))))))
382
383
384     (DEFPROP GETROOT EXPR
385       (LAMBDA NIL
386               (PROG (PROP)
387                     (SETQ PROP (GET LEX CAT))
388                     (RETURN (COND ((NOT (ATOM PROP)) (CAR PROP)) (T LEX))))))
389
390
391     (DEFPROP GETF EXPR
392       (FLAMBDA (FEATURE)
393                (PROG (TEMP)
394                      (COND ((NOT (EQ ARCTYPE (QUOTE CAT))) (ERROR (ERROR5))))
395                      (SETQ TEMP (ASSQ (CAR FEATURE) FEATURES))
396                      (COND
397                        (TEMP (RETURN (COND ((NULL (CDR TEMP)) T) (T (CADR TEMP))))))))))
398
399
400     (DEFPROP VIRCATEGORY EXPR
```

```
401       (LAMBDA (CAT)
402               (EQ CAT (CAR *))))
403
404
405     (DEFPROP PRINTPARSE EXPR
406       (LAMBDA (PARSE TAB)
407               (COND ((NULL PARSE))
408                     ((ATOM (CAR PARSE)) (SKIP TAB) (PRI PARSE))
409                     (T (PRINTPARSE (CAR PARSE) TAB)
410                        (PRINTPARSE (CDR PARSE) TAB)))))
411
412
413     (DEFPROP PRI EXPR
414       (LAMBDA (PARSE)
415               (COND ((NULL PARSE) (TERPRI))
416                     ((ATOM (CAR PARSE)) (PRIN1 (CAR PARSE)) (PRI (CDR PARSE)))
417                     (T (TERPRI)
418                        (PRINTPARSE (CAR PARSE) (ADD TAB 4))
419                        (PRINTPARSE (CDR PARSE) (ADD TAB 4))))))
420
421
422     (DEFPROP DISPLAY EXPR
423       (FLAMBDA (L)
424               (SETQ $REGS
425                     (CAR L)
426                     $STATES
427                     (COND ((NULL (CDR L)) NIL)
428                           (T (CADR L))))
429               T))
430
431
432     (DEFPROP UNDISPLAY EXPR
433       (LAMBDA NIL
434               (SETQ $REGS NIL $STATES NIL)))
435
436
437     (DEFPROP PAINREGS EXPR
438       (LAMBDA (REGLST)
439               (MAPC
440                 (QUOTE
441                   (LAMBDA (X)
442                           (PRIN1 X)
443                           (PRIN1 (QUOTE $))
444                           (PRIN1 (GETRV X))
445                           (TERPRI)))
446                 REGLST)))
447
448
449     (DEFPROP ERROR4 EXPR
450       (LAMBDA NIL
451               (PRIN1
452                 (QUOTE
453                   "***WARNING--ATTEMPT TO ADDL OR ADDR TO ATOMIC VALUED REGISTER="))
454               (PRIN1 (CAR L))
455               (SKIP 4)
456               (PRIN1 (CAR L))
457               (PRIN1 (QUOTE =))
458               (PRIN1 A)
459               (PRIN1 (QUOTE " STATE ="))
460               (PRIN1 STATE)
461               (TERPRI)))
462
463
464     (DEFPROP ADDL EXPR
465       (FLAMBDA (L)
466               (PROG (A)
467                     (SETQ A
468                           (GETRV (CAR L))
469                           REGS
470                           (CONS (CONS (CAR L) (CONS (EVAL (CADR L)) A)) REGS))
471                     (COND ((AND (ATOM A) (NOT (NULL A))) (ERROR4) T)))))
472
473
474     (DEFPROP ADDR EXPR
475       (FLAMBDA (L)
476               (PROG (A)
477                     (SETQ A
478                           (GETRV (CAR L))
479                           REGS
480                           (CONS (CONS (CAR L) (APPEND A (LIST (EVAL (CADR L)))))
481                                 REGS))
482                     (COND ((AND (ATOM A) (NOT (NULL A))) (ERROR4) T)))))
483
484
485     (DEFPROP GETREGS EXPR
486       (LAMBDA NIL
487               (PROG (A)
488                     (MAPC
489                       (QUOTE
490                         (LAMBDA (X)
491                                 (COND ((MEMQ (CAR X) A))
492                                       (T (SETQ A (CONS (CAR X) A))))))
493                       REGS)
494                     (RETURN A))))
495
496
497     (DEFPROP PTREGS EXPR
498       (LAMBDA NIL
499               (COND
500                 ($REGS
```

```
501                 (COND
502                   ((OR (NULL :STATES) (MEMQ STATE1 :STATES))
503                    (COND
504                      ((NULL :TRACE)
505                       (PRIN1 (QUOTE "STATE =")
506                       (PRIN1 STATE1)
507                       (TERPRI))))
508                    (PRINREGS (COND ((EQ :REGS (QUOTE ALL)) (GETREGS)) (T :REGS)))))))))

509
510
511   (DEFPROP DICTERR EXPR
512     (LAMBDA NIL
513             (PERROR ((CAR STRING1)) CANNOT BE MORPHED)))
514
515
516   (DEFPROP INITIALIZE EXPR
517     (LAMBDA NIL
518             (SETQ STRING
519                   STRING1
520                   STATE
521                   STATE1
522                   STACK
523                   STACK1
524                   REGS
525                   REGS1
526                   HOLD
527                   HOLD1)))
528
529
530   (DEFPROP TST EXPR
531     (FLAMBDA (L)
532             (PROG (STRING STATE STACK REGS HOLD)
533                   (INITIALIZE)
534                   (RETURN
535                     (COND ((NULL STRING) NIL)
536                           ((EVAL (CAR L))
537                            (PRINTARC ARCTYPE)
538                            (APPLY (QUOTE PROCNX) (CDR L)))))))))
539
540
541   (DEFPROP ERROR5 EXPR
542     (LAMBDA NIL
543             (PERROR ATTEMPT
544                     TO
545                     EVALUATE
546                     GETF
547                     ON
548                     NON
549                     CAT
550                     ARC
551                     :
552                     CURRENT
553                     STATE
554                     =
555                     (STATE)
556                     :
557                     CURRENT
558                     ARC
559                     =
560                     (ARCTYPE))))
561
562
563   (DEFPROP ERROR6 EXPR
564     (LAMBDA NIL
565             (PERROR ATTEMPT
566                     TO
567                     SENDR
568                     ON
569                     NON
570                     PUSH
571                     ARC
572                     :
573                     CURRENT
574                     STATE
575                     =
576                     (STATE)
577                     :
578                     CURRENT
579                     ARC
580                     =
581                     (ARCTYPE))))
582
583
584   (DEFPROP GLUE EXPR
585     (LAMBDA (X)
586             (COND ((NULL X) NIL)
587                   (T (APPEND (CAR X) (GLUE (CDR X)))))))
588
589
590   (DEFPROP PERROR EXPR
591     (FLAMBDA (&LS)
592             (TERPRI)
593             (PRIN1 (QUOTE ">>>> ERROR--"))
594             (MAPC
595               (QUOTE
596                 (LAMBDA (&XS)
597                         (COND ((EQ &XS (QUOTE :)) (TERPRI) (TAB 12))
598                               ((ATOM &XS) (PRIN1 &XS))
599                               (T (MAPC (QUOTE (LAMBDA (&YS) (PRIN1 (EVAL &YS)))) &XS))))))
```

```
600                 $LS)
601               (TERPRI)))
602
603
604     (DEFPROP PTRACE EXPR
605       (FLAMBDA (SLS)
606               (COND
607                 (:TRACE (TERPRI)
608                         (SKIP :TAB)
609                         (MAPC
610                           (QUOTE
611                             (LAMBDA (SXS)
612                                     (COND ((EQ $XS (QUOTE :)) (TERPRI) (SKIP :TAB))
613                                           ((ATOM $XS) (PRIN1 $XS))
614                                           (T (MAPC (QUOTE (LAMBDA (SYS) (PRIN1 (EVAL SYS)))) SXS)))))
615                           $LS)
616                         (TERPRI)
617                         T))))
618
619
620     (DEFPROP PPARSE? EXPR
621       (LAMBDA NIL
622               (SETQ :PRINTPARSE (NOT :PRINTPARSE))))
623
624
625     (DEFPROP ERRSET EXPR
626       (FLAMBDA ($EXPS)
627                (LIST (EVAL (CAR $EXPS)))))
628
629
630     (DEFPROP ERROR EXPR
631       (FLAMBDA ($VALS)
632                (UNEVAL (QUOTE ERRSET) (CAR $VALS))))
633
634
635     (DEFPROP ABORT EXPR
636       (LAMBDA NIL
637               (PTRACE ABORTED)
638               (SETQ BODY-S (CDR BODY-S))
639               (GO TAG)))
640
641
642     (DEFPROP :REPLACE EXPR
643       (LAMBDA (LST EL N)
644               (APPEND (BUT-NTH LST N) (CONS EL (CDR (NTH LST N))))))
645
646
647     (DEFPROP BUT-NTH EXPR
648       (LAMBDA (L N)
649               (COND ((EQ N 1) NIL)
650                     (T (CONS (CAR L) (BUT-NTH (CDR L) (SUB1 N)))))))
651
652
653     (DEFPROP FINDHEIGHT EXPR
654       (LAMBDA (X)
655               (COND ((NUMBERP X) (COND ((LESSP (LENGTH STACK) X) 0) (X)))
656                     (T (FINDPOS STACK X)))))
657
658
659     (DEFPROP FINDPOS EXPR
660       (LAMBDA (L ST)
661               (PROG (CNT PTR)
662                     (SETQ CNT 0 PTR L)
663                 TAG (COND ((NULL PTR) (RETURN 0))
664                           ((EQ (CAAR PTR) ST) (RETURN (ADD1 CNT)))
665                           (T (SETQ PTR (CDR PTR) CNT (ADD1 CNT)) (GO TAG))))))
666
667
668     (DEFPROP LIFTR EXPR
669       (FLAMBDA (L)
670               (PROG (HEIGHT TEMP)
671                     (COND
672                       ((NULL (CDDR L))
673                        (RETURN (APPLY (QUOTE LIFTR) (APPEND L (LIST 1))))))
674                     (SETQ HEIGHT (FINDHEIGHT (CADDR L)))
675                     (RETURN
676                       (COND ((ZEROP HEIGHT))
677                             (T
678                               (SETQ STACK
679                                     (:REPLACE STACK
680                                               (:REPLACE (SETQ TEMP (CAR (NTH STACK HEIGHT)))
681                                                         (CONS (CONS (CAR L) (EVAL (CADR L)))
682                                                               (CADR TEMP))
683                                                         2)
684                                               HEIGHT)))))))
685
686
687     (SETQ :PRINTPARSE T :REGS NIL)
688
689
690     (UNTRACEPARSE)
691
692
END OF FILE
```