AN AUGMENTED TRANSITION NETWORK GRAMMAR FOR ENGLISH

by

Jean E. Jervis

B.Sc., University of British Columbia, 1972

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 1974
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 May 24, 1974.
ABSTRACT

The use of augmented transition network (ATN) grammars for the analysis of natural language sentences is discussed. A small sample grammar is illustrated and briefly described. An ATN grammar for English was implemented and is described in detail. This grammar uses both semantic and syntactic information to guide the parsing. The value of the ATN model for natural language analysis is critically evaluated.
# TABLE OF CONTENTS

I. INTRODUCTION ................................................. 1  

II. GENERAL OVERVIEW AND BACKGROUND ......................... 3  

1. Transition Network Grammars .............................. 3  
2. Simple Transition Network Grammar ...................... 5  
3. Augmented Transition Network (ATN) Grammars .......... 8  
4. The Implementation Of An ATN Grammar .................. 12  
    4.A Semantic Markers ...................................... 13  
    4.B System Overview ...................................... 14  
    4.C Capabilities Of The Grammar ......................... 14  

III. SYSTEM DESCRIPTION ........................................ 16  

1. Pre-Scan Of The Input Sentence .......................... 16  
    1.A The Dictionary ....................................... 17  
    1.B Morphemic Analysis .................................... 19  
2. The Grammar .................................................. 21  
    2.A The Sentence Level Network .......................... 27  
        2.A.1 Declarative Sentences ......................... 27  
            Transitive Verbs .................................. 27  
            Intransitive Verbs ................................. 29  
            Indirect Objects ................................. 30  
            Predicate Adjectives ............................... 32  
            Predicate Adjective Complements .................. 33  
            Auxiliary Verbs ................................. 33  
            Negation ......................................... 36  
            Verb Particles .................................... 39
"That" And "To" Verb Complements ..................40
2.A.2 Imperative Sentences ...............................41
2.A.3 Passive Sentences ................................42
Finding The Agent Of Passive Sentences ..........43
2.A.4 Questions ............................................45
Yes-No Questions .........................................46
Question Adverbs .........................................47
Question Pronouns .......................................48
Question Determiners ...................................50
2.A.5 Conjoined Sentences .................................51
2.A.6 Noun And Preposition Phrase Utterances ....53
2.B The Noun Phrase Network .............................54
2.B.1 Basic Parsing Strategy .............................54
2.B.2 Simple Noun Phrases ..............................55
2.B.3 The Pre-Nominal Modifiers ......................56
Determiners ................................................56
Ordinals .....................................................57
Quantifiers ...............................................57
Adjectives .................................................58
2.B.4 Post-Nominal Modifiers ............................59
Preposition Phrases ......................................59
Relative Clauses ........................................60
2.B.5 Incomplete Noun Phrases .........................61
2.B.6 "Something" Constructions ......................62
2.B.7 Conjoined Noun Phrases .........................63
2.C Extensions .............................................63
2. Suggestions For Prospective Grammar Writers ..........88
   2.A Common Problems ........................................88
   2.B Ordering The Arcs .......................................89
   2.C Conditions On The Arcs ...............................90
BIBLIOGRAPHY ..................................................92
APPENDIX 1 - HOW TO RUN ....................................94
APPENDIX 2 - SAMPLE PARSES ................................96
APPENDIX 3 - THE GRAMMAR ..................................118
APPENDIX 4 - THE DICTIONARY ..............................139
APPENDIX 5 - THE MORPH TABLE ............................143
APPENDIX 6 - THE PREPASS ROUTINES ....................144
APPENDIX 7 - THE SEMANTICS ROUTINES ..................148
APPENDIX 8 - THE AUXILIARY GRAMMAR ROUTINES ........152
APPENDIX 9 - THE PARSER ....................................154
# TABLE OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Simple Transition Network Grammar</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2</td>
<td>The Grammar</td>
<td>22</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Parse of a Simple Declarative Sentence</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Parse of a Sentence with Indirect Object</td>
<td>32</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Parse of a Sentence in Future Perfect</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Progressive Tense</td>
<td></td>
</tr>
<tr>
<td>Figure 6</td>
<td>Parse of a Sentence Containing the Modal &quot;Can&quot;</td>
<td>36</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Parse of an Imperative Sentence</td>
<td>41</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Parse of a Passive Sentence</td>
<td>43</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Parse of a Yes-No Question</td>
<td>47</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Parse of a Noun Phrase</td>
<td>55</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENT

I would like to thank my supervisors, Dr. Raymond Reiter and Dr. Richard Rosenberg for originally interesting me in computational linguistics and for their many valuable suggestions and criticisms. The financial support of the National Research Council is gratefully acknowledged.
I. INTRODUCTION

The field of computational linguistics is primarily concerned with the study of machine understanding of natural language. The term "natural language" in this thesis refers to written English. Traditionally research into the structure of language has been the domain of the field of linguistics. Linguistics seeks to develop an all-encompassing theory of language, whereas the goals of computational linguistics are somewhat more practical. Linguistics attempts to characterize the structural regularities of language by proposing grammatical theories which enable all well-formed sentences of the language to be generated.

Computational linguistics is more concerned with the recognition of sentences; that is, given a sentence determine its structural components and the relationship among them. Linguistic theory is not of much help in this situation. "Pathological" cases which may be of great importance to the linguist generally do not concern the computational linguist who is more interested in a system which works accurately and efficiently on a "useful" subset of English. The primary objective in computational linguistics is understanding the information which is being communicated; the form of the information is of less importance.

The first step in machine understanding of a sentence is decomposing or parsing the sentence into its structural
components. Many parsing schemes have been suggested\cite{7}. This thesis investigates the use of augmented transition network (ATN) grammars for the analysis of English sentences. The ATN model of natural language was developed by Woods\cite{11}. This model is capable of performing the equivalent of a transformational analysis of English sentences in a computationally efficient manner. This thesis discusses the development of an ATN grammar for English which identifies a subset of English that would be used, for example, in a question answering system. Both semantic and syntactic information are used to guide the parsing.
II. GENERAL OVERVIEW AND BACKGROUND

The first section in this chapter discusses the use of transition networks for the analysis of English sentences. In the next section a small sample grammar is illustrated and explained. Augmented transition network (ATN) grammars are discussed in the third section. The last section gives a brief introduction to the implementation discussed in Chapter III.

1. Transition Network Grammars

This section describes the transition network grammar model developed by Woods[11,12]. A transition network grammar model is an extension of the idea of a finite state transition diagram. A transition network grammar consists of a network of nodes with arcs connecting them. The nodes represent the states of the "parsing machine" and the arcs connecting them represent possible transitions. Each arc is labelled with the events which permit a transition from that state to the next. In a transition network grammar the events correspond to the occurrence of words in the input. The network has a distinguished state called the start state and a set of distinguished states called final states. A state is said to accept or recognize a string if the string permits a sequence of transitions from that state to a final state. The arcs in a
transition network grammar may be either lexical arcs which correspond to transitions permitted by single words or recursion arcs (PUSH arcs) which invoke recursive applications of the network to recognize a phrase of some kind.

Woods[11] considers that there are five basic arc types - PUSH, POP, JUMP, TST, and CAT. A CAT arc represents a transition that may be taken if the current word in the input string is a member of the syntactic category indicated on the arc. After a CAT arc is followed the input string is advanced causing the next word in the input to become the current word.

A TST arc is similar to a CAT arc except that the conditions for the transition are specified by the arbitrary condition on the arc. A CAT arc is actually a special case of a TST arc, but because it is so commonly used it is given a separate name.

A JUMP arc is like a TST arc except the input string is not advanced. This arc is useful for bypassing states where optional constituents are identified.

A POP arc is a pseudo arc because it has no destination state. It is considered to be an arc so that its order with respect to the other arcs leaving the state may be established. The POP arc indicates that the state is a final or accepting state for the transition network.

A PUSH arc invokes a call to the network whose name is on
the arc. The PUSH arc permits a transition to be made if the specified phrase can be recognized at the current position in the sentence. For example, if the NP network identifies noun phrases then the PUSH NP arc must locate a noun phrase at the current position before the transition may be made. The PUSH arc advances the input string beyond the last word of the recognized phrase.

The parsing system also recognizes two other types of arcs. A WRD arc permits a transition if the current word is the same as the word named on the arc. A MEM arc is followed if the current word is a member of the list of words given on the arc.

2. Simple Transition Network Grammar

A simple transition network grammar is shown in Figure 1. This network grammar recognizes simple declarative sentences with noun phrases containing determiners and adjectives. Consider the sentence:

The big dog chased him.

To recognize this sentence, processing starts in state S of the network. A transition may be made to state S-NP if a phrase of the type recognized by the NP network (i.e. a noun phrase) is found. The PUSH arc remembers the state of the top level computation and passes control to state NP.
Figure 1 - Simple Transition Network Grammar
The parser now tries to parse a noun phrase starting at state NP. The current word "the" is a determiner so the CAT DET arc permits a transition to state NP-DET, then the adjective "big" causes a transition to state NP-ADJ and the noun "dog" a transition to state NP-NP. The POP arc at state NP-NP causes a return to the PUSH arc at state S. Since the PUSH was successful, a transition to state S-NP is made.

From S-NP the verb "chased" permits a transition to state S-V. Although S-V is a final state the last word of the input has not been processed so the string cannot be accepted here. The PUSH NP arc is taken to state S-S because the pronoun "him" is recognized in the NP network. Since S-S is a final state and "him" is the last word, the string is recognized as a sentence.
3. Augmented Transition Network (ATN) Grammars

The augmented transition network (ATN) parser described in this section was originally implemented at Harvard and subsequently extended at Bolt, Beranek, and Newman by Woods[11,12,13]. For more detailed documentation of Woods' parser please refer to Woods[12,13]. A simple version of this parser was implemented at the University of British Columbia by R. Reiter. A complete listing of this parser may be found in Appendix 9.

The transition network grammar described above acts only as a sentence recognizer. In order for such a grammar to be of any practical value it must also be able to provide a description of the syntactic structure of a recognized sentence.

The ATN grammar provides this capability by building up a structural description of the sentence as it proceeds from state to state. The components of this description are placed in registers which are maintained at each level of the recursive network. Each POP arc of the grammar has a form associated with it which specifies how these registers are to be assembled into the structural representation which is returned from that state. Structure-building actions on the arcs of the grammar may test, set, or alter the contents of the registers. Each arc of the network may have an arbitrary condition associated with it which must be satisfied before the arc is followed.
The conditions and structure-building actions on the arcs of the network provide the transition network grammar with capabilities equivalent to those of a transformational grammar[1]. By manipulating the contents of the registers the grammar now has the ability to move, copy, and delete fragments of the sentence structure. The conditions on the arcs and the register testing actions permit the grammar to behave in a context dependent manner.

The design of the ATN grammar permits structural descriptions to be built up in a flexible manner. The pieces of sentence structure are stored in the registers until the parser reaches the final state. Since the decision as to the final structure of the parse is postponed until all the components have been identified, the components of the parse may be ordered differently than the order in which they occurred in the input. A tentative decision as to the function of a component may be reversed and the registers re-arranged if a subsequent input indicates that the decision was incorrect.

The VIR arc and HOLD action provide a convenient method of dealing with displaced constituents in the input. Some transformations cause constituents to be moved to the front of the sentence. For example, a simple yes-no question such as:

Is the man talking?

is introduced by an auxiliary verb. The displaced constituent ("is") is found first in a left to right parse and although it
is obvious that it is displaced, its correct position in the sentence is as yet unknown. The HOLD action allows the constituent ("is") to be placed on a HOLD stack where it will remain until a state is encountered where it would normally be identified. At this point the VIR arc can remove the constituent from the HOLD stack and treat it as though it had just occurred in the input. This feature is very useful since it allows the grammar writer to take full advantage of the regularities of natural language.

The transition network is non-deterministic; the input does not uniquely determine a path through the network. Thus there will be English sentences which have several distinct paths through the network. These are said to be ambiguous with respect to the grammar. The parsing algorithm should be capable of following all paths for a given sentence.

The parser considers the arcs leaving a state in the order in which they occur. The first possible arc is followed. If this choice later turns out to be unsuccessful the parser backs up, trying all other alternatives in the order in which they occur, until either a path is found through the network or it is determined that no path exists.

Sentences which are ambiguous with respect to the grammar can be parsed in more than one way. Consider the sentence:

They are flying planes.

This sentence may be parsed in two different ways. Either "are
flying" is the progressive form of the verb "fly" or "flying" is used to modify the noun "planes". The transition network will return the parse generated by the first path through the network. When common usage and context are taken into consideration it is usually clear which of the parses of a syntactically ambiguous sentence is most likely intended. The problem, then, is to find the "most likely" path first. Even when ambiguous sentences are not involved, it is important for reasons of efficiency to try to find the correct path quickly, thereby minimizing the amount of searching that must be done. The ordering of the arcs leaving the states and the conditions on them allow the parser to try the more common or more likely constructions first.

The transition network parser is written in Lisp. This programming language appears to be an excellent choice for this application since recursion is very natural in the language. Lisp is an interpreted language so it is possible to use arbitrary Lisp functions as well as the register manipulating functions supplied by the parser for the conditions and actions on the arcs. The Lisp environment also makes sophisticated interactive debugging aids available to the grammar writer[2].
For many applications in natural language analysis it is not necessary to obtain all possible parsings of the input sentence. Although a sentence may be ambiguous with respect to the grammar, it usually will not be ambiguous when the meaning of the words and the context in which it was uttered are taken into consideration. Thus heuristics can be used in an attempt to select the "most likely" parse in a given context. In an ATN grammar this can be done by ordering the arcs leaving each state in the grammar so that those corresponding to the most probable analyses are tried first. Conditions may be placed on the arcs so that their ordering becomes dependent on the features of the sentence being analyzed. Woods[11] mentions that these conditions may incorporate semantic as well as syntactic features of the words. However he does not appear to use any semantic information to guide the parsing in the LSMLIS system[13].

The ATN grammar described in the next chapter is an extended version of the WALT system grammar[3] using semantic markers to aid in the selection of the "most likely" parse structure.
4. Semantic Markers

The concept of semantic markers was first introduced by Fodor and Katz in "The Structure of a Semantic Theory". They comment that:

"... the semantic markers assigned to a lexical item in a dictionary entry are intended to reflect whatever systematic semantic relations hold between that item and the rest of the vocabulary of the language."\(^1\)

Thus semantic markers provide a semantic classification of words which may be used by the conditions on the arcs of the grammar to exclude semantically meaningless parses.

The determination of the "most likely" parsing of some sentences requires knowledge of the context in which the sentence occurred. Tests are used in addition to the semantic markers for determining whether the parsed structure is reasonable in terms of its context.

---

4.B System Overview

Before the parsing of a sentence is attempted a PREPASS program scans the sentence to ensure that all the words are recognized by the system. The PREPASS program must establish correct entries in the system dictionary for all the words in the sentence. A morphemic analysis routine is used to create dictionary entries for regularly inflected words whose root forms are already in the dictionary. Contractions and abbreviations are expanded to their full forms at this time. The resulting sentence is passed to the parser for analysis.

An ATN grammar is used to parse the sentence into a deep structure-like representation. As discussed above, heuristics involving semantic as well as syntactic information are used to guide the parsing. The dictionary which contains syntactic and semantic information for each word is accessed throughout the parsing process.

4.C Capabilities of the Grammar

The grammar identifies a subset of English constructions that would be used, for example, in a question answering system. Many types of questions and relative clauses as well as declarative and imperative sentences are recognized. Verbs may have direct and indirect objects, prepositional phrases, predicate adjectives, complements, or particles if appropriate.
Various tenses and modal forms of verbs are identified. Sentences may be in the active or passive voice. Some negative forms and conjoined phrases are recognized. Sentence fragments consisting of noun or preposition phrases are identified.
III. SYSTEM DESCRIPTION

The general organization of the parsing system is discussed in this chapter. The first section deals with the pre-scan of the sentence, the second with the grammar itself, and the last with the semantic tests.

1. Pre-scan of the Input Sentence

Before the actual parsing of a sentence is attempted, correct dictionary entries must be established for all the words in the input sentence. A morphemic analyser reduces the number of words necessary in the initial dictionary by removing suffixes from regularly inflected nouns, verbs, adjectives and adverbs. Numbers, lists and other proper nouns are assigned syntactic categories and entered into the dictionary. Contractions and abbreviations are expanded at this time into their full forms. Phrases which are considered as a unit by the system are collapsed into a single hyphenated word. These functions are all handled by the PREPASS routines. Please refer to Appendix 6 for a complete listing of the PREPASS functions.
1. A The Dictionary

The dictionary entry for a word is found on the property list of that word. The property flag DICT indicates that a word has a dictionary entry. The dictionary contains both syntactic and semantic information. The syntactic information will be considered first.

Inflectional features of a word are stored on the word's property list under the property flag of the appropriate syntactic category (N, V, ADV, DET, etc.). For the root form of a word this entry is an atom such as ER-EST, S-ED, etc. This entry is used by the morphemic analyser. Inflected word forms have an entry consisting of a list of inflectional features. The first element of this list is the root form of the word, the other elements are inflectional features such as number, case, person, etc. For verbs a second list under the flag FEATURES has a list of atoms specifying special features of the verb. These indicate whether the verb may take a direct or indirect object or that-complement, whether it may be passivized or act as an auxiliary, modal or copula, etc. Verbs such as "pick" which may be used with a particle (e.g. "pick up") have a list consisting of admissible verb-particle combinations under the flag PARTICLES.

Semantic information also occurs in the dictionary. This information is used in parsing sentences to aid in the selection of the most likely parse and to reject obviously nonsensical
input. Each noun may have a list of semantic markers such as ANIMATE, PHYSOBJ, EVENT, etc. under the flag N-TYPE. These markers may be very specific or they may describe only the general classification into which the noun falls. A Lisp predicate may be found on the property list of a noun under the flag N-SEMANTICS. This predicate which is evaluated when a noun phrase is identified may make more specific syntactic or semantic checks on the components of the phrase. Pronouns, with the exception of "it" and "they", have semantic marker ANIMATE under the flag N-TYPE. An adjective may also have a list of semantic markers under the flag ADJ-TYPE describing the types of nouns that it may modify. A verb may have semantic markers under the flags SUBJ-TYPE, DO-TYPE, and INDO-TYPE describing the classes of subjects, direct and indirect objects respectively that can reasonably be expected to accompany this verb. A Lisp predicate may be found on the property list of a verb under the flag V-SEMANTICS. This predicate can perform any type of semantic checks on the parse tree and is evaluated after the parse is completed. If the semantic markers are absent then the word is assumed to be compatible in all situations. For example the adjective "good" has no semantic markers since it may modify almost any type of noun.

Integers are entered into the dictionary under the category INTEGER on an atom whose printname is the same as the integer. Lists are entered as proper nouns and are assigned a name (LIST1, LIST2, etc.) by the PREPASS routine. The root form in
the dictionary is the list itself.

A word in the dictionary may be flagged SUBSTITUTE. If this word is encountered in an input sentence PREPASS will replace it with the substitute word or words. This feature may be used for alternative spellings, or expanding contractions or abbreviations. For example, "can't" and "cannot" are both replaced by "can not" for ease in processing.

The flag COMPOUND may also be found on a word in the dictionary. Under the flag COMPOUND there is a list of compounds that this word forms. If this word is encountered in an input sentence PREPASS searches the next words in the string to see if any group of words is the same as one of the possible compounds. If so, PREPASS will replace the words by the compounded form. For example, "how many" is compounded to "how-many" since the two words behave as one.

The current system dictionary may be found in Appendix 4.

1.8 Morphemic Analysis

The morphemic analysis facility reduces the number of initial dictionary entries needed in the system. A regularly inflected noun, verb, adjective, or adverb requires only a single dictionary entry containing its root form and a code indicating the type of regular inflection the word uses. The
morphemic analyzer then can recognize all regularly inflected forms of the root word and place the appropriate entry in the dictionary. The analysis is done by using a table called MORPHTABLE. This table is a list, each element of which consists of a suffix to be removed from a word followed by a list of possible syntactic categories for the resulting word. The morphemic analysis makes no attempt to recognize prefixes or more complicated situations such as the example below, where words move from one syntactic category to another adding suffixes each time.

Examples:

transport -> transportation
formal -> formalize -> formalization

A listing of MORPHTABLE may be found in Appendix 5. The morphemic analysis routines are listed in Appendix 6.
2. The Grammar

This section discusses the overall organization of the grammar and examines in some detail the problems encountered and the strategies used in solving these problems. Please see Chapter II for a general discussion of augmented transition network grammars and parsers. The transition network diagram of the grammar in Figure 2 will be useful in understanding this section.

The state names in the grammar indicate the level of the network being processed and the constituents of that network already identified. For example the state NP-N indicates that the NP-level of the network is being processed and that a noun has been identified. Unless the order of the arcs is explicitly indicated by numbers on the arcs, they are numbered clockwise from the top of the state. For further details concerning the tests and actions on the arcs please see the complete grammar listing in Appendix 3.

The parses produced by the grammar are similar to the deep structures produced by Woods[13]. Basically the parse of a sentence consists of the subject, an auxiliary verb segment specifying the tense and modality, and a verb phrase containing the main verb, and the direct and indirect objects, complements, preposition phrases and adverbs, if any of these are present. The deep structure of a passive sentence is identical to its active counterpart. Questions are transformed to their
Figure 2 - The Grammar
Figure 2 continued
Figure 2 continued
declarative forms. The exact specification of the form of the deep structures is contained in the listing of the structure-building routines in Appendix 8 and in the listing of the grammar in Appendix 3. Some sample parses may be found in Appendix 2.

2.4 The Sentence Level Network

The sentence or S-level network is the main network in the grammar. This network identifies the subject, tense, main verb, and post-verbal modifiers, if any, of the sentence, checks the semantic agreement of the major components and builds the associated parse tree. The initial state in the grammar is S.

2.4.1 Declarative Sentences

The parsing of a simple declarative sentence embodies the basic parsing strategy. All the more complicated sentence types are handled with minor variations of this strategy.

Transitive verbs

Consider the sentence:

The boy kicks the ball.

The parser starts by comparing the string to the grammar in
state S. The first word "the" does not look like the beginning of an English question so the predicates which test for tensed auxiliary verbs and question words fail and the first two JUMP arcs are excluded. The input does not start with an untensed verb as an imperative does nor with an introductory preposition phrase so the next two JUMP arcs are also excluded. The JUMP S-DCL arc is taken since the input does not begin like a question. Now it has been established that the input is a declarative sentence, so the TYPE register is set to DCL.

At state S-DCL the parser is still looking at the word "the" since the JUMP arc does not advance the input string. In this state the parser attempts to locate the subject noun phrase. The only arc is a PUSH to the noun phrase (NP) level which is successful and the structure:

(NP (DET THE) (N BOY (NUMBER SG)) (NU SG))

is returned. The parser proceeds to state S-NP to look for verbs. The current word is "kicks". The VIR V arc fails since there is nothing on the HOLD stack. The current word is a tensed verb which agrees in number (singular) and person (third) with the subject so the CAT V arc is taken. The current word becomes "the". The parser proceeds to state S-AUX to look for a main verb. No more verbs are found so the last arc in S-AUX, a JUMP to S-V-SEMANTICS is taken. The semantic agreement between the subject and the verb is checked in S-V-SEMANTICS. Since the verb "kick" has the ANIMATE subject "boy" the check succeeds and the parser jumps to state S-V. At this state in the network the
subject and main verb have been identified and the parser attempts to find the post-verbal modifiers. Since the verb "kick" is transitive the PUSH NP arc attempts to locate an object noun phrase. This arc succeeds and the structure:

(NP (DET THE) (BALL (NUMBER SG)) (NU SG))

becomes the direct object of the verb. The remaining input string is now NIL, so the parse proceeds on a series of JUMP arcs through the remaining states of the S-level network to the final state S-S. Before the final POP arc can be taken the semantic agreement among the principal elements of the parse must be checked. The semantic check succeeds since the verb "kick" has the ANIMATE subject "boy" and the PHYSOBJ direct object "ball". The structure in Figure 3 is produced as the deep structure of the input sentence.

Thus the basic strategy for simple declarative sentences involves finding the subject noun phrase at S-DCL, the verb at S-NP, and the object noun phrase at S-V. The parser then jumps to S-S where the semantics are checked and the parse popped.

Intransitive verbs

Example:

The men talked.

If the main verb is intransitive the parser does not look for an object noun phrase at S-V. Instead the parser jumps
SENTENCE: THE BOY KICKS THE BALL

PARSE:
S
  MOOD DCL
  VOICE ACTIVE
  NP
    DET THE
    N BOY
    NUMBER SG
    NU SG
  AUX
    TNS PRESENT
  VP
    V KICK
    NP
      DET THE
      N BALL
      NUMBER SG
      NU SG

Figure 3 - Parse of a Simple Declarative Sentence

through the remainder of the states to S-S. Minor variations on this strategy recognize post-verbal modifiers such as adverbs and preposition phrases.

Indirect objects

The parsing strategy becomes more complicated when the main verb may take an indirect object. Consider the following sentences:

(1) He gave the book to the boy.
(2) He gave the boy the book.

Jacobs and Rosenbaum in "English Transformational Grammar"[4] explain that these sentences are both generated from the same deep structure, the only difference being that an additional
transformation has been applied to sentence (2). The "indirect object inversion transformation" reverses the order of the direct and indirect object noun phrases and deletes the preposition. This transformation is optional since both sentences (1) and (2) are grammatical. Sentences of the first type, in which there is no indirect object inversion transformation, are parsed in the same manner as other sentences with transitive verbs. The only difference is that the indirect object appears in a prepositional phrase as a post-verbal modifier. The parse produced is shown in Figure 4.

Now consider sentences in which indirect object inversion has occurred. The verb is identified at state S-NP. Since it is transitive the "direct" object is picked up at S-V. The parser jumps to state S-V-NP. The PUSH NP arc is attempted here because the verb "give" is flagged as being able to take an indirect object (INDOBJ). The push to the noun phrase network is successful. At this point the registers containing the deep structure are re-arranged. The noun phrase structure just found becomes the direct object of the verb and a preposition phrase consisting of the previous "direct" object and the preposition "to" is constructed. The parser then jumps to state S-S since the entire string has been analysed. The parse produced for sentence (2) is exactly the same as for sentence (1) since both sentences were produced from the same deep structure.
SENTENCE:  HE GAVE THE BOOK TO THE BOY

PARSE:
S
   MOOD DCL
   VOICE ACTIVE
   NP
      DET NIL
      PRO HE
      NUMBER SG
      SUBJ
      PNCODE 3SG
      NU SG
   AUX
   TNS PAST
   VP
      V GIVE
      NP
         DET THE
         N BOOK
         NUMBER SG
         NU SG
      PP
      PREP TO
      NP
         DET THE
         N BOY
         NUMBER SG
         NU SG

Figure 4 - Parse of a Sentence with Indirect Object

Predicate adjectives

Copula verbs may be followed by predicate adjectives.

Example:

The boy was young.

At state S-V if the main verb was a copula verb, the parser checks for the presence of a predicate adjective. Since the predicate adjective modifies the subject a semantic check is performed on the subject and adjective. If the two are compatible the predicate adjective becomes the verb in the deep
structure. In one theory of transformational grammar[4] the deep structure of this type of sentence has the adjective as the verbal and an obligatory "copula transformation" introduces the copula verb into the sentence.

**Predicate adjective complements**

Consider the sentence:

The boy was younger than his brother.

A predicate adjective complement may follow a comparative predicate adjective. The parser recognizes the predicate adjective in state S-V. Analysis of the input continues at S-PREDADJ. If the word "than" is found and the adjective has the feature COMPARATIVE the scan proceeds to state S-PREDADJ-COMP. Here the parser attempts to find a noun phrase which becomes the object of the predicate adjective "verb". No semantic checking is done at this point, although the predicate adjective must be semantically compatible with the object noun phrase. This check should be added to the grammar.

**Auxiliary verbs**

Thus far only sentences with simple verbs have been discussed. Auxiliary verbs are important in English since they carry the tense and modality, if any, of the sentence.
Examples:

(1) They had talked.
(2) They will have been talking.
(3) They can talk.
(4) They could have talked.
(5) They do talk.

The auxiliary verb is identified at state S-NP. This verb must be tensed and agree in person and number with the subject. The tense (PRESENT or PAST) is saved in register TNS. The parser then proceeds to state S-AUX to look for more verbs. This state recognizes future tenses, the perfect, progressive, and perfect progressive aspects of verbs as well as modal and emphatic "do" constructions. Passive constructions are recognized in this state. These will be discussed later. The verbs "will" and "shall" are both considered to be future tense auxiliaries and their modal meanings are ignored since they seem to be very commonly misused in everyday English. The modal verbs "can", "may", "could", "might", etc. are all assigned PRESENT tense. Rosenbaum and Jacobs[4] consider the syntactic tense of the modals "can", "may", etc. to be present and "could", "might", "would", etc. to be past. The syntactic tense does not appear to carry much semantic significance, so it seemed more reasonable to leave the tenses as present. Later it may be more practical to add a CONDITIONAL mood or tense.

After a passive or an emphatic use of the word "do" is encountered the parser goes to state S-V since no more verbs may follow.
Examples:

They do talk.
The book was given.

Otherwise the parser remains in state S-AUX until all the verbs are found. An emphatic use of the word "do" is parsed as if "do" were a modal.

SENTENCE: THEY WILL HAVE BEEN TALKING

PARSE:

S

MOOD DCL
VOICE ACTIVE
NP
DET NIL
PRO THEY
NUMBER PL
SUBJ
PCODE 3PL
NU PL
AUX
TNS FUTURE PERFECT PROGRESSIVE
VP
V TALK

Figure 5 - Parse of a Sentence in Future Perfect Progressive Tense

The deep structures for sentences (2) and (3) are shown in Figures 5 and 6. These deep structures resemble those discussed by Jacobs and Rosenbaum[4]. Their deep structures have a verbal segment (usually the verb) with features indicating progressive or perfect aspect, or both, and an auxiliary segment. The auxiliary may or may not be a modal and has a tense feature. According to the theory of transformational grammar the surface
structures are generated by applying rather complicated transformations. During this process the auxiliary segment may be deleted. If it immediately precedes the verb and is neither perfect nor a modal then its features are placed on the verb and the auxiliary is deleted. Thus an auxiliary does not appear in the surface structure of present tense sentences like the following:

The man talks.

Negation

The grammar can also identify negative sentences.

Examples:

He does not come.
He must not come.
They won't be coming.
If an auxiliary verb has been found in state S-NP, the grammar will recognize the word "not" in state S-AUX. The flag NEGATIVE is then added to the sentence type.

According to Jacobs and Rosenbaum the constituent NEGATIVE appears in the deep structure of a negative sentence. The "negative placement transformation" places the NEGATIVE constituent immediately after the auxiliary in the deep structure. Further transformations replace NEGATIVE by "not" and the auxiliary, if one is not already present, by "do". An optional transformation contracts the auxiliary and "not". Thus sentences such as the following are formed.

Example:

He doesn't come.

The contractions are expanded to their full forms by the PREPASS routines, so the grammar need not be concerned with them. It can be seen from the transformations described above that all negative sentences contain auxiliary verbs. Therefore the verb "do" in negative sentences is not being used as a modal, so it does not appear in the deep structure.

Klima in "Negation in English"[6] discusses many other aspects of negation. Adverbs such as "never", "hardly", "scarcely", "seldom", etc. convey varying degrees of negativity.
Examples:

He never believed him.
He hardly believed him.

Verbs may also be inherently negative or convey some degree of negativity.

Examples:

He was unable to come.
He doubted he would come.

Negative adjectives and pronouns may also negate the sentence.

Examples:

No men came.
Not much was done.

There are also negative clauses introduced by "not", "neither" and "neither-nor" combinations.

Examples:

He seldom came and neither did she.
It neither rained nor snowed that winter.
I told him not to come today.

Negation does not necessarily imply sentence negation.

Examples:

A not unattractive woman arrived later.
The children were very unhappy.

The rules describing the scope of the negation and the other structures which may accompany the negation become fairly complex. The grammar does not handle these constructions, with the exception of the words "no", "none", and "nothing". There is room for much more work to be done in this area.
Verb particles

A sentence may contain a verb particle in its verb phrase. A particle looks like a preposition, but behaves quite differently. Compare these sentences:

(1) The robot picked up the block.
(2) The man talked on the phone.

In sentence (1) the word "up" may be transposed to the end of the sentence:

(3) The robot picked the block up.

The second sentence may not be transposed in this manner:

(4) *The man talked the phone on.

The word "up" in sentences (1) and (3) is called a verb particle. The transposition of the particle from its position immediately after the verb to the far side of the object phrase is caused by the "particle movement transformation"[4]. This transformation is optional when the object is a simple noun phrase since sentences (1) and (3) are both grammatical. The transformation must occur if the object phrase is a pronoun:

The man put it down.
*The man put down it.

The grammar recognizes particles which occur immediately after the verb (S-V) or object noun phrase (S-V-TOCOMP). The parser checks the dictionary to ensure that the verb-particle combination is valid, then the verb is altered to a verb-particle compound form. Particles are assumed to be prepositions which is perhaps a poor assumption in the case of
particles like "apart".

"That" and "To" verb complements

"That" complements may occur after any verb that is flagged THAT in the dictionary.
Example:
I hope that you come.
Reduced "that" complements are also recognized.
Example:
I hope you come.
The complement is identified at state S-V-WRD=THAT by pushing to the S-DCL network.

"To" complements may occur after any verb. They may be passive.
Examples:
I want to talk to you.
He had to be taken home.
I want John to talk to Mary.
The complement is identified at state S-V-WRD=TO by pushing to the S-NP level network. The subject of the lower level sentence is the object of the top level sentence, if it is present, otherwise it is the subject.
2. A. 2 Imperative Sentences

Imperative sentences begin with an untensed verb, optionally preceded by an adverb or "please".

Examples:

(1) Give the book to the man.
(2) Do not lose the pencil.
(3) Please go home.

The state S-IMP identifies the verb and sets up "you" as the dummy subject. Parsing continues at state S-AUX where a negative construction is identified. The deep structure of sentence (2) is shown in Figure 7.

SENTENCE: DO NOT LOSE THE PENCIL

PARSE:

S

Mood Negative IMP
Voice Active

NP

Det Nil
Pro You
Nu SG-PL

AUX

TNS Untensed

VP

V Lose

NP

Det The

N Pencil

Nu SG

Nu SG

Figure 7 - Parse of an Imperative Sentence
2.A.3 Passive Sentences

Examples:

The function was executed.
The book was given to the boy.

Passive sentences are parsed in the same way as active sentences until state S-AUX is reached. Then if the auxiliary verb is "be" and the current word is a past participle of a verb that may be passivized, the PASSIVEFLAG is set. At this point the registers containing the deep structure must be re-arranged. The "subject" of the sentence is placed on the HOLD stack since it is probably the object of the active sentence. Normally the subject is set to the dummy value "something" and AGFLAG is set to indicate that the agent has not been found, unless a potential agent has already been encountered. The voice of the sentence is set to PASSIVE. The parser then proceeds to state S-V. Here the direct object is removed from the HOLD stack and parsing continues as for an active sentence.

Consider the sentence:

The boy was given the book.

This sentence is more complicated since two transformations are involved, the passive transformation and the indirect object inversion transformation. At state S-AUX it is realized that the sentence is passive. The subject becomes the dummy noun "something" and the noun phrase "the boy" is placed on the HOLD stack. Parsing continues at state S-V where "the boy" becomes
the direct object. The parser jumps to S-V-NP where "the book" is identified as the direct object since "give" is flagged INDOBJ, and "the boy" becomes the indirect object. The parse produced by the grammar is shown in Figure 8.

**SENTENCE:** THE BOY WAS GIVEN THE BOOK

**PARSE:**
```
S
  MOOD DCL
  VOICE PASSIVE
  NP
    DET NIL
    PRO SOMETHING
    NU SG-PL
  AUX
    TNS PAST
  VP
    V GIVE
      NP
        DET THE
        N BOOK
        NUMBER SG
        NU SG
      PP
        PREP TO
          NP
            DET THE
            N BOY
            NUMBER SG
            NU SG
```

**Figure 8 - Parse of a Passive Sentence**

Finding the agent of passive sentences

The agent of a passive sentence, if it is present, occurs as the object of the preposition "by".
Example:

The ball was kicked by the boy.

Usually the agent appears in the verb phrase, but in questions and relative clauses the agent may occur before it is known that the sentence is passive.

Examples:

By which boy was the ball kicked?
The boy by whom the ball was kicked is my brother.

The flag FRONTED-AGFLAG is set before introductory preposition phrases are identified. Then if the preposition phrase begins with "by" the potential agent is lifted to the AGENT register in the S-level network and no deep structure is returned from the preposition (PP) network. When the main verb is identified the AGENT register is examined. If the sentence is passive and the AGENT (subject) agrees semantically with the verb then the AGENT becomes the subject and the AGFLAG is turned off since the agent has been located. Otherwise, as in sentences (1) and (2) the prepositional phrase is re-constructed and added to the post-verbal modifiers.

(1) By which stream was it found?
(2) By which tree did the man lose his hat?

Now consider the situation where the agent occurs in the verb phrase. In state S-AUX when it is discovered that the sentence is passive, the AGFLAG is set to indicate that the agent has not yet been found. The AGFLAG is SENDRed to the preposition and noun phrase networks each time the networks are
searched for post-verbal modifiers. The parse is blocked in the initial PP state if the AGFLAG is set and the preposition "by" is encountered. The parser then fails back through the NP and PP networks to the S-level network without parsing the preposition phrase and jumps to state S-V-PP. A WRD BY arc transfers the search to S-V-PREP=BY, which pushes for a noun phrase. If this agent noun phrase is semantically compatible as the subject of the verb then the agent becomes the subject and AGFLAG is turned off. If the phrase is not compatible the preposition phrase is re-constructed and added to the post-verbal modifiers. The parser continues from S-V-PP. If the real subject appears in the second preposition phrase the judiciously ordered backup through the networks ensures that it will be found.

Examples:

The book was found by the chair by the man.
The book was found under the table by the man.

2.A.4 Questions

The questions recognized by the grammar fall into several categories. These will be discussed in the following sections.
Yes-no questions

Consider the questions:

(1) Did John kick the ball?
(2) Are they coming?

Yes-no questions differ from declarative sentences in that the auxiliary verb introduces the sentence. If the auxiliary is not present then the sentence is introduced by the verb "do". Jacobs and Rosenbaum[4] explain that a QUESTION constituent is present in the deep structure. The "interrogative transformation" interchanges the auxiliary constituent with the subject noun phrase and the QUESTION constituent is deleted. If an auxiliary is not present, the verb "do" must be introduced since the auxiliary constituent no longer directly precedes the verbal. Thus the use of "do" in questions is not modal.

If the input sentence starts with a tensed auxiliary verb then the parser jumps to state S-YESNO. The auxiliary is placed on the HOLD stack to be retrieved with the VIR V arc at state S-NP. The YESNO flag is set to indicate that subject-verb inversion has occurred and the sentence type is set to YESNO. In affirmative questions the parser continues to state S-DCL and parsing continues as for declarative sentences. The deep structure produced for sentence (1) is shown in Figure 9. One complication arises if a contracted auxiliary verb occurs in the input. Since the PREPASS routine substitutes the expanded form directly into the input string the "not" is out of place.
SENTENCE:  DID JOHN KICK THE BALL

PARSE:
S
  MOOD YESNO
  VOICE ACTIVE
  NP
    DET NIL
    NPR JOHN
    NU SG
  AUX
    TNS PAST
  VP
    V KICK
    NP
      DET THE
      N BALL
      NUMBER SG
      NU SG

Figure 9 - Parse of a Yes-No Question

Example:

Won't he come?  \textarrow{\textast} Will not he come?

Thus at state S-YESNO-NEG the grammar checks for a "not" resulting from an expanded contraction and parsing continues as for the affirmative questions.

Question adverbs

These questions are introduced by a question adverb such as "when", "where", "why", "how", etc.

Examples:

Why did he come?
Where are the red blocks?

The parser jumps to state S-WH to pick up the question adverb. The question type is set to QADV. Parsing then continues in the
same manner as for yes-no questions.

**Question pronouns**

These questions are introduced by a question pronoun such as "who", "which", "what", etc. These questions are of three types. Firstly there are questions where the subject noun phrase has been replaced by the question pronoun. No subject-verb inversion occurs in this case.

Example:

Who saw the boy?

Secondly the object noun phrase may be replaced.

Example:

What did he see?

The third question type occurs when the object of a preposition is replaced by a question pronoun.

Example:

To whom was the book given?

Because the WH-question transformation causes the question pronoun phrase to be fronted, subject-verb inversion must occur in the second and third question types.

The first two question types are processed in state S-WH. The object replacement situation is processed first on the assumption that it is more complex and therefore more easily recognizable. The object is placed on the HOLD stack and
processing continues at state S-YESNO to deal with subject-verb inversion. The VIR NP arc in state S-V recovers the object from the hold stack. To ensure that no subject replacement question pronouns slip through the network, a test at S-AUX checks that any main verb which may also be an auxiliary (e.g. "have", "be", "do") also has an auxiliary in the sentence.

Example:

Which book has the man had?

The subject replacement situation is processed by keeping the question pronoun as the subject and jumping to S-NP to continue parsing as for a declarative sentence.

The third situation where the question pronoun is the object of a preposition is processed in state S-PP. If the WH-PHRASE flag was set by the preposition phrase network indicating that a question pronoun was encountered, the parser proceeds to state S-YESNO to deal with subject-verb inversion.

The sentence type becomes QPRO in all these cases. The question pronoun may optionally be followed by a preposition phrase.

Example:

How many of the functions are correct?

A fronted partitive construction may also introduce the question.
Example:

Of the functions how many are correct?

The parsing of these questions may become very complex since several "major" transformations may occur. Consider the sentence:

What was the boy given?

This sentence involves interrogative, wh-question (i.e. The "what" is fronted.), passive, and indirect object inversion transformations. In practical terms this implies that several constituents may be on the HOLD stack at once and that care must be taken to ensure that they are removed in the correct order.

**Question determiners**

These questions have a word such as "which", "what", "whose" or "how many" modifying their introductory noun phrase. Examples:

Which books did you take?
Which man took the books?
To which man were the books given?
By which man were the books taken?

These questions fall into the same three types as the question pronoun questions. The analysis is essentially the same with the exception of the initial few states. State S-WH identifies the question determiner and proceeds to state S-QDET. The sentence type is set to QDET. The determiner of the noun phrase has been identified so it is SENDRed to the noun phrase network.
and the NP network is entered at the state NP-DET to recover the rest of the noun phrase. Processing continues at state S-WH-NP. First the parser assumes the noun phrase is the object of the sentence and goes to S-YESNO. If this fails the parser jumps to S-NP since the noun phrase must be the subject. The process now is identical to the question pronoun situation.

2.4.5 Conjoined Sentences

Only two simple types of conjoined sentences are recognized. The first type involves the conjunction of two or more complete sentences of any type.

Examples:

The boys ran and the girls talked.
Take it or leave it.
Where did John go and when did he leave?

This construction is recognized after a conjunction has been found in state S-MAINCLAUSE. The parser proceeds to state S-CONJ where it pushes to the S-level network.

The second type is a sentence with a conjoined verb phrase.

Examples:

They ran and jumped.
He came and saw and conquered.
Who picked it up and put it on the table?
What was picked up by the man and put on the stack?

The main verb in the second verb phrase is processed in state S-CONJ. If the input string is now NIL the parse is structured in state S-CONJ-BUILD. Otherwise all the necessary flags and
registers are initialized in the lower S-level network and the parser pushes to state S-V to look for post-verbal modifiers in the second verb phrase.

The only conjunctions handled in the grammar are the coordinating conjunctions "and" and "or". The analysis of conjoined forms can become quite complicated. It would be useful in attempting a more detailed analysis to have available a stack describing the constituents parsed. Since coordinating conjunctions join equivalent constructions one needs to know what constructions should be looked for next.

Difficulties arise with this analysis because conjunctions join constituents with equivalent surface structures. When a conjunction is encountered the sentence preceding it has been decomposed into its deep structure. Thus it is necessary to guess what the surface structure must have been using the flags and registers. This information must be used to deduce which elements must be initialized for analyzing the remainder of the sentence. A better scheme is necessary for more complex sentences.

Another problem occurs because of the recursive nature of the grammar. Since each S-level network looks for conjoined phrases the conjoined constituents become nested inside one another. This seems unreasonable since the constituents joined by a coordinating conjunction are all equivalent.
Subordinating conjunctions ("when", "where", "because", etc.) should be fairly simple to add to the grammar. These conjunctions are used only to subordinate one complete sentence to another and with the exception of the verb tenses the structure of one sentence does not affect the other.

Examples:

When she comes home they will leave.
They will leave when she comes home.

2.A.6 Noun and Preposition Phrase Utterances

If the grammar cannot identify the input as a complete sentence it checks to see whether it is either a noun phrase or a preposition phrase. If this is the case the input is parsed as a noun or preposition phrase utterance. These constructions could be useful as answers to questions in a conversational situation.

Examples:

Where is the block?
In the box.
What is next to it?
A cube.
The noun phrase or NP network identifies the components of a noun phrase, checks their semantic agreement, and builds the associated parse tree. This network is entered from the S-level network, the PP-level network, or recursively from itself.

2.8.1 Basic parsing strategy

The parsing of a simple noun phrase embodies the basic noun phrase parsing strategy. The more complicated noun phrases are parsed using minor variations of this strategy. Consider the phrase:

The young boy in the park

The determiner "the" is identified on the CAT DET arc in state NP. The parser then jumps through the states which identify optional constituents to NP-PART where the adjective phrase or ADJP network identifies the adjective "young". The parser proceeds to state NP-ADJ where the head noun "boy" is identified. The parser then jumps to NP-N to look for post-nominal modifiers. The preposition phrase "in the garden" is found and the parser jumps to NP-NP. Here the number agreement between the determiner "the" and the noun "boy" is checked. The semantic agreement between the adjective "young" and the noun "boy" is also checked before the structure in Figure 10 is returned. The possible noun phrase constituents are discussed
in more detail below.

PHRASE: THE YOUNG BOY IN THE PARK

PARSE:
NP

  DET THE
  ADJ YOUNG
  N BOY
  NUMBER SG
  NU SG

PP

  PREP IN
  NP

    DET THE
    N PARK
    NUMBER SG
    NU SG

Figure 10 - Parse of a Noun Phrase

2.8.2 Simple noun phrases

The noun phrase may consist of a single pronoun or proper noun.

Examples:

  He
  John
2.8.3 The pre-nominal modifiers

Determiners

This constituent is recognized at state NP. Determiners include words such as "the", "a", "this", "that", etc. and possessive pronouns ("his", "their", etc.). Another group of words including "some", "every", "all", "no", "any", and "both" are considered to be determiners. This classification was made because these words cannot be preceded by another determiner and may in some cases be followed by an ordinal (i.e. a word like "first", "last", etc.):

Examples:

*The some boys
Every last man

The determiner structure analysis of Stockwell et al.[8] was used in making this classification.

Possessive proper nouns are also identified in this state.

Example:

John's biggest red book

These act as determiners in that they precede the other constituents in the noun phrase.
Ordinals

If an ordinal is present it must follow the determiner and precede the quantifiers and adjectives in the noun phrase. Ordinals indicate the position of the noun in a sequence of objects. There is an infinite sequence of number ordinals ("first", "second", "third",...) and a few others such as "next" and "last". Winograd[10] notes that ordinals may be recognized since they are the only words that may occur between a determiner and a number.
Example:

The first five books

Superlative adjectives may also act as ordinals.
Example:

The biggest five boxes

Ordinals are identified at state NP-DET. The parser then proceeds to state NP-ORD to look for quantifiers.

Quantifiers

A quantifier may follow an ordinal. These constructions may be quite complex. The NP-ORD state pushes to the QUANTP network to identify quantifier constructions. The simplest quantifiers are numbers ("one", "two", "three",...). Other words such as "several", "many", "few", etc. which are entered as quantifiers in the dictionary are also identified.
Examples:

Two young boys
The first few days

More complex constructions are also identified.
Example:

At least a dozen books
Less than two books
At least a few more than four books
Several more books

Adjectives

State NP-PART identifies adjectives and other pre-nominal modifiers which occur after quantifiers and before the noun.
Example:

The big red beautiful flower

Present and past participles of verbs may be used as adjectives.
Examples:

The running boy
The painted table

Superlative and comparative forms of adjectives are recognized.
Examples:

The most beautiful girl
The very oldest man

Nouns may also be used to modify the head noun in the phrase. Winograd[10] calls these nouns "classifiers".
Example:

The Christmas tree ornament counter

Possessive nouns may be used as modifiers. Ordinals may occur after a possessive noun since the possessive noun marks the head of a possessive noun phrase which acts as a sort of determiner structure for the next noun in the phrase.

Example:

His oldest sister's first husband's house

The semantic agreement routines ought to check the agreement among the constituents of these possessive phrases, but at present this is not done.

2.8.4 Post-nominal modifiers

These noun phrase constituents occur after the head noun in the phrase.

Preposition phrases

Preposition phrases may modify the noun.

Example:

The boy in the room

A problem occurs here because the preposition phrase does not always modify the noun preceding it. The phrase may just happen to be positioned directly after the noun although it modifies
some other element in the sentence.

Examples:

(1) Give the book to the man.
(2) Put the book in the box on the table.
(3) Put the book into the box.

In sentences such as (2) it is necessary to have specific knowledge about the situation to determine whether "on the table" modifies "box" or "book". In sentences such as (1) and (3) the choice of preposition indicates that the phrases do not modify the preceding noun. Prepositions such as "into", "onto", and "to" are flagged as MOTION prepositions in the dictionary and if one of these is encountered the preposition phrase is not parsed as a post-nominal modifier, but appears at the sentence level.

Relative clauses

Relative clauses may be used to modify the noun. These are recognized at state NP-N by their introductory preposition or relative pronoun. The noun phrase head is SENDRed to the S-REL network. This noun phrase may be the missing subject, object or object of a preposition in the relative clause. If the relative pronoun is "whose" the missing noun phrase is possessive. Relative clauses may be passive.

Examples:

The man that we saw
The man that saw the dog
The man by whom the book was given
The man whose dog we saw

The S-REL network processes the remainder of the missing phrase and then merges back into the S-level network.

Reduced relative clauses are also recognized. The S-REL-REDUCED network is searched last since these clauses resemble so many other types of constructions.

Examples:

The man given the book
The man we saw
The man not running

2.B.5 Incomplete noun phrases

If the head noun is missing then the noun phrase is said to be incomplete. The dummy noun "ones" becomes the head noun of the phrase. Quantifiers or ordinals preceded by determiners may comprise a noun phrase.

Examples:

Give me at least three.
Give me the first (few).

These constructions can optionally be followed by a partitive construction.

Examples:

He found two of the three books.
Give me the biggest of the books.

There are some determiners ("all", "any", "some", "both", etc.) flagged QUANT in the dictionary, which are used as quantifiers
in these constructions.

Examples:

I need some.
Give me any of the books.

The words "all" and "both" may omit the "of" in the partitive.

Example:

All the boys came.

A possessive pronoun or proper noun may comprise a noun phrase, but may not be followed by a partitive construction.

Examples:

Give me hers.
Give me John's.

A phrase ending with a possessive noun may be used instead of a complete noun phrase.

Example:

Give me his youngest sister's older brother's.

2.8.6 "Something" constructions

The grammar also recognizes noun phrases containing constructions with "something" or "anything". These pronouns are unusual because they are followed by adjectives.

Examples:

Bulls will charge at anything red.
Give me something big which is on the table.

These words are classified as GENPRO (general pronouns) in the dictionary.
2.8.7 Conjoined noun phrases

Any noun phrase may consist of a series of conjoined noun phrases.

Examples:

John and Mary
A boy and his dog
He or she or they

If a conjunction is recognized at state NP-MAINPHRASE the parser proceeds to NP-CONJ where a push to the NP network attempts to find another noun phrase. Since the PUSH NP forms a recursive call to the network, any subsequent noun phrases in the series will be nested inside the returned structure. The noun phrases are all equivalent so it would seem more reasonable to have the conjoined phrases all at the same level in the parse.

2.C Extensions

Some of the constructions which one might wish to add to the grammar are discussed in the following sections.

2.C.1 Existential "There"

Sentences which assert the existence of their subjects are often transformed into sentences introduced by the word "there".
Examples:

There are five books on the table. How many men are there?

This construction would be easy to add to the grammar. Notice that the transformation which produces the existential "there" only applies when the subject is indefinite. Thus the following sentence is a locational not existential use of "there":

There are the books.

2.C.2 Subordinate Clauses

Subordinate clause constructions should be simple to add to the grammar. Subordinate conjunctions behave in a much simpler manner than coordinate conjunctions because they are used only to subordinate one complete sentence to another.

Examples:

(1) When they arrive we shall have dinner.
(2) If the block is on the table, put it on the floor.
(3) He enjoyed the book although it was very long.

A complete analysis of sentences such as (2) and (3) involves finding the antecedent for the pronoun "it". This task involves the use of semantic information.
2.C.3 Comparatives

Constructions of the form "as many as" or "more than" followed by a sentence would be useful, particularly in a question answering system environment. These should not be difficult to identify.

Examples:

I want as many as John has.
I want more books than John gave Jim.

2.C.4 Declarative Questions

In conversation, questions are often expressed in declarative form.

Examples:

(1) You gave the book to whom?
(2) He hit what?
(3) He kicked which chair?

At present the grammar can identify question determiner (QDET) declaratives such as sentence (3). Question pronoun (QPRO) declaratives could be added quite easily.
2.C.5 Coordinate Conjunctions

Extension of the treatment of coordinate conjunctions would be fairly difficult. The scope of these conjunctions must often be determined using semantic tests. Please refer to Section 2.A.5 of this chapter for further discussion of the difficulties involved in this type of analysis.

2.C.6 Relative Clause Disambiguation

The area of relative clause disambiguation has not been considered in the grammar. Semantic information can be used to determine which noun a relative clause modifies. Consider the noun phrase:

The man with the green hat buying a newspaper

Obviously the relative clause "buying a newspaper" modifies "man" not "hat". The investigation of this problem would be interesting.

2.C.7 Punctuation

The grammar does not handle any punctuation. The use of punctuation such as question marks and commas for helping in the disambiguation of sentences should prove valuable.
3. The Semantic and Grammatical Tests

3.A Sentence Network

3.A.1 The Semantic Tests

Semantic agreement tests are performed on the major components of the sentence as they are identified. If a component is semantically incompatible the parser backs up and attempts to re-parse the offending component. Further checks are made when the sentence has been completely parsed to ensure that all necessary parts of the sentence have been identified. For more details please consult the SEMANTICS routines in Appendix 7.

As the parse progresses, information about the subject, direct and indirect objects is saved in the semantic registers S-SUBJ, S-DO, and S-INDO. Each time a noun phrase is parsed the NP network LIFTRs a list consisting of the head noun of the phrase and its features to the next higher level network. This list is placed in the appropriate semantic register to be used in making semantic agreement tests. The S-VERB register contains the main verb and its features. Consider the sentence:

The boy kicked the dog.

When this sentence is parsed the register S-SUBJ contains the
The register S-VERB contains:

(V KICK ((TNS PAST) (PNCODE ANY)))

The register S-DO contains the list:

(N DOG ((NUMBER SG)))

The register S-INDO contains NIL since no indirect object is present. Only the head noun is saved in order to avoid having to analyse the structure returned from the noun phrase network. In some situations this is not the most reasonable solution. Consider the sentence:

Some of the boys ran away.

In the grammatical analysis "ones" is the dummy head noun of this incomplete noun phrase. The structure below is placed in the register S-SUBJ.

(DUMMY ONES NIL)

From a semantic point of view a structure such as:

(N BOYS ((NUMBER PL)))

would provide more information since the real subject is an unspecified subset of "the boys". More work could be done in this area.

In the state S-V-SEMANTICS when the subject and main verb of the sentence have been identified a semantic test checks that the subject and verb are semantically compatible. The semantic markers of the subject head noun and those describing the subject of the verb (i.e. the markers found under the flag
SUBJ-TYPE on the verb) must have a non-null intersection. If the semantic markers on any component are missing then the component is assumed to be compatible in all situations. For example, the verbs "use", "is", and "have" do not have any semantic markers under the flag SUBJ-TYPE since these verbs may take a wide variety of subject types. At present only nouns, pronouns, verbs and adjectives have semantic markers; other categories of words are ignored in the semantic tests.

A similar semantic check occurs when the direct object has been identified. If the main verb may take an indirect object this check is delayed until either the indirect object is found or the complete sentence has been parsed. Consider the sentences:

(1) The boy gave the girl a book.
(2) The boy gave a book to the girl.

In sentence (1) the indirect object occurs in the direct object position because of the indirect object inversion transformation. In sentence (2) the noun phrase occurring in the same position is really the indirect object. The deep structure of sentence (1) is re-arranged when the second noun phrase is identified. Thus the direct object semantics should not be checked in this case until the parser is certain that the noun phrase really is the direct object. In sentence (1) the grammar checks the direct and indirect object semantics when both have been identified and the deep structure re-arranged. The parser does not know that the prepositional phrase in
sentence (2) contains the indirect object so the direct object semantics are checked when the entire sentence has been parsed.

The semantic agreement between the subject head noun and the predicate adjective is checked before the remainder of the sentence is parsed. Although participles may usually act as adjectives the predicate adjective must be classified as an adjective (ADJ) in the dictionary.

If the subject or one of the object noun phrases is a compound noun phrase then a compound head noun structure is LIFTRed from the NP network. The semantic tests are then performed on each head noun in the compound noun phrase.

It should be noted that some question and negative sentences may cause a problem under this scheme. For example, the verb "talk" is flagged in the dictionary as having an ANIMATE subject. Thus the sentences below will fail to parse since their subject "book" is inanimate.

Does the book talk?
The book does not talk.

In this type of sentence a question may be answered or a statement verified just on the basis of the meanings of the words. This type of utterance does not appear to be of much value in a question answering system.

When the entire sentence has been parsed, a few more semantic tests are made to ensure that all the necessary components have been located. For example, transitive verbs
must have a direct object and copula verbs must have some type of post-verbal modifier present.

If a Lisp predicate is present on the verb under the flag V-SEMANTICS it is evaluated when the entire sentence has been parsed. This predicate can perform complex syntactic or semantic checks on the parsed constituents of the sentence. At this time the components of the deep structure are all contained in registers, so these as well as the semantic registers (S-SUBJ, S-VERB, S-DO, and S-INDO) are available for the use of this predicate. This arbitrary test allows the verb "put", for example, to insist that a locational preposition phrase or adverb be found before the parse is completed.

Examples:

(1) *He put the block.
(2) He put the block on the stack.
(3) He put the block there.

As well as eliminating incorrect sentences like sentence (1) this test ensures that the prepositional phrase "on the stack" in sentence (2) is not parsed as a post-nominal modifier of the noun "block".

This predicate can also consult the "real world" to determine whether the parsed structure makes sense in the situation under consideration. In sentence (2) above, it might be desirable to check that blocks can be put on stacks. One might choose to reject the sentence:

Put the block into the cube.
if cubes in this situation are known to be solid. Care must be taken in making this type of decision since a sentence like:

You cannot put the block into the cube.

has the same sort of structure but is semantically quite reasonable. This type of "real world" test is not used in the grammar since no specific situation was considered.

3.A.2 Subject-Verb Number Agreement

The grammatical subject must agree in person and number with the tensed verb in the sentence. The subject, if it is a pronoun, must be in the nominative case (i.e. be flagged SUBJ in the dictionary). If the subject is a compound noun phrase, currently it is classified as either singular or plural (SG-PL). Usually the noun phrase is plural although it may be singular if the conjunction involved is "or".

Examples:

He or she is talking.
He and she are talking.
He or she or they are talking.
3.8 Noun Phrase Network

3.8.1 Semantic Tests

Semantic agreement tests are performed on the adjectives and head noun of the phrase before the noun phrase structure is returned. The register SEM-ADJS contains a list of all the adjectives and register SEM-NOU>M the head noun of the phrase. If the adjectives and noun are incompatible the parser backs up and attempts to re-parse the phrase. Determiners, ordinals, and quantifiers seem to agree with all types of nouns so they are excluded from the tests. Nouns which modify other nouns are excluded since their relationship with the head noun is very complex. This relationship appears to depend on much more subtle properties of the words involved than just the broad general classifications of ANIMATE, PHYSOBJ, etc.

Examples:

Bakery man
Post man
*Man bakery
Man servant
Man hole
*Hole man

If the present participle of a verb is used to modify the noun then the SUBJ-TYPE of that verb must agree semantically with the noun.
Examples:

The running boy
*The running table

When the noun is modified by the past participle of a verb the DO-TYPE of the verb must agree semantically with the noun.
Examples:

The painted table
*The painted party

At present the parser only checks the semantic agreement between the head noun and the adjectives modifying it. The semantic agreement of possessive noun phrases which may act as a sort of determiner structure for the main noun is not checked. Consider the phrase:

The old man's new car
The parser checks that "new" and "car" are compatible, but the components of the phrase "the old man's" are not checked for semantic compatibility. This check ought to be added to the grammar.

If a Lisp predicate is present on the head noun under the flag N-SEANCE, it is evaluated after the other semantic tests are performed but before the noun phrase structure is returned. This predicate may perform syntactic or semantic checks on the parsed constituents of the noun phrase which are contained in the registers. This test could, for example, prevent certain adjectives from modifying the head noun or check that nouns modifying the head noun were appropriate. The predicate can
also look at the "real world" to determine, for example, whether a preposition phrase occurring after the head noun modifies the noun or has some other function in the sentence.

3.B.2 Determiner-Noun Number Agreement

The number agreement of the determiner and noun is checked before the noun phrase structure is returned. A list consisting of the determiner and its feature list is placed in register SEM-DET when the determiner is identified. At present the number agreement only considers determiners classified as DET in the dictionary and nouns classified as N. Thus proper nouns (NPR), possessive pronouns (POSSPRO), etc. are not checked. A singular noun unless it occurs after a conjunction must be accompanied by a determiner. Mass and plural nouns may omit the determiner.

Examples:

A girl
*Girl
The girl and boy
Water
Boys

If a quantifier has been found then the determiner sometimes will not agree with the head noun.

Example:

A few books
IV. COMMENTS

1. Use of ATN Grammars

The ATN model of grammar is generally speaking quite an attractive and useful model for natural language. This section summarizes the advantages and drawbacks to this type of approach.

1. A Advantages

1. A. 1 Ease of Writing

The language in which the grammar is specified is designed to be convenient and natural for the grammar writer, rather than the computer. The grammar is completely separate from the parser so the grammar writer does not need to be aware of exactly how the parser works providing she understands the order in which the network is searched. The parser provides a basic selection of actions and conditions which are suited to the analysis of natural language.
1.4.2 Flexibility

The deep structure of the sentence is built in a very flexible manner. While the sentence is being analysed the pieces of the parse are stored in registers until the final state is reached. This enables a tentative decision to be made about the function of a constituent. Later if a subsequent input indicates that the decision is incorrect the registers may be re-arranged without backing up.

The deep structures produced by the grammar may be easily altered. The POP arc on the final state of the grammar specifies the deep structure that is assigned to the phrase recognized in the network. Therefore the deep structure can be altered just by changing the form specified on the POP arc.

It is usually fairly straightforward to add more conditions to the grammar to improve the quality of the parses produced. If the grammar writer is careful the arcs and states necessary to recognize new constructions can be added with a minimum of difficulty once the main framework is present.
1.4.3 Capturing Regularities

The phrase "capturing the regularities of natural language" is explained by Woods in the following manner:

"One of the linguistic goals of a grammar for a natural language is that the grammar capture the regularities of the language. That is, if there is a process that operates in a number of environments, the grammar should embody that process in a single mechanism or rule and not in a number of independent copies of the same process for each of the different contexts in which it occurs...."

An example of this principle is the representation of a preposition phrase as a sentence constituent. A noun phrase often follows a preposition in English sentences and the two form a sort of structural unit. If the grammar did not treat preposition phrases as sentence constituents it would be missing a regularity of the language.

The ATN grammar appears to be successful in capturing these regularities. Firstly the use of registers to contain flags allows two or more almost identical sub-networks to be merged into a single one, setting flags to direct the parser in places where the original networks differed. These flags may then be tested by the conditions and actions on the arcs of the sub-network to guide the parser through the proper path.

-----------------------------

Secondly the register structure and the use of the HOLD action and VIR arc make the processing of transformations natural in the grammar. The theory of transformational grammar is again an attempt to capture the regularities of natural language. Transformational grammar tries to explain the connection between passive sentences and their active counterparts, interrogative sentences and their declarative counterparts, negative sentences and their affirmative forms, etc. Without the power of a transformational grammar many more states would be necessary to identify these forms. A context free grammar, for example, cannot recognize these similarities.

Because ATN grammars capture the regularities of natural language they maintain a degree of what Woods[11] calls "perspicuity". That is, by looking at the ATN diagram it is reasonably clear what types of constructions are recognized by the grammar. This is largely due to the fact that the ATN grammars capture the regularities of natural language and thus are reasonably compact in representation.

1.B Criticisms

This section discusses some of the problems involved with the use of an ATN grammar for the analysis of natural language. It should be noted that many of the criticisms of the parsing system refer specifically to the simple version of the ATN
parser which was implemented at the University of British Columbia. Woods' parser[13] is much more sophisticated and takes into account some of these objections.

1.B.1 Ordered Backup

Perhaps the greatest problem with the ATN parsing system is that search through the network is conducted in a strictly depth-first manner. The parser considers the arcs leaving a state in the order in which they occur. The first possible arc is followed. If this choice turns out to be unsuccessful (i.e. the path becomes blocked in the destination state) all the other arcs at this state are tried until either a path is found or all arcs are rejected, causing the parser to back up to the previous state. This process continues until a path is found through the network or it is determined that no path exists.

In some circumstances it would be convenient to have a more directed search through the network. Consider a situation where a critical choice must be made at state A. At state B it is discovered that the wrong choice was made at A. All the alternative paths between states A and B are searched before the other choice at A is considered. This situation is inconvenient because first of all much time may be wasted considering unlikely paths. Secondly unless a great deal of thought has been given to the ordering of the arcs and the conditions on
them it is probable that an "unlikely" parse will be generated.

The ATN grammar described in Chapter III has several situations where ordered backup would be desirable. One such situation involves the processing of questions beginning with a question determiner. The noun phrase containing the question determiner may be either the subject or object of the sentence. At state S-WH-NP the question determiner phrase has been identified and a decision must be made as to the function of the phrase in the sentence. If the phrase is the object then subject-verb inversion occurs and the next word must be an auxiliary verb. The parser goes to state S-YESNO to deal with the inversion. Otherwise the phrase is the subject and the next word must be a verb. Parsing continues at state S-NP to locate the verb. Since auxiliary verbs may be used as the main verb or as part of a more complex verb construction it is impossible to ensure that the correct arc is followed. Consider the following questions:

(1) Which man has he seen?
(2) Which man comes?
(3) Which man has come?
(4) Which man has the book?

In sentence (1) subject-verb inversion does occur and the auxiliary verb condition ensures that the appropriate arc is taken. In sentences (2), (3), and (4) the question determiner phrase is the subject. The correct arc is followed for sentence (2) since "comes" is not an auxiliary verb. However the wrong arc is taken in both sentences (3) and (4) since "has" is an
auxiliary.

In the above situation the wrong arc will be followed for some questions regardless of how the arcs are ordered. If the chosen arc turns out to be unsuccessful it would seem reasonable to fail directly back to the point where it is known that a wrong decision could easily have been made. The other arc should then be followed and only if this path too is blocked, should the other alternatives be considered. It would be desirable, for instance, to assign a very low priority to search through the reduced relative clause network. This network is often responsible for the construction of strange parses in a desperate attempt to explain the presence of "extra" verbs.

Without ordered backup, complicated conditions must be added to the grammar in an attempt to prevent sentences which have made an incorrect choice from being parsed into an obscure construction on backup.

1.9.2 Parsed Constituents

It seems somewhat inefficient that there is no mechanism for setting aside parsed constituents (e.g. noun and preposition phrases) which may be analysed many times in the process of finding a path through the network. Whereas the relationship among the various constituents in the sentence may vary as different paths through the network are searched, the
constituents themselves tend to remain as fairly stable units in the sentence. A more efficient approach to the problem might be to have a first pass which would identify the major constituents of the sentence which are usually identified by lower level networks. A second pass corresponding to the top (or S) level network could attempt to arrange these constituents into a reasonable structure. If this was unsuccessful, alternative parsings would be tried for the major constituents.

Woods has experimented with saving these constituents, but he feels that:

"... the savings in parse time for such an approach does not always justify the storage required to store all of the partial substring analyses...."\(^1\)

1.8.3 Recursion

The recursive nature of the ATN grammar is very useful for eliminating duplication of states by taking advantage of the regularities of natural language. However recursion does lead to problems when, for example, coordinate clauses are being parsed. Consider the sentence:

He sang and he danced and he drank.

The parse structure returned by the grammar for this sentence is

of the form:

(S (AND (S 
    (S (AND (S 
        (S)))))))

There is no semantic or syntactic reason for the nesting of the structure but this is the type of parse that is naturally returned by a recursive grammar where each sentence checks for another conjoined sentence. The structure below would seem more reasonable since all the sentences are now at the same level:

(S (AND (S 
    (S 
        (S 
            (S 
                (S)))))))

This type of structure could, of course, be constructed but it is not natural to the system.

The other problem encountered with the recursive nature of the grammar is that preposition phrases tend to become nested inside other noun or preposition phrases. It is definitely convenient to consider that a noun phrase may be followed by a preposition phrase. This is not always the best assumption since preposition phrases often end up nested inside noun phrases and must depend on specific semantic tests to dislodge them.

Examples:

(1) Put the block on the table.
(2) Is the book in the room?
(3) The block on the table is red.

For example, tests insisting that "put" have a locational descriptor and that copula verbs must have a post-verbal modifier are necessary to produce the correct parse for
sentences (1) and (2). Sentence (3) is correctly parsed with the preposition phrase modifying the noun phrase.

A more satisfactory solution might be not to allow a preposition phrase to follow a noun. After each noun is identified the grammar would have to look for one or more prepositional phrases. Semantic tests could then be performed to determine what modifies what and the appropriate structures built. This solution however fails to appreciate one of the regularities of natural language; that is, a noun is often modified by a preposition phrase and the two form a structural unit.

1.8.4 Sentences Which Fail to Parse

Sentences may fail to parse for several reasons. The sentence may be ungrammatical or it may contain a word unknown to the system or used in an unfamiliar manner. If a word is unknown the system notifies the user and she is given an opportunity to stop or enter a synonym or dictionary entry. If the sentence fails to parse (i.e. a path through the network cannot be found) when all the words are known, the system has no way of knowing which path was closest to being correct. This seems especially unreasonable when one considers that the average person understands ungrammatical and other strangely constructed sentences quite well. A possible solution is to re-
parse the offending sentence relaxing the grammatical and semantic constraints. It is likely then that the sentence will have several possible parse structures and that the wrong one will be chosen since all the constraints that normally help determine a likely parse structure have been relaxed.

ATN grammars are not really suitable for dealing with ungrammatical or semantically "strange" constructions. Wilks[9] discusses a system of "preference semantics" where words have preferred contexts. If the preference cannot be satisfied the word will be placed in a less likely context. Wilks' system seems much more practical for dealing with this problem.

1.8.5 Operation of the Parser

The following comments refer specifically to the simple version of the ATN parser implemented at the University of British Columbia. A listing of this parser is in Appendix 9.

There are several features of the parser which are annoying but are not serious shortcomings. First of all, the FEATURES list is only defined on a CAT arc. This is annoying when the parser wishes to look ahead in the sentence checking that words have particular features. For example, it is often necessary to know whether the next word is a tensed verb. At present the problem is solved by setting the necessary local variables and calling the internal parser routines directly. This is not
satisfactory since the grammar is now dependent on the internal mechanism of the parser.

Secondly, the LIFTR action can be rather awkward to use if the grammar knows neither the state nor level from which it PUSHed to the current network. For instance, if the grammar wishes to set a register at the most recent S-level network when a question determiner is encountered in a noun phrase, the level of the most recent S network is unknown (both the sentence and the noun phrase network are recursive) and furthermore the state from which it PUSHed is unknown. The grammar must LIFTR the register to all possible states from which it could have PUSHed. A more reasonable solution would be to allow registers to be global to a certain level in the network. Then a LIFTR action could set a register at the S-level without having to know the applicable state.

Finally in some situations it would be convenient to fail a parse completely. If an impossible arrangement of words is encountered the parser, at present, can only block the path at that point. Much time is then wasted in further searching before the search is abandoned.
2. Suggestions for Prospective Grammar Writers

ATN grammars make it fairly simple to write moderately complex grammars. Debugging the grammars may be somewhat more difficult unless care is taken in writing them.

2. A Common Problems

When new constructions are added to the grammar they usually parse correctly. However there is a tendency for some of the other constructions, which resemble the new ones in perhaps very subtle ways, to parse incorrectly. Consider, for example, the grammar described in Chapter III. When yes-no questions were added to the grammar they worked correctly. The grammar however began to mistake some imperative sentences for questions. The imperative sentence:

Do the work quickly.

was parsed as the yes-no question whose declarative form is:

The work do quickly.

This led to modifications requiring that transitive verbs have objects and that yes-no questions have main verbs other than "do" unless an auxiliary is present.

This type of problem is very common so it is important to check that old constructions still work after new ones are added.
2.8 Ordering the Arcs

The ordering of the arcs in each state is very important. The first arcs should look for the most easily identifiable constructions, leaving the more complex and less identifiable constructions until last. This way the simple arcs such as WRD arcs are tried first. Less time is wasted since no complicated networks are searched unless there is good reason to believe they are applicable. That is, either all simpler constructions have failed or the introductory word(s) or preceding word(s) indicate this is a likely path.

In the grammar described in Chapter III, PUSH NP arcs tend to take a long time to show that no noun phrase is present since they may contain a great variety of optional constituents. This network also frequently produces incorrect parses if forced into the reduced relative clause network. Thus the search of this type of network should be delayed until the simple arcs that can be accepted or rejected on a single test have been tried.

The arcs in the noun phrase network in the grammar were reordered so that the reduced relative clause network is searched only as a last resort. Before this was done the question:

Is the man talking fast?

was parsed as though its declarative form were:

The man, who talks fast, is.

instead of:

The man is talking fast.
The problem occurred because participles (and past tenses with participle forms) which do not occur next to their auxiliaries for one reason or another may be mistaken for participles introducing reduced relative clauses. The auxiliary verb is then parsed as a main verb. It is difficult to devise a test which will recognize this mistake afterwards so it is better to prevent this mistake from occurring. For this reason the arcs in the noun phrase network were re-ordered so that the reduced relative clause network is searched after all else in the network has failed.

2.6 Conditions on the Arcs

It appears to be worthwhile putting very comprehensive conditions on the arcs of the grammar. The model works best if the right decision is made the first time. The grammar writer should attempt to eliminate backup if possible since it cannot be ordered. It is wise not to rely on the backup because predicting the result of backing through a complex network is very difficult. If changes are later made in the grammar the writer may not even remember that the arcs were ordered in a certain way for a special reason.

Any information which is known about the form of the next word or words in the construction should be incorporated into the tests on the arcs. For example, before a PUSH PP arc is
followed to attempt to identify a preposition phrase, the next word should be tested to see if it is a preposition. If it is not, the time involved in pushing to a lower level network is saved. The tests are more important when a path cannot be immediately rejected or accepted but must be searched further.

The conditions on the arcs can immediately eliminate ungrammatical and nonsensical constructions and may help prevent other totally different constructions from just happening to fit into the pattern on backup and producing nonsense.
BIBLIOGRAPHY


APPENDIX I - HOW TO RUN

Under the MTS operating system at the University of British Columbia the parsing system may be run by issuing the following command:

$SOURCE JEJ:PARSER

The file JEJ:PARSER contains the following instructions:

$R CS:LISP PAR=PDS=10,MAX=55
$CONTINUE WITH JEJ:PREPASS RETURN
$CONTINUE WITH CS:PARSE RETURN
($$PRINT$$)
(INIT)
$CONTINUE WITH *MSOURCE*

These commands load the Lisp interpreter, the parser and all the other Lisp routines necessary. The function call, (INIT), initializes the parsing system by reading in the dictionary, grammar and morphemic analysis tables.

A sentence may be parsed by calling the function PARSER with the sentence as an argument. For example, to parse the sentence:

He runs fast.

the following function call is issued:

(PARSER (HE RUNS FAST))

Punctuation should not be entered with the sentence.

The function PARSER assumes that parsing starts in state S of the grammar. The user may specify the state in which the parse is to start by using the function P instead. For example, to parse the noun phrase:
The big man
starting in state NP which is the start of the noun phrase
network call:

(P (THE BIG MAN) NP)
APPENDIX 2 - SAMPLE_PARSES

SENTENCE: JOHN RAN FAST

PREPASS TIME = 5 MS
(JOHN RAN FAST)

PARSE:
S
   MOOD DCL
   VOICE ACTIVE
   NP
      DET NIL
      NPR JOHN
      NU SG
   AUX
      TNS PAST
   VP
      V RUN
      ADV FAST

PARSE TIME = 116 MS
SENTENCE: IS HE A YOUNG BOY

PREPASS TIME = 4 MS
(IS HE A YOUNG BOY)

PARSE:
S
   MOOD YESNO
   VOICE ACTIVE
   NP
      DET NIL
      PRO HE
      NUMBER SG
      SUBJ
      PNCODE 3SG
      NU SG
   AUX
      TNS PRESENT
   VP
      V BE
      NP
         DET A
         ADV YOUNG
         N BOY
            NUMBER SG
            NU SG

PARSE TIME = 186 MS
SENTENCE: WHO TOOK THE BALL WHICH WAS IN THE WATER

PREPASS TIME = 8 MS
(WHO TOOK THE BALL WHICH WAS IN THE WATER)

PARSE:
S
   MOOD QPRO
   VOICE ACTIVE
   NP
      DET NIL
      QPRO WHO
      NU SG-PL
   AUX
      TNS PAST
   VP
      V TAKE
      NP
         DET THE
         N BALL
         NUMBER SG
         NU SG
   S
   MOOD REL
   VOICE ACTIVE
   NP
      DET WH
      N BALL
      NU SG
   AUX
      TNS PAST
   VP
      V BE
     PP
       PREP IN
      NP
         DET THE
         N WATER
         NUMBER MASS
         NU MASS

PARSE TIME = 350 MS
SENTENCE:  THE BOY IN THE PARK HAS THE BALL

PREPASS TIME = 6 MS
(THE BOY IN THE PARK HAS THE BALL)

PARSE:
S
   MOOD DCL
   VOICE ACTIVE
   NP
      DET THE
      N BOY
         NUMBER SG
      NU SG
   PP
      PREP IN
   NP
      DET THE
      N PARK
         NUMBER SG
      NU SG
   AUX
      TNS PRESENT
   VP
      V HAVE
   NP
      DET THE
      N BALL
         NUMBER SG
      NU SG

PARSE TIME = 241 MS
SENTENCE: BY WHOM WERE THE BOOKS GIVEN TO JOHN AND HIS BROTHER

PREPASS TIME = 12 MS
(BY WHOM WERE THE BOOKS GIVEN TO JOHN AND HIS BROTHER)

PARSE:
S
   MOOD QPRO
   VOICE PASSIVE
   NP
      DET NIL
      QPRO WHO
      NU SG-PL
   AUX
      TNS PAST
   VP
      V GIVE
      NP
         DET THE
         N BOOK
         NUMBER PL
         NU PL
      PP
         PREP TO
         NP
            AND
            NP
               DET NIL
               NPR JOHN
               NU SG
               NP
                  DET
                  POSS
                  PRO HE
                  POSSPRO
                  N BROTHER
                  NUMBER SG
                  NU SG

PARSE TIME = 344 MS
SENTENCE: WHICH OF THE MEN SAW MORE THAN TWO DOGS

PREPASS TIME = 10 MS
(WHICH OF THE MEN SAW MORE THAN TWO DOGS)

PARSE:
S
   MOOD QPRO
   VOICE ACTIVE
   NP
      DET NIL
      QPRO WHICH
      NU SG-PL
      PP
         PREP OF
      NP
         DET THE
         N MAN
         NUMBER PL
         NU PL
   AUX
      TNS PAST
   VP
      V SEE
      NP
         DET NIL
         QUANTP
         COMP
         ADV MORE
         QUANT
         INTEGER TWO
         N DOG
         NUMBER PL
         NU PL

PARSE TIME = 480 MS
SENTENCE: PLEASE PUT AWAY ANYTHING BIG WHICH IS ON TOP OF THE TABLE

PREPASS TIME = 8 MS
(PLEASE PUT AWAY ANYTHING BIG WHICH IS ON-TOP-OF THE TABLE)

PARSE:

S
  MOOD IMP
  VOICE ACTIVE
  NP
    DET NIL
    PRO YOU
    NU SG-PL
  AUX
    TNS UNTENSED
  VP
    V PUT-AWAY
    NP
      DET NIL
      ADJ BIG
      GENPRO ANYTHING
      NU SG
    S
      MOOD REL
      VOICE ACTIVE
      NP
        DET WH
        GENPRO ANYTHING
        NU SG
      AUX
        TNS PRESENT
      VP
        V BE
        PP
          PP ON-TOP-OF
          NP
            DET THE
            N TABLE
            NUMBER SG
            NU SG
  ADV PLEASE

PARSE TIME = 566 MS
SENTENCE: THE SMALL GIRL WHOSE SISTER YOU SAW MAY BE YOUNGER THAN MY BROTHER

PREPASS TIME = 18 MS

( THE SMALL GIRL WHOSE SISTER YOU SAW MAY BE YOUNGER THAN MY BROTHER)

PARSE:
S
MOOD DCL
VOICE ACTIVE
NP
DET THE
ADJ SMALL
N GIRL
NUMBER SG
NU SG
S
MOOD REL
VOICE ACTIVE
NP
DET NIL
PRO YOU
SUBJ
OBJ
NUMBER SG-PL
PNCODE 2SGPL
NU SG-PL
AUX
TNS PAST
VP
V SEE
NP
DET WH
POSS
N GIRL
N SISTER
NUMBER SG
NU SG
AUX
TNS PRESENT
MODAL MAY
VP
V
ADJ YOUNG
COMPARATIVE
NP
DET
POSS
PRO I
POSSPRO
N BROTHER
SENTENCE: WHY IS JOHN HOPING THAT MARY WILL FLY TO PARIS

PREPASS TIME = 13 MS

(WHY IS JOHN HOPING THAT MARY WILL FLY TO PARIS)

PARSE:

S
   MOOD QADV
   VOICE ACTIVE
   NP
      DET NIL
      NPR JOHN
      NU SG
   AUX
      TNS PRESENT PROGRESSIVE
   VP
      V HOPE
      COMPL
         CTYPE THAT
   S
      MOOD DCL
      VOICE ACTIVE
      NP
         DET NIL
         NPR MARY
         NU SG
      AUX
         TNS FUTURE
      VP
         V FLY
      PP
         PREP TO
      NP
         DET NIL
         NPR PARIS
         NU SG
   QADV WHY

PARSE TIME = 299 MS
SENTENCE: WHAT WAS PICKED UP BY THE MAN AND PUT DOWN ON THE TABLE

PREPASS TIME = 18 MS
(WHAT WAS PICKED UP BY THE MAN AND PUT DOWN ON THE TABLE)

PARSE:

S
    AND
    S
      MOOD QPRO
      VOICE PASSIVE
      NP
        DET THE
        N MAN
        NUMBER SG
        NU SG
      AUX
        TNS PAST
      VP
        V PICK-UP
        NP
          DET NIL
          QPRO WHAT
          NU SG-PL
    S
      MOOD QPRO
      VOICE PASSIVE
      NP
        DET THE
        N MAN
        NUMBER SG
        NU SG
      AUX
        TNS PAST
      VP
        V PUT-DOWN
        NP
          DET NIL
          QPRO WHAT
          NU SG-PL
        PP
          PREP ON
          NP
            DET THE
            N TABLE
            NUMBER SG
            NU SG

PARSE TIME = 583 MS
SENTENCE: HE PUT THE PYRAMID ON THE TABLE INTO THE BOX

PREPASS TIME = 8 MS
(HE PUT THE PYRAMID ON THE TABLE INTO THE BOX)

PARSE:
S
  MOOD DCL
  VOICE ACTIVE
  NP
    DET NIL
    PRO HE
      NUMBER SG
    SUBJ
    PNCOED 3SG
    NU SG
  AUX
    TNS PAST
  VP
    V PUT
    NP
      DET THE
      W PYRAMID
        NUMBER SG
      NU SG
    PP
      PREP ON
    NP
      DET THE
      W TABLE
        NUMBER SG
      NU SG
    PP
      PREP INTO
    NP
      DET THE
      W BOX
        NUMBER SG
      NU SG

PARSE TIME = 310 MS
Sentence: Some of the dogs were seen by the stream in the park.

Prepass time = 8 ms
(Some of the dogs were seen by the stream in the park)

Parse:

S
    Mood DCL
    Voice Passive
    NP
        Det nil
        Pro something
        Nu sg-pl
    Aux
        Tns Past
    VP
        V see
        NP
            Det some
            N ones
            Nu sg-pl
        PP
            Prep of
            NP
            Det the
            N dog
            Nu number pl
            Nu pl
        PP
            Prep by
            NP
                Det the
                N stream
                Nu number sg
                Nu sg
            PP
                Prep in
                NP
                    Det the
                    N park
                    Nu number sg
                    Nu sg

Parse time = 386 ms
SENTENCE: GIVE ME AT LEAST A FEW MORE THAN A DOZEN BOOKS

PREPASS TIME = 8 MS
(GIVE ME AT LEAST A FEW MORE THAN A DOZEN BOOKS)

PARSE:
S
  MOOD IMP
  VOICE ACTIVE
  NP
    DET NIL
    PRO YOU
    NU SG-PL
  AUX
    TNS UNTENSED
  VP
    V GIVE
      NP
        DET NIL
        QUAN TP
        SUPER
        ADV LEAST
        DET A
        QUAN TP
        QUANT FEW
        COMP
        ADV MORE
        DET A
        QUANT DOZEN
      N BOOK
        NUMBER PL
        NU PL
      PP
        PREP TO
        NP
          DET NIL
          PRO I
          OBJ
          NUMBER SG
          PN CODE 1SG
          NU SG

PARSE TIME = 301 MS
SENTENCE:Didn't you see John's

PREPASS TIME = 7 MS
(DID NOT YOU SEE JOHN'S)

PARSE:
S
  MOOD NEGATIVE YESNO
  VOICE ACTIVE
NP
  DET NIL
  PRO YOU
  SUBJ
  OBJ
  NUMBER SG-PL
  PNCODE 2SGPL
  NU SG-PL
AUX
  TNS PAST
VP
  V SEE
NP
  DET NIL
  POSS
  NPR JOHN
  N ONES
  NU SG-PL

PARSE TIME = 167 MS
SENTENCE: ALL THOSE WOMEN COULD HAVE PICKED THE BOOKS AND PENCILS UP

PREPASS TIME = 12 MS
(ALL THOSE WOMEN COULD HAVE PICKED THE BOOKS AND PENCILS UP)

PARSE:

S
   MOOD DCL
   VOICE ACTIVE
   NP
      DET ALL
      N ONES
      NU SG-PL
   PP
      PREP OF
         NP
            DET THAT
            N WOMAN
            NUMBER PL
            NU PL
   AUX
      TNS PRESENT PERFECT
      MODAL COULD
   VP
      V PICK-UP
      NP
         AND
            NP
               DET THE
               N BOOK
               NUMBER PL
               NU PL
            NP
               DET NIL
               N PENCIL
               NUMBER PL
               NU PL

PARSE TIME = 330 MS
SENTENCE: LIST ALL THE FUNCTIONS THAT CALL THE FUNCTIONS THAT CALL THE FUNCTIONS THAT THIS FUNCTION CALLS

PREPASS TIME = 20 MS
(LIST ALL THE FUNCTIONS THAT CALL THE FUNCTIONS THAT CALL THE FUNCTIONS THAT THIS FUNCTION CALLS)

PARSE:
S
   MOOD IMP
   VOICE ACTIVE
   NP
     DET NIL
     PRO YOU
     NU SG-PL
   AUX
     TNS UNTENSED
   VP
     V LIST
     NP
       DET ALL
       N ONES
       NU SG-PL
     PP
       PREP OF
       NP
         DET THE
         N FUNCTION
         NUMBER PL
         NU PL
   S
      MOOD REL
      VOICE ACTIVE
      NP
        DET WH
        N FUNCTION
        NU PL
      AUX
        TNS PRESENT
      VP
        V CALL
        NP
          DET THE
          N FUNCTION
          NUMBER PL
          NU PL
       S
          MOOD REL
          VOICE ACTIVE
          NP
            DET WH
            N FUNCTION
PARSE TIME = 755 MS
Sentence: The name of the girl kicking the dog is Mary

Prepass time = 13 ms
(The name of the girl kicking the dog is Mary)

Parse:

S
  MOOD DCL
  VOICE ACTIVE
  NP
    DET THE
    N NAME
      NUMBER SG
    NU SG
  PP
    PREP OF
    NP
      DET THE
      N GIRL
        NUMBER SG
      NU SG
  S
    MOOD REL
    VOICE ACTIVE
    NP
      DET WH
      N GIRL
      NU SG
    AUX
      TNS PRESENT PROGRESSIVE
    VP
      V KICK
      NP
        DET THE
        N DOG
          NUMBER SG
        NU SG
    AUX
      TNS PRESENT
    VP
      V BE
      NP
        DET NIL
        NPR MARY
        NU SG

Parse time = 392 ms
SENTENCE: THE FIRST FOUR SURPRISINGLY VERY OLD BOOKS WITH THEIR COVERS PRINTED IN RED INK

PREPASS TIME = 20 MS

PARSE:

S

MOOD NPU
NP
DET THE
ORD FIRST
QUANT
INTEGER FOUR
ADJP
ADV SUPRISINGLY
ADV VERY
ADJ OLD
N BOOK
NUMBER PL
NU PL
PP
PREP WITH
NP
DET
POSS
PRO THEY
POSSPRO
N COVER
NUMBER PL
NU PL
S

MOOD REL
VOICE PASSIVE
NP
DET NIL
PRO SOMETHING
NU SG-PL
AUX
TNS PRESENT
VP
V PRINT
NP
DET WH
N COVER
NU PL
PP
PREP IN
NP
DET NIL
ADJ RED
PARSE TIME = 5268 MS

COMMENTS: The parsing is slow because noun phrase utterances are one of the last constructions the grammar looks for. This situation is further complicated since "printed" first appears to be the main verb of a sentence.

SENTENCE: THE FAT RED PARTY WAS ANGRY

PREPASS TIME = 5 MS
(THE FAT RED PARTY WAS ANGRY)

NIL

PARSE TIME = 955 MS

COMMENTS: This sentence fails to parse because "party" and adjectives are semantically incompatible.

SENTENCE: THE BOX GAVE THE BOY A PENCIL

PREPASS TIME = 5 MS
(THE BOX GAVE THE BOY A PENCIL)

NIL

PARSE TIME = 282 MS

COMMENTS: This sentence fails to parse because the subject of the verb "give" must be animate.
SENTENCE: A RUNNING PYRAMID IS ON THE TABLE

PREPASS TIME = 6 MS
(A RUNNING PYRAMID IS ON THE TABLE)

NIL

PARSE TIME = 564 MS

COMMENTS: This sentence fails to parse because the adjective "running" should not modify "pyramid".

SENTENCE: PUT THE CUBE

PREPASS TIME = 3 MS
(PUT THE CUBE)

NIL

PARSE TIME = 280 MS

COMMENTS: This sentence fails to parse because it is not specified where the "cube" is to be "put".

SENTENCE: ME TALKS

PREPASS TIME = 6 MS
(ME TALKS)

NIL

PARSE TIME = 123 MS

COMMENTS: This sentence fails to parse because "me" cannot be the subject of a sentence.
SENTENCE: THEY WAS COMING TO THE PARTY

PREPASS TIME = 5 MS
(they was coming to the party)
NIL

PARSE TIME = 123 MS

COMMENTS: This sentence fails to parse because the subject and verb do not agree in number.

SENTENCE: HE GAVE HER A BOOKS

PREPASS TIME = 4 MS
(he gave her a books)
NIL

PARSE TIME = 301 MS

COMMENTS: This sentence fails to parse because the determiner "a" does not agree with the noun "books" which it modifies.
APPENDIX 3 - THE GRAMMAR

1. THE GRAMMAR

2. IS

3. THIS IS THE INITIAL STATE OF THE GRAMMAR

4. JUMP S-YESNO

5. IF INPUT LOOKS LIKE A YESNO QUESTION, PROCEED TO

6. IS-YESNO TO LOGX FOR THE TENSED AUXILIARY VERB.

7. (IAUXV LEX)

8. (ISETR TYPE 'YESNO)

9. JUMP S-WH

10. IF INPUT LOOKS LIKE A WH-QUESTION, PROCEED TO S-WH TO

11. LOOK FOR QUESTION DETERMINERS, PRONOUNS OR ADVERBS.

12. (WHOFORM LEX)

13. JUMP S-IMP

14. IF INPUT LOOKS LIKE AN IMPERATIVE, PROCEED TO S-IMP TO

15. LOOK FOR AN UNTENSED VERB. IMPERATIVES MAY NOT BE INTRODUCED

16. BY A PREPOSITION PHRASE.

17. (AND (NULL (GETR PREPPHRASES))

18. (SETL LEX 'UNTENSED)

19. (SETL TYPE 'UNTENSED)

20. EPUSH PP

21. LOOK FOR INTRODUCTORY PREPOSITION PHRASES.

22. IF AN INTRODUCTORY PREPOSITION PHRASE BEGINS WITH "BY"

23. THEN THE SUBJECT OF A PASSIVE QUESTION MAY BE IN THE PHRASE.

24. (AND (NULL (GETR PREPPHRASES))

25. (SETL LEX 'PREP)

26. (SENDL FRONTED-AGFLAG T)

27. ECOND ((NOT (GETR AGENT)) (ADDR PREPPHRASES NIL)

28. (IF S-PP)

29. JUMP S-DCL

30. IF INPUT DOES NOT LOOK LIKE A QUESTION, TRY TO PARSE IT AS

31. A DECLARATIVE SENTENCE.

32. (NOT (QUESTFORM LEX))

33. (SETL TYPE 'DCL)

34. JUMP S-NPU

35. IF ALL ELSE FAILS, TRY TO INTERPRET THE INPUT AS A NOUN PHRASE

36. SENTENCE. PREPOSITION PHRASES OR ADVERBS MAY NOT PRECEDE

37. THIS CONSTRUCTION.

38. (AND (NULL (GETR PREPPHRASES))

39. (NULL (GETR AGENT))

40. (NULL (GETR ADVERBS))

41. JUMP S-PPU-POP

42. IF A PREPOSITION PHRASE HAS BEEN FOUND AND ALL ELSE

43. IFALLS, THE INPUT IS A FRAGMENT CONSISTING OF A

44. PREPOSITION PHRASE.

45. (OR (GETR PREPPHRASES)

46. (GETR AGENT)

47. ECOND (GETR AGENT)

48. (ADDR PREPPHRASES (BUILD (PP (PREP BY) AGENT) T))

49. (IF S-PP)

50. EPUSH ADV

51. LOOK FOR INTRODUCTORY ADVERBS.

52. (GETL LEX 'ADV)

53. (ADDR ADVERBS P)

54. (TO S1)

55. )

56. )

57. IS-PPU-POP

58. CONSTRUCT A PREPOSITION PHRASE UTTERANCE PARSE AND RETURN.

59. (PP (BUILD-PPU)

60. T)

61. )

62. IS-PP

63. IF AN INTRODUCTORY PREPOSITION PHRASE HAS BEEN FOUND, IF A QUESTION

64. (GETEXTRINSIC OR A QUESTION PRONOUN HAS BEEN FOUND IN THE PHRASE,

65. THE FLAG WH-PHRASE IS SET AND THEN PARSE CONTINUES AS FOR

66. A QUESTION.
IE.C. IN HOW MANY FUNCTIONS IS FOB CALLED?

IF A QUESTION DETERMINER OR PRONOUN HAS BEEN FOUND THE NEXT
WORD MUST BE AN AUXILIARY VERB.

(AND (GETR WH-PHRASE)
(AUX LEFT)))

I.JUMP S

PROCESSING A SIMPLE DECLARATIVE SENTENCE. PROCEED TO STATE S.

S-DCL

THE INPUT APPEARS TO BE A DECLARATIVE SENTENCE, LOOK FOR THE
SUBJECT OF THE SENTENCE.

PUSH NP

THE INPUT APPEARS TO BE A DECLARATIVE SENTENCE, LOOK FOR THE
SUBJECT OF THE SENTENCE.

I.RECOGNIZE THE SUBJECT OF THE SENTENCE.

I.PROCEED TO S-NP TO LOOK FOR VERBS.

T

SET SUBJ #)

(SET NP SUBJ (GETR HEADNOUN))

(I TO S-NP)

S-IMP

THE INPUT APPEARS TO BE AN IMPERATIVE SENTENCE.

IE.G. LIST THE FUNCTIONS.

I.RECOGNIZE THE SUBJECT OF THE SENTENCE.

I.PROCEED TO S-VERB TO CHECK FOR A NEGATIVE IMPERATIVE.

I.E.G. DO NOT LIST THE FUNCTIONS.

I.PROCEED TO S-VERB TO LOOK FOR A TENSED AUXILIARY VERB.

I.RECOGNIZE THE SUBJECT OF THE SENTENCE.

I.PROCEED TO S-VERB TO LOOK FOR A TENSED AUXILIARY VERB.

I.RECOGNIZE THE SUBJECT OF THE SENTENCE.

I.PROCEED TO S-VERB TO LOOK FOR A TENSED AUXILIARY VERB.
THE INPUT IS INTRODUCED BY A WH-NOUNPHRASE.
E.G. WHICH MAN WHAT BOOKS

JUMP S-YESNO

THE WH-NP IS THE OBJECT OF THE SENTENCE. AN AUXILIARY VERB MUST BE THE NEXT ELEMENT SINCE SUBJECT-VERB INVERSION OCCURS.
E.G. WHAT BOOKS DID HE GIVE YOU?
PLACE THE OBJECT NOUNPHRASE ON THE HOLD STACK.
S ProcedE To S-YESNO TO DEAL WITH INVERSION.

JUMP S-SP

THE WH-NOUNPHRASE IS THE SUBJECT OF THE SENTENCE. A VERB MUST BE THE NEXT ELEMENT. NO SUBJECT-VERB INVERSION OCCURS.
E.G. WHICH MAN CAME?
S ProcedE TO S-SP TO LOOK FOR AUXILIARY VERBS.

GET LEX

_SETA S-DO (GETR HEADNOUN)
_(HOLD (GETR WH-NP))
_(SETA GOIJ (GETR HEADNOUN))

JUMP S-REI

THIS IS THE START OF THE RELATIVE CLAUSE NETWORK.
RELATIVE CLAUSES MAY START WITH "WHICH", "WHO", "THAT", "WHOM" or a preposition.

REP (WHICH WHO THAT)

PROCESSING A RELATIVE CLAUSE STARTING WITH "WHICH", "WHO", or "THAT", "WHICH" or "THAT" MAY REPLACE THE SUBJECT OR OBJECT OF THE CLAUSE.
E.G. THE MAN THAT WE SAW....
E.G. THE MAN THAT SAW THE DOG.....

TO S-REL-WH)

END WHOH

RELATIVE CLAUSE STARTING WITH "WHOH" MEANS THAT THE PRESSING NOUN-PHRASE IN THE CLAUSE IS NOT THE SUBJECT.
SO PUT THE SENDA-ED REGISTER WH ON THE HOLD STACK.
E.G. THE BOY WHOH I SAW....

TO

(HOLD (GETR WH))
_SETA S-DO (GETR HEADNOUN)
_SETA WH-NP
.TO S-REL-WH))

END WHOSE

THE RELATIVE CLAUSE STARTS WITH "WHOSE", THIS MEANS THAT A NOUN PHRASE FELOWS. THIS NOUN PHRASE MAY BE EITHER THE SUBJECT OR OBJECT OF THE CLAUSE.
E.G. THE MAN WHOSE DOG WE SAW....

TO

I TO S-REL-WHOSE)

ECAT PREP

THE RELATIVE CLAUSE IS INTRODUCED BY A PRONOUN.
E.G. THE WATER INTO WHICH HE FELL....

TO

(SETA PREP *)
.TO S-REL-PREPI)

END WHOSE

IF THE REST OF THE NOUNPHRASE MODIFIED BY "WHOSE".
S ProcedE TO S-REL-WH TO LOOK FOR THE REST OF THE CLAUSE.
IF THE CLAUSE WAS INTRODUCED BY A PREPOSITION THEN CONSTRUCT THE PREPOSITION PHRASE.
E.G. THE GIRL TO WHOSE BROTHER THE BOOK WAS GIVEN...
IF THE PREPOSITION WAS "BY" THEN SAVE THE NOUN PHRASE IN CASE THE CLAUSE IS PASSIVE.
E.G. THE GIRL BY WHOSE BROTHER THE BOOK WAS TAKEN...

(PUSH NP-DET
   (SENQ DET 'WH)
   (SENQ DET 'WH)
   (COND (GETR PREPWH)
      (COND (EQU (GETR PREP) 'BY)
      (SETQ AGENT *)
      (SETQ HEAD AG ICEIR HEADNOUN))
   (T (SETQ WH-HEADNOUN (GETR HEADNOUN))
      (SETQ WH *))
   (TO S-REL-WH)))
)

(S-REL-PREP
  IA PREPOSITION INTRODUCED THE RELATIVE CLAUSE.
  (PEN WHICH WH)
  THE MISSING NOUN-PHRASE IS THE OBJECT OF THE PREPOSITION.
  E.G. THE WATER INTO WHICH HE FELL...
  BUILD THE PREPOSITION PHRASE.
  IF THE PREPOSITION IS "BY" THEN SAVE THE NOUN PHRASE IN CASE THE CLAUSE IS PASSIVE.
  (COND ((GETR WH) (HOLD (GETR WH))))
  (SETQ AGENT (GETR WH))
  (SETQ HEAD AG ICEIR HEADNOUN))
  (T IADDR PREPPHRASES (BUILDO (PP (PREP •) •! PREPWHMII)
  (SETQ WH-HEADNOUN (GETR HEADNOUN))
  (SETQ WH »))
  (TO S-REL-WH))
)

(S-REL-WHOSE
  THE MISSING NOUN PHRASE IS A POSSESSIVE MODIFYING A FOLLOWING NOUN PHRASE.
  E.G. THE GIRL TO WHOSE BROTHER THE BOOK WAS GIVEN...
  (COND ((GETR WH) (HOLD (GETR WH))))
  (SETQ AGENT (GETR WH))
  (SETQ HEAD AG ICEIR HEADNOUN))
  (T IADDR PREPPHRASES (BUILDO (PP (PREP •) •! PREPWHMII)
  (SETQ WH-HEADNOUN (GETR HEADNOUN))
  (SETQ WH »))
  (TO S-REL-WHOSE))
)

(S-REL-NP
  THE RELATIVE PRONOUN HAS BEEN IDENTIFIED. LOOK FOR THE REST OF THE CLAUSE.
  (PUSH NP
   (COND (GETR WH) (HOLD (GETR WH)))
   (GETR WHOJI))
   (SETQ WHOJ I)
   (SETQ SUBJ I)
   (SETQ SUBJ (GETR HEADNOUN)))
   (TO S-NP))
   (JUMP S-NP)
)

(S-REL-REDUCED
  NO NOUN-PHRASE FOUND, SO KEEP REGISTER WH AS THE SUBJECT.
  IF WE ARE NOT PROCESSING A REDUCED RELATIVE CLAUSE WHICH MUST START WITH A NP IF WE GET TO HERE.
  E.G. THE BOY WHO CAME...
   (NOT (GETR REDUCED))
   (SETQ SUBJ (GETR HEADNOUN))
   (GETR SUBJ))
   (TO S-REL-WH))
)

(PUSH PP
   (GET LEX 'PREP)
   (ADDQ PREPPHRASES *)
   (TO S-REL-WH))
)

(S-REL-REDUCED
  THIS IS THE START OF THE REDUCED RELATIVE CLAUSE NETWORK.
)

(WH NOT
  THE VERB IS NEGATED.
)
(CAT V

THE REDUCED RELATIVE CLAUSE IS PASSIVE.
E.G., "THE BOOK GIVEN TO YOU"

THE OBJECT), SET AGFLAG AND PASSIVEFLAG TO INDICATE A PASSIVE.

SET SUBJ TO "SOMETHING". DEFAULT TENSE TO PRESENT. SAVE

THE VERB IN V.

(PASSIVE)

ISETR V *

ISETR S-VERB (BUILDQ @ (V *) (#3) FEATURES)

(SETNI HEADNOUN)

ISETR AGFLAG T

ISETR PASSIVEFLAG T

ISETR SUBJ

"SOMETHING" (PROP SOMETHING) (HWH SG-PL)

ISETR S-SUBJ (GETR WH-HEADNOUN)

ISETR TENSE (PRESENT)

ISETR VOICE (VOICE PASSIVE)

(ITO S-V)

(IJUMP S-REL-WH)

OTHERWISE THE REDUCED CLAUSE MUST START WITH A NP.
E.G., "THE BOOK THE MAN GAVE TO THE BOY"

T

ISETR REDUCED T

) (S-NP

A SUBJECT NOUN-PHRASE HAS BEEN FOUND. LOOK FOR AN AUXILIARY VERB.

IVIR V

ISETR VERB FROM THE HOLD STACK IF PROCESSING A QUESTION.

AND (ISETR YESNO)

(S-V-NUMBER-CMK (GETR S-SUBJ) (GETR S-VERB) )

(ISTV (CAOR *))

(ITO S-AUX)

(CAT V

A VERB MUST BE TENSED, UNLESS PROCESSING A TO-COMPLEMENT.

OR (AND (ISETR TNS)

(S-V-NUMBER-CMK (GETR S-SUBJ)

(BUILDQ @ (V *) (#3) FEATURES))

(AND (ISETR TOFLAG)

(SETNI UNSET))

(SETNI TENSE (LIST (GETF TNS)))

(SETNI S-VERB (BUILDQ @ (V *) (#3) FEATURES))

(ISTV V *)

(ITC S-AUX)

(IJUMP S-REL-WH)

A NOUN-PHRASE UTTERANCE HAS BEEN FOUND. RETURN THE PARSE.

(AAD ADVH "ADV")

) (ITD S-NP)

)

(S-NP-POP

A NOUN-PHRASE UTTERANCE HAS BEEN FOUND. LOOK FOR MAIN VERB.
\textbf{SET TENSE AND ASPECT ACCORDINGLY.}\\
\textbf{WHO NOT}\\
\textbf{THE VERB IS NEGATED. THE MAIN VERB SO FAR MUST BE AN AUXILIARY.}\\
\textbf{FEATURE (GETR V) 'AUX}\\
\textbf{(SET NEG 'NEGATIVE}\\
\textbf{(TO S-AUX))}\\
\textbf{ICAT V}\\
\textbf{PERFECT}\\
\textbf{(SET V =')}\\
\textbf{ASPECT 'PERFECT}\\
\textbf{S-VERB (BUILDQ (V *) (#1) FEATURES}\\
\textbf{(TO S-AUX))}\\
\textbf{ICAT V}\\
\textbf{PROGRESSIVE}\\
\textbf{(SET V =')}\\
\textbf{ASPECT (APPEND (GETR ASPECT) 'PROGRESSIVE}\\
\textbf{S-VERB (BUILDQ (V *) (#1) FEATURES})\\
\textbf{(TO S-AUX))}\\
\textbf{ICAT V}\\
\textbf{FUTURE}\\
\textbf{(SET V =')}\\
\textbf{TENSE 'FUTURE}\\
\textbf{S-VERB (BUILDQ (V *) (#1) FEATURES}\\
\textbf{(TO S-AUX))}\\
\textbf{ICAT V}\\
\textbf{SENTENCE IS PASSIVE. PLACE SUBJ ON HOLD STACK SINCE IT IS}\\
\textbf{REALLY AN OBJECT. SET AGFLAG AND PASSIVEFLAG TO INDICATE PASSIVE.}\\
\textbf{SET SUBJ TO 'SOMETHING" UNLESS AGENT HAS BEEN FOUND ALREADY.}\\
\textbf{PASSIVE}\\
\textbf{(SET V =')}\\
\textbf{S-VERB (BUILDQ (V *) (#1 FEATURES}\\
\textbf{(TO S-AUX))}\\
\textbf{ICAT V}\\
\textbf{DO-AUX}\\
\textbf{(SET V =')}\\
\textbf{TENSE 'DO-AUX}\\
\textbf{S-VERB (BUILDQ (V *) (#1) FEATURES}\\
\textbf{(TO S-AUX))}\\
\textbf{ICAT V}\\
\textbf{MODAL}\\
\textbf{(SET V =')}\\
\textbf{S-VERB (BUILDQ (V *) (#1) FEATURES}\\
\textbf{(TO S-AUX))}\\
\textbf{ICAT V}\\
\textbf{WHEN} 'DO' IS THE AUXILIARY AND NOT PROCESSING A QUESTION WITH SUBJECT-VERB INVERSION OR A NEGATIVE 'DO' IS USED AS A MODAL.}\\
\textbf{DO-AUX}\\
\textbf{(SET V =')}\\
\textbf{S-VERB (BUILDQ (V *) (#1) FEATURES}\\
\textbf{(TO S-AUX))}\\
\textbf{ICAT V}\\
\textbf{IF 'DO' IS THE AUXILIARY AND NOT PROCESSING A QUESTION WITH SUBJECT-VERB INVERSION OR A NEGATIVE}\\
\textbf{DO-AUX}\\
\textbf{(SET V =')}\\
\textbf{S-VERB (BUILDQ (V *) (#1) FEATURES}\\
\textbf{(TO S-AUX))}
ICAT PREP

ICAT PREP

THE MAIN VERB MAY NOT BE A MODAL.

COND \[(AND \{GETR QOBJ\} \(\{FEATURE \{GETR VI \"AUX\}\}\) \(\{OR \{GETR ASPECT\}\) \(\{GETR MODAL\}\) \(\{FUTURE\}\) \{GETR TENSE\} \{MODAL\} NIL)\]\]

COND \[(GETR AGENT) \{MATCH PREPHRASES \{BUILD \{PP \{PREP \"BY\}\} \{AGENT\}\} \{AGENT NIL\}\}]\]

IS-V-SEMANTICS

THIS STATE CHECKS THE SEMANTIC AGREEMENT OF THE SUBJECT AND THE MAIN VERB.

COND \[(IS-V-SEMANTICS \{GETR S-SUBJ\} \{GETR S-VERB\} NIL)\]}

IS-V

AT THIS POINT THE MAIN VERB HAS BEEN IDENTIFIED. LOOK FOR POST-VERBAL MODIFIERS.

ICAT NP

RETRIEVE THE NP PLACED ON HOLD STACK AS OBJECT OF THE VERB WHEN PROCESSING A RELATIVE CLAUSE, PASSIVE, FOR WH-QUESTION.

T

COND \[(NOT \{V-DO-AGREEMENT\}) (ABORT)\]

SETR OBJ «T

TO S-V-TOCOMP)】

PUSH NP

LOOK FOR A DIRECT OBJECT OR SUBJECTIVE COMPLEMENT FOR TRANSITIVE OR COPULA VERBS.

COND \[(\{FEATURE \{GETR VI \"TRANS\}\} \{FEATURE \{GETR VI \"COPULA\}\} \{SEND\} \{NOT\} \{feature\} \{GETR \{HEAD\} \{NUM\}\})\]

SETR S-DO \{GETR HEAD\} \{NUM\})]

COND \[(NOT \{V-DO-AGREEMENT\}) (ABORT)\]

SETR OBJ \{V\}

TO S-V-TOCOMP)

PUSH AJP

LOOK FOR PREDICATE ADJECTIVES AFTER COPULA VERBS.

COND \[(\{AND\} \{FEATURE \{GETR VI \"COPULA\}\} \{OR \{GET \{LEX \"ADV\}\} \{GET \{LEX \"ADJ\}\}\})\]

SETR V \{\}

COND \[(NOT \{PRED-ADJ-SEMANTICS \{LIST \{GETR HEAD\} \{ADV\}\}\}) \{GETR S-SUBJ\} \{GETR S-VERB\}\]

(ABORT)\]

SETR S-VERB \{\}

TO S-PRED-ADJ)

ICAT PREP

LOOK FOR A PARTICLE IMMEDIATELY AFTER THE VERB, EXPEL THE PREDICATE TO THE NEW COMPOUND.

SEND \{NOT \{GET \{VI \"PARTICLE\}\}\]

SEND \{NOT \{GET \{GETR VI \"PARTICLES\}\}\]

SETR V \{SEND \{ASSQ \{GET \{GETR VI \"PARTICLES\}\}\\]\}

SETR S-VERB \{BUILD \{V\} \{V\}\}

SETR PARTICLE T)

TO S-V)】

ICAT PREP

LOOK FOR THAT-COMPLEMENTS.

SEND \{NOT \{GET \{VI \"THAT\}\}\]

SETR S-V-THAT)

ICAT PREP

LOOK FOR TO-COMPLEMENTS.

SEND \{NOT \{GET \{VI \"TO\}\}\]

SETR S-V-WTO)

ICAT PREP

LOOK FOR THAT-COMPLEMENTS.

SEND \{NOT \{GET \{VI \"THAT\}\}\]

SETR S-V-WTHAT)

ICAT PREP

LOOK FOR TO-COMPLEMENTS.

SEND \{NOT \{GET \{VI \"TO\}\}\]

SETR S-V-WTO)

ICAT PREP

LOOK FOR THAT-COMPLEMENTS.
126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126

126
(ADDR ADVERBS *)
(TO S-V-NP)
(JUMP S-V-PP 1)
)

(S-V-PREP-BY

THE PREPOSITION "BY" HAS BEEN FOUND AND THE SENTENCE IS
PASSIVE. THE FOLLOWING NP MAY BE THE AGENT OF THE PASSIVE
SENTENCE AND THUS THE SUBJECT OF THE ACTIVE SENTENCE.
IE.G. THE BOOK WAS GIVEN BY THE MAN.

Otherwise the noun phrase may be the subject of the locational
PREPOSITION "BY".
IE.G. THE MAN STOOD BY THE STREAM.

(PUSH NP

(SENR AGFLAG (GETR AGFLAG))
(SENR MOTIONFLAG T)
(COND (NOT IS-V-SEMMANTICS (GETR HEADNOUN) (GETR S-VERB))

(IF (ADDR PREPPHRASES BUDDIO PP (PREP BY) #)))

(SETR SUBJ *)

(SETR S-SUBJ (GETR HEADNOUN))

(SETR AGFLAG NIL))
(TO S-V-PP))
)

(S-V-PP

LOCK FOR FINAL POST-VERBAL MODIFIERS.

(IWRD BY

THE AGENT OF THE PASSIVE SENTENCE MAY FOLLOW "BY".

(GETR AGFLAG)
(TO S-V-PREP-BY))

(PUSH PP

(GET LEX *PREP)
(SENR MOTIONFLAG)
(COND (NOT (ASSO *MOTION (CDR (GET LEX *PREP))))

(SENR AGFLAG (GETR AGFLAG))

(IF (ADDR PREPPHRASES #))
(TO S-V-PP))
)

(PUSH ACV

(GET LEX *ADV)

(IF (ADDR ADVERBS #)

(TO S-V-PP))
)

(JUMP S-MAINCLAUSE)
)

(S-V-WRD=THAT

SLOCK FOR THAT-COMPLEMENTS.

IE.G. I HOPE THAT HE COMES.

(PUSH S-OCL

(SENR TYPE *OCL*)

(IF (ADDR COMPL

(BUDDIO COMPL (TYPE THAT) #))

(IF (AND (FEATURE (GETR V) *TRANS)

(SETR S-DO (CAR (GETR COMPL))))

(TO S-MAINCLAUSE)))
)

(S-V-TOCOMP

SLock FOR TO-COMPLEMENTS.

IE.G. I WANT TO GO.

(ICAT PREP

(ADDR V (GETR V (GET (GETR V) *PARTICLES)))

(IF (AND (NOT (GETR PARTICLES))

(ASSO # (GET (GETR V) *PARTICLES))))

(SETR V (CDR (ASSO # (GET (GETR V) *PARTICLES))))

(SETR S-VERB (BUDDIO V * V))

(SETR PARTICLE 1)

(TO S-V-TOCOMP))
)

(IWRD TO

(TO S-V-WRD=THAT))

(JUMP S-V-NP

T)
)

ENDC )
1001 (S-V-ARQ-T0)
1002
1003 (S-V-ARQ-T0)
1004 1005 LOOK FOR THE REMAINDER OF THE TO-COMPLEMENT. THE SUBJECT OF THE TO-COMPLEMENT IS THE OBJECT OF THE TOP LEVEL SENTENCE. IF IT IS PRESENT, OTHERWISE IT IS THE SUBJECT OF THE TOP-LEVEL.
1006 1007 (PUSH S-NP  
1008  (SEND TDFLAG T)  
1009 (SEND TYPE H-T0CL1)  
1010 (SEND SUBJ  
1011 (COND (GETR OBJ) )  
1012 (GETR SUBJ) )  
1013 (SEND TENSE (GETR TENSE))  
1014 (SEND S-SUBJ  
1015 (COND (GETR OBJ) (GETR S-D0))  
1016 (GETR SUBJ) (GETR S-SUBJ) )  
1017 1018 (SEND ADDR COMPL 
1019 (BUILD (COMP ICTYPE TO) *))  
1020 (AND (FEATURE (GETR VI) *TRANS)  
1021 (SETR S-DO (CAR (GETR COMPL))))  
1022 1023 (TO S-MAINCLAUSE))  
1024 1025 )
1026 1027 (S-MAINCLAUSE  
1028 1029 1030 THE MAIN CLAUSE HAS BEEN IDENTIFIED. LOOK FOR A CONJUNCTION FOR THE END OF SENTENCE.
1031 1032 (CAT CONJ  
1033 T  
1034 (SETR CONJ *)  
1035 (TO S-CONJ))  
1036 1037 1038 (JUMP S-S  
1039  
1040 )
1041 1042 (S-CONJ  
1043 1044 A CONJUNCTION HAS BEEN FOUND AFTER THE MAIN CLAUSE. LOOK IF THE CONJUNCTED PHRASE.
1045 1046 (S-MAINCLAUSE  
1047 (S-CONJ-V-ACTIVE  
1048 ITHE MAIN CLAUSE IS AN ACTIVE DECLARATIVE SENTENCE WITH NO SUBJECT-VERB INVERSION.  
1049 (PUSH S-V-SEMANTICS  
1050 1051 SEND DOWN ALL THE NECESSARY REGISTERS CONCERNING TENSE, TYPE, SUBJECT, AND VERB INFORMATION.)  
1052 1053 1054 (SETR S-DO (CAR (GETR COMPL)))  
1055 1056 (AND (GETR VI))  
1057 (SEND PASSIVEFLAG)  
1058 (SEND TYPE H-MIMP)  
1059 (SEND YESNO)  
1060 (SETR CONJ-V *)  
1061 (TO S-CONJ-V-PASSIVE))  
1062 1063 (S-CONJ-V-PASSIVE  
1064 ITHE MAIN CLAUSE IS PASSIVE OR HAS SUBJECT-VERB INVERSION.  
1065 ITHE CONJUNCTED PHRASE IS A COMPLETE VERB PHRASE.
1066 1067 (AND (GETR VI))  
1068 (SEND PASSIVEFLAG)  
1069 (SEND YESNO)  
1070 (SEND TYPE H-MIMP)  
1071 (SEND YESNO)  
1072 (AND (GETR OBJ) (GETR OBJ))  
1073 (SETR CONJ-V *)  
1074 (TO S-CONJ-V-PASSIVE))  
1075 (IPUSH S  
1076 (SETR S-CONJ *)  
1077 (TO S-S))  
1078 1079 )
1080 1081 (S-CONJ-V-PASSIVE  
1082 ITHE MAIN CLAUSE IS AN ACTIVE DECLARATIVE SENTENCE WITH NO SUBJECT-VERB INVERSION.  
1083 (PUSH S-V-SEMANTICS  
1084 1085 SEND DOWN ALL THE NECESSARY REGISTERS CONCERNING TENSE, TYPE, SUBJECT, AND VERB INFORMATION.
1086 1087 1088 1089 ITHE MAIN CLAUSE IS AN ACTIVE DECLARATIVE SENTENCE WITH NO SUBJECT-VERB INVERSION.  
1090 (IPUSH S-V-SEMANTICS  
1091 1092 SEND DOWN ALL THE NECESSARY REGISTERS CONCERNING TENSE, TYPE, SUBJECT, AND VERB INFORMATION.
1093 1094 1095 1096 1097 1098 (SEND VI)  
1099 (SEND VI)  
1100 (SEND VI)  
1101 (SEND VI)  
1102 1103 )
1104 1105 )
1106 1107 )
1108 )
1109 )
1110 )
1111 )
1112 )
IF THE STRING IS NULL THE "PUSH" FAILS. THEREFORE ALL CONSTITUENTS MUST BE PRESENT AT THIS LEVEL.

IF THE BOOKS WERE PICKED UP AND TAKEN.

PROCEED TO S-CONJ-BUILD TO CONSTRUCT THE PARSE.

THE MAIN CLAUSE HAS EITHER SUBJECT-VERB INVERSION OR A PASSIVE TRANSFORMATION.

THE CONSTITUENTS OF THE CONJUNCTED SENTENCE ARE ALL PRESENT AT THIS LEVEL, SO CONSTRUCT THE PARSE, CHECK THE SEMANTICS, AND PROCEED TO S-S.

THE NETWORK RECOGNIZES NOUN-PHRASES.
THIS STATE CHECKS FOR DETERMINERS, PRONOUNS, PROPER NOUNS, AND POSSESSIVE PRONOUNS AND PROPER NOUNS. DETERMINERS MAY BE MISSING.

ICAT GENPRO

THIS ARC IDENTIFIES GENERAL PRONOUNS SUCH AS SOMETHING, ANYTHING ETC.

T (SET PROPER-BUILD (ICATPRO 1))
ICAT DET
1207 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1208 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1209 E.G. SOME OF THE BOYS...

1211 (NOT (GETF POSSPRO))
1212 (SETA SEM-DET (BUILDQ POSS *)))
1213 (SETA SEM-NOUN (GETA SEM-NOUN))
1214 (SETA OFLAG (GETF QUANT))
1215 (TO NP-DET)

1217 ICAT DET
1219 IPROCESS POSSESSIVE PRONOUN DETERMINERS.
1221 E.G. HIS BOOK... THEIR DOG...
1223 (GETF POSSPRO)
1224 (SETA SEM-DET (BUILDQ POSS *)))
1225 (SETA DET (BUILDQ (POSS (PRO *)) #)) (FEATURES))
1226 (TO NP-DET)

1227 ICAT PRO
1229 (NOT (GETF POSS))
1230 (SETA SEM-DET (BUILDQ POSS *)))
1231 (SETA SEM-NOUN (BUILDQ (POSS (PRO)) (FEATURES))
1232 (SETA HEADNOUN (GETA SEM-NOUN))
1233 (SETA NU (GETF NUMBER))
1234 (SETA PRO E)
1235 (TO NP-N)

1236 ICAT PRO
1238 (NOT (GETF POSS))
1239 (SETA PROPER (BUILDQ (POSS (PRO)) (FEATURES))
1240 (SETA SEM-NOUN (BUILDQ (PRO *)) (FEATURES))
1241 (SETA HEADNOUN (GETA SEM-NOUN))
1242 (SETA NU *SG)
1243 (TO NP-N)

1244 ICAT DET
1246 (NOT (GETF POSS))
1248 (SETA PROPER (BUILDQ POSS *)))
1249 (SETA SEM-DET (BUILDQ (PRO) (FEATURES))
1250 (SETA SEM-NOUN (BUILDQ (PRO *) (FEATURES))
1251 (SETA HEADNOUN (GETA SEM-NOUN))
1252 (SETA NU *SG)
1253 (TO NP-N)

1254 ICAT PRO
1256 (NOT (GETF POSS))
1258 (SETA PROPER (BUILDQ POSS *)))
1259 (SETA SEM-DET (BUILDQ POSS *)))
1260 (SETA SEM-NOUN (BUILDQ POSS (PRO *)) (FEATURES))
1261 (SETA HEADNOUN (GETA SEM-NOUN))
1262 (SETA NU *SG)
1263 (TO NP-N)

1264 ICAT DET
1266 (NOT (GETF POSS))
1268 (SETA PROPER (BUILDQ POSS *))
1269 (SETA SEM-DET (BUILDQ POSS *))
1270 (SETA SEM-NOUN (BUILDQ POSS *))
1271 (SETA HEADNOUN (GETA SEM-NOUN))
1272 (SETA NU *SG)
1273 (TO NP-N)

1274 ICAT QDET
1276 (NOT (GETF POSS))
1278 (SETA PROPER (BUILDQ POSS *)))
1279 (SETA SEM-DET (BUILDQ POSS *))
1280 (SETA SEM-NOUN (BUILDQ POSS *))
1281 (SETA HEADNOUN (GETA SEM-NOUN))
1282 (SETA NU *SG)
1283 (TO NP-N)

1284 ICAT PRO
1286 (NOT (GETF POSS))
1288 (SETA PROPER (BUILDQ POSS *))
1289 (SETA SEM-DET (BUILDQ POSS *))
1290 (SETA SEM-NOUN (BUILDQ POSS *))
1291 (SETA HEADNOUN (GETA SEM-NOUN))
1292 (SETA NU *SG)
1293 (TO NP-N)

1294 ICAT PRO
1296 (NOT (GETF POSS))
1298 (SETA PROPER (BUILDQ POSS *))
1300 (SETA SEM-DET (BUILDQ POSS *))
1301 (SETA SEM-NOUN (BUILDQ POSS *))
1302 (SETA HEADNOUN (GETA SEM-NOUN))
1303 (SETA NU *SG)
1304 (TO NP-GENPRO)

1306 ICAT DET
1308 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1309 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1310 E.G. SOME OF THE BOYS...

1311 (NOT (GETF POSSPRO))
1312 (SETA DET *)
1313 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1314 (SETA OFLAG (GETF QUANT))
1315 (TO NP-DET)

1316 ICAT DET
1318 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1319 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1320 E.G. SOME OF THE BOYS...

1321 (NOT (GETF POSSPRO))
1322 (SETA DET *)
1323 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1324 (SETA OFLAG (GETF QUANT))
1325 (TO NP-DET)

1326 ICAT DET
1328 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1329 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1330 E.G. SOME OF THE BOYS...

1331 (NOT (GETF POSSPRO))
1332 (SETA DET *)
1333 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1334 (SETA OFLAG (GETF QUANT))
1335 (TO NP-DET)

1336 ICAT DET
1338 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1339 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1340 E.G. SOME OF THE BOYS...

1341 (NOT (GETF POSSPRO))
1342 (SETA DET *)
1343 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1344 (SETA OFLAG (GETF QUANT))
1345 (TO NP-DET)

1346 ICAT DET
1348 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1349 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1350 E.G. SOME OF THE BOYS...

1351 (NOT (GETF POSSPRO))
1352 (SETA DET *)
1353 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1354 (SETA OFLAG (GETF QUANT))
1355 (TO NP-DET)

1356 ICAT DET
1358 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1359 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1360 E.G. SOME OF THE BOYS...

1361 (NOT (GETF POSSPRO))
1362 (SETA DET *)
1363 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1364 (SETA OFLAG (GETF QUANT))
1365 (TO NP-DET)

1366 ICAT DET
1368 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1369 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1370 E.G. SOME OF THE BOYS...

1371 (NOT (GETF POSSPRO))
1372 (SETA DET *)
1373 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1374 (SETA OFLAG (GETF QUANT))
1375 (TO NP-DET)

1376 ICAT DET
1378 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1379 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1380 E.G. SOME OF THE BOYS...

1381 (NOT (GETF POSSPRO))
1382 (SETA DET *)
1383 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1384 (SETA OFLAG (GETF QUANT))
1385 (TO NP-DET)

1386 ICAT DET
1388 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1389 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1390 E.G. SOME OF THE BOYS...

1391 (NOT (GETF POSSPRO))
1392 (SETA DET *)
1393 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1394 (SETA OFLAG (GETF QUANT))
1395 (TO NP-DET)

1396 ICAT DET
1398 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1399 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1400 E.G. SOME OF THE BOYS...

1401 (NOT (GETF POSSPRO))
1402 (SETA DET *)
1403 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1404 (SETA OFLAG (GETF QUANT))
1405 (TO NP-DET)

1406 ICAT DET
1408 IF THE DETERMINER HAS FEATURE "QUANT", THE OFLAG IS SET
1409 TO INDICATE THAT A PARTITIVE MAY FOLLOW.
1410 E.G. SOME OF THE BOYS...

1411 (NOT (GETF POSSPRO))
1412 (SETA DET *)
1413 (SETA SEM-DET (LIST *DET #)) (FEATURES))
1414 (SETA OFLAG (GETF QUANT))
1415 (TO NP-DET)
THE DETERMINER HAS BEEN IDENTIFIED. NOW LOOK FOR ORDINALS.

ORDINALS ARE WORDS SUCH AS FIRST, LAST, OR NEXT. THESE ALWAYS OCCUR BEFORE THE ADJECTIVES AND QUANTIFIERS IN THE NOUN PHRASE.

E.G. THE FIRST LARGE BOOK...

ORDINALS MAY BE MISSING.

(CAT ORD)

LOOK FOR ORDINALS SUCH AS FIRST, LAST ETC.

T

(CAT ADJ)

SUPERLATIVE ADJECTIVES MAY ACT AS ORDINALS.

E.G. THE BIGGEST RED BOOK...

SUPERLATIVE (ADJ) #1 FEATURES)

(TO NP-ORD)

(CAT ORD)

ORDINALS MAY BE MISSING.

T)

(INP-ORD)

THE ORDINAL HAS BEEN IDENTIFIED. NOW LOOK FOR QUANTIFIERS.

(PUSH QUANT)

THE QUANTIFIER MAY BE A MORE COMPLICATED STRUCTURE.

E.G. AT LEAST FIVE OF THE BOYS....

T

(SETR QUANT T)

(SETR SEM-DEF NIL)

(TO NP-QUANT)

(CAT ORD)

THE QUANTIFIER MAY BE MISSING.

T)

(INP-QUANT)

THE QUANTIFIER HAS BEEN IDENTIFIED. NOW LOOK FOR A PARTITIVE CONSTRUCTION AFTER A DETERMINER AND ORDINAL OR A QUANTIFIER, OR A QUANTIFIER DETERMINER.

E.G. AT LEAST FIVE OF THE BOYS....

E.G. THE FIRST OF THE PARTIES....

E.G. SOME OF THE BOOKS....

(INP-QUANT)
1*01 IGETK  OUANTII
1*02 (10 NP-QUANT-PREP-GF)>
1*0* (JUMP  NP-QUANT-PREP"OF
1*05 ;IF THE QUANTIFIER DETERMINER IS "ALL" OR "BOTH" ANO
1*07 '  :N0 ORDINAL CP. QUANTIFIER IS PRESENT, THE "OF" IN THE
1*08 (PARTITIVE  CONSTRUCTION HAY BE UMIITEO.
1*0? IE.G- ALL THESE BOYS....
1*10
1*11 IANO INOT  IGETR OROIIINOT IGETR OUANTI)
1*12 IGETR  OFLAG)
1*13 I  MEMO (GETR DET) MALL  B0THII  )
1*1*
1*15 IJUMP  NP-PART
1*16 STHERE IS NO  PARTITIVE  CONSTRUCTION.
1*18
1*19 INOT IEQ »  'OF)I)
1*20
1*21 IJUMP  NP-MAINPHRASE
1*22
1*23  ;THE NOUN PHRASE MAY BE INCOMPLETE. IF A QUANTIFIER
1*24  IOR A QUANTIFIER DETERMINER, OR A DETERMINER AND AN ORDINAL,
1*25 ;OR A POSSESSIVE HAS BEEN FOUND, TRY TO POP.
1*26 IE.G. GIVE ME AT LEAST FOU.
1*27
1*28 [OR (GETR QUANT)(GETR QFLAG)(GETR POSSFLAG)]
1*29 (AND IGETR ORDGET) GETR GETR DET)]
1*30 (SETR N 'ONES)
1*31 (SETR NU 'SG-PL)
1*32 )
1*33
1*35
1*37 INP-QUANT-PREP-GF
1*38
1*39
1*40
1*41
1*42
1*43
1*44
1*45
1*46
1*47
1*48
1*49
1*50
1*51
1*52
1*53
1*54
1*55
1*56
1*57
1*58
1*59
1*60
1*61
1*62
1*63
1*64
1*65
1*66
1*67
1*68
1*69
1*70
1*71
1*72
1*73
1*74
1*75
1*76
1*77
1*78
1*79
1*80
1*81
1*82
1*83
1*84
1*85
1*86
1*87
1*88
1*89
1*90
1*91
1*92
1*93
1*94
1*95
1*96
1*97
1*98
1*99
IF A POSSESSIVE NOUN HAS BEEN FOUND, ANY PREVIOUS NOUNS MUST HAVE BEEN ACTING AS ADJECTIVES, SO THE PARSE IS REARRANGED. E.G. THE TALL BOY'S FIRST RED HAT...

PROCEED TO NP-DET TO LOOK FOR ORDINALS, ETC.

IF A PREPOSITIONAL PHRASE MAY MODIFY A NOUN, E.G. THE BOY IN THE PARK...

IF A RELATIVE CLAUSE MODIFYING THE NOUN, RELATIVE CLAUSES MUST START WITH A RELATIVE PRONOUN OR A PREPOSITION, THE HEAD NOUN IS SEND-RD TO THE CLAUSE.

IF POSTNOMINAL MODIFIERS MAY BE MISSING.

LASTLY LOOK FOR REDUCED RELATIVE CLAUSES. AVOIDS CONFUSION WITH OTHER VERBS IN SENTENCE.
THE MAIN PHRASE HAS BEEN FOUND. LOOK FOR A CONJUNCTION OR POP.

CAT CONJ

LOOK FOR ANOTHER NOUN PHRASE AFTER THE CONJUNCTION.

E.G. THE BOOKS AND THE PENCIL.

A CONJUNCTION HAS BEEN FOUND. LOOK FOR ANOTHER NOUN PHRASE.

PUSH NP

<table>
<thead>
<tr>
<th>THE BOOK THE MAN GAVE...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(NOT (GETR PRO))</td>
</tr>
<tr>
<td>SENDR WH</td>
</tr>
<tr>
<td>(COND (GETR N))</td>
</tr>
<tr>
<td>BUILD NP (GETR WH (N #) (NU #) N NU))</td>
</tr>
<tr>
<td>(GETR PROPER)</td>
</tr>
<tr>
<td>BUILD NP (GETR WH (NU #) (NU #) PROPER NU))</td>
</tr>
<tr>
<td>SENDR WH-HEADNOUN (GETR SEM-NOUN))</td>
</tr>
<tr>
<td>SENDR TYPE <em>REL</em>)</td>
</tr>
<tr>
<td>ADDR REL #</td>
</tr>
<tr>
<td>{TO NP-MAINPHRASE}</td>
</tr>
<tr>
<td>THE BOOK THE MAN GAVE...</td>
</tr>
</tbody>
</table>

THE MAIN PHRASE HAS BEEN FOUND. LOOK FOR A CONJUNCTION OR POP.

CAT CONJ

LOOK FOR ANOTHER NOUN PHRASE AFTER THE CONJUNCTION.

E.G. THE BOOKS AND THE PENCIL.

A CONJUNCTION HAS BEEN FOUND. LOOK FOR ANOTHER NOUN PHRASE.

PUSH NP

<table>
<thead>
<tr>
<th>THE BOOK THE MAN GAVE...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(NOT (GETR PRO))</td>
</tr>
<tr>
<td>SENDR WH</td>
</tr>
<tr>
<td>(COND (GETR N))</td>
</tr>
<tr>
<td>BUILD NP (GETR WH (N #) (NU #) N NU))</td>
</tr>
<tr>
<td>(GETR PROPER)</td>
</tr>
<tr>
<td>BUILD NP (GETR WH (NU #) (NU #) PROPER NU))</td>
</tr>
<tr>
<td>SENDR WH-HEADNOUN (GETR SEM-NOUN))</td>
</tr>
<tr>
<td>SENDR TYPE <em>REL</em>)</td>
</tr>
<tr>
<td>ADDR REL #</td>
</tr>
<tr>
<td>{TO NP-MAINPHRASE}</td>
</tr>
<tr>
<td>THE BOOK THE MAN GAVE...</td>
</tr>
</tbody>
</table>

THE MAIN PHRASE HAS BEEN FOUND. LOOK FOR A CONJUNCTION OR POP.

CAT CONJ

LOOK FOR ANOTHER NOUN PHRASE AFTER THE CONJUNCTION.

E.G. THE BOOKS AND THE PENCIL.

A CONJUNCTION HAS BEEN FOUND. LOOK FOR ANOTHER NOUN PHRASE.

PUSH NP

<table>
<thead>
<tr>
<th>THE BOOK THE MAN GAVE...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(NOT (GETR PRO))</td>
</tr>
<tr>
<td>SENDR WH</td>
</tr>
<tr>
<td>(COND (GETR N))</td>
</tr>
<tr>
<td>BUILD NP (GETR WH (N #) (NU #) N NU))</td>
</tr>
<tr>
<td>(GETR PROPER)</td>
</tr>
<tr>
<td>BUILD NP (GETR WH (NU #) (NU #) PROPER NU))</td>
</tr>
<tr>
<td>SENDR WH-HEADNOUN (GETR SEM-NOUN))</td>
</tr>
<tr>
<td>SENDR TYPE <em>REL</em>)</td>
</tr>
<tr>
<td>ADDR REL #</td>
</tr>
<tr>
<td>{TO NP-MAINPHRASE}</td>
</tr>
<tr>
<td>THE BOOK THE MAN GAVE...</td>
</tr>
</tbody>
</table>

THE MAIN PHRASE HAS BEEN FOUND. LOOK FOR A CONJUNCTION OR POP.

CAT CONJ

LOOK FOR ANOTHER NOUN PHRASE AFTER THE CONJUNCTION.

E.G. THE BOOKS AND THE PENCIL.

A CONJUNCTION HAS BEEN FOUND. LOOK FOR ANOTHER NOUN PHRASE.

PUSH NP

<table>
<thead>
<tr>
<th>THE BOOK THE MAN GAVE...</th>
</tr>
</thead>
<tbody>
<tr>
<td>(NOT (GETR PRO))</td>
</tr>
<tr>
<td>SENDR WH</td>
</tr>
<tr>
<td>(COND (GETR N))</td>
</tr>
<tr>
<td>BUILD NP (GETR WH (N #) (NU #) N NU))</td>
</tr>
<tr>
<td>(GETR PROPER)</td>
</tr>
<tr>
<td>BUILD NP (GETR WH (NU #) (NU #) PROPER NU))</td>
</tr>
<tr>
<td>SENDR WH-HEADNOUN (GETR SEM-NOUN))</td>
</tr>
<tr>
<td>SENDR TYPE <em>REL</em>)</td>
</tr>
<tr>
<td>ADDR REL #</td>
</tr>
<tr>
<td>{TO NP-MAINPHRASE}</td>
</tr>
<tr>
<td>THE BOOK THE MAN GAVE...</td>
</tr>
</tbody>
</table>
This network identifies adverbs.

This network identifies adjective phrases.
(ADJP-ADJP
ICONSTRUCT AND RETURN PARSER

1801 (POP (BUILD-ADJP)
1802 T)
1803 )
1804 (PP)

1805 THIS NETWORK IDENTIFIES PREPOSITION PHRASES.
1806 (END BY
1807 (IF THE PREPOSITION IS "BY" AND IT IS NOT KNOWN WHETHER
1808 THE SENTENCE IS PASSIVE (I.E., THE PHRASE IS FRONTED)
1809 SET THE BYFLAG SO THE NOUN PHRASE CAN BE LIFTR-EQ.
1810 (GETR FRONTED-AGFLAG)
1811 (GETR BYFLAG T)
1812 (TC PP-PREP))
1813 (CAT PREP
1814 (IF THE SENTENCE IS PASSIVE AND THE PREPOSITION IS "BY",
1815 THEN THE AGENT OF THE SENTENCE MAY BE IN THE PHRASE. THE PHRASE
1816 IS NOT PARSED AS A PREPOSITION PHRASE.
1817 (IF THE PREPOSITION IS A MOTION PREPOSITION AND THE MOTIONFLAG
1818 IS SET, THEN THE PHRASE IS NOT PARSED AT THIS LEVEL.
1819 SET THE BYFLAG.
1820 (GETR MOTIONFLAG)
1821 (GETR AGFLAG)
1822 (TC PP-PREP))
1823)
1824 (GETR MOTIONFLAG))
1825 (SETP PREP *)
1826 (TO PP-PREP)
1827)
1828)
1829 (PP-PREP
1830 (IF THE OBJECT OF THE PREPOSITION.
1831 (SEND AFLAG (GETR ACFLAG))
1832 (SEND MOTIONFLAG (GETR MOTIONFLAG))
1833 (GETR NP *)
1834 (ICOND (GETR BYFLAG))
1835 (LIFTR AGENT * (LIFTR HEADAG (GETR HEADNOUN)))
1836 (TO PP-PREP))
1837)
1838)
1839)
1840)
1841)
1842)
1843)
1844)
1845)
1846)
1847)
1848)
1849)
1850)
1851)
1852)
1853)
1854)
1855)
1856)
1857)
1858)
1859)
1860)
1861)
1862)
1863)
1864)
1865)
1866)
1867)
1868)
1869)
1870)
1871)
1872)
1873)
1874)
1875)
1876)
1877)
1878)
1879)
1880)
1881)
1882)
1883)
1884)
1885)
1886)
1887)
1888)
1889)
1890)
1891)
1892)
1893)
1894)
1895)
1896)
1897)
1898)
1899)
1900)

1906)
1907)
1908)
1909)
1910)
IQP-DPR

I A QUESTION PRONOUN HAS BEEN FOUND. LOOK FOR AN OPTIONAL
PREPOSITION PHRASE BEFORE POPPING.

PUSH PP

GET LEX 'PREP

SET PP 1

ETO GP-QP)

JUMP CP-QP

)

IQP-QP

RETURN THE QUESTION PRONOUN PHRASE.

POP [BUILD-GP]

)

IQUANTP

THIS NETWORK RECOGNIZES QUANTIFIER CONSTRUCTIONS.

ICAT INTEGER

THE QUANTIFIER MAY BE A NUMBER.

IE.G. FIVE YELLOW BOOKS...

T

[SET Q 'BUILDQ INTEGER *1)]

ETO QUANTP-Q1)

ICAT QUANT

IDENTIFY QUANTIFIER WORDS SUCH AS SEVERAL, MANY, FEW....

T

[SET R Q 1]

[ETO QUANTP-Q1)]

]JUMP QUANTP-Q

LOOK FOR EXPRESSIONS NOT PRECEDED BY A QUANTIFIER.

IE.G. MORE THAN 6....

]

IQUANTP-Q

IWORD AT

LOOK FOR "AT" EXPRESSIONS.

IE.G. FIVE AT LEAST...

IE.G. AT LEAST FOUR...

T

[ETO QUANTP-PREP-AT])

ICAT CCOMP

LOOK FOR COMPARATIVE EXPRESSIONS.

IE.G. MORE THAN 5...

IE.G. SEVERAL MORE...

T

[SET CCOMP 'BUILDQ (ADV *1)]

[ETO QUANTP-CCOMP)

]JUMP QUANTP-QUANTP

IF A QUANTIFIER HAS BEEN FOUND, TRY TO POP.

IE.G. IF A QUANTIFIER HAS BEEN FOUND, TRY TO POP.

GET R Q1

[SET R QUANT-Q])

IQUANTP-PREP-AT

PROCESSING AN "AT" CONSTRUCTION. A SUPERSLATIVE FORM SUCH AS
LEAST, MOST, OR FEWEST MUST BE FOUND.

ICAT SUPER

T

[SET SUPER 'BUILDQ (ADV *1)]

[ETO QUANTP-MODS)

IQUANTP-CCOMP
2001 PROCESSING A COMPARATIVE
2002
2003 (WORD THAN)
2004
2005 THIS IS A "MORE THAN" TYPE EXPRESSION.
2006
2007 T
2008 (TO QUANTP-MODS))
2009
2010 \(\text{JUMP QUANTP-QUANTP}
2011 \)
2012 IA COMPARATIVE CONSTRUCTION HAS BEEN FOUND.
2013 I.E.G. SEVERAL MORE... FIVE FEWER...
2014 IIF THE WORD "MORE" HAS BEEN FOUND AND THE NEXT WORD
2015 IIS AN ADJECTIVE, THE ADJECTIVE IS A COMPARATIVE
2016 IF NOT, SO DO NOT POP.
2017 I.E.G. THE MORE BEAUTIFUL GIRLS...
2018 \(\text{COND (AND (GETR (COMP)) \MORE)}
2019 \(\text{EOT (EAD (GETR (COMP)) \MORE)}
2020 \(\text{GET (GETR (\ADJ)) \#)}
2021 \(\text{NIL)}
2022 \(\text{)}
2023 \(\text{)}
2024 \(\text{\SETR QUANT (NOT (AND (GETR (COMP)) (GETR Q))})\)
2025 
2026 
2027 
2028 \(\text{QUANTP-MODS}
2029 \)
2030 \(\text{LOOK FOR THE MAIN QUANTIFIER.}
2031 \)
2032 \(\LEAT DEF)
2033 \)
2034 \(\text{LOOK FOR A PRECEDING DETERMINER. THESE OCCUR IN EXPRESSIONS}
2035 LIKE "MORE THAN A FEW".
2036 \)
2037 \(\text{NOT (GETR DEF)}\)
2038 \(\text{SETR DEF \#)}
2039 \(\text{(TO QUANTP-MODS))}
2040 \)
2041 \)
2042 \(\text{PUSH QUANTP}
2043 \)
2044 \(\text{LOOK FOR THE MAIN QUANTIFIER CONSTRUCTION.}
2045 \)
2046 \)
2047 \(\text{SETR QUANT \#)}
2048 \(\text{(TO QUANTP-QUANTP))}
2049 \)
2050 \(\text{JUMP QUANTP-QUANTP}
2051 \)
2052 \(\text{PROCESSING A CONSTRUCTION LIKE "FIVE AT LEAST". TRY TO POP.}
2053 \)
2054 \(\text{(AND (GETR Q) (NOT (GETR COMP))}))\)
2055 \)
2056 \)
2057 \)
2058 \(\text{QUANTP-QUANTP}
2059 \)
2060 \(\text{END OF QUANTIFIER NETWORK. CONSTRUCT PARSE AND RETURN.}
2061 \)
2062 \(\text{POP (BUILD-QUANTP)}
2063 \)
2064 \)
2065 \)
2066 \)
2067 \)
<table>
<thead>
<tr>
<th>THE DICTIONARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>21</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>23</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>27</td>
</tr>
<tr>
<td>28</td>
</tr>
<tr>
<td>29</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>31</td>
</tr>
<tr>
<td>32</td>
</tr>
<tr>
<td>33</td>
</tr>
<tr>
<td>34</td>
</tr>
<tr>
<td>35</td>
</tr>
<tr>
<td>36</td>
</tr>
<tr>
<td>37</td>
</tr>
<tr>
<td>38</td>
</tr>
<tr>
<td>39</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>41</td>
</tr>
<tr>
<td>42</td>
</tr>
<tr>
<td>43</td>
</tr>
<tr>
<td>44</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>46</td>
</tr>
<tr>
<td>47</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>49</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>51</td>
</tr>
<tr>
<td>52</td>
</tr>
<tr>
<td>53</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>55</td>
</tr>
<tr>
<td>56</td>
</tr>
<tr>
<td>57</td>
</tr>
<tr>
<td>58</td>
</tr>
<tr>
<td>59</td>
</tr>
<tr>
<td>60</td>
</tr>
<tr>
<td>61</td>
</tr>
<tr>
<td>62</td>
</tr>
<tr>
<td>63</td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>65</td>
</tr>
<tr>
<td>66</td>
</tr>
<tr>
<td>67</td>
</tr>
<tr>
<td>68</td>
</tr>
<tr>
<td>69</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>71</td>
</tr>
<tr>
<td>72</td>
</tr>
<tr>
<td>73</td>
</tr>
<tr>
<td>74</td>
</tr>
<tr>
<td>75</td>
</tr>
<tr>
<td>76</td>
</tr>
<tr>
<td>77</td>
</tr>
<tr>
<td>78</td>
</tr>
<tr>
<td>79</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>81</td>
</tr>
<tr>
<td>82</td>
</tr>
<tr>
<td>83</td>
</tr>
<tr>
<td>84</td>
</tr>
<tr>
<td>85</td>
</tr>
<tr>
<td>86</td>
</tr>
<tr>
<td>87</td>
</tr>
<tr>
<td>88</td>
</tr>
<tr>
<td>89</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>91</td>
</tr>
<tr>
<td>92</td>
</tr>
<tr>
<td>93</td>
</tr>
<tr>
<td>94</td>
</tr>
<tr>
<td>95</td>
</tr>
<tr>
<td>96</td>
</tr>
<tr>
<td>97</td>
</tr>
<tr>
<td>98</td>
</tr>
<tr>
<td>99</td>
</tr>
<tr>
<td>100</td>
</tr>
</tbody>
</table>
3C0 (MORE

ADV •

3C1 (COMP •)

3C2 (MOST

ADV •)

3C3 (SUPER •)

3C4 (QUANT •)

3C5 (MOST (NUMBER PL))

3C6 (FAR

ADV •)

3C7 (FURTHER

ADV •)

3C8 (FURTHEST

ADV •)

3C9 (RECURSIVELY

ADV •)

3CA (WELL

ADV •)

3CB (PLEASE

ADV •)

3CC (SOMETHING

GENPRO •)

3CD (ANYTHING

GENPRO •)

3CE (NOTHING

GENPRO •)

3CF (EVERYTHING

GENPRO •)

3D0 (FIRST

ORD •)

3D1 (SECOND

ORD •)

3D2 (THIRD

ORD •)

3D3 (NEXT

ORD •)

3D4 (LAST

ORD •)

3D5 (TWO

INTEGER •)

3D6 (THREE

INTEGER •)

3D7 (FOUR

INTEGER •)

3D8 (FIVE

INTEGER •)

3D9 (NOT NEG •)

3DA (ISN'T

SUBSTITUTE •)

3DB (AREN'T

SUBSTITUTE •)

3DC (WASN'T

SUBSTITUTE •)

3DD (WEREN'T

SUBSTITUTE •)

3DE (HASN'T

SUBSTITUTE •)

3DF (HADN'T

SUBSTITUTE •)

3E0 (HAVEN'T

SUBSTITUTE •)

3E1 (HASN'T

SUBSTITUTE •)

3E2 (HADN'T

SUBSTITUTE •)

3E3 (HASN'T

SUBSTITUTE •)

3E4 (HADN'T

SUBSTITUTE •)

3E5 (ISN'T

SUBSTITUTE •)

3E6 (AREN'T

SUBSTITUTE •)

3E7 (WASN'T

SUBSTITUTE •)

3E8 (WEREN'T

SUBSTITUTE •)

3E9 (HASN'T

SUBSTITUTE •)

3EA (HADN'T

SUBSTITUTE •)

3EB (HAVEN'T

SUBSTITUTE •)

3EC (HASN'T

SUBSTITUTE •)

3ED (HADN'T

SUBSTITUTE •)

3EE (ISN'T

SUBSTITUTE •)

3EF (AREN'T

SUBSTITUTE •)

3F0 (WASN'T

SUBSTITUTE •)

3F1 (WEREN'T

SUBSTITUTE •)

3F2 (HASN'T

SUBSTITUTE •)

3F3 (HADN'T

SUBSTITUTE •)

3F4 (HAVEN'T

SUBSTITUTE •)

3F5 (HASN'T

SUBSTITUTE •)

3F6 (HADN'T

SUBSTITUTE •)

3F7 (ISN'T

SUBSTITUTE •)

3F8 (AREN'T

SUBSTITUTE •)

3F9 (WASN'T

SUBSTITUTE •)

3FA (WEREN'T

SUBSTITUTE •)

3FB (HASN'T

SUBSTITUTE •)

3FC (HADN'T

SUBSTITUTE •)

3FD (HAVEN'T

SUBSTITUTE •)

3FE (HASN'T

SUBSTITUTE •)

3FF (HADN'T

SUBSTITUTE •)
### APPENDIX 5 - MORPH TABLE

```lisp
; THE MORPH TABLE

1 (IS **)  
2   (NIL N T (POSS))  
3 (NIL N PRT (POSS))  
4 (IS)  
5   (NIL N (INTYPE S) (NUMBER PL))  
6   (NIL V (VBTYPE S-EO) (TNS PRESENT) (POPCODE 3SG))  
7 (IS E)  
8   (T) N (INTYPE ES) (NUMBER PL))  
9   (T) V (VBTYPE ES-ED) (TNS PRESENT) (POPCODE 3SG))  
10 (IS E V)  
11   (IS FI N (INTYPE S) (NUMBER PL))  
12   (FP) N (INTYPE ES) (NUMBER PL))  
13 (IS E)  
14   (NIL N (INTYPE ES) (NUMBER PL))  
15   (NIL V (VBTYPE ES-ED) (TNS PRESENT) (POPCODE 3SG))  
16 (IS N I *)  
17   (NIL V T (PRESPART))  
18 (IS)  
19   (NIL V (VBTYPE S-D) (TNS PAST) (PASTPART))  
20 (IS E *)  
21   (NIL V (VBTYPE S-EO) (VBTYPE ES-ED)) (TNS PAST) (PASTPART))  
22 (IS E)  
23   (NIL V (VBTYPE S-D) (TNS PAST) (PASTPART))  
24 (IS E #)  
25   (NIL V (VBTYPE S-EO) (VBTYPE ES-ED)) (TNS PAST) (PASTPART))  
26 (IS E)  
27   (NIL V (VBTYPE S-D) (TNS PAST) (PASTPART))  
28 (IS E #)  
29   (NIL V (VBTYPE S-EO) (VBTYPE ES-ED)) (TNS PAST) (PASTPART))  
30 (IS E)  
31   (NIL V (VBTYPE S-D) (TNS PAST) (PASTPART))  
32 (IS E #)  
33   (NIL V (VBTYPE S-EO) (VBTYPE ES-ED)) (TNS PAST) (PASTPART))  
34 (IS E)  
35   (NIL V (VBTYPE S-D) (TNS PAST) (PASTPART))  
36 (IS E #)  
37   (NIL V (VBTYPE S-EO) (VBTYPE ES-ED)) (TNS PAST) (PASTPART))  
38 (IS E)  
39   (NIL V (VBTYPE S-D) (TNS PAST) (PASTPART))  
40 (IS E #)  
41   (NIL V (VBTYPE S-EO) (VBTYPE ES-ED)) (TNS PAST) (PASTPART))  
42 (IS E)  
43   (NIL V (VBTYPE S-D) (TNS PAST) (PASTPART))  
44 (IS E #)  
45   (NIL V (VBTYPE S-EO) (VBTYPE ES-ED)) (TNS PAST) (PASTPART))  
46 (IS E)  
47   (NIL V (VBTYPE S-D) (TNS PAST) (PASTPART))  
48 (IS E #)  
49   (NIL V (VBTYPE S-EO) (VBTYPE ES-ED)) (TNS PAST) (PASTPART))  
50 )  
END OF FILE
```
APPENDIX 6 - THE PREPASS ROUTINES

1: THE SYSTEM INITIALIZATION ROUTINES

1 (DEFUN $STOPPRINTS () (REPEAT *EVAL READ) LOOP) )
2 ($STOPPRINTS)
3 (STATUS (47 0))
4 (DEFUN $STOPPRINTS () (UNEVAL $STOPPRINTS **PRINT ON**))
5 (DEFUN READ-DICT (FILE) (PROG (ENTRY)
6 (LET ((ENTRY (READ-RDR FILE))
7 (COND ((EOF RDR))
8 (EOFILE **SOURCE**)
9 ((REAO-DICT ENTRY) ENTRY))
10 (GO LOOP)))
11 (DEFUN READ-GRAMMAR (FILE) (PROG (INPUT)
12 (LET ((INPUT (READ-RDR FILE))
13 (COND ((EOF RDR))
14 (EOFILE **SOURCE**)
15 (EOFILE **SOURCE**)
16 ((REAO-GRAMMAR INPUT) INPUT))
17 (GO LOOP))))
18 (DEFUN READ-MTABLE (FILE)
19 (LET ((CORDER (READ-RDR FILE))
20 (COND ((EOF RDR))
21 (EOFILE **SOURCE**)
22 (EOFILE **SOURCE**)
23 ((REAO-MTABLE CORDER) CORDER))
24 (GO LOOP)))
25 (DEFUN P FEXPR (LIST)
26 (LET ((SENT (PREPASS (CAR LIST)))
27 (PRINT (CAR SENT))
28 (PRINT (PREPASS TIME - (ETIME))
29 (PRINT SENT))
30 (PRINT (PARSE SENT)
31 (PRINT (PARSE TIME - (ETIME))
32 (PRINT (PREPASS FAILURE))))
33 (GO LOOP)))
34 (DEFUN PARSER FEXPR (L)
35 (LET ((SENT (PARSE (CAR L)))
36 (DEFUN INIT ()
37 (LET ((SENT (PREPASS (CAR L)))
38 (PRINT (PREPASS FAILURE))
39 (GO LOOP)))))
40 (DEFUN CHECKWORO (WORD) (PROG (NEWSENT)
41 (LET ((NEWSENT (LIST **START**)))
42 (APPEND NEWSENT)
43 (COND ((EOF RDR)) (EOFILE **SOURCE**)
44 (EOFILE **SOURCE**)
45 ((SETQ SENT (PREPASS NEWSENT))
46 (PRINT (PREPASS TIME - (ETIME))
47 (PRINT NEWSENT))
48 (PRINT (PARSE SENT)
49 (PRINT (PARSE TIME - (ETIME))
50 (PRINT (PREPASS FAILURE))))
51 (GO LOOP)))
52 (DEFUN CHECKWORD (WORD) (PROG (NEWSENT)
53 (LET ((NEWSENT (LIST **START**)))
54 (APPEND NEWSENT)
55 (COND ((EOF RDR)) (EOFILE **SOURCE**)
56 (EOFILE **SOURCE**)
57 ((SETQ SENT (PREPASS NEWSENT))
58 (PRINT (PREPASS TIME - (ETIME))
59 (PRINT NEWSENT))
60 (PRINT (PARSE SENT)
61 (PRINT (PARSE TIME - (ETIME))
62 (PRINT (PREPASS FAILURE))))
63 (GO LOOP)))
64 (DEFUN CHECKWORD (WORD) (PROG (NEWSENT)
65 (LET ((NEWSENT (LIST **START**)))
66 (APPEND NEWSENT)
67 (COND ((EOF RDR)) (EOFILE **SOURCE**)
68 (EOFILE **SOURCE**)
69 ((SETQ SENT (PREPASS NEWSENT))
70 (PRINT (PREPASS TIME - (ETIME))
71 (PRINT NEWSENT))
72 (PRINT (PARSE SENT)
73 (PRINT (PARSE TIME - (ETIME))
74 (PRINT (PREPASS FAILURE))))
75 (GO LOOP)))
76 (DEFUN PREPASS (SENTENCE)
77 (PROG (NEWSENT)
78 (LET ((NEWSENT (LIST **START**)))
79 (APPEND NEWSENT)
80 (COND ((EOF RDR)) (EOFILE **SOURCE**)
81 (EOFILE **SOURCE**)
82 ((SETQ NEWSENT (SUSTITUTE-FIND-CPOMP (CHECKWORD CPOMP)
83 (CHECKWORD SENTENCE))
84 (RETURN NEWSENT)))
85 (RETURN NEWSENT)
86 (RETURN NEWSENT))))
87 (DEFUN PREPASS (SENTENCE)
88 (PROG (NEWSENT)
89 (LET ((NEWSENT (LIST **START**)))
90 (APPEND NEWSENT)
91 (COND ((EOF RDR)) (EOFILE **SOURCE**)
92 (EOFILE **SOURCE**)
93 ((SETQ NEWSENT (SUSTITUTE-FIND-CPOMP (CHECKWORD CPOMP)
94 (CHECKWORD SENTENCE))
95 (RETURN NEWSENT)))
96 (RETURN NEWSENT))))
97 (DEFUN PREPASS (SENTENCE)
98 (PROG (NEWSENT)
99 (LET ((NEWSENT (LIST **START**)))
100 (APPEND NEWSENT)
101 (COND ((EOF RDR)) (EOFILE **SOURCE**)
102 (EOFILE **SOURCE**)
103 ((SETQ NEWSENT (SUSTITUTE-FIND-CPOMP (CHECKWORD CPOMP)
104 (CHECKWORD SENTENCE))
105 (RETURN NEWSENT)))
106 (RETURN NEWSENT))))
107 (DEFUN PREPASS (SENTENCE)
108 (PROG (NEWSENT)
109 (LET ((NEWSENT (LIST **START**)))
110 (APPEND NEWSENT)
111 (COND ((EOF RDR)) (EOFILE **SOURCE**)
112 (EOFILE **SOURCE**)
113 ((SETQ NEWSENT (SUSTITUTE-FIND-CPOMP (CHECKWORD CPOMP)
114 (CHECKWORD SENTENCE))
115 (RETURN NEWSENT)))
116 (RETURN NEWSENT))))
117 (DEFUN PREPASS (SENTENCE)
118 (PROG (NEWSENT)
119 (LET ((NEWSENT (LIST **START**)))
120 (APPEND NEWSENT)
121 (COND ((EOF RDR)) (EOFILE **SOURCE**)
122 (EOFILE **SOURCE**)
123 ((SETQ NEWSENT (SUSTITUTE-FIND-CPOMP (CHECKWORD CPOMP)
124 (CHECKWORD SENTENCE))
125 (RETURN NEWSENT)))
126 (RETURN NEWSENT))))
(defun find-compound (sentence)
  (cond (not (setf entry (get (car sentence) "compound")))
        (return nil)
        ((mapc lambda (entry) t)
          ; REMOVE THIS
          (cond (>(length sentence) (length (car entry)))
                ((equal (nth (reverse sentence) (- (length sentence) (sub (length (car entry)))) (reverse (car entry))))
                 (setq found (cadr entry)
                       sentence (nth sentence (length (car entry))))
                 ; REMOVE THIS
             ) )
        )
    )
)

(defun sublist (list)
  (mapcar (function (lambda (word)
                          (cond (eq (car word) "substitute")
                                (put word '0ct 'dictionary)
                                (repeat 1 and (put word (car clist) (cadr clist)))
                                (setq clist (cond (eq (car type) 's) (atom)
                                                   (*function )
                                                   )
                                )
                           )
                        )
                      )
    )
)

(defun special (type word)
  (mapcar (function (lambda (word)
                      (cond (eq (car word) "substitute")
                            (put word (car clist) (cadr clist))
                            (setq clist (cond (eq (car type) 's) (atom)
                                               (*function )
                                               )
                            )
                      )
                  )
            )
    )
)

(defun notfound (word)
  (prog (new)
    (print (the word (word) is not in the dictionary )
    (enter either a new dictionary entry or nil to cancel )
    (return)
    (cond (atom (setq new (read))) nil)
    (t (enterclist (car new) (cda new)))
    (checkword word )
    )
)

(defun enterclist (word clist)
  (cond (eq (car clist) "substitute")
        (put word (car clist) (dictionary))
        (repeat 1 and (put word (car clist) (cadr clist)))
        (setq clist (cond (eq (car type) 's) (atom)
                           (*function )
                           )
        )
    )
)

; THE MORPHEMIC ANALYSIS ROUTINES

(defun morph? (aword)
  (prog (word ret)
    (setq word (reverse (explode aword)))
    (setq ret )
    )
)

(defun sublist (list)
  (mapcar (function (lambda (word)
                      (cond (eq (car word) "substitute")
                            (put word '0ct 'dictionary)
                            (repeat 1 and (put word (car clist) (cadr clist)))
                            (setq clist (cond (eq (car type) 's) (atom)
                                               (*function )
                                               )
                            )
                      )
                  )
            )
    )
)

(defun special (type word)
  (mapcar (function (lambda (word)
                      (cond (eq (car word) "substitute")
                            (put word (car clist) (cadr clist))
                            (setq clist (cond (eq (car type) 's) (atom)
                                               (*function )
                                               )
                            )
                      )
                  )
            )
    )
)

(defun notfound (word)
  (prog (new)
    (print (the word (word) is not in the dictionary )
    (enter either a new dictionary entry or nil to cancel )
    (return)
    (cond (atom (setq new (read))) nil)
    (t (enterclist (car new) (cda new)))
    (checkword word )
    )
)

(defun enterclist (word clist)
  (cond (eq (car clist) "substitute")
        (put word (car clist) (dictionary))
        (repeat 1 and (put word (car clist) (cadr clist)))
        (setq clist (cond (eq (car type) 's) (atom)
                           (*function )
                           )
        )
    )
)

; THE MORPHEMIC ANALYSIS ROUTINES

(defun morph? (aword)
  (prog (word ret)
    (setq word (reverse (explode aword)))
    (setq ret )
    )
)
(DEFUN SUFFIXTEST (ENTRY) (PROG (STEM FOUND DOUBLE) (COND ((NOT (SETQ STEM (CESTM (CAR ENTRY) WORD))) (RETURN NIL)) (COND (RESULT (ERROR "LISP ERROR IN MORPH?")) ((EQ RESULT 'FOUND) 'WORD) (RETURN NIL))) (DEFUN CETSTEM (SUFFIX WORD) (COND ((NULL SUFFIX) WORD) ((EQ CAR SUFFIX) HAND (GREATERP (LENGTH WORD) 3)) (EQ CAR WORD) (CONSONANT (CAR WORD)) ((NULL WORD) NIL) (EQ (CAR SUFFIX) (CAR WORD)) (GETSTEM (CDR SUFFIX) (CAR WORD)) (T NIL)) (DEFUN MAKE-ENTRY (S-ENTRY) (PROG (TESTWORD ATTRIBUTES TYPE) (SETQ TESTWORD (IMPL ode (APPEND (REVERSE STEM) (CAR S-ENTRY) ) ) ) (SETO ATTRIBUTES (CDR S-ENTRY) ) (SETO TYPE (CADR S-ENTRY) ) (COND ((AND (GET TESTWORD 'N) (GET TESTWORD 'V))) (SETQ FOUND T) (ENTERWORD 'WORD TYPE 'CONSONANT (CAR WORD)) (RETURN T) (T (RETURN NIL))))) (DEFUN NTYPE FEXPR (TYPE) (EQ (CAR TYPE) (GET TESTWORD 'N))) (DEFUN VTYPE FEXPR (TYPE) (EQ (CAR TYPE) (GET TESTWORD 'V))) (DEFUN ADJTYPE FEXPR (TYPE) (EQ (CAR TYPE) (GET TESTWORD 'ADJ))) (DEFUN ADVTYPE FEXPR (TYPE) (EQ (CAR TYPE) (GET TESTWORD 'ADV))) (DEFUN CONSONANT (LL) (MEMO LMBCOFGHJKLM NPQRSTVWX2))) (DEFUN HORPH (WORD) T) (** MISCELLANEOUS LISP ROUTINES **) (NULL (ADDPROP *DEFUN *EXPR *FLAMBOA (DEF)) (PRCG (NAME TYPE) (SETQ NAME (CAR DEF)) (SETQ DEF (CDR DEF)) (SETQ TYPE (SELECT (CAR DEF) (EXPR (SETQ DEF (CDR DEF)) *LAMBOA) (EXPR (SETQ DEF (CDR DEF)) *FLAMBOA) (*NEP (SETQ DEF (CDR DEF)) *LAMBOA) *LAMBOA)) (COND ((GETNAME NAME (EXPR SUBR FSUSR NSUBR)) (PRIN (PRIN NAME)) (PRINT "HAS BEEN PREVIOUSLY DEFINED. NEW DEFINITION WILL BE USED") (PRIN))) (ADDPROP NAME *EXPR 'CONS TYPE DEF) (SETQ DEF (GET 'FUNCTIONS 'FUNCTIONS) *FUNCTIONS) (SCOND (AND DEF (PRINTF NAME DEF)) (T (PRINTF "FUNCTIONS" 'FUNCTIONS) (RETURN NAME)))))
(DEFUN UNDEFUN FEXPR (L) (PRDX L) (DEFUN FEXPR (L) (MAPC 'LAMBDA (FUN) (COND ((SETQ LS (GET FUN 'EXPR)) (PRINT FUN AS AN EXPR AS A REMOVED) (PRINT FUN (CAR LS) ??) (PRINT FUN NO PREVIOUS DEFINITION OF FUN TO REMOVE) ) L ) ) )

(DEFUN VERBUSY?)

(COND ((? (STATUS (X71)) ) ((? (STATUS (X70)) ) ) )

(STATUS (X71) LISPOUT 1201)

(DEFUN PUTPROP (AI SI FLAGI (PUT AI FLAG S))

(PUT *FLAGP 'SUBR (GET *SUBR))

(PUT *GENSYM 'SUBR (GET *GENSYM 'SUBR))

 DEFINE (SCLCCK 'SUBR (LIBRARY SCLOCK))

(DEFUN TRACE FEXPR (L) (PUT L 'BUG (DEFUN TRACEREP (SETO XX) (REM XX 'BUG) ) )

(DEFUN IMplode (L) (READ (IMplodeBUFFER) )

(OPEN (IMplodeBUFFER 1001) )

(DIFUN ERNSET FEXPR (SEXPS) (LIST (EVAL (CAR SEXPS)) )

(DIFUN ERROR FEXPR (VAL) (UNEVAL 'ERRSET (CAR VAL) )

(DIFUN PRINTL FEXPR (L) (TERPRI (MAPC *(LAMBDA (X) (ATOM (PRINL X)) (PRINL (EVAL X))) ) L )

(DIFUN PARSETRACE) (SETQ TRACE (NOT TRACE) )

(DIFUN ETIME) (FIX (TIMES ISCLOCK 5SCLCCK) 1001) )

(DIFUN STIME) (SETQ 5SCLCCK (SCLOCK 0.0)) )

SCONTINUE WITH JESSCLUSION's JESSAUX RETURN

END OF FILE
APPENDIX 7 - THE SEMANTICS ROUTINES

1 ; THE SEMANTICS ROUTINES
2
3 ; THE SENTENCE SEMANTICS ROUTINES
4
5 (DEFUN S-SEMANTICS (SUBJ VERB DO INDO)
6 ; THIS ROUTINE CHECKS THE SEMANTIC AGREEMENT AMONG THE SUBJECT, VERB,
7 ; DIRECT AND INDIRECT OBJECTS IN A SENTENCE.
8 (PROG (RET)
9 (COND (INCT EQ (CADR VERB) 'V3)
10 (EQ (S-V-SEMANTICS SUBJ VERB))
11 (RETURN NIL))
12 (INCT (V-DO-SEMANTICS VERB DO))
13 (RETURN NIL))
14 (EVFEATURE (CADR VERB) 'REDO)
15 (AND (EVFEATURE (CADR INDO) 'REDO))
16 (RETURN NIL)))
17 (COND (EVFEATURE (CADR VERB) 'V3)
18 (AND (EQ (N-OVERLAP (CADR VERB) SUBJ))
19 (RETURN NIL))
20 (RETURN T))))

21 (COND (EQ (CADR VERB) 'V3-SEMANTICS)
22 (PTRACE SPECIAL VERB SEMANTICS TEST #
23 (EQ VERB VERB)
24 (SETQ RET (EVAL (GET (CADR VERB) 'V-SEMANTICS)))
25 (SETQ VAL (SUCCESS?)
26 (PTRACE " " (VAL) - VALUE IS (RET))
27 (RETURN RET))))

28 (RETURN T))

29

30 (DEFUN S-V-SEMANTICS (SUBJ VERB)
31 ; THIS ROUTINE CHECKS THE SEMANTIC AGREEMENT BETWEEN THE SUBJECT AND VERB
32 (PROG (RET VAL)
33 (PTRACE SUBJECT-VERB SEMANTIC AGREEMENT
34 » " SUBJECT « (SUBJ)
35 " VERB = (VERB) "
36 (SETO RET (COND (OR (NULL SUBJ)
37 (NULL VERB) "...
38 NIL)
39 (AND (MEMQ (CAR SUBJ) 'NP)
40 (COND (OR (NULL SUBJ)
41 (MEMQ (CADR SUBJ) 'IN PROD)
42 (COND (ATOM (CADR SUBJ))
43 (INTERSECT (LIST (GET (CADR SUBJ) 'N-TYPE)
44 (GET (CADR VERB) 'SUBJ-TYPE))
45 (GET (CADR VERB) 'SUBJ-TYPE)))))
46 (SETQ VAL (SUCCESS?)
47 (PTRACE " " (VAL) - VALUE IS (RET))
48 (RETURN RET))))

49

50 (DEFUN V-DO-SEMANTICS (VERB DO)
51 ; THIS ROUTINE CHECKS THE SEMANTIC AGREEMENT BETWEEN THE VERB AND DIRECT
52 OBJECT.
53 (PROG (RET VAL)
54 (PTRACE VERB-DIRECT-OBJECT SEMANTIC AGREEMENT #
55 » " VERB = (VERB) "
56 " DIRECT-OBJECT = (DO) "
57 (SETO RET (COND (NULL VERB) NIL)
58 (AND (NULL DO)
59 (OR (EVFEATURE (CADR VERB) 'INTRANS)
60 (EVFEATURE (CADR VERB) 'V3)
61 (NULL DO) NIL)
62 (MEMQ (CADR DO) 'NP)
63 (COND (ATOM (CADR DO))
64 (INTERSECT (LIST (GET (CADR DO) 'N-TYPE)
65 (GET (CADR VERB) 'SUBJ-TYPE))
66 (GET (CADR VERB) 'DO-TYPE)))))
67 (SETQ VAL (SUCCESS?)
68 (PTRACE " " (VAL) - VALUE IS (RET))
69 (RETURN RET))))

70

71 (DEFUN V-INDO-SEMANTICS (VERB INDO)
72 ; THIS ROUTINE CHECKS THE SEMANTIC AGREEMENT BETWEEN THE VERB AND INDIRECT
73 OBJECT.
74 (PROG (RET VAL)
75 (PTRACE VERB-INDIRECT-OBJECT SEMANTIC AGREEMENT #
76 » " VERB = (VERB) "
77 " INDIRECT-OBJECT = (INDO) "
78 (SETO RET (COND (NULL VERB) NIL)
79 (AND (NULL INDO) T)
80 (MEMQ (CADR INDO) 'NP)
81 (COND (ATOM (CADR INDO))
82 (INTERSECT (LIST (GET (CADR INDO) 'N-TYPE)
83 (GET (CADR VERB) 'INDO-TYPE))
84 (GET (CADR VERB) 'INDO-TYPE)))))
85 (SETQ VAL (SUCCESS?)
86 (PTRACE " " (VAL) - VALUE IS (RET))
87 (RETURN RET))

88

89

90
This routine checks the number and case agreement between the grammatical subject and the verb.

This routine checks that each head noun of a compound noun phrase agrees semantically with the verb.

This routine checks that a copula verb has some type of post-verbal modifier present.

This routine checks the semantic agreement between the subject and predicate adjective.

This routine checks that a copula verb has some type of post-verbal modifier present.

This routine checks the semantic agreement between the subject and predicate adjective.

This routine checks the number and case agreement between the grammatical subject and the verb.

This routine checks that each head noun of a compound noun phrase agrees semantically with the verb.

This routine checks that a copula verb has some type of post-verbal modifier present.
\begin{verbatim}
(defun intersect (l)
  ;; This routine first removes the null sublists of L and then produces
  ;; the intersection of the remaining sublists.
  (prog (indic int)
    (mapc (lambda (x) (cond (null x) (seto ll (append ll (list x))))))
    ;;
    (cond (null ll) (return t)
          (null (cdr ll)) (return (car ll)))
    (mapc (lambda (x)
      (cond ((eq x (car ll))
        (setq indic nil)
        (mapc (lambda (sublist)
          (cond ((eq sublist x)
            (seto indic t)
            (uneval "mapc nil")
            ))
          (cdr l)
          (cond (null indic)
            (seto indic nil)
            (uneval "mapc nil")
            )))
        )))
    (car ll))
    (return int))
  )

;; NOUN PHRASE SEMANTICS ROUTINES
(defun np-semantics (det adjs noun)
  (check number agreement and semantic agreement
    (cond (np-number-chk det noun)
          (np-semantic-chk det adjs noun)
          ))
)(defun np-number-chk (det noun 1
  (check number agreement between the determiner and noun
    (prog (ret val)
      (pytrace np number agreement s
        "determiner = " det r
        " noun = " noun s
        " "
        (setq val (success?))
        (pytrace " " val r
          " value is " ret s
          " "
        (return ret))
      )
    )))
(defun np-semantic-chk (det adjs noun)
  (check semantic agreement between the determiner and noun
    (cond (np-semantic-chk det adjs noun)
          ))
)
(DEFUN NP-SEMANTIC-CHECK (DET ADJS NOUN)
  ; THIS ROUTINE CHECKS THE SEMANTIC AGREEMENT BETWEEN THE DETERMINER,
  ; ADJECTIVES, AND THE NOUN IN A NOUN PHRASE.
  (PROG (RET VAL)
    (PTRACE NP SEMANTIC AGREEMENT ;
      (COND ((OR (NULL NOUN)
                  (NULL ADJS)
                  (NOT (EQ (CAR NOUN) 'NIL)))))
      (SETQ RET (INTERSECT (APPEND (LIST (GET (CADR NOUN) 'N-TYPE))
                                 (MAPCAR 'CONSTRUCT ADJS)) (COND ((AND RET
                                                      (EQ (CAR NOUN) NIL)
                                                      (GET (CAAR NOUN) 'N-SEMANTICS))
                                                      (PTRACE SPECIAL NOUN SEMANTICS TEST ;
                                                      (SETQ RET (EVAL (GET (CADR NOUN) 'N-SEMANTICS)))
                                                      (SETQ RET (SUCCESS?))
                                                      (RETURN RET)))))
  )

(DEFUN CONSTRUCT (ADJ)
  ; THIS ROUTINE CONSTRUCTS A LIST OF THE ADJECTIVES' SEMANTIC MARKERS.
  (SELECTC (CAR ADJ)
    (*ADJ (GET (CADR ADJ) 'ADJ-TYPE))
    (+V (GET (CADR ADJ) 'ADJ-TYPE))
    (*PRED (SELECT (CADADR ADJ))
      (*PREDPART *SUBJ-TYPE)
      (*PASTPART *DO-TYPE)
      NIL)
    (NIL)))

END OF FILE
APPENDIX 8 - THE AUXILIARY GRAMMAR ROUTINES

1: THE STRUCTURE-BUILDING ROUTINES

(defun build-s ()
  (setq type
    (buildq (modd) (* 1) neg type))
  (cond (getr voice) (setq c-voice (getr voice)))
  (setq c-s
    (buildq (aux) * (tense aspect))
    (cond (getr modal) (setq aux (getr modal))))
  (setq c-v
    (buildq (iv)) v))
  (setq c-dj
    (cond (getr obj) (list (getr obj))))
  (setq c-vp
    (buildq (a) v obj comp phrasess adverbs))
  (cond (getr s-conj)
    (buildq (s (++ (+)) conj s-sconj)))
  (if (getr s) ))

(defun build-np ()
  (setq det
    (buildq (get) det))
  (setq poss
    (cond (getr poss) (list (getr poss))))
  (setq n
    (cond (getr n) (buildq (n) (n feat))
      (getr proper))))
  (setq nu
    (buildq (nu) nu))
  (setq np
    (buildq (np) (np feat))
    (for nil)
    (buildq (n) (n feat))
    (for super))))
  (setq quant (cond (getr quant) (list (getr quant))))
  (buildq (q (quantp)) ++ ++))
  (setq quant (cond (getr quant) (list (getr quant))))
  (buildq (q (quantp)) ++ ++))
  (setq c-type
    (buildq (s modd) (* 1) neg type))
  (cond (getr voice) (setq c-voice (getr voice)))
  (if (getr c-voice) (getr c-voice))
  (setq c-aux
    (buildq (aux) * (tense aspect))
    (cond (getr conj-passive)
      (setq c-obj (cond (getr obj) (list (getr obj))))
      (setq c-adv adv)
      (for adv))))
  (setq c-adjp
    (cond (getr adj) (add adj (getr adj)))
    (buildq (adj) (* 1) adj))
  (if (getr adj) ))
APPENDIX 9 - THE PARSER

The code appears to be a part of a parser implementation, likely for a programming language. It includes functions for parsing expressions, building expressions, and other related operations. The code is written in a lisp-like syntax, with functions defined using lambda expressions and conditionals. The file appears to be part of a larger text, possibly a book or manual, given the context of the code and the title "THE PARSER".

The code includes functions such as `DEFPROP`, `IFLAMBDA`, `IFPROG`, `IFDEFPROP`, and various other lisp constructs. The functions are likely used to define the parser's behavior, handling cases such as error reporting, expression evaluation, and building of parsed expressions.
(DEFPROP NRC EXPR
(IFLAMBA (D
(IFPROC ISTRINO STATE STACK REGS HOLD)
UNITIALIZE)
(RETURN
(CONO ((NULL STRING) NIL)
(APPLY (QUOTE PROGNX) ICAR D)))
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
201 (INITIALIZE)
202 (RETURN)
203 (COND (NOT (EVAL (CDR L)) NIL))
204 (IT (NAPC (QUOTE EVAL) (CDR L)))
205 (PTRACE JUMP TO (CAR L))
206 (EVAL STRING (CAR L) STACK REGS HOLD (CONS STATE :LEVEL)))
207
208
209 (DEFPROP CAT EXPR)
210 (FLAMBA (L))
211 (EPROG (STRING STATE STACK REGS HOLD FEATURES))
212 (INITIALIZE)
213 (RETURN)
214 (COND (NULL STRING) NIL)
215 (IF (EVAL (QUOTE (CAR L)))
216 (PTRACE ARC (ARC (CAR L)))
217 (APPLY (QUOTE PROGRAM) (CDR L)))))
218
219
220 (DEFPROP PUSHPROGRAM EXPR)
221 (FLAMBA (L))
222 (COND (NULL L) ERR (ERROR))
223 (COND (EPROG (QUOTE (CAR L))) (ERROR (ERROR)))
224 (IF (EVAL STRING (CAR L) STACK REGS HOLD (CONS STATE :LEVEL)))
225 (IF (EVAL (CAR L)) (APPLY (QUOTE PUSHPROGRAM) (CDR L)))))
226
227
228
229 (DEFPROP TRACEPARSE EXPR)
230 (FLAMBA (L))
231 (SETQ (TRACE T))
232
233
234
235 (DEFPROP UNTRACEPARSE EXPR)
236 (FLAMBA (L))
237
238
239 (DEFPROP INIT-TRACE EXPR)
240 (FLAMBA (L))
241 (COND (SETQ (TRACE T) 0)
242 (PRINT (QUOTE SENTENCES))
243 (PRINT SENTENCE)
244 (ERROR)))
245
246
247
248 (DEFPROP PRINTARC EXPR)
249 (FLAMBA (ARCTYPE))
250 (PTRACE TAKING (ARCTYPE (CAR L) ARC)))
251
252
253
254 (DEFPROP PRINTPUSH EXPR)
255 (FLAMBA (L))
256 (SETQ :LEVEL (ADD1 :LEVEL))
257 (COND (PTRACE ABOUT TO PUSH) (SETQ STAB (ADD STAB 2)))))
258
259
260
261 (DEFPROP PRINTPOP EXPR)
262 (FLAMBA (L))
263 (SETQ :LEVEL (SUB1 :LEVEL))
264 (COND (PTRACE ABOUT TO POP - )
265 (SETQ ITAB (SUB :ITAB 2))))
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
(DEFPROP PRINTPARSE EXPR
  (LAMBDA (PARSE TAB)
    (COND ((NULL PARSE) (TERMPRT)
           (PRINTPARSE CAR PARSE) TAB)))
)

(DEFPROP PRI EXPR
  (LAMBDA (PARSE)
    (COND ((NULL PARSE) (TERMPRT)
           (PRINTPARSE CAR PARSE) TAB)))
)

(DEFPROP DISPLAY EXPR
  (LAMBDA (L)
    (SETQ :REGS NIL
e  ; states
    (COND ((NULL (CDR L)) NIL)
    T)))
)

(DEFPROP UNDISPLAY EXPR
  (LAMBDA NIL
    (SETQ :REGS NIL
e  ; states
    (COND ((NULL (CDR L)) NIL)
    T)))
)

(DEFPROP PRINREGS EXPR
  (LAMBDA (L)
    (MAPC *(QUOTE *]
    (LAMBDA (X)
      (PRINT X)
    )
    (TERMPRT)))
)

(DEFPROP ADR EXPR
  (LAMBDA (L)
    (MAPC *(QUOTE *]
    (LAMBDA (X)
      (GETRV (CAR X) X)
    )
    (TERMPRT)))
)

(DEFPROP ACOL EXPR
  (LAMBDA (L)
    (MAPC *(QUOTE *]
    (LAMBDA (X)
      (GETRV (CAR X) X)
    )
    (TERMPRT)))
)

(DEFPROP GETREGS EXPR
  (LAMBDA NIL
    (MAPC *(QUOTE *]
    (LAMBDA (X)
      (GETRV (CAR X))
    )
    (TERMPRT)))
)

(DEFPROP PTREGS EXPR
  (LAMBDA NIL
    (COND ((NULL (CDR L)) NIL)
    T)))

(DEFPROP PTAGIT EXPR
  (LAMBDA (L)
    (MAPC *(QUOTE *]
    (LAMBDA (X)
      (GETRV (CAR X))
    )
    (TERMPRT)))
)
IDEFPROP DICERR EXPR
  (LAMBDA NIL
    (ERROR ((CAR STRING)) CANNOT BE MORPHED))

IDEFPROP INITIALIZE EXPR
  (LAMBDA NIL
    (SETQ STRING
      STATE
      STACK
      REGS
      HOLD)
    (RETURN)
    (COND ((NULL STRING) NIL)
       (T (PRINTARC ARCTYPE)
          (APPLY (QUOTE PROCNX) (CONS L))))

IDEFPROP ERRORS EXPR
  (LAMBDA NIL
    (ERROR ATTEMPT
to
EVALUATE
GETA
ON
NON CAT
ANC
CURRENT
STATE
* (STATE)
* CURRENT
* ANC
* (ARCTYPE)))

IDEFPROP GLUE EXPR
  (LAMBDA (X)
    (COND (EQUAL X NIL)
       (T (APPEND (CAR X) (GLUE (CDR X))))))

IDEFPROP PERROR EXPR
  (LAMBDA (X)
    (COND ((ATOM X) (PRINT (QUOTE "ERROR—"))
       (T (MAPC (QUOTE LAMBDA (Y) (PRINT Y)) (CONS L))))

IDEFPROP SYL EXPR
  (LAMBDA (X)
    (COND ((EQ X (QUOTE 2)) (LAMBDA (Y) (COND ((ATOM Y) (PRINT Y))
       (T (MAPC (QUOTE LAMBDA (Z) (PRINT (QUOTE (EVAL Z)))) (CONS Y))))))))
(DEFPROP PTRACE EXPR (FLAMBDA NIL (COND ((EQL (QUOTE PTRACE) (LIST (QUOTE "TERM"))) actor))))
(DEFPROP PPARSE? EXPR (FLAMBDA NIL (COND ((EQL (QUOTE PPARSE?) (QUOTE (PRINT)))))
(DEFPROP ERRSETI EXPR (FLAMBDA (SEXPSsteller) (LIST (QUOTE ERRSETI) (CAR SEXPSsteller))))
(DEFPROP LIFTCHEI EXPR (FLAMBDA (L) (IPROC (CUT PTR) 3)
  (TAG (COND ((EQL (CUT PTR) NIL) 1)
    (EQ (CUT PTR) PTR) 3)
  (RETURN (APPLY (QUOTE LIFTR) (APPEND L (LIST NIL)))))
  (RETURN (SETQ HEIGHT (FINDHEIGHT (CAR L))))
  (RETURN (SETQ STACK (REPLACE STACK (CAR (NTH STACK HEIGHT)) (CONS (CAR L) (EVAL (CAAR L)))))
  (RETURN (COND ((EQL (QUOTE PTRACE) (QUOTE "TERM")))
    (COND ((EQL (QUOTE PPARSE?) (QUOTE (PRINT)))))
    (COND ((EQL (QUOTE ERRSETI) (QUOTE (PRINT)))))
    (UNTRACEPARSE))))
END CF FILE