CONCEPTUAL ANALYSIS OF NOUN GROUPS IN ENGLISH

Anatole V. Gershman
Department of Computer Science
Yale University
New Haven, Connecticut 06520

Abstract

An expectation-based system, NGP, for parsing English noun groups into the Conceptual Dependency representation is described. The system is a part of ELI (English Language Interpreter) which is used as the front end to several natural language understanding systems and is capable of handling a wide range of sentences of considerable complexity. NGP processes the input from left to right, one word at a time, using linguistic and world knowledge to find the meaning of a noun group. Dictionary entries for individual words contain much of the program's knowledge. In addition, a limited ability for the handling of slightly incorrect sentences and unknown words is incorporated.

0. Introduction

Every natural language processor has to have the ability to interpret noun phrases. This paper describes a set of programs called NGP (Noun Group Processor) which is an integral part of ELI, the English Language Interpreter (Riesbeck and Schank 1976) which serves as the front end to three of the Yale natural language understanding systems, SAM, PAM and WEIS. SAM is a system capable of understanding stories such as various newspaper reports by using scripts (Schank and Abelson 1975, 1977; Cullingford 1975, 1977). PAM is an understanding system which uses general knowledge about peoples' goals and plans (Wilensky 1976). WEIS is a system which understands and classifies a great variety of isolated newspaper headlines on international relations. Thus, our task was to process not only noun phrases of considerable complexity but also to interpret newspaper headlines, which are not always grammatically correct. The following two examples illustrate the kind of sentences our system is able to handle.

1. A CONNECTICUT MAN, JOHN DOE, AGE 23, OF 342 COLLEGE AVENUE, NEW HAVEN WAS PRONOUNCED DEAD AT THE SCENE BY DR. DANA BLAUCHARD, MEDICAL EXAMINER.
2. FUNERAL OF INDIA'S SHASTRI ATTENDED BY USSR KOSYGIN AND USA HUMPHREY.

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract N00014-75-C-1111

To process such a large scope of sentences the program makes extensive use of its knowledge of the problem domain and the redundancy of natural language expressions. This saves effort and permits correct processing of such irregularities of input text as missing commas and articles, or slightly incorrect word order. It also provides for the ability to ignore unknown words or (in some cases) to make plausible interpretations of unknown words. This knowledge is kept in the dictionary. The control mechanisms remain domain independent.

NGP is a production-like system which uses expectations as its basic control mechanism. The problem with every production-like system is the tendency for the accumulation of a large number of expectations fighting for a chance to be tested. In this work I have tried to develop a theory of how various expectations are organized and processed, which, I believe, is in fact a theory of how people process natural language. The basic guiding principle for this theory was its intuitive plausibility.

!• Noun Group Semantics

In this paper we will discuss two classes of noun groups according to the conceptual structures they generate: PP - Picture Producers and CTP - Concept Producers.

PP's are defined by Schank (Schank 1975) as concepts which tend to produce pictures of real world items in the mind of a hearer. For example,

(1) A BIG RED APPLE

is a Picture Producing noun group. To understand such an item means to identify the structure in the memory which corresponds to this item if such a structure exists or to create one according to some frame. This Is done in two stages. In the first stage, we analyze the input phrase and translate it into an expression in Conceptual Dependency (Schank 1972, 1973, 1975). This expression should preserve in a language independent form all the information contained in the surface phrase. Thus (1) will generate

```
(#PHYS0BJ TYPE (APPLE) COLOR (x)
          SIZE (y) DETERM (INDEF)),
```

where x and y are points on the color and size scales. In the second stage, we identify the CD expression with the existing memory structures by performing the necessary memory search and feature matching.

A CD expression for a PP consists of a header followed by a property list. The header is similar to a superset pointer in hierarchically organized memory systems. It points to a frame of properties that the PP is expected to have. The property list explicitly given in the CD expression must be compatible with this frame. Thus a (#PERS0N) is expected to have FIRSTNAME,

LASTNAME, RESIDENCE, etc. but a (#PHYSOBJ) is not. All properties not included in the frame must be specified by a REL clause. For example,

(2) JOHN DOE, THE PASSENGER OF THE CAR

is represented by

```
X: (//PERSON FISTNAME (JOHN) LASTNAME (DOE)
           REL ((<-> ($DRIVE PASSENGER X))))
```

SAM's memory program accepts 7 general classes of PP's: //PERSON, //PHYSOBJ, //ORGANIZATION, //LOCALE, //ROAD, //GROUP, and //POLITY, which can be illustrated by the following examples:

(3) JOHN - (//PERSON FIRSTNAME (JOHN))

(4) TABLE = (//PHYSOBJ TYPE (*TABLE*))

(5) NAVY - (//ORGANIZATION BRANCH (NAVY))

```
(6) 5 FOXON RD  - (//LOCALE STREETNUMBER (5)
                           STREETNAME (FOXON)
                           STREETTYPE (ROAD))

(7) ROUTE 69 = (//ROAD ROADNUMBER (69)
                       ROADTYPE (HIGHWAY))

(8) JOHN AND  - (//GROUP
    MARY              MEMBER (//PERSON FIRSTNAME (JOHN))
                      MEMBER (//PERSON FIRSTNAME (MARY)))
```

(9) USA - (//POLITY TYPE (COUNTRY) NAME (USA))

Very often noun groups do not describe any real world items. Consider the following sentence:

(12) JOHN VOTED IN THE 1976 PRESIDENTIAL ELECTION.

THE 1976 PRESIDENTIAL ELECTION does not produce a single "picture" in the mind of the hearer. Rather, it points to a complicated concept involving the names of the candidates, primaries, voter registration, etc. The knowledge about typical elections is normally organized in a script-like form. The verb VOTED specifies the role John played in the election script. Thus, the meaning of (12) is the invocation of the election script and the instantiation of the script roles. The CD representation of THE 1976 PRESIDENTIAL ELECTION produced by the parser looks as follows:

```
SELECTION TYPE (PRESIDENTIAL) TIME (1976)
                                  REF (DEF)),
```

where $ELECTION is a script name and TYPE and TIME are script parameters. This output is interpreted by the Script Applier. All script names and parameters which appear in the CD expression must be recognizable by the Script Applier.

2* Basic Noun Group Parser

The goal and the general methods of the Noun Group Parser (NGP) are identical to the rest of ELI, i.e. the goal of NGP is the extraction of the conceptualizations that underlie the input. Expectations are its basic mechanisms of operation. (See Riesbeck and Schank 1976). However, the control structure and the order in which the expectations are stored and tested in NGP are very different from those of ELI. To put it briefly, in ELI all the expectations are placed In one pool and are tested whenever a new word or concept is considered. NGP takes advantage of the relatively rigid structure of English noun groups to select and order suitable expectations at each point of the process. The program examines the words of the input string from left to right. The basic loop of the analyzer consists of two steps:

1. The dictionary definition of the current word is loaded into the active memory.
2. Relevant expectations are selected and tested. If an expectation is satisfied, the actions associated with it are executed.

This basic loop is similar to the monitoring control program of ELI or any other production-like system. The difference is in the selection and ordering of expectations. This process is rather complicated and I will try to describe it systematically and in increasingly greater detail throughout the rest of the paper. I will begin by presenting the analysis of a simple example:

(1) LARGE CHINESE RESTAURANT

First, NGP sees the word LARGE. The dictionary definition of LARGE is a program which can test the environment when LARGE is brought into the active memory and build the initial SEMANTIC NODE for it. These semantic nodes (called NGP nodes in the program) are the construction sites where various parts of the future CD expression are being assembled. The node for LARGE, say NGP1, has an expectation attached to it which says "if the next semantic node is an Inanimate PP then attach modifier SIZE (x) to it". NGPI is saved in a stack called MODLIST.

The word CHINESE builds the semantic node NGP2, whose SEMANTIC VALUE is (*CHINA*) and which has an expectation saying "if the next semantic node is a //PHYSOBJ then attach the modifier MADEIN (*CHINA*) to it, if it is a #PERSON or an #ORGANIZATION then attach the modifier PARTOF (*CHINA*) to it". Having done this, the monitor checks the expectation attached to NGPI. It fails and NGP2 is placed on the top of MODLIST.

Next comes the word RESTAURANT. It builds the semantic node NGP3 whose semantic value is (#ORGANIZATION OCCUPATION (RESTAURANT)) and which has an expectation: "if the PREVIOUS semantic node can be a restaurant type then attach it to the current node". Now the monitor goes into the expectation testing mode of operation. It sees

two sets of expectations: those attached to NGP2 looking "forward" at NGP3 and those attached to NGP3 looking "backward" at NGP2. Expectations attached to NGP1 are not considered because NGP1 is hidden by NGP2. First, the monitor tests those expectations of the current node which look "backward" (called BACKWARD in the program). If there are no such expectations or if all of them fail, the monitor tests the "forward" expectations (called FORWARD in the program) attached to the previous semantic node. If an expectation is satisfied, the stack is popped and the process is repeated until no expectations are satisfied. Intuitively, MODLIST contains those modifiers which have not yet been attached. The current node, which is kept in NGAP, is the focus of assembling activities at each step. In our example (*CHINA*) can be a restaurant type, the expectation is satisfied, the value of NGP3 is modified, and NGP2 is removed from MODLIST. The following diagram illustrates the transition:

```
BEFORE: MODLIST - NGP2, NGP1
        NGAP - NGP3
        NGP3 - (//ORGANIZATION
                    OCCUPATION (RESTAURANT))
AFTER:  MODLIST - NGP1
        NGAP - NGP3
        NGP3 - (//ORGANIZATION
                    OCCUPATION (RESTAURANT)
                    TYPE (*CHINA*))
```

Now the monitor sees NGP1 on the top of the stack. Since NGP3 does not have any BACKWARD expectations left, the FORWARD expectation of NGP1 is tested. Note that at this point, NGP3 does not correspond to any particular word, but represents the combined meaning of CHINESE RESTAURANT. LARGE can be attached to NGP3 and the resulting structure is:

```
MODLIST - EMPTY
NGAP - NGP3
NGP3 - #ORGANIZATION  OCCUPATION (RESTAURANT)
                      TYPE (*CHINA*)
                      SIZE (x))
```

So far, we have introduced the following concepts:

SEMANTIC NODES - are the nuclei around which all construction activities are done. The value of a semantic node is a piece of conceptual structure which might be used in assembling the CD expression for the whole noun group.

BACKWARD and FORWARD - are the two groups of expectations attached to a semantic node.

NGAP - holds the current semantic node.

MODLIST - is a stack which holds all previous semantic nodes.

The basic control algorithm of NGP, which was informally described with the help of the above example, now can be stated in more precise terms:

STEP1 Read new word. Execute its definition and put the resulting semantic node in NGAP.

STEP2 If MODLIST is empty then go to STEP7 else go to STEP3.

STEP3 If NGAP does not have any BACKWARD expectations go to STEP5, otherwise go to STEP4.

STEP4 Evaluate BACKWARD expectations of NGAP. In case of failure go to STEP5, otherwise pop the stack and go to STEP2.

STEP5 If the semantic node on the top of MODLIST does not have any FORWARD expectations then go to STEP7, otherwise go to STEP6.

STEP6 Evaluate FORWARD expectations. In case of failure go to STEP7, otherwise pop the stack and go to STEP2.

STEP7 Put the content of NGAP (current semantic node) on MODLIST and go to STEP1.

The underlying assumptions of this algorithm are:

(a) People read noun groups from left to right.

(b) People do not passively accumulate words until they decide that they have reached the head noun. Instead, they make decisions about the interpretations and combinations of words as soon as it becomes possible (I.e. as soon as an expectation is satisfied). Thus, in a phrase MEAT SHOP OWNER, MEAT SHOP is interpreted before OWNER is read.

(c) Expectations attached to words which come later in the phrase usually are stronger than those of preceding words. In the sequence of words of a simple noun group (like FEARLESS CHINESE SOLDIER) words on the left are usually modifiers of some word on the right. A modifier normally has FORWARD expectations for a fairly large class of items it can modify. On the other hand, it is relatively seldom that a word is looking for a particular modifier on its left.

So far, I have carefully avoided one very important problem. My basic control algorithm does not have a STOP statement. Where does a noun group end? This problem is discussed in the next section.

3# The Problem of Boundaries

One problem that any noun group processor has to solve is the problem of boundaries. Where does a noun group end? In most cases the answer to this question is quite simple: things like verbs, commas, prepositions, and articles terminate most noun groups. In practice, however, none of these indicators is very reliable. Consider the following example that NGP had to deal with:

(1) THE U.S. FORCES FIGHT IN VIETNAM IS HOPELESS.

This example illustrates the difficulties arising from the ambiguity of the part of speech classification of the words FORCES and FIGHT. When the context does not provide an early disambiguation we have to make a guess and then later correct it if necessary. As a first guess, NGP collects the maximum number of elements into a noun group. Thus it includes both FORCES and FIGHT rather than stopping after THE U.S.

(2) BILL, JOHN, AND MARY LEFT.

(3) BILL KICKED JOHN, AND MARY KICKED BILL.

BILL, JOHN, AND MARY in the second example constitute one semantic unit -

```
(0GROUP MEMBER (//PERSON FIRSTNAME (BILL))
        MEMBER (//PERSON FIRSTNAME (JOHN))
        MEMBER (#PERSON FIRSTNAME (MARY)))
```

But is it reasonable to consider this phrase as a single noun group on the surface level? Example (3) shows that JOHN, AND MARY might be different groups. Expectations external to the noun group must decide whether these three words can be clustered in one group. The same is true for examples (4) and (5), where the phrase ON THE TRAY may or may not be attached to the noun phrase THE GLASS.

(4) JOHN SAW THE GLASS ON THE TRAY.

(5) JOHN PUT THE GLASS ON THE TRAY.

On the other hand, the preposition OF in the phrase OF STATE in example (6)

(6) U.S. ASSISTANT SECRETARY OF STATE
MARSHALL GREEN

is predicted by the noun SECRETARY, and can be interpreted by the noun group processor without outside help. This brings in the following principle of noun group processing:

ANY UNEXPECTED WORD WHICH IS INCOMPATIBLE WITH THE CURRENT NOUN GROUP TERMINATES THE GROUP ON THE PRECEDING WORD.

Control is returned to the higher level routine which called the noun group and which decides how the group will be used. It might be attached to a preceding noun group or used otherwise.

Semantically, a phrase like

(7) A RECENT YALE GRADUATE, JIM MEEHAN, 27,
ASSISTANT PROFESSOR OF COMPUTER SCIENCE
AT UCI (was awarded ...)

is one PP and, therefore, should be considered one noun group. From the processing point of view, we need a more restricted definition of SURFACE noun groups. A SURFACE NOUN GROUP (or, simply, noun group) is a string of words which can be processed by NGP without relinquishing control to the higher processor.

What are the rules of compatibility which determine the boundaries of a surface noun group? All semantic nodes that can be used in a noun group must belong to one of the following classes: ADJECTIVE, ADVERB, NOUN, TITLE, NAME, NUMBER, DETERM, and BOGUS. (This information is stored on the node under the property MARKER). Class BOGUS is reserved for unknown words and will be discussed later. Class TITLE contains all the words which can be followed by a name: professor, doctor, patrolman, president, etc. The noun group is processed from left to right as long as the following conditions are satisfied:

(1) Each word which is not specifically expected must belong to one of the classes mentioned above.
(2) No word can precede a DETERM.
(3) ADJECTIVES, ADVERBS, and NUMBERS cannot be preceded by either NOUNS, TITLES, or NAMES.
(4) TITLES and NOUNS cannot be preceded by a NAME.
(5) A NAME cannot be immediately preceded by a NOUN.
(6) A NAME cannot be preceded by a DETERM.

For example, phrase (7) will be processed as four separate noun groups:

(a) A RECENT YALE GRADUATE - ends with a comma, but even if this comma were missing, the phrase would have been terminated at the same place by NAME, using rules 5 and 6
(b) JIM MEEHAN - ends with a comma
(c) 27 - special case of a noun group - an age group
(d) ASSISTANT PROFESSOR OF COMPUTER SCIENCE AT UCI - ends with WAS which is a verb

Noun groups OF COMPUTER SCIENCE and AT UCI are processed without leaving NGP since the word PROFESSOR sets up expectations for them.

Rules (1) - (6) are much looser than the usual syntactic rules for noun groups (see, for example, Winograd 1972). But our goal is not the rejection of syntactically incorrect sentences. We introduce restrictions only where they help, where their absence creates disambiguation or processing difficulties.

The other distinctive feature of our rules is that they are generated dynamically and can be changed by actions of any expectation. This is how, for example, possessives are handled:

(8) POLICE CHIEF'S NEW CAR

First, the node for POLICE CHIEF is build:

```
NGP1:
 VALUE  - (#PERSON OCCUPATION (POLICE-CHIEF))
 MARKER - TITLE
```

Then the program sees the possession mark which satisfies a special default expectation. The action of this expectation transforms NGP1 into:

```
NGP1:
 VALUE  - (#PERSON OCCUPATION (POLICE-CHIEF))
 MARKER - ADJECTIVE
 FORWARD - "If the next node is a #PHYSOBJ then
   make it POSSBY the value of NGPI (i.e. by
   (^PERSON OCCUPATION (POLICE-CHIEF)))"
```

4• Putting Pieces Together

In the previous section I described the basic noun group processor. Complex noun groups are broken into simpler phrases which are processed separately. Separately, however, does not mean independently. The previously built part of the noun group can affect the analysis of the remaining parts. In this section I will describe the mechanism of this interaction and how various parts of a noun group are put together.

In accordance with our general principles, this process is driven by a hierarchically organized set of expectations. There are two kinds of expectations: (1) those dynamically generated by the input and (2) default expectations supplied by the control mechanism. These default expectations are designed to catch such unexpected things as appositives, addresses, age groups, etc. For example, when we hear A CONNECTICUT MAN in

```
(1) (The award was given to)
    A CONNECTICUT MAN, JOHN DOE, AGE 23,
    OF 234 COLLEGE AVENUE, NEW HAVEN.
```

we do not necessarily immediately expect to hear his name, age, and address, although we know that as a person he has these characteristics. These are secondary, default expectations which are tested only if other, explicit expectations fail. In the above example the processing goes as follows. First, A CONNECTICUT MAN is collected, generating:

```
(2) (#PERSON GENDER (MALE)
          RESIDENCE (//LOCALE STATE (*CONN*)))
```

At this point, control returns to ELI which tests the expectations which were pending before we reached this phrase. One of these expectations is satisfied and its action puts structure (2) into the waiting slot in a larger frame:

```
(ACTOR (NIL) <-> (*ATRANS*) OBJECT (*AWARD*)
  TO (#PERSON GENDER (MALE)
        RESIDENCE (#/LOCALE STATE (*CONN*))))
```

The slot that (2) filled is remembered in the variable called LASTNG. Then comes JOHN DOE. No explicit expectations are satisfied. The monitor goes to a special mode called TRAP. TRAP checks whether LASTNG was a person and, if so, checks the default expectations about a person. The NAME expectation is satisfied and the specialized action which collects personal names is executed. As a result name modifiers are attached to the male Connecticut resident:

```
(#PERSON GENDER (MALE)
         RESIDENCE (#LOCALE STATE (*CONN*))
         FIRSTNAME (JOHN) LASTNAME (DOE))
```

After this, control goes back to the top level processor. This reads the next word, "27". Again, no expectations are immediately satisfied and the monitor traps into the secondary expectations. The AGE expectation is satisfied and the specialized action which collects AGE specification groups is executed. The result is an AGE modifier which is attached to .John. OF 234 COLLEGE AVENUE also goes to TRAP, which calls the address group processor. The final result is:

```
(#PERSON GENDER (MALE)
   RESIDENCE (#LOCALE STATE (*CONN*)
                      STREETNUMBER (234)
                      STREETNAME (COLLEGE AVENUE))
   FIRSTNAME (JOHN) LASTNAME (DOE))
```

The following example illustrates a slightly different problem:

```
(3) LOUIS CAPPIELLO, YALE POLICE CHIEF
```

In order to figure out that being a YALE POLICE CHIEF is LOUIS CAPPIELLO's occupation we first have to collect both noun groups. This is done with the help of another secondary expectation called EXTRA-NOUNGR trap. LOUIS CAPPIELLO generates:

```
(//PERSON FIRSTNAME (LOUIS) LASTNAME (CAPPIELLO))
```

YALE POLICE CHIEF generates:

```
(//PERSON OCCUPATION (YALE-POLICE-CHIEF))
```

Then another secondary expectation tests to see if LASTNG and EXTRANG could be the same thing. If so, the two groups are merged.

Appositives can be arbitrarily complex: from simple name groups to complicated prepositional phrases and relative clauses. Very rarely are they explicitly expected. They are handled by the secondary expectations based on the general properties of things and the knowledge about the ways these things can be expressed in English. TRAP represents an attempt to implement the mechanism controlling the interaction between these expectations.

TRAP is still in the experimental stage of development. Its flow of control is rather complex. In general, first, it tries to find and test expectations about general properties of the item in LASTNG. For example, for a person it tries to collect special modifiers such as name, age, and address. If all these expectations fail, TRAP checks for possible appositives such as simple EXTRA noun groups, prepositional phrases, or relative subclauses. If one of these appositives is collected, TRAP first checks the explicit expectations which may have been pending (for example, a WHICH-clause might want to be attached to a particular physical object) and then checks the secondary expectations again. This time, it may catch some properties which it missed the first time because they were encoded in a more complicated form. In order to clarify this description let us follow a few more examples:

```
(3) JOHN DOE OF GENERAL MOTORS
```

The subgroup OF GENERAL MOTORS is caught by TRAP's prepositional phrase expectation. Since there are no specific expectations which can link JOHN DOE and GENERAL MOTORS, the default one, attached to OF is checked. Its action links the two groups as follows:

```
(^PERSON FIRSTNAME (JOHN) LASTNAME (DOE)
         SOMEREL ^ORGANIZATION
                 ORGNAME (GENERAL-MOTORS)))
```

SOMEREL means that we do not really know the

exact nature of the relations between JOHN DOE and GENERAL MOTORS.

In the following example

(4) US NAVY TASK FORCE WHICH HAS BEEN ON PATROL DUTY IN THE INDIAN OCEAN (left the area)

the WHICH clause is collected by TRAP's subclause expectation and is attached to US NAVY TASK FORCE by an expectation associated with WHICH. The result is:

```
X: (#GR-ORG PARTOF
     (#ORGANIZATION BRANCH (NAVY) PARTOF (*USA*))
     REL ((ACTOR X
           <-> ($PATROL PLACE (*INDIAN-OCEAN*)))))
```

Subclause processing represents a difficult problem on its own. The problem of subclause boundaries, for example, is as complex as that of noun groups. In solving it, I used the same philosophy as for noun groups boundaries: the current subclause is finished when the next word is not expected by any expectations from that subclause.

The traditional stumbling block of all parsers - AND conjunction - is also handled by a series of TRAP expectations. Although, in difficult cases we cannot avoid backtracking, simple cases like

(5) JOHN AND MARY ATE SOUP AND LASAGNA AND LEFT.

can be processed by the program with the help of the following heuristics. If AND is not specifically expected and occurs in the sentence between two noun groups which can be combined in one semantic unit then it is interpreted as a link between the two noun groups. Otherwise, if AND occurs in the sentence after the verb it is Interpreted as a link between two clauses.

All examples presented so far deal with noun groups describing Picture Producers. The next example shows how Concept Producers are handled.

(6) (Castro condemned) THE EXECUTION OF THOUSANDS OF COMMUNISTS IN INDONESIA.

THE EXECUTION refers to the script $EXECUTION. This script has among its roles the VICTIM of the execution. Among the expectations associated with the script there is one which expects the victim to be a person (or a group of people) introduced by the preposition OF. Hearing the word EXECUTION sets up an expectation for the word OF (someone). THOUSANDS OF is another unit which creates a group whose members follow. This expectation is satisfied by COMMUNISTS. When IN INDONESIA comes it is not expected by anybody. Hence, the noun group collection is suspended and THE EXECUTION which is now transformed into:

```
($EXECUTION VICTIM (//GROUP MEMBER
          (#PERSON OCCUPATION (COMMUNIST)
                   COMPNUM (ORDER VAL (1000)))))
```

is placed in the MOBJECT slot of MTRANS for "condemned". After this, IN INDONESIA is collected:

(LOC VAL (*INSIDE* PARTOF ("INDONESIA)))

Now the processor must decide whether Indonesia was the place where the execution occurred or where it was condemned by Castro. In the absence of other expectations, the program picks the first alternative.

To conclude this section, I would like to discuss the treatment of words unknown to the program. People have a limited ability to interpret such words from context, or, at least, to ignore them. We tried to put some of this kind of intelligence in our programs. The problem has two aspects. First, we have to figure out what role the unknown word (or words) might play in the sentence and then interrogate the context to find out what meaning this word might have. The borderline between these two tasks is very vague. As of now, most of the first part is handled by NGP and most of the second part by Rick Granger's program called FOUL-UP (Granger 1977). The following examples illustrate how the NGP part works.

(7) JOHN ATE A F00 FISH.

FOO is interpreted as an unknown modifier and ignored.

(8) JOHN ATE A BLUE FOO.

The output of NGP

(//BOGUS COLOR (BLUE) LEXVAL (FOO) REF (1NDEF))

is handed to FOUL-UP for further investigation.

(9) DR FOO BAZ ATE A BLUE FISH.

FOO BAZ are interpreted as the first and the last names of a person whose occupation is DOCTOR.

(10) FOO'S FISH WAS BAD.

FOO is interpreted as the last name unless (9) and (10) occurred in the same story, in which case FOO would have already been defined as a first name.

(11) JOHN WAS TAKEN TO THE HOSPITAL BY FOO AMBULANCE.

FOO is interpreted to be a name of an ambulance company, since AMBULANCE has a BACKWARD expectation looking for a company name.

(12) 593 FOO BAZ AVENUE

FOO BAZ is interpreted as the name of an avenue.

5* Comparison with other Work and Conclusions

The work, presented in this paper is a further development of ELI. The main difference between this program and most other parsers (see, for example, Winograd 1972, Woods and Kaplan 1971) is that it does not separate its linguistic knowledge from its general world knowledge. In other programs the analysis is done in two stages. First the input is analyzed syntactically and then the result is interpreted semantically. For example, LUNAR (Woods and Kaplan 1971) uses the Augmented Transition Network Grammar (Woods 1970) to generate possible syntactic interpretations of a given sentence and then applies its domain knowledge to determine whether the interpretation is meaningful. Thus, noun groups are parsed purely syntactically and their meaning is not established until the whole sentence is parsed. In each noun group the first noun is assumed to be the head noun. If later this turns out to be incorrect, the system backs up and tries to accumulate more elements into the noun group. For example, the correct processing of the phrase PRESIDENT JIMMY CARTER which contains three nouns will require LUNAR to back up twice. This means that a great deal of unnecessary effort is spent in finding syntactically plausible but meaningless parses. This is especially true when one tries to relax some syntactic rules to allow for slightly incorrect sentences. In NGP the parsing is done by the use of rules most appropriate in a given situation, semantic or syntactic. Thus, in the example above, the programs contained in the dictionary entry for the word PRESIDENT will immediately collect JIMMY CARTER. Most of the program's linguistic knowledge is not built into its control structure but stored in the dictionaries and used as a part of its general knowledge. This makes the program very flexible, easily extensible, and provides for the correct processing of "ungrammatical" sentences.

Another important difference between this program and both Winograd's and the LUNAR system is In the representation of meaning. The meaning of a sentence in Winograd's system is a program for manipulating blocks. The meaning of a sentence in the LUNAR system is a request for information about some properties of the rocks from the Moon. Both these systems are very specialized and not easily extensible to other domains. Our analyzer is based on the Conceptual Dependency representation system which is not limited to any particular domain. The same program can handle a wide variety of topics, from car accident reports to state visits to China.

The results presented in this paper show that both linguistic and world knowledge are required for correct and efficient handling of noun groups. The program demonstrates the possibility and the advantages of the simultaneous application of both kinds of knowledge, without separating the process of understanding into syntactic and semantic stages. The program provides an intuitively plausible model for a hierarchically organized, expectation based control mechanism for analyzing noun groups.

6. References

I] Cullingford, R.E. (1975). An Approach to the Representation of Mundane World Knowledge: The Generation and Management of Situational Scripts. American Journal of Computational Linguistics. Microfiche 44.

2] Cullingford, R.E. (1977). Organizing World Knowledge for Story Understanding by Computer. Unpublished Doctoral Dissertation. Department of Engineering and Applied Science, Yale University.

3] Granger, R.H. (1977). FOUL-UP. Paper to be presented at the 5-th International Joint Conference on Artificial Intelligence. Cambridge, Massachusetts.

4] Riesbeck, C.K. and Schank, R.C. (1976). Comprehension by Computer: Expectation-based Analysis of Sentences in Context. Yale Dept. of Comp. Sci. Research Report #78.

5] Schank, R.C. (1972). Conceptual Dependency: A Theory of Natural Language Understanding. Cognitive Psychology 3(4):552-631, 1972.

6] Schank, R.C. (1973). Identification of Conceptualizations Underlying Natural Language. In R. C. Schank and K. Colby, eds. Computer Models of Human Thought and Language. W. H. Freeman, San Francisco.

7] Schank, R.C. et al. (1975). Conceptual Information Processing. North Holland, Amsterdam.

8] Schank, R.C. and Abelson, R.P. (1975). Scripts, Plans, and Knowledge. Proceedings of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR.

9] Schank, R.C. and Abelson, R.P. (1977). Scripts, Plans, and Understanding. Lawrence Erlbaum Associates, Hillsdale, N.J.

10] Wilensky, R. (1976). Machine Understanding of Human Intentionality. Proceedings of the ACM Annual Conference. Houston, Texas.

II] Winograd, T. (1972). Understanding Natural Language. Academic Press, New York.

12] Woods, W.A. (1970). Transition Network Grammars for Natural Language Analysis. Comm. ACM 13(10):591-606, 1970.

13] Woods, W.A. and Kaplan, R.M. (1971). The Lunar Sciences Natural Language Information System. BBN Report No. 2265. Bolt Beranek and Newman Inc. Cambridge, Massachusetts.