COMPUTER GENERATION OF NATURAL LANGUAGE FROM A DEEP CONCEPTUAL BASE

Neil Murray Goldman

Stanford University

Prepared for:

Advanced Research Projects Agency
National Science Foundation

January 1974

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER: STAN-CS-74-461	2 GOVT ACCESSION NO.
3 RECIPIENT'S CATALOG NUMBER: AD/A-005941	
4 TITLE (and Subtitle): COMPUTER GENERATION OF NATURAL LANGUAGE FROM A DEEP CONCEPTUAL BASE.	5 TYPE OF REPORT & PERIOD COVERED: technical, January, 1974
	6 PERFORMING ORG. REPORT NUMBER: STAN-CS-74-461 (AIM 247)
7 AUTHOR(s): Neil Murray Goldman	8 CONTRACT OR GRANT NUMBER(s): DAHC 15-73-C-0435
9 PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Department, Stanford University, Stanford, California 94305	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS: ARPA/IPT, Attn.: Stephen D. Crocker, 1400 Wilson Blvd., Arlington, Va. 22209	12 REPORT DATE: JANUARY 1974
	13 NUMBER OF PAGES: 321
14 MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): ONR Representative: Philip Surra, Durand Aeronautics Bldg., Rm. 165, Stanford University, Stanford, California 94305	15 SECURITY CLASS (of this report): Unclassified
	15a DECLASSIFICATION/DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Releasable without limitations on dissemination.

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

PRICES SUBJECT TO CHANGE

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

For many tasks involving communication between humans and computers it is necessary for the machine to produce as well as understand natural language. We describe an implemented system which generates English sentences from Conceptual Dependency networks, which ar unambiguous, language-free representations of meaning. The system is designed to be task independent and thus capable of providing the language generation mechanism for such diverse problem areas as question answering, machine translation, and interviewing.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Stanford Artificial Intelligence Laboratory
Memo AIM-247

JANUARY 1974

Computer Science Department
Report No. STAN-CS-74-461

Computer Generation of Natural Language From a Deep Conceptual Base

by

Neil Murray Goldman

ABSTRACT

For many tasks involving communication between humans and computers it is necessary for the machine to produce as well as understand natural language We describe an implemented system which generates English sentences from Conceptual Dependency networks, which are unambiguous, language-free representations of meaning The system is designed to be task independent and thus capable of providing the language generation mechanism for such diverse problem areas as question answering, machine translation, and interviewing

A dissertation submitted to the Department of Computer Science and the Committee on Graduate Studies of Stanford University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC 15 73 C 0435 and the National Science Foundation under Contract NSF GJ42906 The views and conclusions contained in this document are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Stanford University, ARPA, nsf, or the U S Government

Reproduced in the U S A Available from the National Technical Information Service, Springfield, Virginia 22151

COMPUTER GENERATION OF NATURAL LANGUAGE FROM A DEEP CONCEPTUAL BASE

Neil Murray Goldman, Ph.D.
Stanford University, 1974

For many tasks involving communication between humans and computers it is necessary for the machine to produce as well as understand natural language. We describe an implemented system which generates English sentences from Conceptual Dependency networks, which are unambiguous, language-free representations of meaning. The system is designed to be task independent and thus capable of providing the language generation mechanism for such diverse problem areas as question answering, machine translation, and interviewing.

The meaning representation which is our starting point contains neither words nor English syntax. Thus selecting words and placing a syntactic structure on the selected words is a major problem to be solved. Because of the language-free nature of the representation, this cannot generally be done by associations between meaning elements and words. Nor can pieces of the meaning structure simply be replaced by words, since the meaning relations have no direct correspondence to any useful relations such as verb-object between English words.

To encode meanings into English both language-independent and language-specific knowledge are required. The former is provided by an already existing memory-inference model. Knowledge of conceptual categories, time relations, idiosyncratic beliefs, and contexts set up by previous language processing or inference may all affect the words and syntactic structures selected. It is shown that a wide variety of world knowledge is needed for language generation. Unlike analysis, where such information is used for disambiguation, generation uses this knowledge for determining appropriateness of words and linguistic relationships.

There are several sources of English-specific knowledge. Discrimination networks permit efficient retrieval of words which express complex meaning relationships and interaction with the memory model. Information associated with word senses provides a method for mapping language-independent meaning relationships into language-dependent syntactic relationships. This knowledge is used to make predictions which guide the construction of an intermediate structure, called a syntax net. This net is neither unambiguous nor language-free. To deal with grammaticality, a formal grammar is incorporated. This grammar describes those aspects of surface English syntax which are required to complement the model's vocabulary and conceptual domain. The final sentence generated is a result of a 'linearization' of the syntax net by the grammar.

Many paraphrases can be generated from single meaning representations. The members of these sets are not syntactic paraphrases of one another, but quite different ways of expressing an underlying meaning. The basic processes and data structures of the system provide an alternative to previously proposed models for language generation. Such an alternative model was necessitated by the use of a language-free meaning representation. Some of the reasons for employing such a representation in computer programs are considered. Many features of language are not dealt with by this system, and some desired extensions are discussed.

ACKNOWLEDGEMENTS

I would like to acknowledge my indebtedness to Professor Roger Schank, who served as my adviser in this research. It was Dr. Schank who first interested me in computational linguistics, and it was his own work which provided the foundation upon which my own work has been built.

I would also like to thank Professors Jerome Feldman and Cordell Green for serving on my reading committee. Their time spent reading the manuscript and their suggestions are greatly appreciated.

The ideas which make up this thesis were not developed in isolation, and I thank all those students who have collaborated in the work on natural language processes at the Stanford Artificial Intelligence Laboratory. In particular, I would like to thank Charles Rieger and Chris Riesbeck for many valuable discussions during the past two years.

My further thanks go to the Istituto per gli Studi Semantici e Cognitivi in Lugano, Switzerland for providing the time and facilities which enabled me to complete the writing of this thesis.

I would certainly be remiss were I not to acknowledge the debt I owe my parents, who instilled in me my very first

notion of "performance" grammar. They have long since given up debugging their creation, and should not be held liable for those "competence" flaws which remain.

PREFACE

To look through a large text in search of some small portion of interest is not an inviting task for any reader. To those who were not deterred from opening this thesis by its sheer bulk, I give

i) my thanks, and

ii) this preface, in the belief that a few paragraphs of outline are worth more than the best Tables of Contents and Lists of Illustrations. It is hoped that a few minutes spent reading this preface will save much time later, both in finding material which is of interest and discarding that which is not.

The INTRODUCTION provides a brief history of attempts at mechanized language processing, stressing those aspects of natural language which have been particularly troublesome. It requires no knowledge of linguistics or computer science to be understood, but will provide no new insights to those moderately well-versed in the problems of computational linguistics.

Chapter 1 presents a basic approach to language processing which has been adopted in this work. It delimits that portion of the problem which we call GENERATION, and provides examples of computer generated English produced by our model.

Chapter 2 describes several approaches to mechanical language generation which have been previously implemented. Of course not all work in the area could be included; we have tried to present a fair cross-section of work, with particular emphasis on those ideas which influenced this research. The discussion is limited to MACHINE generation; no attempt is made to cover the literature of generative grammar, as developed by theoretical linguistics over the past decade.

Chapter 3 is devoted to the fundamentals of Conceptual Dependency representation. Those familiar with the literature in this theory <30, 31, 32> may wish to skip this chapter or merely skim the material in it. For others, a more thorough reading will be necessary in order to understand the material which follows.

Chapter 4 discusses the considerations which come into play when one is faced with the problem of generating language from meaning. We describe very generally a process which produces English sentences from conceptual structures. This chapter thus provides an overview of the material presented in the following two chapters. It should be sufficient to give a basic, if somewhat crude, understanding of 'conceptual generation'.

In order to produce English sentences from their meanings, several distinct sorts of knowledge are required.

Some of this is linguistic in nature; e.g., how words and meanings are related, the notion of the syntax of a natural language. Other information is not linguistic in nature, but concerns world knowledge and beliefs. In Chapter we detail this knowledge and its organization in our model.

Chapter 6 describes the process which utilizes this knowledge to produce natural language sentences from meaning representations. We show how a simple refinement of the process enables the program to produce paraphrases by finding different natural language encodings of a single meaning.

In Chapter 7 an extension to the implemented program is described. The extension concerns the generation of nouns which describe events, and demonstrates the need for a new form of interaction between previously separate parts of the generative process.

Chapter 8 contains a comparison of conceptual with non-conceptual representations, and presents some theoretical arguments favoring conceptual representations for certain tasks. The thesis is concluded with a brief summary and a look at some important problems which are not handled adequately (or at all) by current theories.

TABLE OF CONTENTS

INTRODUCTION

> "And the Lord came down to see the city and the tower, which the sons of men had built. And the Lord said, "Behold, they are one people, and they have all one language; and this is only the beginning of what they will do; and nothing that they propose to do will now be impossible for them. Come, let us go down, and there confuse their language, that they may not understand one another's speech. So the Lord scattered them abroad from there over the face of all the earth, and they left off building the city. Therefore its name was called Babel, because there the Lord confused the language of all the earth . . ."

Thus the Bible explains the origins of the world's many languages. It would appear that the Lord's efforts were in vain -- the sons of man have been little dissuaded from highrise building (and other evildoing) by lack of communication. But the job of creating a confusing set of languages was indeed masterfully done.

Through the years man has remained fascinated by language. He has studied its origins and development. He has shown that an individual can, with a moderate amount of effort, learn to communicate in more than one language, thus making language less of a hindrance to him.

When the digital computer came into widespread use, and its potential as a general symbol processor was realized, it was only natural to try to teach it to deal with human languages.

The first human, or 'natural' language problem to which computers were seriously applied was translation. The approach used was to read sentences (via a teletype or punched cards) in language L, and produce a sentence by sentence translation in language M. But despite much persistence and many varied attempts, the translation problem remained the domain of man and not the machine.

Since one of the prime difficulties in translation seemed to be the lack of one to one correspondence between the words of one language and those of another, it was hoped that language tasks involving only a single language might be more easily solved. Could a computer, for instance, take in information expressed in English and later answer questions about that information? Or could the computer's vast memory be used like an encyclopedia to store information, and the machine then commanded to retrieve all it 'knew' about a given subject from this store?

The results of work on 'uni-lingual' problems, like the work on translation, failed to justify early hopes. But in this work it was seen that the difficulties which proved to be the ultimate stumbling blocks were the same ones which had stymied the machine translators. The confusion in human language lies not in the multitude of human languages, but in the nature of language itself.

More particularly, all the early programs ran into trouble because they failed to 'understand' the language they were dealing with <45>. In some cases the ambiguity of language caused problems. A translation program could not translate "Smith went to the dentist for a gum infection" into another language without first understanding the English. Here 'understanding' includes recognizing "gum infection" as a medical problem with a part of the mouth. Without this level of understanding the phrase could be read as analogous to "virus infection" (a medical problem caused by a virus) and thus translated into something like "a medical problem caused by a stick of Juicy Fruit".

In other cases, the many-to-one relationship between language forms and meanings is the obstacle. A question answerer, having been told

"Brutus and his cohorts killed Caesar by stabbing him"

might easily be expected to answer the question

"Who killed Caesar?"

A human could answer this question equally well having been told

"Brutus and his cohorts stabbed Caesar to death"

because he 'understands' the relationship between "killing" and "stabbing to death". But how can we make the <u>computer</u> see this relationship?

Our goal is to make the computer use natural language in human-like ways.

The problems machines have had to date with natural language emphasize that, while we know how to use language, we don't yet understand how it works. At least two paths to our goal might be tried. Perhaps using language is like driving a car -- inferring the operations needed to drive from the mechanical design of the automobile is, at best, an indirect approach to learning how to drive. If this is the case, our efforts should be directed toward finding heuristics to deal with immediate problems, setting aside questions of underlying language theory.

On the other hand, perhaps the problem is more analogous to building the car from a roomful of parts. Unless the principles of operation are understood, the chances of stumbling across the right sequence of moves to get it all together are rather dim.

Both approaches have been, and are being, investigated. At one extreme are approaches which focus on making the computer accomplish a particular task, employing whatever heuristics appear to help when obstacles arise. At the opposite end of the spectrum are approaches which ignore both specific tasks and computational methods, focussing only on the formal properties of language itself.

The model of language processing incorporated in our program, like many other models, lies somewhere between these extremes.

We have tried to avoid two major pitfalls of the extremes. Task oriented approaches run the risk of finding solutions which fail to generalize to new problems. The prime cause of this seems to be the tendency to continually redefine the task domain, narrowing it in order to eliminate particularly sticky language problems.

Statistical 'word crunching' -- making decisions based on frequency counts of words and word stems in text -- is an approach to information retrieval which displays this fault. Interesting, and even, useful, results can be obtained as long as the data base is appropriately limited. But problems arise if the domain widens. And the techniques used do not appear even minimally relevant to other tasks, such as machine translation.

The second danger we try to avoid is that inherent in a pure linguistic approach. In ignoring the computer and particular tasks, in trying to separate language from its use, attention too often becomes focussed on the question "What strings of symbols constitute the language?" This question is a difficult one; in fact, it is not even well defined. However, it is not a question which arises in any of the tasks we would like computers to deal with. A process which could answer this question might well contain subprocesses useful in a performance program. There is no guarantee of this, however.

We do not believe that people have specialized ways of dealing with language for each of the problems they face. We don't believe the computer should do this either. Our goal is to find general language processing techniques with a wide range of applicability. In the next chapter we introduce a model based on such techniques, briefly discuss each of its components, and define language generation, the main topic of this thesis.

CHAPTER 1

LANGUAGE PROCESSING AND THE ROLE OF GENERATION

What sorts of natural language tasks would we like computers to deal with? Several have been proposed:

1) Machine Translation (MT) -- It would be useful to have machines which could read scientific documents, newspaper articles, novels, etc., and translate them into other languages.

2) Information Retrieval -- The computer would have access to a large body of information on some subject and find that portion of it relevant to a specific topic. For example, it might be used like a law library to help a lawyer find precedents for a case.

3) Information Summaries -- similar to (2), but the computer would summarize the relevant information which it was able to find. Humans demonstrate the ability to summarize in preparing abstracts for articles and in headline writing (at least in those cases in which headlines are used as an indication of article content).

4) Question Answering (QA) -- The machine would answer specific questions about its data base. A newcomer to a computer center could sit down at a terminal and find out how to get an account, how to log on, how to edit files, etc. by typing queries to the computer in English.

c) Medical Interviewing -- A machine could take a patient's medical history and conduct an initial interview to compile lists of symptoms and other standard information.

d) Computer Aided Instruction (CAI) -- Computers are already being used to aid classroom instruction. But the student who uses such a machine today must mold his answers and questions to its limited language handling capabilities. Natural student-teacher interactions are not yet possible.

e) Home Terminals -- McCarthy <21> has suggested that serious consideration be given to supplying the public with home access to computer stored information. No more telephone books, TV Guides, bus schedules, recipe books, etc. cluttering up the house. Many specialized question answering and information retrieval programs could be a part of such a system. A great many simple things could be done with little use of natural language by the computer. But, in the long run, if tens of millions of people are to be communicating with computers this way, it would be preferable to have command and response languages which were much like English and thus required little training of the users.

1.1 Basic Components

Three basic mechanisms are involved in these tasks. One is language analysis, which maps surface language strings into some other form which we shall call their 'underlying

representation'. A second process is language generation, which maps 'underlying representations' into surface strings. Finally, there are cognitive processes, which operate on the result of language analysis and produce material for language generation.

THE DIFFICULTY OF LANGUAGE GENERATION AND ANALYSIS, as defined here, IS HEAVILY DEPENDENT ON THE NATURE OF THE UNDERLYING REPRESENTATION. The closer this representation is to natural language, the easier will be the task of generating language from the representation. Within this thesis we shall discuss several possible forms for this representation, and show how the representations which make analysis and generation simpler tend to make cognitive processing more difficult.

We shall present and work with a model which employs a conceptual underlying representation. The notion of conceptual representation will be explained in detail in Chapter 3. For now we may just think of it as a representation of meaning abstracted from natural language. This conceptual representation is designed to facilitate the processing of meanings rather their derivation from or expression in natural language.

Let us digress for a moment to discuss some terminology which might make the notion of conceptual representations clearer. We shall frequently have occasion to refer to

the syntax of natural language. For our purposes, the most important aspect of syntax is surface syntax; in particular, constituent structure -- the grouping of the words of a sentence into units which grammarians call noun phrases, verb phrases, clauses, etc. Syntax also covers such aspects of language as agreement and voice.

We shall also talk about the form in which meaning is expressed. Form includes both the syntax and the individual words used in a sentence. Many forms may have the same meaning:

"Burton tried the butterscoth fondue"
"Burton tasted the butterscoth fondue"

In such cases we speak of the multiple realizations of a meaning. On the other hand, when a single form has more than one meaning, we have ambiguity:

Alec had thrown the game. (and the gamblers were pleased)
Alec had thrown the game. (and the checkers lay scattered about the room)

Finally, we shall speak of the content, or meaning of a sentence. Just as syntax is defined in terms of abstract concepts, so we shall define content only in terms of abstract concepts. These concepts are the units of meaning provided by a conceptual representation. And just as the syntactic units seem to have some sort of 'reality' to language users, so these meaning units should have a reality for language understanders. The meaning of a sentence is in part determined by the syntax used to construct the sentence.

It is also affected by the context in which the sentence occurs. When talking about meaning, however, we shall not be including the notion of the intent of the utterance. We shall be designing a generator which always "says what it means"; not one which says "Your hair was very pretty when it was long" when it means "Your short hairdo is outrageous." Ultimately the best definition of meaning we can give will be the representation used for it. It is content, without any remnants of form, that conceptual representation attempts to capture.

We shall try to avoid using the term semantics in our descriptions. It has been used in many ways in the literature; in fact, almost anything which has to do with the relation of natural language to meaning has been termed semantics at some time. Katz and Fodor <11> defined it as "linguistic description minus grammar" which is a satisfactory definition if we know what linguistic description and grammar are. This definition points up one feature of most semantic representations: they are by nature linguistic. That is, they attempt to represent meanings expressed by a particular natural language. Conceptual representations are not linguistic in nature. They are meant to describe information derived from sensory experience and mental processing as well as linguistic sources.

The term analysis will be used to refer to the discovery of the conceptual representation of the meaning of a sentence. Parsing, on the other hand, will refer to the discovery of the syntactic structure of a sentence.

Finally, we shall call expressing a meaning representation in natural language realizing that representation. Realization is thus a special case of language generation, distinguished by its use of a meaning representation as a source.

Figure 1-1 outlines the basic components and interactions of a conceptually based language processing system. The three main components are those mentioned earlier. First, there is a language analyzer, which maps surface strings into conceptual representations:

$$A: \quad S \rightarrow C$$

There is a language generator, which maps conceptual structures into surface strings:

$$G: \quad C \rightarrow S$$

Finally, there is a 'memory model' which manipulates conceptual structures:

$$M: \quad C \rightarrow C$$

Both analysis and generation are meaning preserving processes. The memory model is probably the least understood of the three components. With analysis and generation we have fairly concrete ideas of what the

desired input - output relationships should be, even though we don't know how to achieve all of them. But for many tasks, particularly those which involve some sort of dialogue situation, it is not even clear what conceptual response would be appropriate for a conceptual input to the memory.

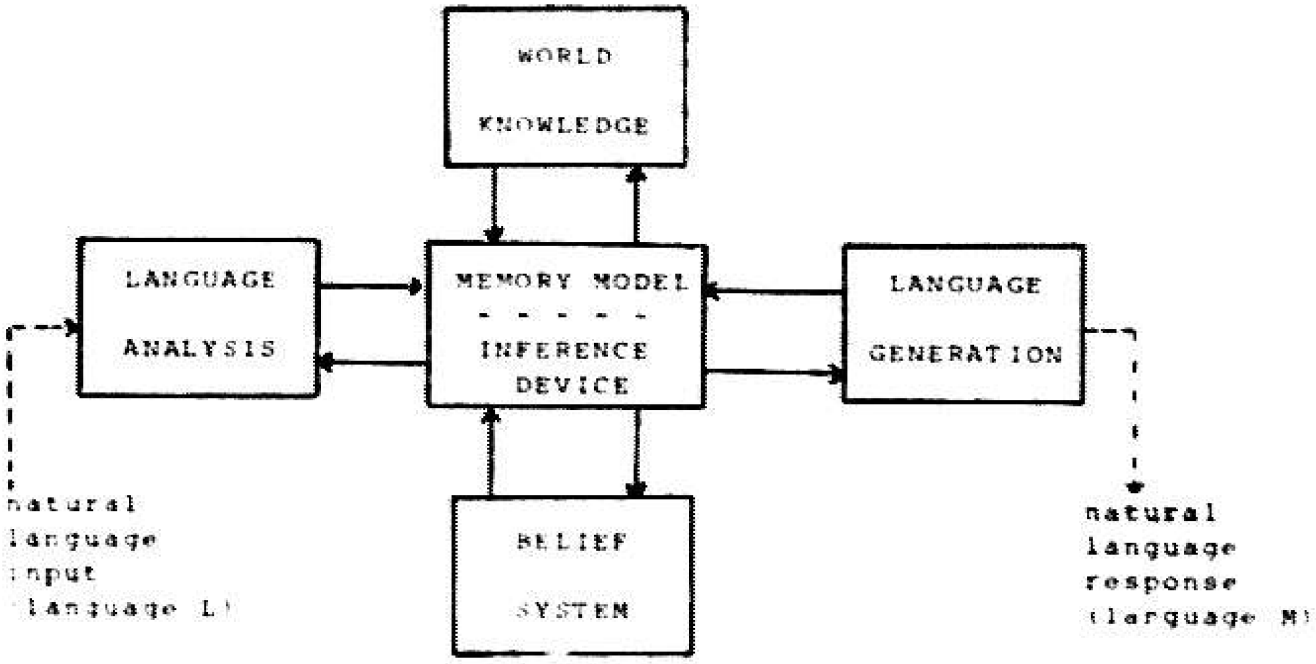


FIGURE 1-1

Let us see how these processes combine to perform some of the tasks mentioned earlier. For MT, a surface string in language L is analyzed to produce a conceptual representation. Since translation requires preservation of meaning, the memory operation reduces to the identity function -- it merely passes the analysis result along to the generator. The original surface string can be discarded. The generator must produce an appropriate string in language M to express this meaning. Translation is the only one of the tasks suggested that requires the output language M to differ from the input language L.

For question answering the analysis would be identical to that performed for MT. In 'information gathering' mode, the memory would not be producing material for the generator to express. It would, however, be integrating the analysis result into its knowledge store[1]. In 'questioning' mode, the analysis result will indicate the request for some sort of information. The memory, depending on its sophistication, will either try to find the requested information, or, failing this, attempt to deduce it from the stored information. In any case, if an answer is obtained, it will be in conceptual form and will be passed to the generator for linguistic expression.

In these two tasks the analysis and generation processes depend only on the natural language being used, not on the particular task at hand. We intend for this to hold true across a broad variety of tasks. A different analyzer would be needed for English than for German strings, but the basic content of the analysis should not depend on whether the string is to be translated or used as new information. A given generator will only express conceptual information in one language. The mapping, however, should be independent of the reason the memory model has for expressing the information.

In this thesis we attack the problem of generation. Generation is defined here to be the mapping of conceptual representations into surface strings -- that is, deciding HOW TO SAY IT. We define the question of choosing or building a conceptual representation for expression -- that is, the problem of deciding WHAT TO SAY -- as not being part of the generation process, but of another which our model places temporally prior to generation. We shall not discuss this problem in this thesis. Nevertheless, we shall assume it has been solved. For some tasks, like MT (where source of generation = result of analysis), the assumption is valid. For others, like interviewing, a great deal of work remains.

BABEL is a computer implementation of a conceptual generator. It assumes a particular conceptual representation (described in Chapter 3) and is intended to operate in a configuration like that shown in Figure 1-1. BABEL has been developed in conjunction with implementations of conceptual analysis and memory operations. The combined system is known as MARGIE (Memory, Analysis, Response Generation, and Inference on English) <3> . BABEL has also been developed operating in a mode in which a human performs the conceptual encodings of meaning and the deduction required for generation.

BABEL has been tested in three task domains. The first is sentence paraphrasing. This is described in some detail in Chapter 6. In this task a sentence is typed by a human, analyzed by a conceptual analysis program, and paraphrases, or multiple realizations, are produced by BABEL from its conceptual representation. The second area might be termed inference expression. In this task, a sentence is typed to the computer and, following conceptual analysis, the memory model produces a set of inferences. These inferences, themselves conceptual representations, are passed to BABEL for expression in English. Finally, we have given BABEL sufficient German linguistic data to perform English → German machine translation for a small subset of the conceptual structures accepted by the present English conceptual analyzer.

1.2 Annotated Examples of BABELing

We present in this section several examples of the program's responses in the various modes of operation. With each example is a brief description of how the result is achieved and notes on points of particular interest. Throughout this section upper case is used to indicate input to and output from the program; lower case is used for comments.

We first consider several paraphrase examples. These are produced with BABEL running as one component of the MARGIE system. Conceptual analysis of the input sentence is not performed by BABEL, but by a program written by Riesbeck <25> .

```
TYPE INPUT
*(JOHN GAVE MARY A BICYCLE)          the input sentence, typed
                                     by a human

OUTPUT FROM PARSER:                  The result of conceptual analysis.
                                     It consists of two parts. First,
                                     the 'meaning' of the utterance:

((ACTOR (JOHN) <=> (*ATRANS*) OBJECT (BIKE REF (INDEF))
FROM (JOHN) TO (MARY) TIME (TIM01) FOCUS ((ACTOR)))
```

This is the conceptual representation employed by BABEL. Chapter 3 describes this representation in detail. For this particular example, the analysis is roughly: "An Actor (who was the individual JOHN) changed the

'possessionship' of an Object (which was an instance of a bicycle) from the possession of the individual JOHN to the individual MARY. This event occurred at time TIM01 and the utterance focussed on the Actor of the event."

The second part of the analysis relates the times used in the representation to each other and to the time of utterance:

```
TIM00 : ((VAL *NOW*))          TIM00 has value *NOW* (which is
                               always the time of utterance)
TIM01 : ((BEFORE TIM00 X))     TIM01 is before TIM00 by an
                               unspecified amount X
```

next we have an internal memory format of this information:

```
PARTIALLY INTEGRATED RESULT:
((*ATRANS* (@JOHN1) (G0004) (@JOHN1) (@MARY1)) (TIME_(G0006)))
```

At this point the original English input is discarded. Only the conceptual analysis is used for further processing. The memory model could proceed to make inferences from the new information. In paraphrase mode, however, it merely saves the information and reconstructs it in the format used by BABEL (which is virtually identical to that produced by the analyzer.) It then becomes the only item of a list of THINGS-TO-SAY passed to BABEL.

```
THINGS TO SAY:

((ACTOR (JOHN) <=> (*ATRANS*) OBJECT (BIKE REF (INDEF))
FROM (JOHN) TO (MARY)) TIME (G0006) FOCUS ((ACTOR)))
```

PARAPHRASE:

BABEL now produces English sentences which express the meaning encoded in the conceptual representation. The first step in this process consists of producing a 'syntax network':

```
G0020:   OBJ2     (G0023)          G0021: LEX   (JOHN)
         OBJ      (G0022)
         ACTSBJ   (G0021)
         VOICE    (ACT)            G0022: DET   (A)
         FORM     (SIM)                   LEX   (BICYCLE)
         TENSE    (PAST)
         MOOD     (INDIC)
         LEX      (GIVE)           G0023: LEX   (NARY)
```

Unlike the conceptual representation, the syntax network utilizes English words and English syntactic relations. Finally, this network is used to produce an English sentence.

```
(JOHN GAVE MARY A BICYCLE)
```

which in this case is identical to the original input.

The program goes on to produce further syntax nets and sentences to express the same meaning.

```
G0022:   IOBJ     (G0025)          G0024:   DET    (A)
         OBJ      (G0024)                   LEX    (BICYCLE)
         ACTSBJ   (G0023)
         VOICE    (ACT)
         FORM     (SIM)            G0025:   POBJ   (G0026)
         TENSE    (PAST)                    PREP   (TO)
         MOOD     (INDIC)
         LEX      (GIVE)
                                   G0026:   LEX    (MARY)

G0023:   LEX      (JOHN)
```

```
(JOHN GAVE A BICYCLE TO MARY)
```

This paraphrase is the result of the fact that 'give' can express the "change possession" meaning in either of two syntactic frames: "X gave Y Z" and "X gave Z to Y".

```
G0023:   IOBJ     (G0026)          G0025:   LEX    (MARY)
         ACTSBJ   (G0025)
         OBJ      (G0024)
         VOICE    (ACT)            G0026:   POBJ   (G0027)
         FORM     (SIM)                     PREP   (FROM)
         TENSE    (PAST)
         MOOD     (INDIC)
         LEX      (GET)            G0027:   LEX    (JOHN)

G0024:   DET      (A)
         LEX      (BICYCLE)
```

```
(MARY GOT A BICYCLE FROM JOHN)
```

By ignoring the fact that the Actor (JOHN) was focussed, a paraphrase which focusses on MARY is produced.

```
G0024:   TOBJ     (G0027)          G0026:   LEX    (MARY)
         ACTSBJ   (G0026)
         OBJ      (G0025)
         VOICE    (ACT)            G0027:   POBJ   (G0028)
         FORM     (SIM)                     PREP   (FROM)
         TENSE    (PAST)
         MOOD     (INDIC)          G0028:   LEX    (JOHN)
         LEX      (RECEIVE

G0025:   DET      (A)
         LEX      (BICYCLE)

(MARY RECEIVED A BICYCLE FROM JOHN)
```

'Receive' is conceptually synonymous with the sense of 'get' used in the preceding paraphrase.

Each of the paraphrases is produced directly from the conceptual representation of the meaning being expressed. BABEL contains no rules which explicitly transform "give" sentences into "receive" sentences, etc.

In the remaining examples, we shall not show the conceptual representation or syntax nets produced by the process. Rather, an 'Englishy' version of the conceptual representation will be given to provide the reader with a fairly good idea of the nature of the information which BABEL is trying to express.

```
TYPE INPUT
*(OTHELLO KILLED DESDEMONA BY CHOKING DESDEMONA)
the current analyzer does not deal with pronouns; thus we
cannot type in "by choking her".
```

The conceptual analysis breaks this information down into the conjunction of two causative relationships: (AND X Y). The first of these, X, relates Othello's grasping Desdemona's neck to its result; namely, that she was not able to take in air:

X=(CAUSE A B) A=(GRASP OTHELLO NECK(DESDEMONA))
B=(UNABLE(INGEST DESDEMONA AIR))

The second element of the conjunction, Y, relates the result of the first, B, to its result: namely, that her 'health' state changed to the lowest possible value (-10)

Y=(CAUSE B C) C=(BECOME DESDEMONA HEALTH -10)

After being passed through the memory, the conceptual analysis is given to BABEL, and the following paraphrases are produced:

PARAPHRASE:

(OTHELLO STRANGLED DESDEMONA)
One of BABEL's major goals is to find words which express large amounts of conceptual structure. In this case, English provides a verb, 'strangle', which expresses almost everything. This is the first realization BABEL produces. In general, the most compact way of expressing meanings also seems to be the way most natural for English speakers (and, we would expect, for speakers of other languages). Because of BABEL's organization of linguistic knowledge, the most compact realization is virtually always the first one created.

(OTHELLO CHOKED DESDEMONA AND SHE DIED BECAUSE SHE WAS UNABLE TO BREATHE)
This and the remaining paraphrases use English conjunction to express the two elements of the conceptual representation. The verb 'choke' is found to express the "prevention of breathing by grasping the neck" idea. The second causal relation is actually expressed using 'because'. BABEL realizes that English provides a special verb, 'breathe' to express 'taking in air", and that the change in health can be expressed as "die".

(OTHELLO CHOKED DESDEMONA AND SHE DIED BECAUSE SHE WAS UNABLE TO INHALE AIR)
Virtually identical to the previous paraphrase. The fact that English provides 'breathe' is now ignored, and a word for the intake of any gaseous substance, 'inhale' is chosen. Of course, use of this more general verb requires explicit mention of its object.

(OTHELLO CHOKED DESDEMONA AND SHE DIED BECAUSE SHE COULD NOT BREATHE)
'Unable' breaks up into the negating of ability.

(OTHELLO CHOKED DESDEMONA AND SHE BECAME DEAD BECAUSE SHE WAS UNABLE TO BREATHE)
Even 'die' can be broken down into separate units in English which express the "health" change. The word 'become' can be used with a 'predicate adjective' to express state changes whenever the resulting state can be named; this holds true even when no word is provided for the notion of changing to that state (e.g., "to become dirty").

(OTHELLO PREVENTED DESDEMONA FROM BREATHING BY GRABBING HER NECK AND SHE DIED BECAUSE SHE WAS UNABLE TO BREATHE)
'Choke' also gets paraphrased eventually. Since its representation matches the pattern "someone doing something which causes someone else not to be able to perform some action' the verb 'prevent' is appropriate. Of course, two pieces of information which were expressed by 'choke' must now be explicitly given; namely, what was prevented ('breathing') and how it was prevented ('by grabbing the neck').

(DESDEMONA WAS UNABLE TO BREATHE BECAUSE OTHELLO GRABBED HER NECK AND SHE DIED BECAUSE SHE WAS UNABLE TO BREATHE)
'Prevent' is itself a pretty complicated idea, and can be expressed in more primitive terms as a causal relation.

(DESDEMONA COULD NOT INHALE AIR BECAUSE OTHELLO GRABBED HER NECK AND SHE BECAME DEAD BECAUSE SHE COULD NOT INHALE IT)
If allowed to run on for awhile, BABEL eventually breaks everything down into small chunks. That is, insofar as possible, each conceptual unit and relation is individually expressed by an appropriate English unit. Of course, interspersed among these last few realizations is a large number of sentences which merely combine paraphrase in different ways.

(OTHELLO CHOKED DESDEMONA AND SHE DIED BECAUSE SHE COULD NOT INHALE AIR)

(OTHELLO CHOKED DESDEMONA AND SHE BECAME DEAD BECAUSE SHE WAS UNABLE TO INHALE AIR)

(OTHELLO CHOKED DESDEMONA AND SHE BECAME DEAD BECAUSE SHE COULD NOT BREATHE)

(OTHELLO CHOKED DESDEMONA AND SHE BECAME DEAD BECAUSE SHE COULD NOT INHALE AIR)

(OTHELLO PREVENTED DESDEMONA FROM BREATHING BY GRABBING HER NECK AND SHE DIED BECAUSE SHE WAS UNABLE TO BREATHE)

(OTHELLO PREVENTED DESDEMONA FROM BREATHING BY GRABBING HER NECK AND SHE DIED BECAUSE SHE COULD NOT BREATHE)

(OTHELLO PREVENTED DESDEMONA FROM BREATHING BY GRABBING HER NECK AND SHE DIED BECAUSE SHE COULD NOT INHALE AIR)

(OTHELLO PREVENTED DESDEMONA FROM BREATHING BY GRABBING HER NECK AND SHE BECAME DEAD BECAUSE SHE WAS UNABLE TO BREATHE)

(OTHELLO PREVENTED DESDEMONA FROM BREATHING BY GRABBING HER NECK AND SHE BECAME DEAD BECAUSE SHE COULD NOT BREATHE)

(OTHELLO PREVENTED DESDEMONA FROM BREATHING BY GRABBING HER NECK AND SHE BECAME DEAD BECAUSE SHE COULD NOT INHALE AIR)

(OTHELLO PREVENTED DESDEMONA FROM INHALING AIR BY GRABBING HER NECK AND SHE DIED BECAUSE SHE WAS UNABLE TO INHALE IT)

(OTHELLO PREVENTED DESDEMONA FROM INHALING AIR BY GRABBING HER NECK AND SHE DIED BECAUSE SHE COULD NOT BREATHE)

(OTHELLO PREVENTED DESDEMONA FROM INHALING AIR BY GRABBING HER NECK AND SHE DIED BECAUSE SHE COULD NOT INHALE IT)

(OTHELLO PREVENTED DESDEMONA FROM INHALING AIR BY GRABBING HER NECK AND SHE BECAME DEAD BECAUSE SHE WAS UNABLE TO INHALE IT)

(OTHELLO PREVENTED DESDEMONA FROM INHALING AIR BY GRABBING HER NECK AND SHE BECAME DEAD BECAUSE SHE COULD NOT BREATHE)

(OTHELLO PREVENTED DESDEMONA FROM INHALING AIR BY GRABBING HER NECK AND SHE BECAME DEAD BECAUSE SHE COULD NOT INHALE IT)

(DESDEMONA WAS UNABLE TO BREATHE BECAUSE OTHELLO GRABBED HER NECK AND SHE DIED BECAUSE SHE WAS UNABLE TO BREATHE)

(DESDEMONA WAS UNABLE TO BREATHE BECAUSE OTHELLO GRABBED HER NECK AND SHE BECAME DEAD BECAUSE SHE WAS UNABLE TO BREATHE)

(DESDEMONA WAS UNABLE TO INHALE AIR BECAUSE OTHELLO GRABBED HER NECK AND SHE DIED BECAUSE SHE WAS UNABLE TO INHALE IT)

(DESDEMONA WAS UNABLE TO INHALE AIR BECAUSE OTHELLO GRABBED HER NECK AND SHE BECAME DEAD BECAUSE SHE WAS UNABLE TO INHALE IT)

(DESDEMONA COULD NOT BREATHE BECAUSE OTHELLO GRABBED HER NECK AND SHE DIED BECAUSE SHE COULD NOT BREATHE)

(DESDEMONA COULD NOT BREATHE BECAUSE OTHELLO GRABBED HER NECK AND SHE BECAME DEAD BECAUSE SHE COULD NOT BREATHE)

(DESDEMONA COULD NOT INHALE AIR BECAUSE OTHELLO GRABBED HER NECK AND SHE DIED BECAUSE SHE COULD NOT INHALE IT)

(DESDEMONA COULD NOT INHALE AIR BECAUSE OTHELLO GRABBED HER NECK AND SHE BECAME DEAD BECAUSE SHE COULD NOT INHALE IT)

*TYPE INPUT
(JOHN PAYED 2 DOLLARS TO THE BARTENDER FOR SOME WINE)
The conceptual analysis of this sentence expresses two events, each being the cause of the other. One event is the transfer of possession of two dollars from the individual JOHN to a (known) bartender. The second event is the transfer of an unspecified quantity of wine from this bartender to JOHN. Furthermore, the analysis claims there was a focus on the 2 dollars.

PARAPHRASE:

(JOHN PAYED THE BARTENDER 2 DOLLARS FOR SOME WINE)
BABEL's first realization employs the verb 'pay', as did the input.

(THE BARTENDER SOLD JOHN SOME WINE FOR 2 DOLLARS)
The next realization uses the verb 'sell'. It was chosen as a result of ignoring the focus on the money and choosing a verb which focuses on the 'seller'.

(JOHN BOUGHT SOME WINE FROM THE BARTENDER FOR 2 DOLLARS)
The verb 'buy' expresses the same meaning, but with yet a different focus.

(JOHN PAYED THE BARTENDER 2 DOLLARS TO GIVE HIM SOME WINE)
When a mutual causation between the giving of money and the of an object exists, 'pay for' is appropriate. More generally, a mutual causation between giving money and another action can be expressed as 'pay to', where this other action must be explicitly stated.

(THE BARTENDER TRADED JOHN SOME WINE FOR 2 DOLLARS)
Mutual causation between two possession changes is the meaning underlying 'trade' for BABEL.

(THE BARTENDER GAVE JOHN SOME WINE AND HE GAVE HIM 2 DOLLARS)
BABEL also expresses this meaning as a simple conjunction of the two events, ignoring the relationship between them.

BABEL also will combine the different paraphrases of 'give' with these basic forms, but this produces no other realizations of particular interest so we shall not bother to show them. In the remaining examples, we shall follow this practice of showing only those paraphrases which demonstrate some interesting feature of conceptual generation.

The examples presented up to this point are ones which can be run through the entire MARGIE system. BABEL is capable of producing sentences from meanings which are more complex than those which are produced from any sentences in the competence of the current conceptual analyzer. To run such examples through the generator, we must hand code the conceptual structure:

```
((CON ((ACTOR (IAG) <=> (*MTRANS*) FROM (*CP*PART (IAG))TO
(*CP*  PART (OTH) NOBJECT ((ACTOR (HANDKERCHIEF *OWN* (DES))
<=>   (*POSS* VAL  (CAS))) TIME (T-3))) FOCUS ((TO PART))
TIME (T-3)) <= ((CON ((CON ((  CON ((ACTOR (OTH) <=>
(*DO*)) TIME (T-1)) <= ((ACTOR (DES)   <=>T (*HEALTH*
VAL (-10))) TIME (T-1)))) <=C ((ACTOR (OTH)  <=>T (*JOY*))
INC (3)  TIME (T-1)))) <=> (*MLOC* VAL (*LTM* PART (OTH))))
TIME (T-?))))
```

expresses a meaning which is basically:

"An event caused Othello to believe that if he performed some unspecified action which resulted in Desdemona's becoming dead it would increase Othello's happiness. The event which made Othello believe this was a communication of some information by Iago to Othello. This information was that Cassio was in possession of a handkerchief owned by Desdemona."

BABEL combines chunks of this primitive meaning into

English words and comes up with the sentence:

PARAPHRASE:

(OTHELLO WANTED TO KILL DESDEMONA BY DOING SOMETHING BECAUSE HE HEARD FROM IAGO CASSIO HAD HER HANDKERCHIEF)

which does not make us believe that computers are on the

threshhold of becoming great playwrights, but does express

the meaning of the conceptual structure reasonably clearly.

```
TYPE INPUT
*(BILL LOANED MARY A BOOK)
```

This is analyzed conceptually as "Bill transferred possession of a book from Bill to Mary, and at the time he did this he believed that at some future time Mary would transfer possession of the book from Mary to Bill"

PARAPHRASE:
(BILL GAVE MARY A BOOK AND HE EXPECTED HER TO RETURN IT TO HIM)
Generating 'give' from a change of possession has already been demonstrated. BABEL knows that beliefs about future events can be realized in English using the verb 'expect'. The most interesting part of this example is the use of 'return' to express the second possession change, although the conceptual units which encode it do not differ from those which encode the initial transfer. The distinction lies in the context of the action: the second transfer of the book is to an individual who previously possessed it. The use of 'return' is not a result of the fact that the particular word 'loan' was used in the input, nor even of the fact that the information which made 'return' appropriate to express this event was originally encoded in the same sentence as the event itself.

TYPE INPUT
*(SOMEONE TOLD CAESAR BRUTUS WOULD KILL CAESAR)
The analysis of this one is fairly straightforward. It is simply the communication, at a past time, from an unspecified person to CAESAR, that an event would occur at some time in the future of this communication. That event is the killing -- i.e., the doing of something to cause a particular change in health -- of CAESAR by BRUTUS.

PARAPHRASE:
(SOMEONE WARNED CAESAR BRUTUS WOULD KILL HIM)
BABEL chooses 'warn' as an appropriate realization of the communicative event in this case. This choice requires conceptual knowledge: in this case, knowledge of the potential effect of the communicated event on the individual to whom it was communicated. Now consider another case:

TYPE INPUT
*(FALSTAFF TOLD HAL FALSTAFF DRANK HALS WINE
Again the analysis indicates communication of an event. This time it is an event in the past of the communication. The meaning of what was told dictates a very different realization from the 'telling' in the previous example:

PARAPHRASE:
*(FALSTAFF ADMITTED TO HAL HE DRANK HIS WINE)
This time the verb 'admit' is selected. In order to choose 'warn', 'admit', or just 'tell' to express communicative events very sophisticated inference processes are required. Neither the memory model now operating with BABEL, nor any other currently available computer program, is capable of performing these processes in much generality.

However, BABEL presents a solution to the linguistic portion of this problem: that is, it knows when such processes should be activated, and what are the appropriate questions to ask.

We present only one example of inference expression. From the standpoint of conceptual generation it makes no difference that a conceptual representation being expressed was created as an inference rather than as an analysis of natural language input. The ability to express inference demonstrates two important points, however:

1) The primary motivation behind the use of a conceptual representation is its facilitation of meaning-oriented processes. The ability of a program to make inferences confirms this.

2) In paraphrasing, BABEL is dealing with conceptual representations created by a conceptual analyzer, a language oriented program. One might believe that this process somehow leaves linguistic elements in the meaning structure it produces. When these structures are manipulated by the inference program, however, no knowledge of language is being used. It seems far less likely that language-specific information is being preserved or introduced by this process. The ability of BABEL to express inferences thus provides a stronger test of the claim that BABEL is truly expresssing a meaning representation than

does paraphrasing.

Since inference expression does not demonstrate any new features of the generator, we shall merely give a simple unannotated example here.

```
TYPE INPUT
*(JOHN PREVENTED MARY FROM HITTING BILL BY CHOKING MARY)
INFERENCES:
(JOHN CHOKED MARY)
(MARY DID NOT HIT BILL BECAUSE MARY WAS UNABLE TO BREATHE)
(JOHN GRABBED MARY'S NECK)
 MARY WAS UNABLE TO BREATHE)
(MARY BECAME HURT)
(MARY DID NOT HIT BILL)
(MARY WANTED TO HIT BILL)
(MARY WANTED BILL TO BECOME HURT)
(MARY WAS ANGRY AT BILL)
(JOHN POSSIBLY CAME TO BE ANGRY AT MARY)
```

We conclude this section with several examples of English → German machine translation. The use of a more exotic foreign language would undoubtedly be a more convincing demonstration of the language-free nature of our conceptual representation and the cross-linguistic validity of the generative process. Inevitably, the mechanical translation capabilities of the program are limited by the human translation capabilities of its

designer; thus the choice of German for these examples.

We emphasize that BABEL does not claim to be an MT program. We believe, though, that the effort needed to enable the program to produce German realizations is roughly the same as that which is required for English generation. Little effort has been expended on German generation. The computer examples below were chosen not to demonstrate the program's generality as an English → German translator, but to show how conceptual analysis and generation solve some of the problems which cause trouble for more conventional approaches to MT

```
TYPE INPUT
*(JOHN ATE A BANANA)
TRANSLATION:
(HANS HAT EINE BANANE GEGESSEN)
```

The conceptual analysis specifies a past event consisting of John ingesting an object which is a banana. The German realization requires the same basic considerations as would an English one, although German includes a few additional syntactic constraints, such as the gender of the nouns in the sentence.

```
TYPE INPUT
*(A MONKEY ATE A BANANA)
TRANSLATION:
(EIN AFFE HAT EINE BANANE GEFRESSEN)
```

The conceptual analysis of this sentence differs from that above only in the concept which becomes the actor of the ingesting event. German requires the selection of a different verb, 'fressen', to translate eat in this case, because of the fact that this actor is an animal rather than a human. Thus the German generator must take into consideration information not relevant to a corresponding English realization.

```
TYPE INPUT
*(JOHN WANTS TO EAT A BANANA)
TRANSLATION:
(HANS WILL EINE BANANE ESSEN)
```
The conceptual analysis is basically "John believes that it would cause him some sort of benefit if he ingested a banana". German provides a verb, 'wollen', which expresses this sort of 'want'. A fairly direct translation, without meaning analysis, would suffice in this case.

```
TYPE INPUT
*(JOHN WANTS A BANANA)
TRANSLATION:
(HANS WUNSCHT DASS MAN IHM EINE BANANE GEBEN WIRD)
```
The conceptual analysis of the English expresses that what John wants (believes would benefit him) is someone to transfer possession of a banana to him. (What he ultimately wants, most likely, is to eat the banana, but discovery of this is not a linguistic task.) English allows the use of 'want' in this case as well, with a single noun phrase direct object. No corresponding construction for this meaning is provided in German; an entire embedded sentence is required.

Like most programs which deal with language use, BABEL is just a toy. It cannot, either alone or in conjunction with other programs currently available, perform any useful function. Nor has any attempt been made to formalize a 'miniature world' in which BABEL could be in some sense 'complete'. We have tried to take a broad view of language production and solve some of those problems which are inherent in language itself rather than those specific to a small domain.

1.3 Remarks

Language generation has taken a back seat to language analysis in most computational linguistics research. There are several reasons for this:

1) Language understanding came to be seen as the major stumbling block in language processing. Understanding = analysis.

2) A problem which has always concerned both computational and pure linguists is ambiguity. It was always a problem of analysis, but could be ignored in generation, either because the underlying representation from which generation took place was assumed to be unambiguous, or because any ambiguity present there could be allowed to remain in the surface.

3) Many tasks which have been attempted require sophisticated language analysis, but little or no language generation. 'Woods' information retrieval system <4> is an example. In general, applications which are not intended to simulate human language use can be quite inflexible or even 'canned' in their output. Much greater variety must be handled in the input domain, however.

4) As a result, the problem of language generation has never been well defined in computational linguistics.

This thesis will attempt to provide a clear picture of what is involved in language generation, when that generation takes place from a conceptual underlying representation. One of our goals will be to present one method by which this form of generation can be accomplished. We will show that generation, like analysis, is dependent on context and on 'world knowledge'. We shall also show how deduction is used by a generation program, albeit for different purposes than those to which analyzers put deductive capability.

Another goal of this work is to make clear a distinction between linguistic knowledge and conceptual knowledge. Both are used in the generative process. Conceptual knowledge, however, is shared by analysis and memory processes, as well as by the generator. It therefore resides in the memory and is stored in a non-linguistic format. Other knowledge is used only in generation. We categorize this knowledge into several distinct classes, and show how each may be represented in the computer and how each contributes to the formation of the surface string.

Before forging ahead with this description, however, we will look back briefly at other work on computer generation of language. None of this work was designed to tackle the precise problem which concerns us here. It is

worthwhile, however, to see where BABEL fits in with, and overlaps, these other efforts. In so doing some of the issues which convince us of the need for this different model of language processing, with its inherently different model of generation, will be clarified.

CHAPTER 2

PREVIOUS WORK ON AUTOMATIC LANGUAGE GENERATION

The notion of using a computer to produce natural language output is not new. In this chapter we review several previous approaches to the problem and point out some of the major strengths and weaknesses of each.

It was realized very early in the development of computational linguistics that a computer equipped with

(1) a random number generator, and
(2) a generative grammar,

could be programmed to pour out English sentences in great profusion. To do this, it is sufficient to take the grammar's sentence symbol S and randomly choose applicable rewriting rules.

Victor Yngve [44] wrote such a random generator using a context free grammar. He chose ten sentences from a children's story and wrote a 77 production grammar which was capable of generating each of them. The grammar contained several types of recursion, including noun phrase conjunction, adjective sequences, and prepositional phrases whose objects are modified by other prepositional phrases -- e.g., "the ball on the table in the room." This grammar was then used to randomly generate sentences, a small sample of which is reproduced below.

When Engineer Small keeps Small and four fire-boxs, he keeps driving wheels, his steam, it, and four black and oiled fire-boxs.

Water is big.

When he is oiled, the shiny smoke stack is proud of four engines.

When he makes its little and polished bell, Small is proud of the four bells under his four black headlights.

The water under the wheels in oiled whistles and its polished shiny big and big trains is black.

Such examples point up the difficulty of deciding the grammaticality of meaningless sentences. One can often say little more than that the sentence is grammatical according to the grammar used to generate it. A more serious defect of such programs is that even if they were able to generate random sentences which English speakers agreed were grammatical, due to their randomness they could not be used directly for computational models of real natural language tasks such as MT. Such considerations have motivated several efforts at computer production of language in the last fifteen years.

3.1 Klein's Paraphrase Program

One of the most glaring deficiencies of early attempts at machine generation of language was the lack of any theoretical basis for the production of sentences. Many ad hoc procedures could be devised to handle different situations as they arose, but such an approach was too cumbersome for any but exceedingly tiny fragments of language.

Sheldon Klein [18] designed one of the first programs which applied a linguistic theory to the problem of natural language generation in a moderately systematic fashion. The goal of Klein's program was to produce coherent paraphrases of English paragraphs. The program accepted as input one or more paragraphs of English text. This text was analyzed, sentence by sentence, with a dependency grammar. The important attribute of a dependency grammar as used by Klein is that it specifies not only a constituent structure (nested bracketing) for a sentence, but also a tree of 'dependency' relationships between the words of the sentence.

The program then used this dependency parse and applied a second dependency grammar (not necessarily distinct from the first) in a generative mode to produce an English paraphrase of the original text.

From a generative viewpoint, then, the problem is to find a method of controlling the 'output' grammar so that only paraphrases are produced. To describe Klein's solution to this problem, it will be necessary to first describe the structure and operation of his dependency grammars.

The formal grammar used can be thought of as a context-free grammar with dependency information attached to each production. In analysis mode, each word of a sentence is considered to be a single constituent. The head of each of these constituents is defined to be the single word which it contains. Let H(C) represent the head of constituent C, and let $g(w_i, w_j)$ be the predicate "word w_i governs word w_j" ("w_j is dependent on w_i"). Then a grammar rule:

$$L = R_1 \; R_2 \; \ldots \; R_n$$

must have associated with it specifications

(1) of an h, $1 \leq h \leq n$, such that $H(L) = H(R_h)$; i.e., the head of the new constituent L is the head of one of the constituents which were grouped to make L

(2) for each i, $1 \leq i \leq n$, $i \neq h$, an index j, $1 \leq j \leq n$, $j \neq i$, such that $g(H(R_j), H(R_i))$; i.e., the head of each constituent which does not become the head of the new constituent must be put in a dependency relation with the head of one of the other constituents being produced.

Furthermore, Klein requires the dependencies to be such that

$$G(w_i, w_j) \Rightarrow w_j \neq w_i$$

where G is the transitive closure of q.

A sample Klein dependency grammar is shown in Figure 2-1. His implementation of phrase structure rules made special use of subscripts on the non-terminals. A non-terminal X_i on the left hand side of a rule essentially subsumes all X_j, $j \in i$. The subscripting enables the grammar to be written more concisely, but does not distinguish it with respect to either power or any formal properties from other formulations of context free grammars.

---

1. Art $\overset{0}{\cdot}$ *N $\overset{2}{\cdot}$ N $\overset{3}{}$

2. Adj $\overset{0}{\cdot}$ *N $\overset{2}{\cdot}$ N $\overset{2}{}$

3. *N $\overset{1}{\cdot}$ Mod $\overset{1}{\cdot}$ N $\overset{1}{}$

4. *V $\overset{1}{\cdot}$ N $\overset{2}{\cdot}$ V $\overset{2}{}$

5. *Prep $\overset{0}{\cdot}$ N $\overset{3}{\cdot}$ Mod $\overset{1}{}$

6. *N $\overset{1}{\cdot}$ V $\overset{3}{\cdot}$ S $\overset{1}{}$

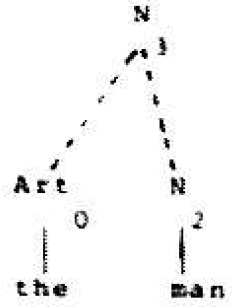
FIGURE 2-1

A '*' prefix on a non-terminal on the left hand side of a rule indicates the 'governor' of the constituent.

For example, the units:

could be combined by rule 1 of the sample grammar to yield

Constituent Structure



Dependency Structure

This formulation is sufficient to enable a slightly modified phrase structure analyzer to perform a dependency parse at the same time it creates a phrase-marker for a sentence. The dependency analysis can be represented as a tree giving the governor-dependency relationships between the words of a sentence. Figure 2-2 shows the phrase marker and dependency tree assigned to the sentence "The fierce tigers in India eat meat".

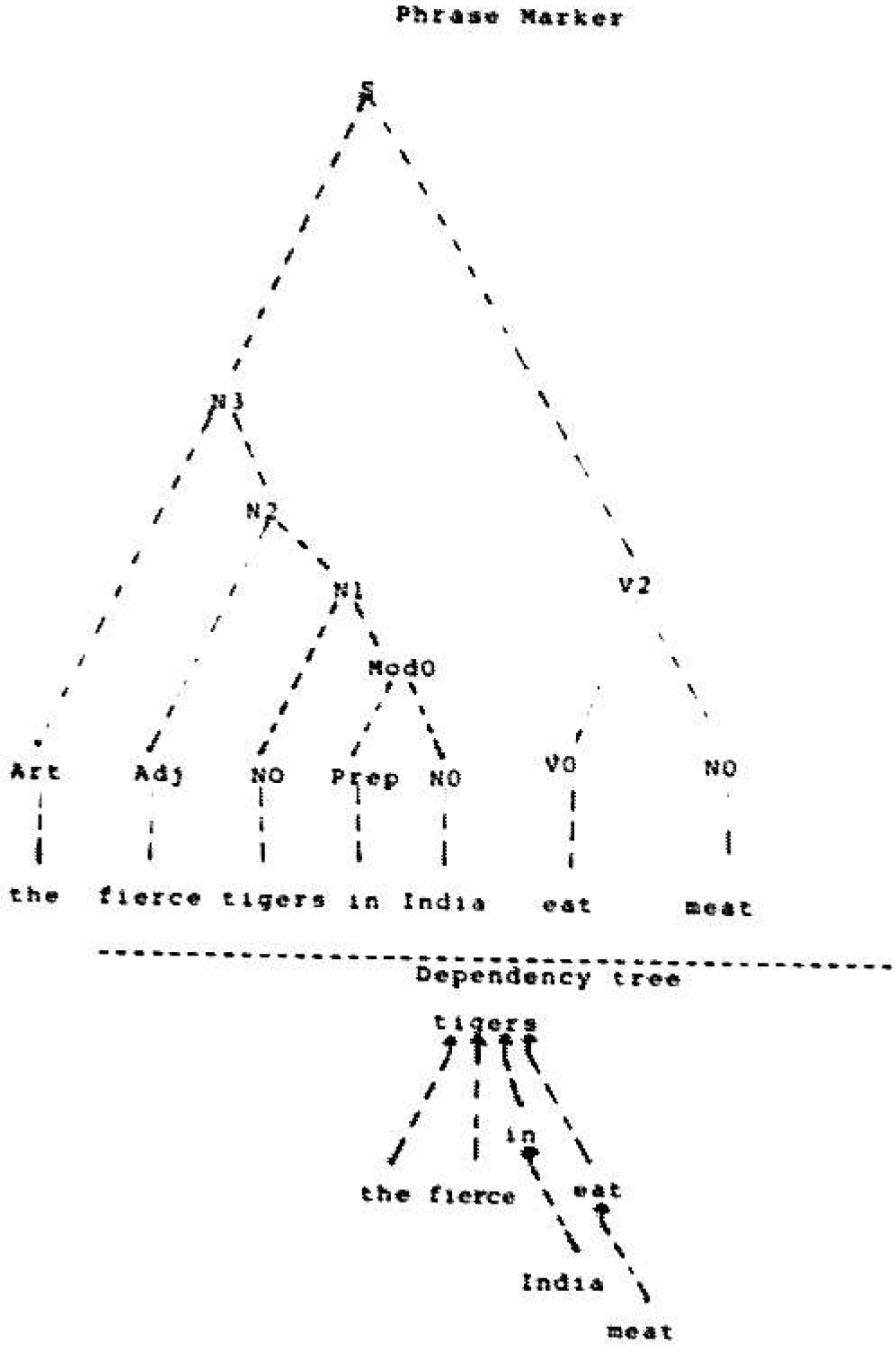
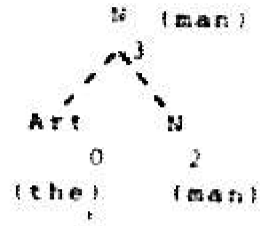

Phrase Marker
S
N3
N2
N1
V2
ModO
Art
Adj
NO
Prep
NO
VO
NO
the
fierce
tigers
in
India
eat
meat
Dependency tree
tigers
in
the
fierce
eat
India
meat

FIGURE 2-2

Klein's system performs such an analysis of each sentence in the input, and then discards the original sentences and the associated phrase-markers. The final pre-generative step is to convert the set of dependency trees into a large dependency network by adding two-way dependency links between all like noun tokens. (These are the only two-way links in the network.) Paraphrase generation is then accomplished through semi-random generation from the output grammar. Generation consists of a simultaneous construction of a sentence syntax (constituent structure) tree and word dependency tree. Initially the dependency tree is empty and the syntax tree consists of the symbol S. At each stage of the generation non-terminal elements at the leaves of the syntax tree are expanded by random choice of an applicable rewriting rule in the dependency grammar. Whenever a new node is produced in the tree, a word must be associated with it. Depending on the production used and the dependency information associated with it, the word chosen may be that associated with the parent of the new node or a new word of the appropriate grammatical category. Rule 1 of Figure 2-1 might be used to expand the node N_3 with associated word 'man' into



Simultaneously, the node 'man' in the dependency tree would have a dependent node 'the' attached to it.

The random nature of the generation is controlled in two ways. Most importantly, no word dependencies are permitted in the generated sentence which did not exist in the dependency network produced by the analysis. In order to achieve paraphrase, rather than a regeneration of the original text, it is necessary to consider the dependency relationship to be transitive across most dependency links. Klein used the heuristic:

$$g(w_1, w_2) \wedge g(w_2, w_3) \Longrightarrow g(w_1, w_3),$$

provided that w_2 is not

i) a verb other than a form of 'to be', or
ii) a preposition other than 'of'.

The second mechanism used to reduce randomness in the output is the use of an 'essay format' to guide the logical sentence flow in the paraphrase paragraphs. This format consists of a list of nouns to be used as sentence subjects and a list of verbs to be used as main verbs. Two modes of creation of essay formats were tried: directly from the nouns and verbs occurring in the source text, or from the nouns and verbs in a human generated 'outline' paragraph.

Following is an example of the paraphrases produced by this program. Input paragraph:

> Clever John met Mary in the park. John married Mary. Mary loved John. Mary wanted a child. Mary had a child. Mary raised a child. John was a successful business man who worked for a corporation. Mary was penniless. John secretly loved Helen who was beautiful. Helen who also loved John was married to Peter. Mary was a friend of Helen. Peter was a buddy of John. Helen who was friendly often ate lunch with Mary. John played golf with Peter. John wanted Helen. Helen wanted John. Divorce was impossible. The solution was simple. John liked Mary. Helen liked Peter. John killed Peter. Helen killed Mary. The end was happy.

Paraphrase paragraphs:

> John who married penniless Mary met her. Clever John was a business man. He loved friendly Helen. He played golf. He wanted Helen. John who killed a buddy liked penniless Mary.
>
> Mary in the park who wanted a child loved clever John. She had a child. She raised it. She was a friend of friendly beautiful Helen.
>
> Beautiful Helen loved successful John. Beautiful Helen was married. Helen who wanted John ate lunch. She liked a buddy. She killed Mary.
>
> Peter was a buddy.

Klein's work provided little improvement over that of Yngve in the realm of syntactic quality. It is important because it took into account the need for <u>meaningful</u> generated sentences. Its major drawback is that the meaning present in the output must have its source in the input.

In fact, this meaning must be reflected in the constituent structure of the input. There is no provision for understanding or reasoning in the model; referential and meaning ambiguity are allowed to remain in the dependency structures created by the analysis routine.

2.2 Friedman's Transformational Generator

Joyce Friedman <12,13,14> has written a generation program based on Noam Chomsky's <5> transformational model of language. While Chomsky's work is theoretically taken as a description of linguistic knowledge rather than of linguistic process, it is formulated as a system of production rules and structure transforming rules. It is therefore natural to use the theory as the basis for a generative procedure.

Friedman's system contains a set of context-free phrase structure rules, a set of transformational rules, and a lexicon. A random generation algorithm is capable of expanding the phrase structure rules to produce a base tree (phrase marker) whose root node is the sentence symbol S. The base tree then undergoes a lexical insertion process which expands the terminal nodes into 'complex symbols' and attaches lexemes to them. The complex symbols contain both syntactic information -- e.g., [+ TRANSITIVE] -- and some categorical information -- e.g., [+ HUMAN].

Following lexical insertion, transformations are applied to the phrase marker to produce a surface tree.

Such a system could be used to randomly generate English sentences in the same fashion as early random generators. The quality of the sentences generated would depend to a large extent on the sophistication of the lexical insertion process. A linguist could judge the sentences produced as acceptable or non-acceptable, and modify the grammar and lexicon appropriately.

But the program would be of little use once the grammar became large and complex because the probability of production of a particular construction which might be of interest would be very low. In order to give the user greater control over the types of sentences generated, Friedman allowed him to initialize the process not just with the sentence symbol S, but with a partially specified phrase marker. By increasing the specificity of this initial phrase marker, it is possible to increase the probability that a particular transformation will be applied in the generative process. In the implementation described in <1> , the initial tree may specify:

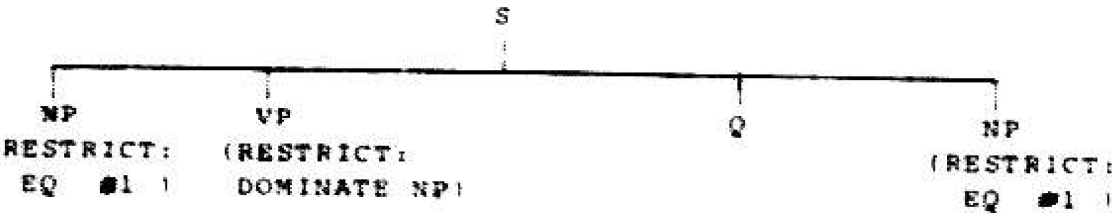
(1) branching structure, including the non-terminals to appear at particular nodes,

(2) dominance restrictions -- a node must be the ancestor of a node labeled with a particular non-terminal in the completed base tree,

(3) non-dominance restrictions -- a node may not be the ancestor of a node labeled with a particular non-terminal in the completed base tree,

(4) Equality -- two or more nodes must have identical subtrees in the completed base tree.

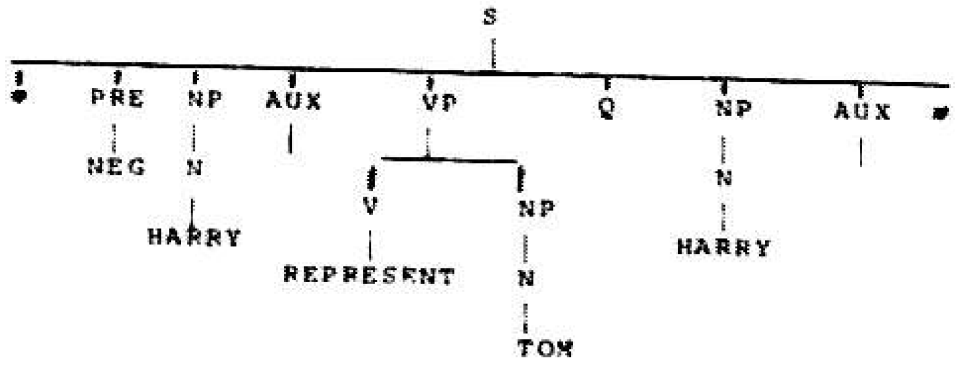
For example, the skeletal phrase marker might be



(which guarantees that the final phrase marker will have the main verb phrase governing a noun phrase and will have identical subtrees for two top level noun phrases).

The random phrase structure generator is constrained by the initial tree to produce a phrase marker satisfying all restrictions of types (1) - (4) specified. The base tree produced undergoes lexical insertion and the transformational cycle to generate a sentence.

The above skeleton could lead to a completed phrase marker which looks something like



and a surface string "Harry doesn't represent Tom, does Harry/"

Transformational grammar provides a syntactic description of language which is far superior to that obtained by simpler grammars. It also has the advantage of producing syntactic variants of a single meaning from a 'canonical' underlying deep structure. Friedman's system is oriented toward linguistic research (the development of better transformational grammars). It thus does not provide a method for using transformational grammars in a computational language processing application. The question of the usefulness of transformational deep structures for semantic processes is left open.

2.3 Generation in Winograd's Blocks World

Terry Winograd's "Computer Program for Understanding Natural Language" <41>, although oriented toward natural language understanding and memory modeling, does a limited amount of generation in order to carry on a conversation.

The basic generative paradigm of the system is the patterned response. A patterned response is a string of English words stored in memory when the system is initialized. The string of words may contain some blank slots to be filled in with strings of words chosen when it

has been determined that the pattern is an appropriate response. Variations of this response paradigm have been commonly used in conversational and interviewing programs <40> , but Winograd introduces some new 'fill in the blank' operations which are relevant to the general generation problem considered in the following chapters.

In Winograd's program a response pattern is activated by either the syntactic form of the sentence input -- a 'when' question, a command, a declarative sentence -- or a special condition arising during the interpretation of the input -- unknown word, ambiguous word, undecidable anaphoric inferences.

The simplest blank-filling operation is to insert a phrase directly, or with a minor transformation, from the input which stimulated it; e.g. "I don't know the word _____." When the necessity to resolve ambiguity arises, a list of senses of the ambiguous word or phrase is taken from the lexicon and included in the response. The human conversant can then resolve the ambiguity by choosing the appropriate sense.

In answering questions, situations arise where objects and events stored in memory formats must be expressed in English. Still, the type of input determines the response pattern:

Question type	Response pattern
Why	"because <event>", or "in order to <event> "
How	"by <event>"
When	"while <event>", or "before <event>"
What	<list of indefinite object descriptions>
Which	<list of definite object descriptions>

here Winograd faces the problem of generating language from a memory representation of information. His solution consists basically of a set of heuristics designed to provide reasonable responses for the 'blocks world' domain.

Each object in the world has a unique internal name associated with it. Each object has a predicate giving its class. Finally, each object class has an English noun associated with it. For example, there might be an object with internal name OBJ21, a predicate (ISA OBJ21 #BALL), and the association ENGLISH (#BALL)=BALL. To describe an object, this noun is combined with adjectives and relative clauses to create an English noun phrase. First a color is attached -- "blue ball" -- and then a size -- "big blue ball" -- and finally relative clauses -- "big blue ball which is to the right of <noun phrase> ".

The description is deemed complete if at any time the phrase specifies a unique internal object, in which case the determiner 'the' is attached. Otherwise, depending on the syntactic environment (see above), a definite or indefinite article is attached to the noun phrase. The selection of color and size adjectives is made in a fashion analogous to the noun selection. Similarly, the English relations "to the right of" and "support" which are used in the relative clauses are directly associated with the memory's internal relations.

Events are described by associating a small program, or pattern, with each internal event type. For example:

Internal event	generation pattern
(#PUTON OBJ1 OBJ2)	(<correct form of 'to put'>, <noun-phrase for OBJ1>, ON, <noun-phrase for OBJ2>)

The correct form of 'to put' is to be decided on syntactic grounds -- "by putting" for 'how' questions, "to put" for 'why' questions. The noun-phrases for OBJ1 and OBJ2 are generated as described above.

While these techniques are capable of generating correct syntactic responses in the situations Winograd's model expects to encounter, they tend to produce unnatural discourse. Three devices were used to make the dialogue less machine-like.

First, in lists of noun-phrases, identical ones can be combined into a single noun-phrase with a proper numerical modifier, producing, for example, "three red blocks". Second, when an object referenced in a question is also referenced in the answer, rather than repeat part or all of a noun-phrase it is desirable to use "one" in the response. Thus:

Q. Is there a red block on the table?

A. Yes, a large one. (not, "Yes, a large red block.")

This is accomplished by directly comparing the two English noun phrases. Finally, a set of heuristics enables the generator to use pronouns 'it' and 'that' in responses.

Winograd's work is significant in that it demonstrates the usefulness of combining syntactic analysis with powerful semantic processes and world knowledge. He shows that a great deal can be accomplished when language analysis results in more than syntactic description. The major drawback of Winograd's work is that many problems of language are avoided by the severe constraints of the world with which he deals.

From the viewpoint of generation, however, Winograd basically adopted an approach mentioned in Chapter I -- that output can be left fairly rigid and needn't be

capable of handling all the syntax or meaning handled by the analysis routines. The 'natural' quality of the generated language is achieved as a byproduct of the constraints of the program's domain. Because he fully understood the relatively small set of potential forms to be expressed, Winograd was able to design extremely clever heuristics for each case and thus produce clean responses for each form.

We have seen four different approaches to language generation. Yngve demonstrated through pure random generation from a context free grammar that a computer could produce sentences from a formal syntactic description, and could be used to test the adequacy of a grammar. Transformational grammars were developed to provide improved descriptions of natural language. Friedman applied Yngve's idea to the new model. She added some controls on randomness, but these were designed to aid the grammar writer rather than adapt the computer model to use in a real task.

Klein tried to use the syntactic structure of the language to derive some semantic structure. To do this he used the notion of 'word dependency'. The device was used only for the preservation of meaning for use in generating paraphrases; no operations on the meaning were performed.

The paraphrases used the same words as the input, and thus were more syntactic than semantic in nature.

Winograd, rather than working on a general theory of generation, used task specific procedures to perform the generation he required for his blocks program. Significantly, he found no use for a random element in generation. For a few simple, well-defined situations fill-in-the-blank responses sufficed. In other cases, special purpose routines expressed in English the meaning of predications used in other parts of the program to control cognitive processes.

The last formulation of the problem of generation which we shall consider attempts to incorporate some of the best aspects of these other systems. It employs a formal representation and generative grammar, and is also designed to be applicable to interesting linguistic and cognitive tasks.

2.4 Simmons' Semantic Networks

Robert Simmons <34,35,36> in recent work on natural language processing has designed a system for analyzing sentences into a semantic representation and generating sentences from such a representation. Since Simmons' approach points up some of the basic distinctions between

the generative approach embodied in BABEL and those of other generators, as well as fundamental differences between conceptual and other meaning representations, it will be worthwhile to describe this work in some detail.

Simmons calls his deep structures semantic networks. These networks consist of concept nodes connected by semantic relations. Each concept node is distinguished by the relation TOKen whose value is a lexical word sense. Among other defining properties, a word sense has a context-free mapping onto an English word. (By a context-free mapping from word sense L1 we mean one which is independent of the semantic network containing the node whose TOKen is L1.) The semantic networks also contain information which is not necessarily reflected in choice of words, but in morphology and syntax -- e.g., MOOD-INTERROGATIVE, TENSE-PAST, VOICE-ACTIVE. Some of this information, such as TENSE, is clearly semantic in nature. Some of it, such as VOICE, is used only for syntactic purposes in generation.

The choice of semantic relations reflects the work of Fillmore <9> on deep semantic case structure of language. Each case relation is presumed to have certain semantic properties. The AGENT of action A, for example, must be an animate instigator of A. This constraint is independent of any particular agent, action, or syntactic structure

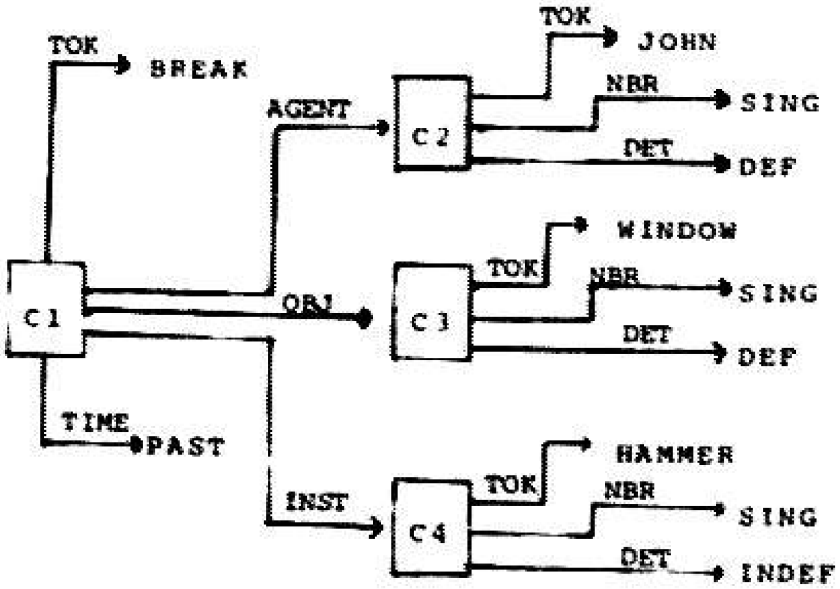
used to encode the relation.

A verb in a semantic network (more precisely, a concept node whose TOKen is a lexical word sense which can be mapped onto a verb) has associated with it a set of case arguments, each case argument being a case relation, such as AGENT or GOAL, and a value. The value of a case relation is another concept node, which may be expressed linguistically as either a noun phrase or an embedded sentence.

The basic semantic network representation for

"John broke the window with a hammer"

might look like:



This can be more concisely written as:

C1	TOK	BREAK	C3	TOK	WINDOW
	AGENT	C2		NBR	SING
	OBJ	C3		DET	DEF
	INST	C4			
	TIME	PAST	C4	TOK	HAMMER
				NBR	SING
C2	TOK	JOHN		DET	INDEF
	NBR	SING			
	DET	DEF			

Three distinct processes operate on these networks. An analyzer encodes English sentences into semantic networks. A generator produces English sentences from the networks. A transformational process maps semantic networks into other networks, and is required for certain types of paraphrases, as well as for inference.

Both the analyzer and the generator are implemented as Augmented Finite State Transition Networks (AFSTNs), as described by Woods [42]. An AFSTN has the structure of a finite state transition network. However, the arc labels no longer name terminal elements to be produced in the output (or scanned in the input stream), but may specify (i) predicates which must be true if the arc is to be followed, (ii) 'subroutine' transfers to other pieces of network, and (iii) storage of information in special registers. The added mechanism enables AFSTNs to perform the same sorts of operations as transformational grammars, but with certain computational and conceptual advantages

over the transformational grammar formalism.

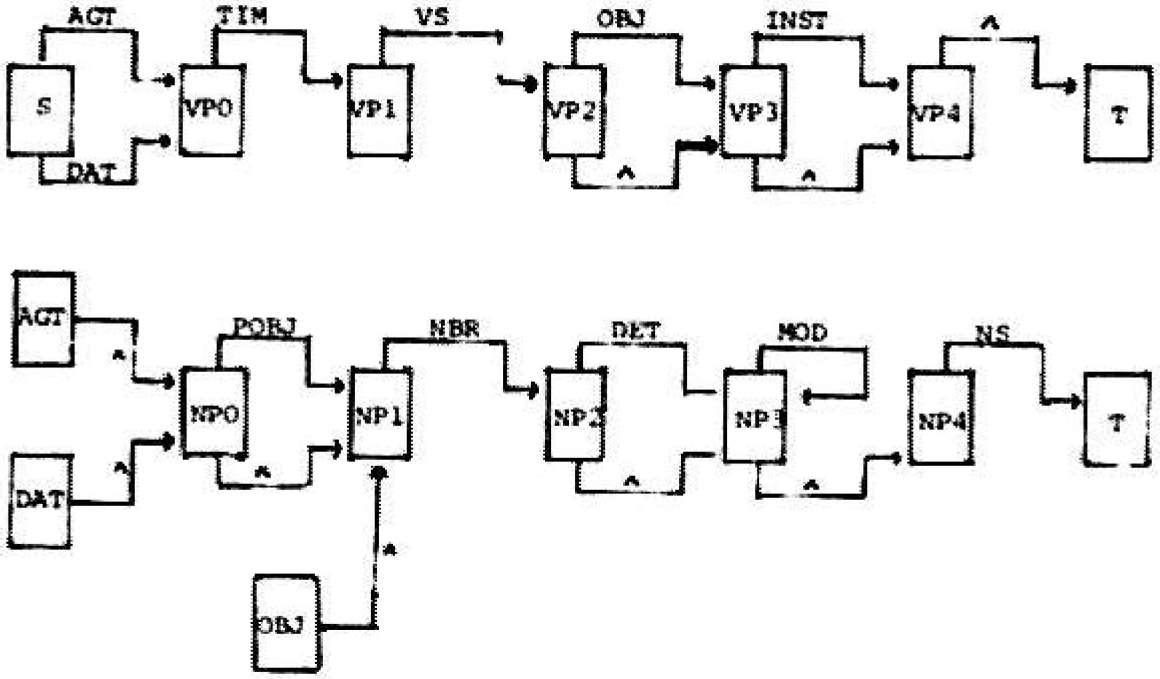
To see how APSTNs can be used to map semantic networks into English, consider the following network which Simmons uses for

"Mary was wrestling with a bottle"

C1	TOK	WRESTLE	C3	TOK	WITH
	TIM	PROG PAST		POBJ	C4
	AGT	C2			
	OBJ	C3	C4	TOK	BOTTLE
				DET	INDEF
C2	TOK	MARY		NBR	SING
	NBR	SING			

In Simmons' formulation, the relations marked TOK do not have words as their values, but pointers to word-sense lexical entries. These entries in turn are associated with individual words, as well as syntactic information (such as past tense forms) and semantic information (such as synonyms) about the words.)

The generation can be accomplished with the simple grammar shown graphically below:



Suppose it is desired to generate a sentence from the semantic concept structure C1. This structure is labelled S and the grammar is entered at the node with the corresponding label. There are two paths leaving this node. The one labelled AGT can be followed only if a corresponding semantic relation exists in the current semantic structure. In this case it does, so several actions take place:

> Structure C1 is relabelled as VP0 (the node at the end of the path followed)
>
> Structure C1 is 'pushed' onto a list of structures to come back to

The grammar is re-entered at node AGT with semantic concept C2 (the value of the AGT relation in the semantic net) as the active node

The path from AGT is labelled (A). This means it can be unconditionally followed to NP0. Two paths leave NP0. The first is labelled POBJ, but cannot be taken because no POBJ relation exists in concept C2. The other can be unconditionally followed to node NP1. To leave this node a relation NBR must be present in the semantic structure. It is, and has value (SING). NBR is a function which is then applied, and creates a noun from the TOK and NBR values (in other words, goes off to the lexicon and finds the singular or plural form of the noun). In this case, the noun will be "MARY". NBR then adds a relation NS (Noun String) to the active node (in this case C2) of the semantic network with this noun as its value. The path DET from NP2 cannot be taken, since the corresponding relation does not exist in the active semantic structure. (If it had, 'a' or 'the' would have been added to the NS). The unconditional path is thus taken to NP3. Here a sequence of MODs (adjectives) is permitted. Eventually (in our case, immediately) the unconditional path to NP4 is taken. The relation NS exists, and the function NS is applied, placing the value of the relation (in our example, "MARY") in the output string for the sentence being formed. The node labelled T in the grammar is thus reached.

This is a terminal; the effect is to 'pop' the pushdown list of interrupted structures, restoring C1 to active status.

C1 was labelled VP0 before being 'pushed'; the generation reverts to that node of the grammar. The only relation leaving VP0 is TIM. TIM is a function similar to NBR; it creates a Verb String (VS) which in this case is "was wrestling". The next transition in the grammar adds the VS to the output string. The transition from VP1 to VP1 requires processing the semantic relation OBJ. This results in activating concept nodes C3 and C4 and adding the string "with a bottle" to the output string. Nothing else of interest occurs and the final output string is "Mary was wrestling with a bottle".

The grammar used in this example consisted of two basic parts. One was a 'noun phrase' grammar which generated noun phrases from appropriate semantic structures. The other was a 'sentence' grammar which generated the verb string and caused activation of the noun phrase grammar at the proper time for the appropriate semantic structures in order to perform a left-to-right generation of the sentence.

Sentence paraphrase may be accomplished in several ways in such a system. If the generation AFSTN is non-deterministic (i.e., there exist distinct paths through

the network which may be followed for a given semantic network's syntactic paraphrase should result. TOKen relations may specify not just a single word sense, but a set of synonymous word senses. Simmons views paraphrase as being handled, at least in part, by a transformational component operating on the semantic networks. Paraphrase transformations would allow mappings between sets of case relations, and might introduce TOKen substitution as well. Given the semantic structure C1 for

"John bought the boat from Mary"

C1	TOK	BUY	C2	TOK	MARY
	SOURCE	C2			
	GOAL	C3	C3	TOK	JOHN
	THEME	C4			
			C4	TOK	BOAT

the rule P1:

BUY			SELL
SOURCE	(V1)	<==>	SOURCE (V1)
GOAL	(V2)		GOAL (V2)
THEME	(V1)		THEME (V3)

can be applied to produce the structure C1'

C1'	TOK	SELL	C2	TOK	MARY
	SOURCE	C2			
	GOAL	C3	C3	TOK	JOHN
	THEME	C4			
			C4	TOK	BOAT

from which the paraphrase "Mary sold the boat to John" might be generated. The rule P1 is interpreted as bi-directional, thus enabling paraphrase from 'sell' to 'buy' as well. Such a rule could also be used to

paraphrase "John gave Mary the book" as "John gave the book to Mary." Whether the latter transformation should exist, or whether this should be handled by non-determinacy in the generation grammar, depends in part on the amount of word dependency allowed in the generation process. More about this problem will be mentioned in Chapter 8.

Simmons distinguishes two types of paraphrase transformations; those which change the choice of lexical entries are termed 'semantic', all others are termed 'syntactic'. The 'buy - sell' rule above is an example of a semantic paraphrase, and is one in which only the TOKen is altered. (In another paper [36], 'John' is the AGENT of 'buy' and 'Mary' the AGENT of 'sell'. With this configuration, the 'buy - sell' transformation involves a change of case relations as well as TOKens.) An example of a syntactic paraphrase transformation would be a change from active to passive VOICE.

Simmons' semantic networks provide a representation of the content of natural language utterances which is appealing for machine implementation on several grounds:

1) In the realm of syntax, these networks, combined with AFSTN analysis and generation, provide the descriptive advantages of transformational grammar.

2) It is possible to define the networks in such a way that they are unambiguous.

3) The same representation serves as a result of analysis and a source for generation.

4) Formal rules can be written to perform inferences and deduction within the network structures. These rules can be used to produce network 'responses' in applications, providing for a natural input-process-respond cycle with no need for random generation.

5) Since a single theoretical framework is provided for generating from any semantic net, ad hoc rules for expressing particular meanings do not appear necessary.

The major drawback of these semantic nets is their language-dependency. That is, the set of basic meanings and relations between these meanings provided by the networks is determined by the particular language to which they are applied. Nevertheless, we shall see how a portion of this generation system has been adapted for use as part of BABEL.

CHAPTER 3

CONCEPTUAL DEPENDENCY REPRESENTATION

3.1 Conceptual Representation: basic requirements

Each of the endeavors reviewed in Chapter 2 was based on a different underlying representation for the content of natural language. Klein and Friedman used representations which explicate syntactic structure. The PLANNER assertions of Winograd and semantic nets of Simmons are oriented toward explicating meaning. A question which thus arises is "What are the desirable properties of a representation of linguistically encoded information, when this information is to be used in a computer application?"

Syntactic formulations are unsatisfactory because inferences and actions cannot readily be based on syntactic structure. The semantic formulations work well on small vocabularies in highly restricted domains. We shall see, however, that when they depend on representing meaning by directly associating language units with executable programs and implicational rules, they make unreasonable processing and storage requirements as the domain of

discourse expands.

Conceptual representation has been proposed as a solution to some of the problems inherent in semantic representations. A level of meaning distinct from any linguistic expression of that meaning is hypothesized. Language units are defined in terms of combinations of meaning units. Only the meaning units are actually associated with inferences and actions for the computer model to carry out. Conceptual representation is distinguished by several features:

(A) A conceptual representation must be 'language-free' -- that is, the same set of units and relations must be used to describe meanings which may be encoded in any human language.

(B) The representation must be unambiguous. This must be true even if the words or word combinations which express that meaning are themselves ambiguous.

(C) The representation provided for natural language sentences which are 'similar' in meaning should directly exhibit this 'similarity'. Closeness of meaning need not be formally defined; it is simply the feeling of speakers of English, for instance, that 'running' and 'walking' are closer in meaning than 'running' and 'killing'.

(D) The representations are oriented toward use in a computational memory model and inference system. One ramification of this is that the units and relations used to represent meanings derived from language must be the same ones used for internally generated information.

(E) The representations are frequently proposed as psychological models of human cognitive structures. The psychological ramifications of the representations will not concern us in this work. It is certainly not clear that a conceptual model must have any psychological validity in order to achieve successful

results in a computer application.

3.2 Conceptual Dependency: representation details

CONCEPTUAL DEPENDENCY (C.D.) is a conceptual representation which encompasses a particular set of primitive conceptual units and relations. It has been developed and described by Schank <31> . We shall not delve into the distinctions between C.D. and other conceptual systems <28> here. This section is devoted to a quick overview of C.D. and examples of its use to encode sentence meanings. This presentation has two main purposes:

i) to give the reader a feeling for the flavor of conceptual representations.

ii) to introduce terminology which will be used in the description of BABEL in Chapters 4 through 6.

We defer until Chapter 8 a theoretical comparison of this conceptual representation, and the language processes which it necessitates, with other approaches, such as those described in Chapter 2.

3.2.1 EVENTS

Natural language often uses single words to convey many pieces of information. This makes for efficient communication, but can cause problems if the individual pieces are needed rather than the entire conglomerate.

English verbs demonstrate this phenomenon. 'Sell', in its most common usage, indicates that some object came into the possession of the buyer and that some money was transferred to the seller. It is easy to construct situations in which a single one of these events, rather than the entire 'sell' complex, becomes central.

In C.D. all actions described in language are broken down into a set of primitive ACTs. ACTs are performed by ACTORs, and this relationship is symbolized:

<ACTOR> <===> <ACT>

'Eating' is represented by the primitive ACT '*INGEST*'; 'John eats' is represented as:

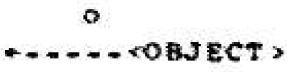
JOHN <===> *INGEST*

Not all ACTOR-ACT relationships describe physical events; 'giving' is an abstract notion involving change of possession and is represented by the ACT '*ATRANS*'. For 'John gives' we have the representation:

JOHN <===> *ATRANS*

The concepts of 'eating and 'giving' involve more than just ACTORs and ACTs. One must eat some physical object. An object cannot just be given by an ACTOR; there must also be some recipient of the giving. To represent relationships between ACTs and entities other than ACTORs, C.D. provides a set of conceptual CASEs. Each ACT requires the presence of a particular subset of CASEs.

Most ACTs require an OBJECTIVE case symbolized:



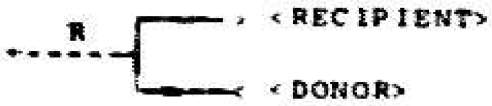
Examples of this relationship include:

"John drinks milk" *JOHN* <==> *INGEST* ---o *MILK*

"Fred breathes" *FRED* <==> *INGEST* ---o *AIR*

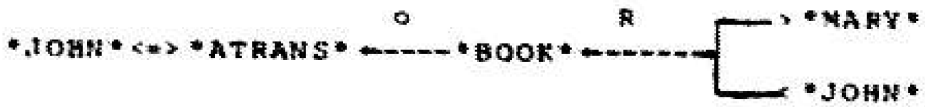
(the latter example desmonstrates how required conceptual cases will be present in representations even if no corresponding surface case exists.)

When the 'possession-ship' of an object is changed by an action, there must be both a DONOR and a RECIPIENT of the possession. The RECIPIENT CASE is provided to represent this relationship, and is denoted



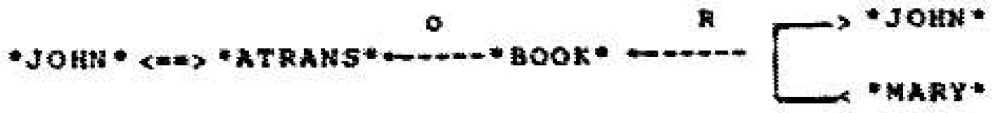
The ACT *ATRANS* requires the RECIPIENT CASE. Some examples:

"John gives Mary a book", or
"Mary receives a book from John"



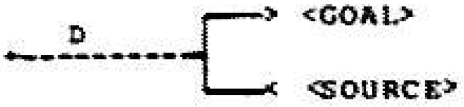
In this example we see how conceptual representation may provide identical analyses of sentences which differ not only in syntax, but in the actual words used. Different words, like 'give' and 'receive', may map into identical conceptual structures even if they are not synonyms in the normal sense. All that is required is that they convey the same meaning in the context in which they occur.

"John takes the book from Mary"

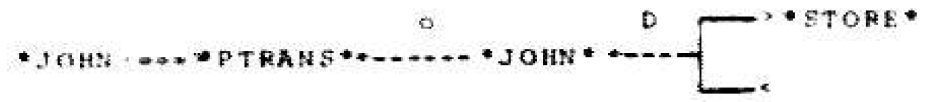


Here the conceptual analysis captures the similarity between 'give' and 'take', both of which communicate a possession change. In English these words are considered 'antonyms'; conceptually they differ by a reversal of recipient case roles.

The ACT *PTRANS* is used to represent actions of changing location. *PTRANS* requires an OBJECT (whose location is changed) and a SOURCE and GOAL location. The DIRECTIVE case provides slots for these locations, and is symbolized:



"John goes to the store"



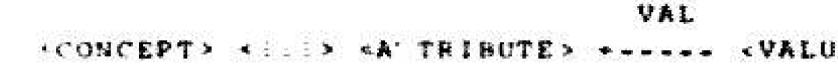
C.D. postulates the existence of fourteen primitive mental, physical and abstract ACTs <30>. Although we will only need to use a few of these in examples for the purpose of describing our generative model, we list all fourteen here for completeness:

ACT	meaning
ATRANS	change of possession ACT
PTRANS	change of location ACT
MTRANS	information transfer ACT
PROPEL	ACTOR applies a force to some object
MOVE	ACTOR moves a bodypart
INGEST	ACTOR takes something 'into' his inside
EXPEL	ACTOR takes something from his inside
GRASP	ACTOR grasps an object
SMELL	ACTOR takes in sense data from nose
SPEAK	ACTOR produces sound
LOOK-AT	ACTOR takes in sense data from eyes
LISTEN-TO	ACTOR takes in sense data in form of sound
CONC	ACTOR 'thinks about' some information
MBUILD	ACTOR performs processing which combines conceptual information to produce new information

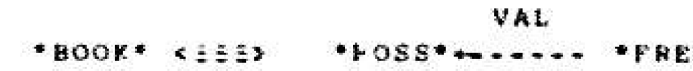
An ACTOR-ACT relationship, together with all the cases required by the ACT, is called an EVENT.

STATEs and STATE-CHANGEs

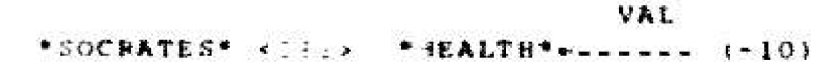
Some of the information stored in a memory and communicated in language is not represented as EVENTs, but as STATEs. The notation used in C.D. for such information is:



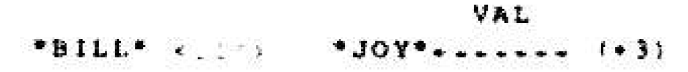
For example, "Fred has the book" is represented as



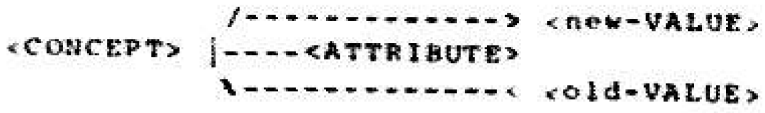
A subset of the ATTRIBUTEs used in C.D. are SCALEs. When the ATTRIBUTE of a STATE relation is a SCALE, the VALUE will be an integer representing a point on the SCALE. "Socrates is dead"



"Bill is happy"

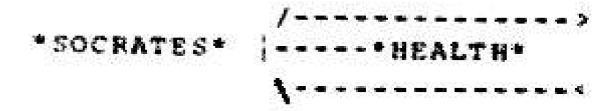


In other cases, changes in state must be represented. The STATE-CHANGE notation is:



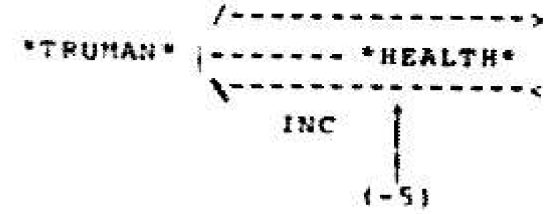
Commonly only the terminal state (ATTRIBUTE + new-VALUE) of a STATE-CHANGE relation is known, and we will not bother putting anything in the initial state slot.

"Socrates dies"



When the change of state is along a scale, it is common that neither the precise initial or terminal state is known, but only the direction, and perhaps amount, of change. A STATE-CHANGE can be modified by an INCrement to show this:

"Truman's condition deteriorates"



No one has yet proposed a closed set of state relations and scales for conceptual representation. While such a set is necessary for theoretical completeness of the

representational system, it has no bearing on the methods used in conceptual generation. For BABEL we have assumed a fairly small set of such units, sufficient for testing the various sorts of English structures which must be generated from state relationships.

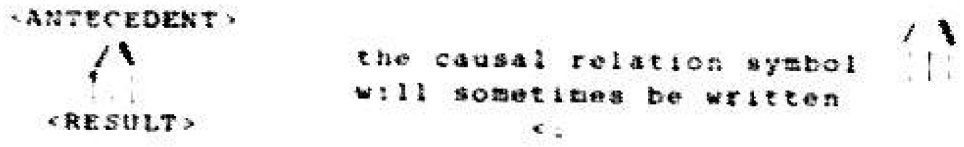
SCALE	dimension measured
HEALTH	physical health
JOY	mental pleasure
ANGER	anger
EXCITE	mental excitation
PSTATE	general physical state
BENEFIT	general well being; affected by change on any other scale
SIZE	size

non-scale states	property
POSS	<CONCEPT> possessed by <VALUE>
OWN	<CONCEPT> owned by <VALUE>
LOC	<CONCEPT> located at <VALUE>
MLOC	mental location; see section 3.2.4

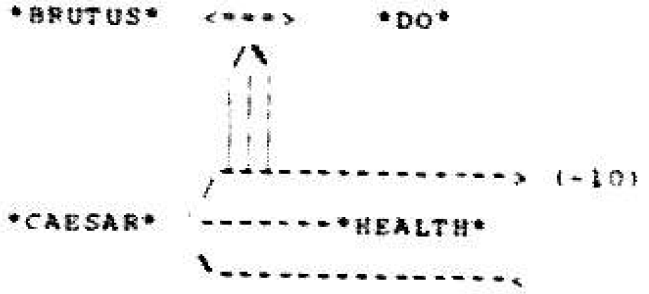
EVENTs, STATEs, and STATE-CHANGEs are all types of relationships which are termed conceptualizations .

3.2.3 CAUSALs and CONJUNCTIONs

Three types of causal relationship are provided. The first is a relation in which the occurrence of an ANTECEDENT conceptualization causes a RESULT conceptualization:

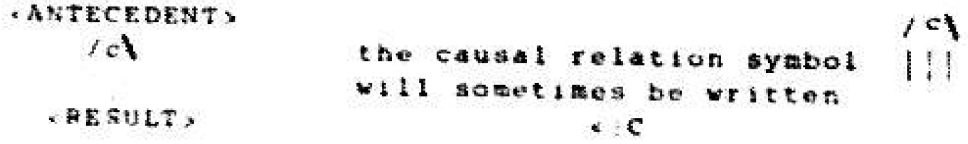


"Brutus killed Caesar"



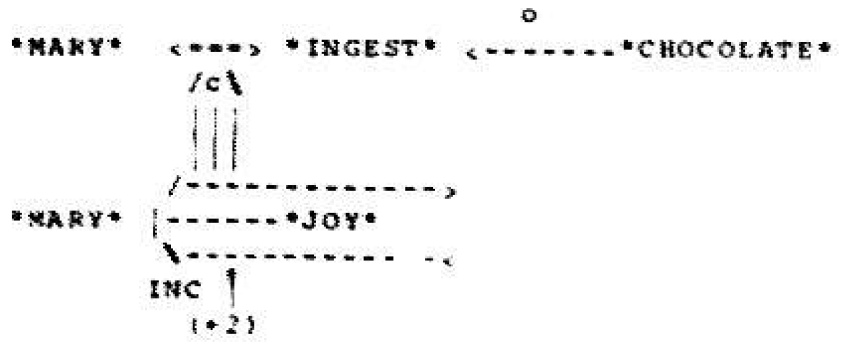
(*DO* is a 'dummy' ACT used to hold the place of some actual, but unknown, ACT and its required cases.)

The second causal relationship provided for is the CAN-CAUSE relation:

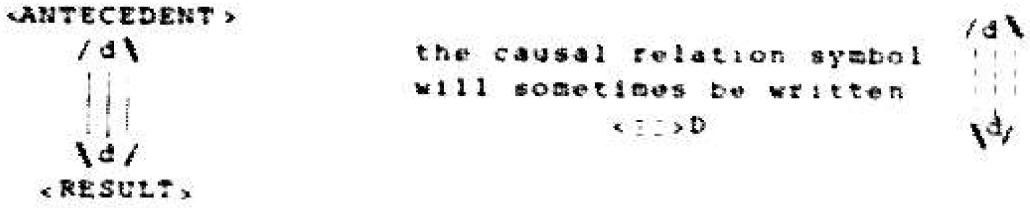


This relation indicates that the occurrence of the ANTECEDENT conceptualization would cause the RESULT conceptualization, but does not indicate the actual occurrence of either.

"Mary likes to eat chocolate"

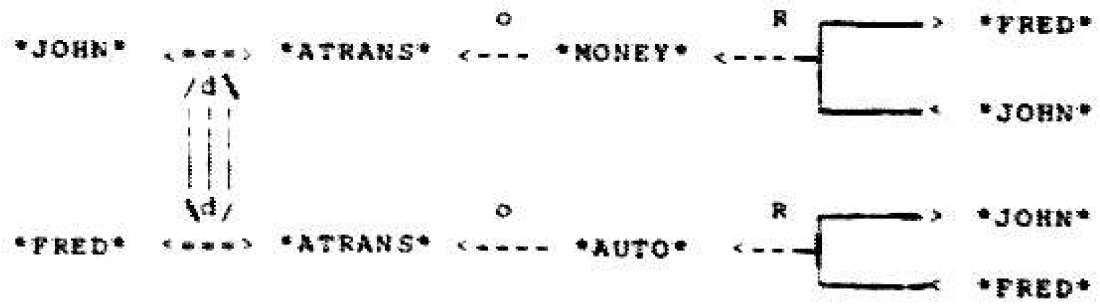


The third type of causal relationship is 'mutual causation':



This relation indicates that the ANTECEDENT and RESULT conceptualizations were caused by each other. The relationship is completely symmetric (and thus the terms ANTECEDENT and RESULT do no have the mnemomic value they have in the other forms of causal relationship). Mutual causation is used to represent 'buying', as in

"John bought the car from Fred"



The same representation is used for "sell" and "pay for" as for "buy".

All the CAUSAL relationships are themselves conceptualizations. Furthermore, any two conceptualizations can be joined by the symbol '^' to form a CONJUNCTION, or by the symbol 'v' to form a DISJUNCTION. Both CONJUNCTIONs and DISJUNCTIONs are also conceptualizations.

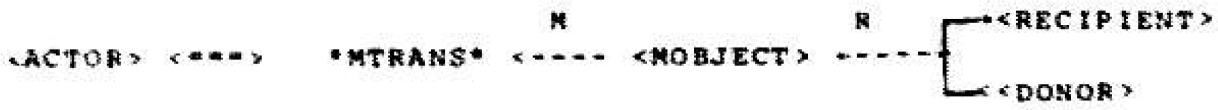
<conceptualization> ^ <conceptualization>

<conceptualization> v <conceptualization>

3.2.4 Mental ACTs and LOCations

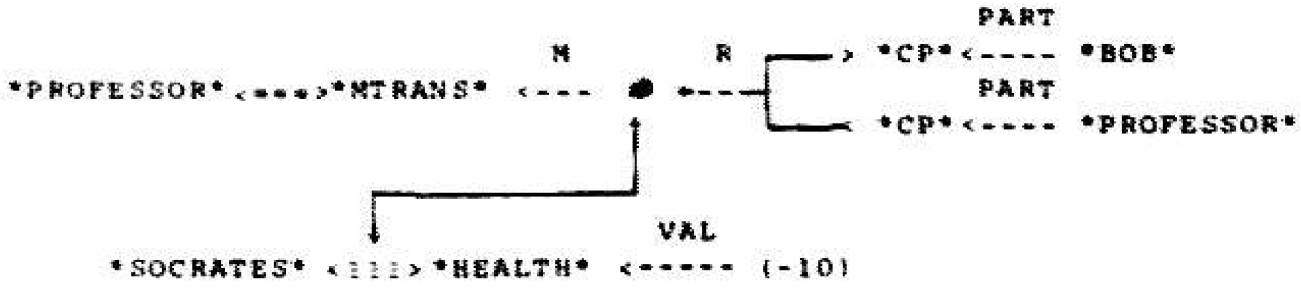
Many English verbs -- tell, remember, teach, read -- involve the transfer of information. Conceptual primitives for representing these meanings are discussed in <32>. The 'mental' ACT *MTRANS* is used to represent transfers of information. This act requires a new CASE, the MENTAL-OBJECT (MOBJECT).

An MOBJECT must itself be some conceptualization. *MTRANS* also requires the RECIPIENT CASE, with the DONOR and RECIPIENT being 'mental locations.' Allowable mental locations include 'conscious processors' (*CP*) of human beings (the conscious mind), the 'long-term memories' (*LTM*) of human beings,[1] and physical objects which in some sense serve as information stores (books, televisions, . . .) The notation for an EVENT using *MTRANS* is:



MTRANS is an abstract ACT which indicates the transfer of the information contained in the MOBJECT from the DONOR to the RECIPIENT.

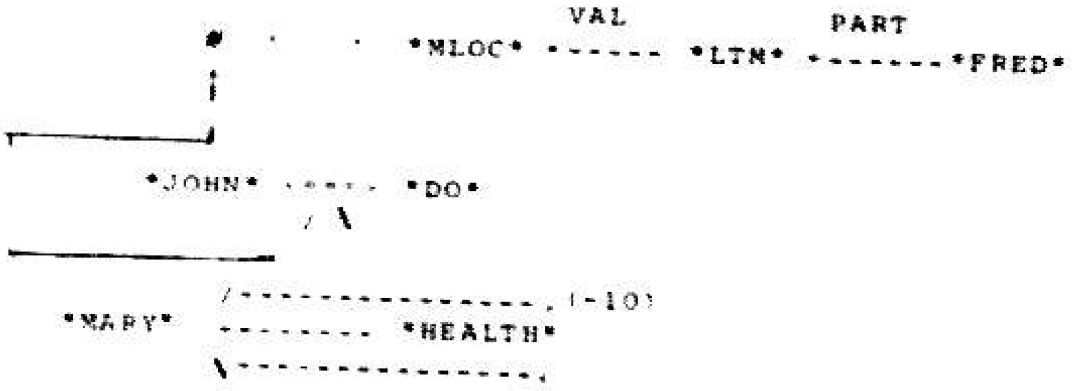
"The professor tells Bob that Socrates is dead"



(The notation *CP* <----[PART] *BOB* indicates the conscious processor of the individual *BOB*. When conceptualizations are embedded in other conceptualizations, a # will often be used as a 'place holder' and will be connected to the

main relational link of the embedded conceptualization.)

Mental locations can also fill the <VALUE> slot of STATE relations which have as their <ATTRIBUTE> *MLOC* (Mental-LOCation). The <CONCEPT> in such relations must be an entire conceptualization. For example,

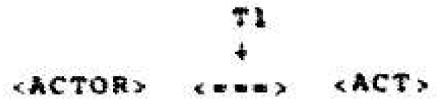


represents the meaning of "Fred believes that John killed Mary."

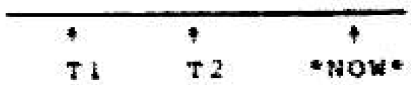
The *CP* can contain arbitrary conceptualizations. The *LTM*, on the other hand, contains only 'believed' conceptualizations, although they may be stored with a 'certainty' rating. A non-believed conceptualization will only be stored embedded in another conceptualization, as in "Politicians claim they are interested in our welfare."

3.2.5 TIMEs and other modifications

Still to be accounted for is the concept of the time of occurrence of an event, which usually is reflected in verbal tensing in language. BABEL deals only with points in time, not intervals. The symbols (T1, T2, T3, . . .) will be used for times, and drawn with pointers to some conceptual link:



The special symbol *NOW* represents the 'current' time -- i.e., the time of an utterance or, more exactly, the time of creation of a conceptualization. TIME relations will be shown on a time line, left representing PAST; right, FUTURE.

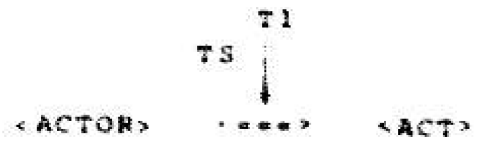


(indicates the relations
T1 < T2 < *NOW*
where ' < ' means 'BEFORE')

In the implementation, every EVENT, STATE, and STATE-CHANGE has a TIME associated with it. In our diagrams however, TIME will be left out unless it is relevant to the point being discussed.

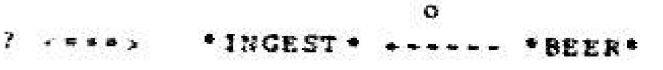
Although BABEL does not deal with time intervals, it is necessary to talk about the beginning or end of an

EVENT or STATE-CHANGE in order to represent some of the verbs in our examples (e.g., "arrive"). This is done by a modifying link labeled TS ('time start') or TF ('time finish') with a time point as its value:

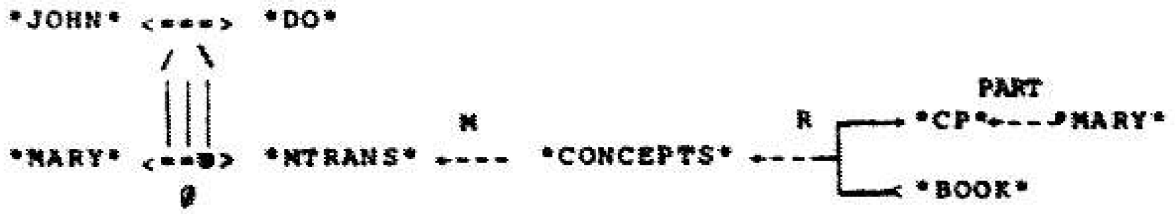


Negation is indicated by a "/" through the main link of the conceptualization -- <=/=>, <f/=>, etc. Interrogatives of two categories are dealt with. When the truth of a conceptualization is being questioned, this will be symbolized by a "?" attached to the main link: "Did John drink the beer?"

If the content of a particular conceptual role is questioned, that role is filled with a "?": "Who drank the beer?"



Another modification is the NODE 'CANNOT' which can modify an EVENT, and is symbolized by a ∅ on the <=>.



is the representation provided for "John prevented Nary from reading the book". (*CONCEPTS* is another 'dummy' conceptual unit. It represents unspecified conceptual information. One property of C.D. which is important for making inferences is its explicit representation of 'missing' conceptual information. For instance, the *DO* in this example might lead the model to try to discover "how did John prevent . . ."; the *CONCEPTS* might lead it to wonder what sort of information was in the book.)

Any conceptualization may be modified by a FOCUS relation. FOCUS always specifies one particular slot in a conceptualization, such as 'the ACTOR of the RESULT'. FOCUS will not be noted in our diagrams; while it is anticipated that the memory model will find uses for FOCUS, it is currently used only by the generation routine to choose between words like "give" and "receive".

3.2.6 Conceptual nominals

The reader may have wondered about the use of units *JOHN*, *BOOK*, etc., in conceptualizations. C.D. has provided a great deal of analysis of verbs and relations found in language, but little analysis of concrete and abstract nominals. The current program does not deal with words like "happiness" and "involvement", but is limited to nouns which name physical objects and people. The unit *JOHN* in a conceptualization is a pointer to a memory node, at which are pointers to all conceptualizations involving *JOHN*, including such conceptual information as

(HUMAN *JOHN*) and (MALE *JOHN*)

The relation most used by the generation system, however, is

(ENGLISH-NAME *JOHN* JOHN)

Where we write *BOOK* in a conceptualization, we really have a pointer #B to a set of relations which includes

(TOKEN-OF #B *BOOK*)

BOOK is the conceptual concept of 'book' and is itself a node associated with all the information about this concept (not about a particular book, however). Included in this information is

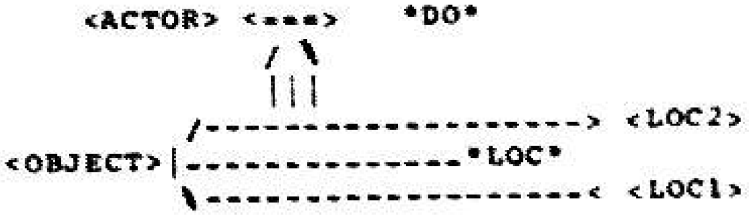
(ENGLISH-NAME *BOOK* BOOK)

In other words, we are assuming that for people and physical objects, we will find an 'ENGLISH-NAME' either directly

associated or associated via a level of indirection (and found by following a TOKEN-OF relation). In writing conceptualizations in this thesis, however, we shall not generally bother to distinguish these two cases, but rather will just represent the pointer to a memory node by the English name with '*' tacked onto the front and back. Chapter 7 discusses the sorts of extensions which will be necessary to deal with more complex English nominalizations.

3.2.7 Remarks

It is obvious that the conceptual representation presented here is based to a great extent on intuition about language and psychology. No proof of the adequacy of the representation to deal with a given data base is provided. Nor is there any test for the independence of the various units and relations. From a computational viewpoint the ACT *PTRANS* could be replaced with the representation



with no loss in the set of meanings representable. However, the goals of conceptual dependency are in part psychological. The representations are not intended as models of the physicist's universe. They are meant to model the world as perceived and described by people; particularly those aspects of the world dealt with in natural language. The conceptual approach to language processing is clearly a cognitive processing model rather than a pure A.I. approach.

An intuitive approach has been traditional in linguistic studies of both syntax and semantics. Whether describing sentences in terms of 'noun phrases' and 'verb phrases' or meanings in terms of 'agents', 'sources', and 'goals', the representations proposed are based on an intuitive choice of units and relations. A superstructure of operations is then placed on this representation and used as a test of its adequacy.

C.D. is not presented here as a finished product to which language processing must conform. It is nevertheless useful as a basis for testing models of analysis, memory functions, and generation. The representations must be allowed to change in detail as inadequacies are uncovered.

The details of C.D. are not important to the generative model presented in this thesis. Only the most basic aspects of conceptual representation -- the use of language

free units, the existence of patterns relatable to linguistic units -- determine the nature of the generative process.

A natural question to ask when first presented with the conceptual approach to language is "Why bother?". Breaking down language into conceptual units rather than syntactic or 'semantic' units adds one more level of complexity to language analysis and one more level to generation. The fact that other approaches have not yet succeeded in 'solving' the natural language problem is not in itself evidence that this additional complexity is required. In short, what are the advantages to this approach which override the handicaps it introduces?

Several points of a theoretical nature can be made favoring the use of conceptual representations over language-based ones. We shall defer a general discussion of this matter until Chapter 8, after the BABEL model of conceptual generation has been fully presented. Hopefully this presentation will itself point up certain advantages of conceptual representation, although we will not dwell on such points.

We conclude this chapter with an example of conceptual processes; that is, how a conceptual memory might manipulate conceptual structures to achieve results difficult to obtain with a language-based meaning representation.

Since none of the material of Chapters 4-7 is necessary for understanding the material in Chapter 8, the reader still bothered by "Why bother?" may wish to read that chapter immediately after this one.

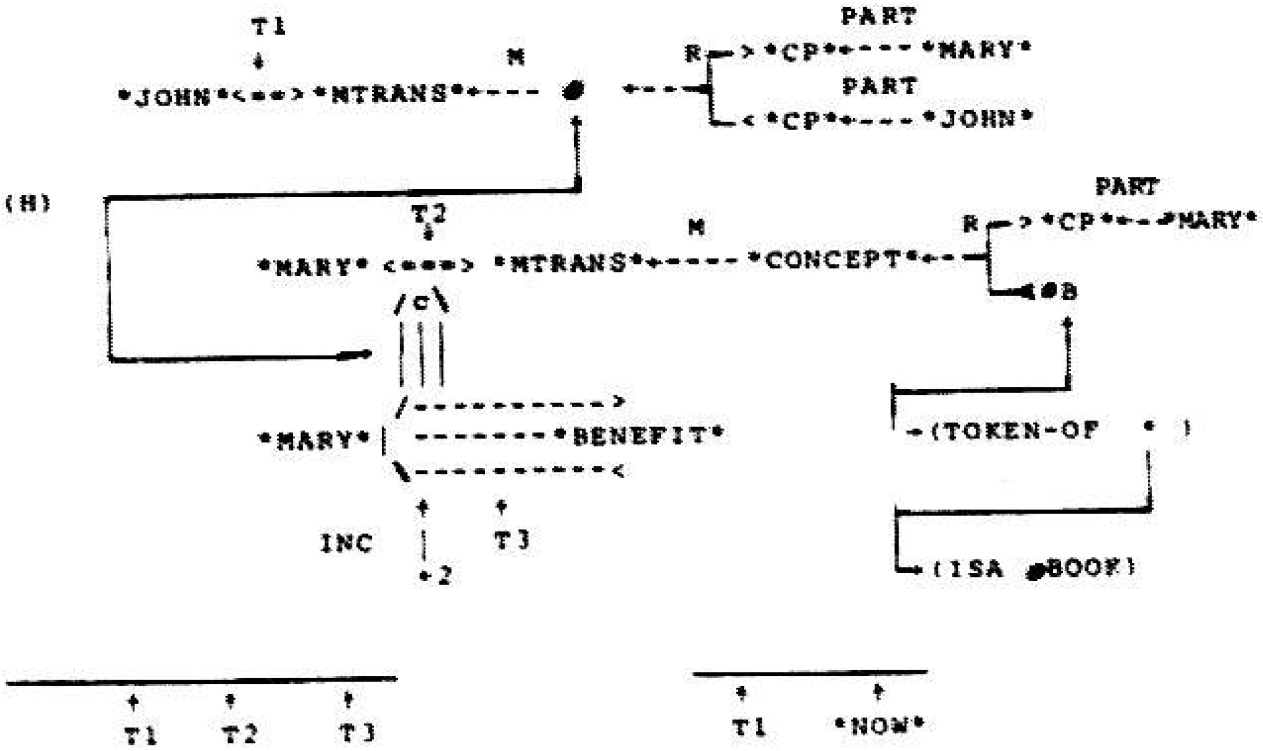
3.3 Conceptual processing: an example

Conceptual representation is really designed to separate meaning from language. We should therefore expect to see it put to greatest advantage in that portion of a linguistic task which involves operations on meaning rather than language -- namely, memory processes. Consider a conceptually based system operating in a dialogue format. We shall follow through a sample exchange and see how the breakdown of language into non-linguistic units, the same units in which knowledge and beliefs are stored, affects the process.

HUMAN: John advised Mary to read the book.

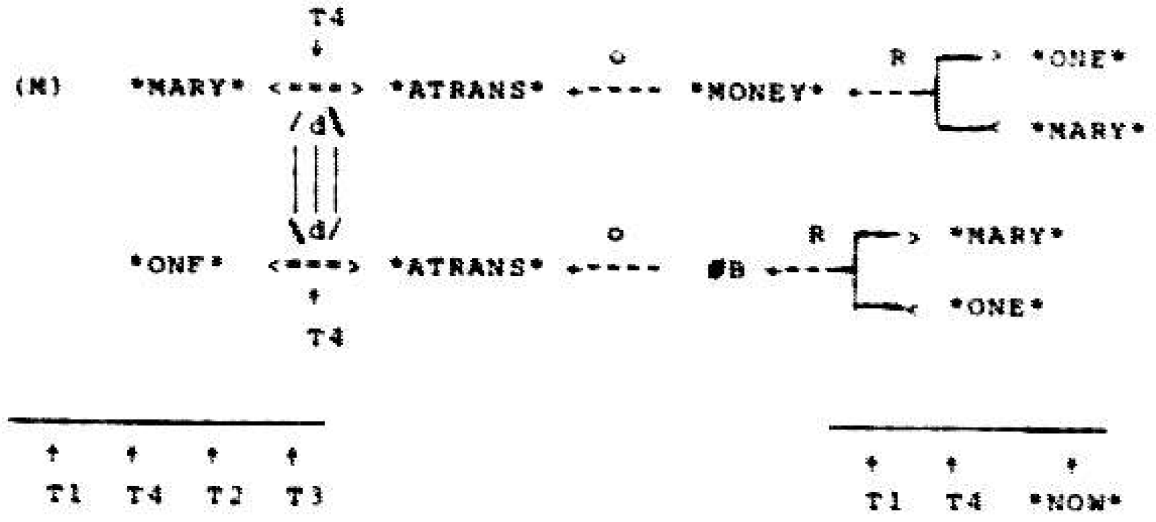
MACHINE: Did Mary buy a copy of the book?

A conceptual analyzer would produce a meaning representation of the input, which would look something like:



(The unit #B is a pointer to a memory node, which represents a token of a concept such as "J.L. Seagull" which is itself a member of the class #BOOK. An 'Englishy' version of (H) would be "John communicated to Mary that if she were to transfer information from a particular book to herself, this would result in some sort of benefit for her.")

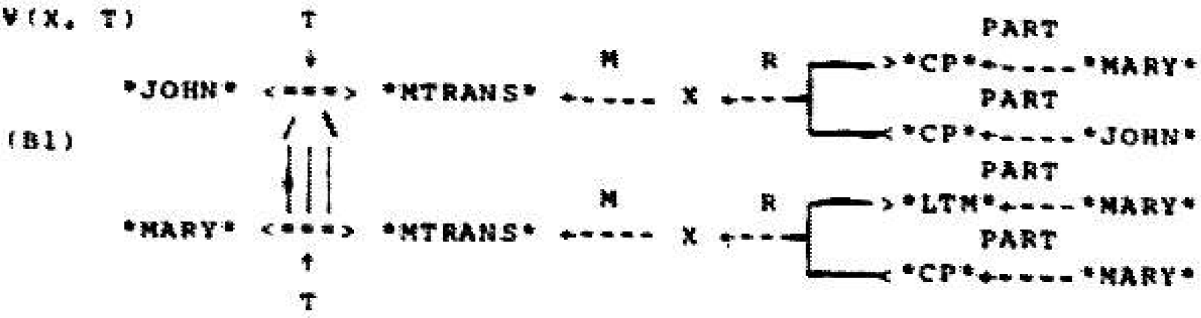
The output would be produced by questioning (verbally) the validity of one of the inferences made from (H) -- in this case, the inference whose conceptual representation is:



(which is approximately "Mary transferred some money to someone, and that someone transferred a particular book to Mary, and these two events mutually caused each other.")

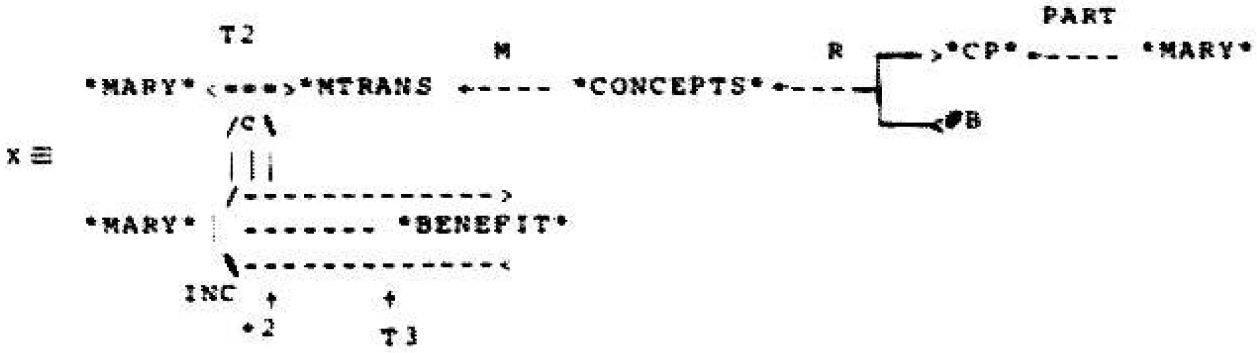
The production of (M) from (B) -- the 'what-to-say' problem -- is the problem which conceptual representations are designed to facilitate. Processing is done entirely at a conceptual level. The linguistic problems of ambuigity and multiple representation are eliminated before this process begins.[2]

Now suppose the machine has a belief that "Mary believes John". This is not stored linguistically, but as a conceptual causal relationship:



("Any information communicated by John to Mary will result in Mary's storing that information in her *LTM*")

This belief can be used as an inference rule. (H) matches the antecedent of (B1), where the bindings $T \equiv T1$ and



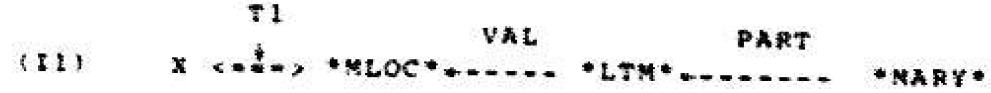
have been made. Therefore the consequent of (B1), CR

3
("Mary stores X in her LTM") , can be inferred.

Note that the machine does <u>not</u> need a separate belief (inference rule) to cover each type of communication (tell, advise, warn, etc.), since they all get converted to *MTRANS* to which (B1) could apply. A representation based on language units would need either a separate rule for each of these verbs, or a rule for 'communication verbs' with appropriate senses marked [+communicative]. The problem of keeping the number of such markers finite appears difficult to surmount. A system which captured sufficient generalities to keep the number of markers reasonable would probably end up looking very much like a conceptual system which broke down information

into its conceptual form each time it was used instead of storing it in that form.

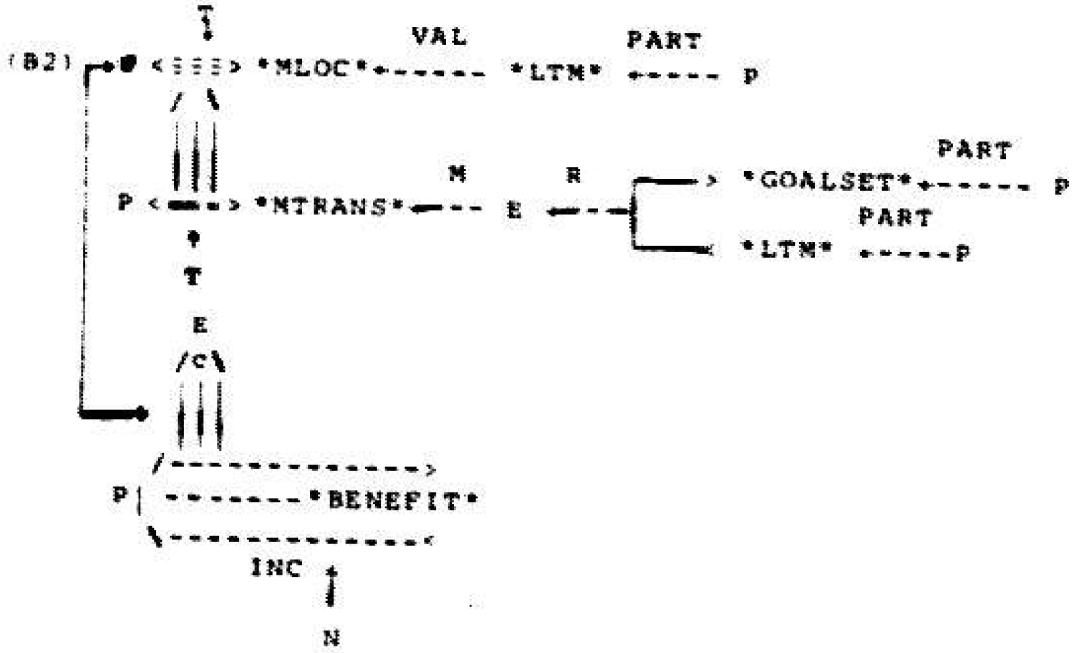
An immediate consequence of CR ("Mary stores X in her LTM") is CR("X is located in Mary's LTM")



where X has the binding found earlier, i.e., X is the CR ("Mary could benefit from reading the book") This inference could be expressed verbally as "Mary wanted to read the book".

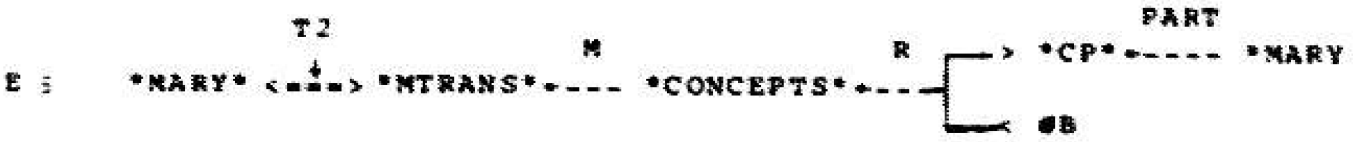
Another belief the machine might have is CR ("If someone believes that some event will benefit him, he makes that event a goal"):

∀ (P, E, T, N > 0)

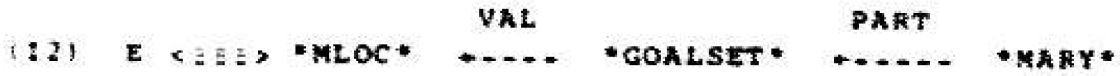


(Here '*GOALSET*' is that set of things which a person currently is acting to bring about.)

(I1) matches the antecedent of (B2), with P = *MARY* and



so the program can infer CR("Mary puts E into her goalset"), and its immediate consequent, CR("E is in Mary's goalset"):



Now all the beliefs about how a person's behaviour is affected by the presence of a goal in his goalset come into play. Among these is the fact that actions sufficient for achieving the goal may be added to the goalset and carried out. This may involve using inference rules 'backwards' -- if the result is in the goalset, then the antecedent action may be taken. One enabling condition for *MTRANS*ing information from a book is being in possession of the book. But this is not an action which Mary could take, so the machine may infer that she added that to her goalset as well.

There are many ways to come into possession of an object. Which is most appropriate depends, among other things, on the nature of that object.

For books one particular way is to go to the library, and the machine might wish infer CR("Mary went to the library") and question that inference. For a much larger class of objects the natural way to come to possess them is through an *ATRANS*. If no reasonable way exists for Mary to *ATRANS* the book to herself (such as by stealing it) then it may be inferred that she adds to her goalset a goal for someone else to *ATRANS* the book to her. One way to cause that is to *ATRANS* some money to someone who has the book. Through such a chain of reasoning (N) might be reached.

Of course many problems have been overlooked in this quick analysis. A real memory model will have to consider strength of beliefs and probabilities associated with can-cause relations. And at every stage of this deduction, alternative paths could have been followed. Some would lead to interesting results, others would not. Effective management of such a search is a classic A.I. problem, but not one which we shall touch upon here. Even the question of knowing whether a given inference is 'interesting' does not seem to have any simple solution. The main points of this sample analysis are:

A) In providing a linguistic response to a linguistic input, a great deal of processing which is not inherently linguistic takes place.

B) A representation based on linguistic units could perform these processes. However, the multiple representation problem alone would expand both the search space and the necessary base of inference

> rules tremendously These problems would be aggravated by the problem of leaving 'similar' meanings implicit in the representation as they are in language, since the similarity could be made explicit only through more inference and deduction.

Conceptual representation provides a framework within which this non-linguistic processing can be formalized. It is specifically designed to avoid multiple representations of meaning and to explicitly represent related meanings. The conceptually based memory should require fewer rules in its rule base to perform a given set of inferences than would a memory based on some 'shallower' representation. This not only saves space, but since fewer rules will be applicable to a given structure, the conceptual memory will have a smaller 'inference space' branching factor. Of course, there is always the possibility that this advantage could be offset by the necessity of performing a deeper search with a conceptual memory to reach a given inference. Unfortunately, no examples of conceptual and non-conceptual memories with reasonably broad and comparable inference domains exist. Thus no data is available which might shed more light on the nature of this breadth-depth tradeoff.

3.4 Summary

We have presented a system for conceptual representation and examples of its use to represent meanings encoded in natural language. We have seen how a conceptually-based memory model might operate with the result of a conceptual analysis of an English input. This operation would take place in a language-free domain and would result in a conceptual response to the input.

Perhaps the greatest price paid for the benefits of conceptual representation is the necessity of performing language generation from a non-linguistic base representation. The performance of this task by BABEL is described in the next three chapters. Among other things, the existence of a program like BABEL demonstrates that conceptual representations do not break down information so far as to render it inexpressable in language. Conceptual generators are indeed feasible; in building them, a great deal can be learned about the nature of language generation, about the relation of syntax and meaning, and about the relationship between linguistic knowledge and conceptual knowledge.

CHAPTER 4

WHAT BABEL DOES --- HOW BABEL DOES IT

BABEL is the generative component for a conceptually based language processor. More specifically, BABEL is a process for carrying out a representational change -- from meaning structure to natural language sentence. The only natural language we shall be concerned with is English. We shall indicate later what portion of the generative process is really English dependent, and what portion is interlingual -- in other words, what must be changed or added to enable BABEL to produce realizations in languages other than English.

Although other generative systems also perform transformations from underlying representations to English, we noted several reasons why these were not applicable to conceptual generation:

A) Syntax based representations (like Friedman's) utilize units such as Noun Phrase, Verb, Auxiliary, etc. in the underlying structure. BABEL STARTS WITHOUT A SYNTACTIC REPRESENTATION OF THE SENTENCE TO BE GENERATED.

B) Semantics based representations (like Simmons'), even if they can eliminate syntactic relations, still incorporate linguistic units in the form of word senses. BABEL STARTS WITHOUT KNOWLEDGE OF THE WORDS TO BE USED IN THE SURFACE SENTENCE.

We can recognize at least three major problems which must be solved in transforming a conceptual representation

into surface English:

(i) Words must be chosen to use in the sentence. These should be the words (of those in the program's vocabulary) which 'best' convey the meaning represented by the conceptual structure.

(ii) The words must be tied together by English syntax relations (or relations from which the syntax can be produced).

(iii) The words and relations must be linearized to form an English sentence.

4.1 Word Selection

Consider first the problem of word selection. By far the most interesting of the words to be chosen (at least with respect to English) are the verbs, since they generally carry a large amount of conceptual information which is spread throughout the underlying structure [1]. But this information is not marked in any way at the conceptual level as being relevant to verb selection. BABEL must somehow notice the presence of the relevant information units and realize that they can be encoded into an English verb.

Let us look at some examples to better understand this problem:

"John drinks milk"

(C4-1) *JOHN* <==> *INGEST* <------ *MILK*
o

In order to generate an English realization from this conceptualization, the fact that *MILK* is a FLUID is of interest, since English makes verb distinctions on the basis of physical properties of INGESTed objects. That is, an INGEST event may be realized with 'eat', 'drink', 'inhale', or one of several other verbs based on the nature of the conceptual OBJECT. However, in

"The bear eats fish"

(C4-2) *BEAR* <===> *INGEST* <----- *FISH*
o

it is not important that BEARs are ANIMALs and not HUMANs. However, to generate a German realization of (C4-2) the distinction is important, since German makes a differentiation which English does not. (German uses the verb 'fressen' to describe eating when done by an animal, but the verb 'essen' when a human agent is involved.)

Although the fact that *MILK* is a FLUID is relevant in (C4-1), it is irrelevant in

"John put a cup of milk in the refrigerator."

(C4-3) *JOHN* <===> *PTRANS* <-o- *CUP* -D-> *REFRIG*
CONT
MILK

Thus the relevance for generation of a conceptual pattern or relation is dependent:

A) on the language chosen (examples (C4-1) and (C4-2)), and

B) on the conceptual context in which it occurs (examples (C4-1) and (C4-3))

In general, every verb (actually, every verb sense) has associated with it a set of Defining Characteristics, or DCs. These are predicates which must be satisfied by a conceptual representation in order for it to be realizable using that verb. To make the notion of DCs clear, we present some examples, each consisting of

1) an English verb

2) an English sentence which should put across the sense of the verb we are interested in

3) a 'skeletal' conceptual dependency representation for that sense

4) the associated DCs

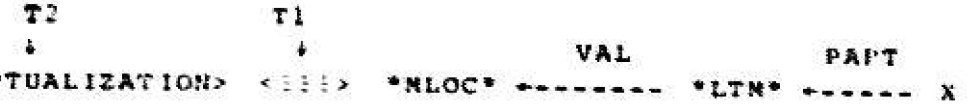
(V1) DRINK as in "Umpires should drink carrot juice."

DCs:
i) structure of the representation is an EVENT
ii) <ACT> = *INGEST*
iii) <OBJECT> has the property FLUID

English provides another sense of "drink", as in "U.S. Grant drank even more than most Presidents".

This sense has the same DCs as listed above, but requires that the OBJECT be the conceptual unit *ALCOHOL* (which of course, is the substance with the English name "alcohol"). Since *ALCOHOL* has the property FLUID, the DCs of this sense of "drink" are a special case of the DCs of the more general sense. For the generator this means that any meaning expressable by the more specific sense could also be expressed using the more general one, although possibly at the cost of making additional information (in this case, the ingested substance) explicit in the generated sentence.

(V2) EXPECT as in "Lear expected his daughters to grant his every wish".

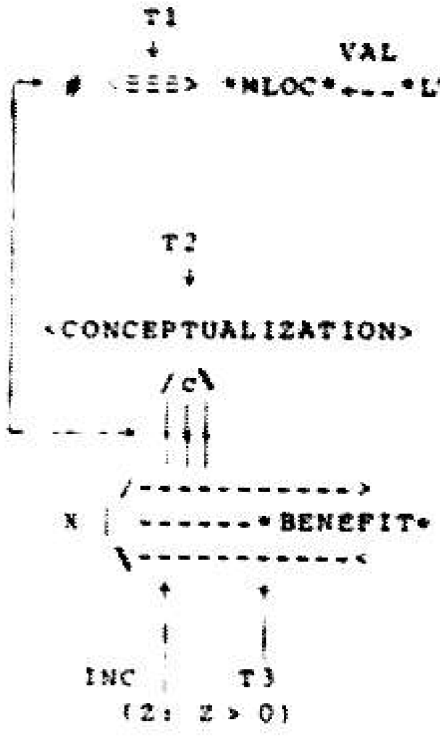


DCs: i) structure is a STATE
ii) <ATTRIBUTE> is *NLOC*
iii) <VALUE> is *LTM*
iv) the times satisfy T1 < T2

It is the last DC which makes "expect" mean "to believe something about the <u>future</u>." Some dialects use 'expect' interchangeably with 'believe'. We can have BABEL speak this dialect by eliminating the fourth DC, which will permit sentences like.

"I expect he is at the race track."
"I expect the butler did it."

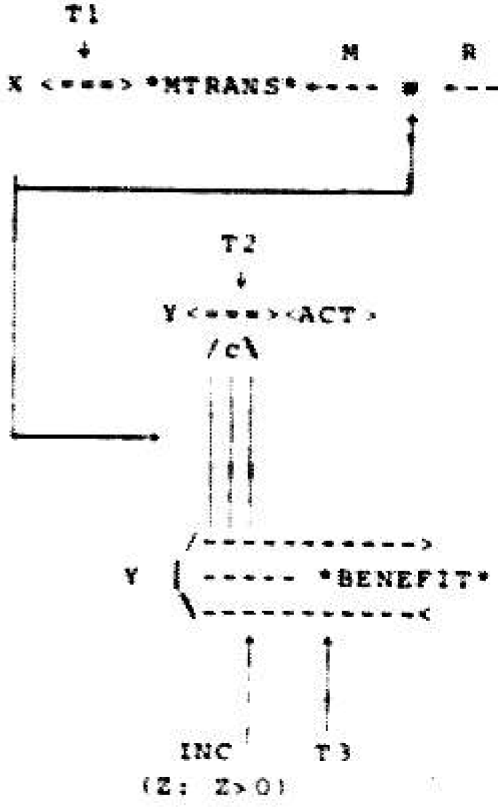
(V3) WANT as in "Lady Macbeth wanted to become the queen of Scotland"



DCs:
i) structure is a STATE
ii) <ATTRIBUTE> = *MLOC*
iii) <VALUE> = *LTM*
iv) <CONCEPT> is a <:C structure
v) <RESULT> of <CONCEPT> is a STATECHANGE on the *BENEFIT* scale by a positive increment
vi) the two positions filled by X in the skeleton match (i.e., the 'wanter' is the 'benefitee'
vii) the times satisfy T1 < T2

Some very interesting things happen when conditions i-vi are satisfied but T2 comes before T1. Remember that the :C (can-cause) relation says nothing about the actual occurrence of its antecedent. In the situation we are hypothesizing (T2 < T1) it becomes important whether X believes that the <CONCEPTUALIZATION > has not occurred, or believes that it may have occurred. If X knows it did not occur, then the verb "wish" is appropriate -- "Alex wishes he had read the book". On the other hand, if X is 'in the dark' as to the actual occurrence of <CONCEPTUALIZATION>, then the verb 'hope' may be chosen -- "Alex hopes his sister read the the book".

(V4) ADVISE as in "Polonius advised Leartes to be truthful."



DCs:
i) structure is an EVENT
ii) <ACT> = *MTRANS*
iii) structure of <MOBJECT> is < C relation between EVENT and STATECHANGE
iv) this STATECHANGE is on the *BENEFIT* scale with a positive increment
v) the two positions filled by X in the skeleton match
vi) the three positions filled by Y in the skeleton match
vii) the times satisfy T1 < T2

If predicate (vii) is not satisfied, the use of the verb 'advise' is prohibited. If T2 <T1 the realization must become something like "...should have..." or "...would have benefitted from ..."

Suppose, now, that all the predicates except (vi) were satisfied. In particular, suppose the "Y" in the STATECHANGE part of the relation were changed to an X. We would have a skeleton expressing "X communicated to Y that X would benefit if Y (did something)." This might get realized as "X requested . . ." or "X asked Y to . . ."

We have shown how local changes in conceptual structures may result in vastly different surface realizations. Defining Characteristics are properties which must be met if a word is to be utilized in realizing a conceptualization. Thus we cannot expect to choose words by defining a 'matching metric' and choosing a word whose DCs are the 'best match' to the idea being expressed.

The DCs we have found useful in choosing words fall naturally into two classes. Class I predicates perform pattern matching within the stimulus conceptualization. These include tests for the identity of two conceptual fields, e.g., a predicate ACTOR = RECIPIENT which would be needed to distinguish "take" from "give". Other predicates in this class test for the presence of particular conceptual elements in the meaning representation -- e.g., is the ACT of a conceptualization *ATRANS*? -- or test its structure -- e.g., is it of the form EVENT-CAUSE-EVENT?

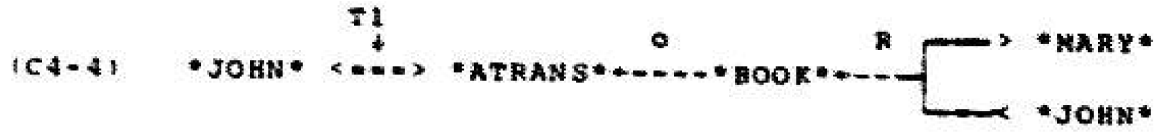
Intraconceptual pattern matching is itself a sufficiently powerful tool to make a crude choice of words to express a conceptualization. But many of the most interesting distinctions between words are encoded not in the structure but in the content of their conceptual representations. Class II predicates test properties which are conceptual in nature. They all involve interaction with the memory model.

The simplest example of such predicates is the use of what is generally considered categorical information. It was shown earlier that the fact that *MILK* is a FLUID is important to the generator in certain instances. *MILK*, when it appears in a conceptualization, is not an English word, but a pointer to a node in memory. And FLUID is not a property shared by the English word "milk" and the German "Milch", etc., but a property of the concept *MILK*. Thus this information is not stored as linguistic information in a lexicon, but is stored in the memory and accessed through the node *MILK*. There are two reasons for such a design. From a generative viewpoint, it turns out that in choosing a verb for a meaning structure BABEL may need to access the information in this way. In distinguishing between "eat" and "drink", for instance, the distinction is made on the basis of whether the OBJECT of *INGEST* is a FLUID. This OBJECT of course is a conceptual, not a linguistic, unit. Even more importantly, this sort of knowledge is also needed in the system for entirely non-linguistic purposes -- e.g., if a substance is dropped on the floor, is a broom or mop the appropriate tool to get? By making properties like FLUID conceptual information, located in the memory model, the information is sharable by language analysis and generation, as well as non-linguistic processes. Categorical information is therefore

NOT a form of LINGUISTIC KNOWLEDGE in a conceptual system.

In addition to categorical information of this sort, the memory is the sole repository of relational information, such as BEFORE-AFTER time relationships. When a conceptualization is passed to BABEL, such relational information is not included unless it is specifically desired that it be expressed. However, linguistic choices may be dependent on this information. We saw in examples (V2)-(V4), for instance, how time relationships were relevant to choosing verbs like "advise", "want", and "request".

Still other linguistic choices are made on the basis of non-linguistic context. Making such choices involves another form of interaction between BABEL and the memory model. Consider.



This can, of course, be realized as

(S4-4) "John gave Mary the book."

But if it is known that there is some time T previous to the time of this event (specified here only as 'past' but

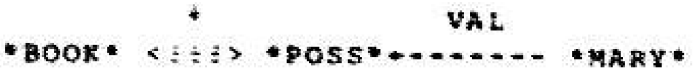
potentially more explicitly given, e.g., "at two o'clock last Saturday") such that Mary was in possession of the book at time T, then (C4-4) may be realized as:

(S4-4') "John returned the book to Mary".

The decision is made on the basis of the context existing in the memory at the time the generation takes place. In this case, the generator passes to memory the request:

FIND:
∃ T0, T0 < T1, such that:

(C4-5)

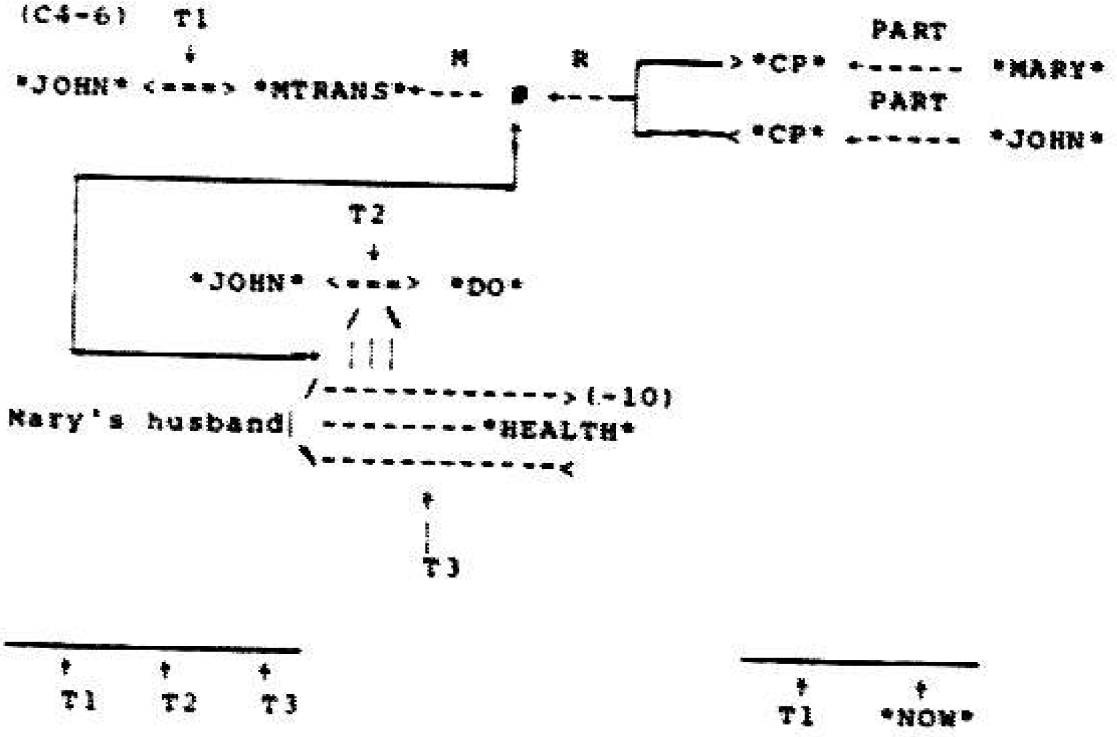


(where T1, *BOOK*, and *MARY* are the same pointers as in the stimulus (C4-4))

i.e., was there a time previous to T1 at which the book was in Mary's possession? If memory finds such a time, (S4-4') may be generated; otherwise, (S4-4) will result[2].

In this example a piece of information about the world in which the generator is operating has been used to make a linguistic decision. English provides many such pairs like 'give-return' which are distinguished on the basis of such knowledge. Examples like 'go - return' and many verbs with 're' prefixes such as 'resubmit', and 'restate' come immediately to mind.

These examples all use information which could reasonably be presumed to be findable in memory rather than requiring deduction. But situations exist in which linguistic considerations require access to deductive capabilities of memory as well as its information retrieval capacity. Consider the conceptualization:



This can be realized as

(S4-6) "John told Mary that he was going to kill her husband".

A reasonable paraphrase might be

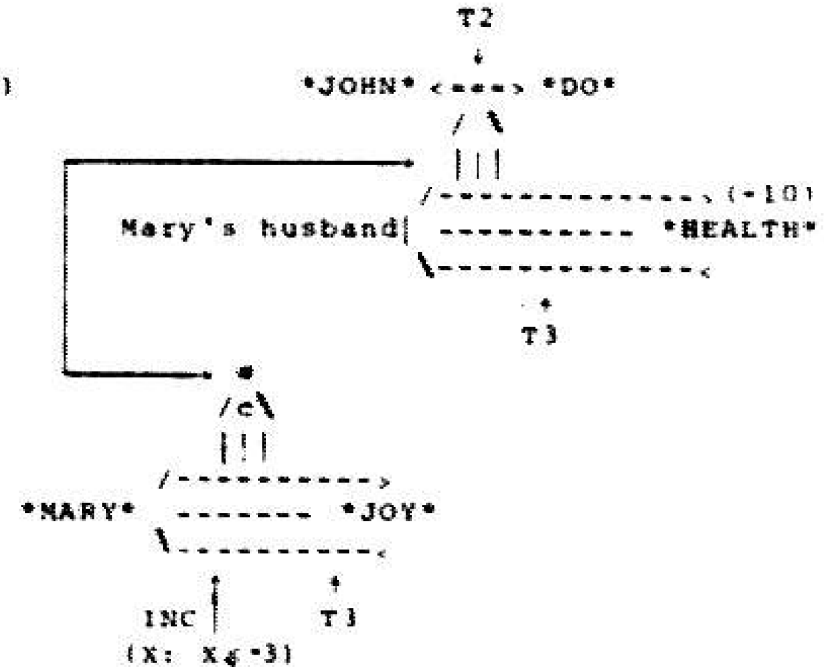
(S4-6') "John treatened to kill Mary's husband".

But one can imagine circumstances in which (S4-6') would be a very poor realization and a much better one would be

(S4-6'') "John promised to kill Mary's husband".

In order to choose between 'tell', 'threaten', and 'promise' BABEL must interrogate its world model. The distinction is made on the basis of whether the NOBJECT of the *MTRANS* could cause the RECIPIENT of the *MTRANS* to become much more unhappy (or much happier). A conceptualization:

(C4-7)



is formed, and if it can be proved then 'threaten' is chosen. On the other hand, if this conceptualization with INCrement (X: X◆3) on the resulting state-change can be proved, then 'promise' may be selected [3].

It is not being claimed that (S4-6'') should be considered a <u>paraphrase</u> of (S4-6). But the BABEL model of generation makes a claim that this is only because

SENTENCE PARAPHRASE HAS GENERALLY BEEN CONSIDERED ONLY IN A NULL CONTEXT. Of course a truly 'null' context would not even permit (S4-6') as a paraphrase of (S4-6). So what is meant here by null context might better be described as a neutral (for a given group) context. (There may exist groups in current U.S. society whose null context would paraphrase "tell . . . would kill husband" with "promise" as often as with "threaten".)

The memory-inference model in the present program is not capable of proving relations of this complexity -- i.e., whether an arbitrary conceptualization describes something which could please or harm a particular individual. Such theorem proving is in fact beyond the current capacities of all language processing systems. Our program resorts to human intervention to answer such questions; a conceptual structure like that above is typed out at the console when the program needs the information and a human informant responds TRUE or FALSE.

It is important to realize that such a capability is <u>not</u> specific to the task of language generation. It is in fact needed to disambiguate the sentences:

"The Nets are threatening to fine Willie Mays"
"The Nets are threatening to win the pennant"

A psychiatric interviewing program would very likely need the ability to analyze what was said to it and determine if it was 'threatening', 'hostile', etc.

The desire to perform such an action has nothing to do with the program's expressing in English the fact that what was said was a threat. Nor does it have to do with performing language analysis, at least in so far as this is defined as transforming language strings into conceptual structures. Since the need for such a capacity can be justified on grounds independent of generation, no unreasonable assumption is being made in making it available to the generator. It demonstrates one interesting interaction between linguistic knowledge -- that English provides a verb "threaten" to describe an information transfer meeting certain conditions -- and non-linguistic capability -- the ability to decide whether a given piece of information has particular implications in a particular context.

This use of the powerful deductive capabilities of a memory model during generation cannot be left undefended. It is certainly not the only way of accomplishing the same ends, and has several ramifications which stand in opposition to previous assumptions about generation. Foremost of these are:

Generation not only fails to be a stepwise inverse of analysis, but is not even a functional inverse -- that is, it is not universally true that ANALYZE(GENERATE (C)) = C.

Now if we consider the context within which language processes occur as well as the words and information being transmitted, analysis and generation do look more like theoretical inverses. Even here, though, there are some differences, due to the fact that the context for analysis includes partial, but not complete, information about the context in which an utterance was generated. The fact that the processes are not stepwise inverses is of greater importance for a performance model, since it means that a solution to one problem will not be a solution to the other. A stepwise inverse of BABEL would end up accessing information about the word "trade" in order to analyze "buy". A stepwise inverse of most analyzers would end up making considerations about possession in generating "give a party". Both situations are undesirable. Finally, from a practical point of view, a computer model which forced a human user to understand sentences generated from a fairly limited syntax would be making no unreasonable demands. A model which forced a human to produce only such sentences would be.

A conceptual analyzer must encode both the event being related by a verb like 'return' and 'promise' and the connotations inherent in their use. If it did not, it would be impossible to correctly understand statements like

"Bertha threatened to give Norman a kiss"

BABEL may choose these same words to express a conceptualization which encodes only the event being described, however. Thus the analysis of a generated sentence may contain more information than the conceptual source from which the same sentence is to be generated.

It therefore makes no sense to speak of THE conceptual representation of a word-sense.

It was always realized that a given sentence may have multiple interpretations due to syntactic and semantic ambiguity which must be resolved by the use of 'context'. It was also realized that the mapping from meaning representation to language representation was one to many; there are many ways to say the same thing in a given language. In this model, the set of ways of expressing something is DEPENDENT ON THE CONTEXT in which the generation is taking place.

The notion of sentence realization takes on a new character, being seen as a linguistic problem which depends on a conceptual context.

It is not the intent of BABEL to provide a 'competence' model <5> of the ideal human speaker's capacity for paraphrase. People have different standards for what they consider

paraphrase; furthermore, a given individual will accept different paraphrase sets for a given utterance in different contexts. BABEL is concerned with the problem of finding linguistic encodings of conceptual information in conceptual contexts. This is certainly related to paraphrase, but is not meant to be a formalization of what linguists and speakers mean by paraphrase.

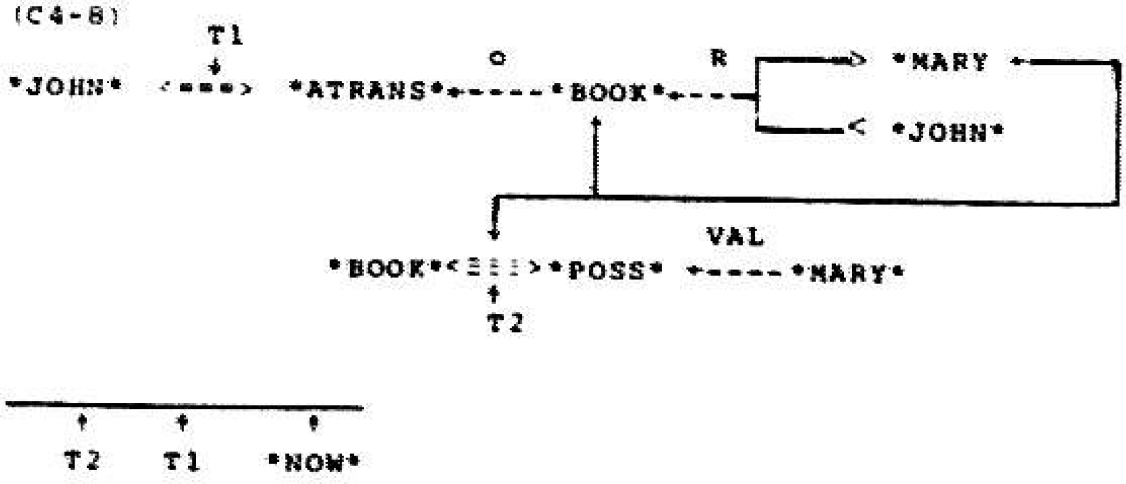
The importance of a conceptual context cannot be overemphasized. It is necessary to draw conceptual diagrams as if they were isolated entities. Such a presentation is sufficient for most explanatory purposes. But the commitment to a conceptual representation includes a commitment to an associative memory storing these conceptualizations and an inference mechanism operating on them. In such a system no conceptualization is truly isolated.

It might be assumed that

"John returned the book to Mary"

should be generated from the conceptual structure:

(C4-8)



which is something like what we would expect an analyzer to produce from the sentence. This seems more natural than generating the sentence from a representation which encodes only the ATRANS event, particularly in view of the associative memory assumption, since at generation time the links between occurrences of *MARY* and *BOOK* in the *ATRANS* conceptualization and their occurrences in the *POSS* conceptualization already exist. It would only be necessary for the 'WHAT-TO-SAY' device -- the process which builds or selects a conceptualization to be expressed -- to choose to attach these links to the conceptual structure being built in order to produce structure (C4-8) for expression.

This course has been rejected because of a basic assumption that the WHAT-TO-SAY decision should be made on non-linguistic grounds. Given that some motivation exists for expressing the *ATRANS* conceptualization, the WHAT-TO-SAY process will fan out across associative links deciding whether associated conceptualizations should be expressed as well. For instance it might be necessary to give further information about *BOOK*, such as the fact that it is about mathematics, to avoid referential ambiguity. But of the potentially enormous set of associated conceptualizations, what are the appropriate grounds for choosing the *POSS* relationship?

The grounds are almost certainly LINGUISTIC; English provides a compact way of expressing this relationship, namely, the word 'return'. English does not provide a concise way of saying that an object was 'purchased at a drugstore'; thus we do not expect such information to be mentioned generally. The exchange:

Q: "Has Fred read the book yet?"
A: "No, he lost the book, which he bought at the drugstore."

would be unusual, even though the information about the book's source may be new to the questioner. On the other hand, the exchange:

Q: "Does John still have Fred's math book?"
A: "No, he returned the book to Fred."

is perfectly acceptable, even though the use of 'returned' instead of 'gave' clearly provides no new information to the questioner. In fact, it seems much more natural to use the redundant 'return' in this case.

Since BABEL has as one of its underlying assumptions a restriction against language dependence in the WHAT-TO-SAY mechanism, the course of 'discovering' linguistically relevant information during the course of generation has been adopted.

In going from meaning representation to sentence, a great deal of compacting is taking place. A single word, like the verb "poison", may encode a large conceptual structure ("to do something which causes someone to ingest a poisonous substance"). BABEL must recognize such conceptual

patterns which have special word encodings. The process of word selection is basically one of putting back together pieces of conceptual structure which the target language provides words to express. There are, in general, many ways to accomplish this compacting. In developing BABEL, we have taken it as an axiom that a good generator will maximize the amount of structure encoded in the words it chooses, thus producing the most concise realization possible for a conceptualization.

4.2 Syntax Representation

Although language <u>understanding</u> may not require the detailed syntax analysis predicated by most existing linguistic models, <u>generation</u> of natural language sentences certainly does require a detailed knowledge of syntax. Since the study of syntactic rules is not the focus of this work, and since a great deal of work has already been done in this area, it was decided to design BABEL so that it could employ an existing formulation of English syntax.

The two best models now available for dealing with the syntax of natural language are transformational grammar, as developed by Chomsky et al., and the APSTNs of Woods, et al. Either approach could have been adopted. Transformational deep structures were rejected because the tree format which they assume does not naturally arise in

conceptual generation. The tree representation is a direct result of the production (or description) of sentences by a context free phrase structure base grammar. Since such a grammar has no place in a conceptually based system, there is no natural source for tree structures.

Simmons' work, described in Chapter 2.4, has shown that AFSTNs can be used to generate natural language from networks which include the words to be used in the sentence and sufficient structural information to deal with natural language syntax. Such networks turn out to be a much more natural intermediate step for BABEL than do phrase markers. We can take the conceptual structures to be realized, convert them to networks, and then linearize the network with an AFSTN.

What BABEL does is to tie together the words it chooses and put them into a SYNTAX NETWORK. Like semantic nets, these syntax nets can be represented as a set of 'structure' nodes (named G1, G2, G3, . . .). With each node will be associated a set of relation-value pairs. The relations are elements of a small set of syntactic relationships handled by the grammar; the value of a relation may be another structure node, a lexical entry pointer, or one of a set of terminal grammar elements. As an example consider the sentence

"John advised Mary to read the book"

which would be generated from the syntax network:

(N4-1)

G1:	LEX	ADVISE	G4:	LEX	READ
	ACTSBJ	G2		ACTSBJ	G3
	OBJ2	G3		OBJ	G5
	INF2	G4		TENSE	PAST
	TENSE	PAST		MOOD	INDICATIVE
	MOOD	INDICATIVE		VOICE	ACTIVE
	VOICE	ACTIVE			
G2:	LEX	JOHN	G5:	LEX	BOOK
				DET	THE
G3:	LEX	MARY			

This network consists of five nodes (G1-G5). The syntax relations included are: TOK, ACTSBJ, OBJ2, INF2, TENSE, MOOD, VOICE, DET. Five lexical entry pointers, ADVISE, JOHN, MARY, READ, BOOK, and THE are present, and the only terminal elements used are PAST, INDICATIVE, and ACTIVE.

THE LEXICAL ENTRIES 'ADVISE' and 'READ' DO NOT CORRESPOND TO WORD SENSES as they do in Simmons' networks. The sentences

"The Lone Ranger mounted Silver and rode off"
"The lepidopterist mounted his Danaus neotippe"

will have the same lexical entry pointer MOUNT as the value of a LEX relationship in their syntax networks. The notion of word sense still exists, as will be seen shortly; its existence, however, is not at the lexical level. Only syntactic information is contained in BABEL's lexicon.

Such information as irregular past and perfect forms for verbs and plurals for nouns will be stored in a lexical entry; the fact that "mount" has at least two distinct meanings will not be found in the lexicon.

A second major theoretical difference between these syntax networks and Simmons' semantic networks is the set of relationships allowed. THE SYNTAX RELATIONS OF BABEL's NETWORKS HAVE NO CONCEPTUAL SIGNIFICANCE whatsoever. JOHN, MARY, and BOAT will have different syntax relationships to BUY and SELL in the sentences

"John sold the boat to Mary"
"Mary bought the boat from John"

although a semantic network might assign the same roles in both sentences <35>.

In BABEL's syntax nets the relationships between embedded sentences and embedding sentences are chosen on syntactic grounds. Thus

"John told Mary Bill drank the beer"

will have as its syntax net:

G1:	LEX	TELL
	.	.
	.	.
	SNT	G4
	.	.

G4:	LEX	DRINK
	ACTSBJ	G5
	OBJ	G6
	.	.
	.	.

since SNT is a relation which causes its value structure (in this case G4) to be realized as an entire sentence. However,

"John advised Mary to read the book"

had a network (N4-1, above) which contained the structure G4 for

"Mary read the book"

embedded as an INF relation. This will result in the verb string of G4 being transformed into an infinitive ("to read") and having its ACTSBJ ("MARY") 'deleted'. However the syntax network for

"John wanted Mary to read the book"

is

G1:	LEX	WANT	G3:	LEX	READ
	ACTSBJ	G2		ACTSBJ	G4
	INF2	G3		OBJ	G5
	.	.		.	.
	.	.		.	.
	.	.		.	.

INF2 is a syntax relation which, like INF, performs an infinitive transformation on a verb string. The generation grammar also skips the ACTSBJ of a structure embedded in an INF2 relation if it matches the ACTSBJ of the structure to which it is related by INF2. The two fail to match in the above example but would match in the networks for

"John wanted to read the book"
"John expected to get a raise"

The syntax relations have two basic effects on the generative process. They determine transformations, like the infinitive and optional deletion transformations just mentioned, and they determine the left-right order of realization of noun and verb phrases and embedded sentences in the generated language string. It is necessary to have a structure for "Mary" related as an OBJ2 to a 'GIVE' structure to generate

"John gave Mary the book"

but the structure "to Mary" related as an IOBJ to a 'GIVE' structure to generate

"John gave the book to Mary"

In a semantic network we would expect to find MARY in the same case, say GOAL, in both examples. Having different relationships in the syntax networks enables the generation grammar to handle the different word orders simply.

Conceptually, of course, both examples have the same representation, with Mary being the RECIPIENT. Similarities and identities in meaning must be expressed in the conceptual representation; the syntax networks are used as an intermediate representation in the generative process and need not meet this requirement.

4.3 Syntax Net Production

BABEL's syntax nets, then, are related to Fillmore's early proposals on case grammar. The basic net consists of a verb and a set of relation-value pairs which relate noun phrases and embedded sentences to the verb. The grammar has the job of choosing 'subjects', 'direct objects', etc., of performing 'deletions' and providing for 'agreement', and carrying out other syntactic functions.

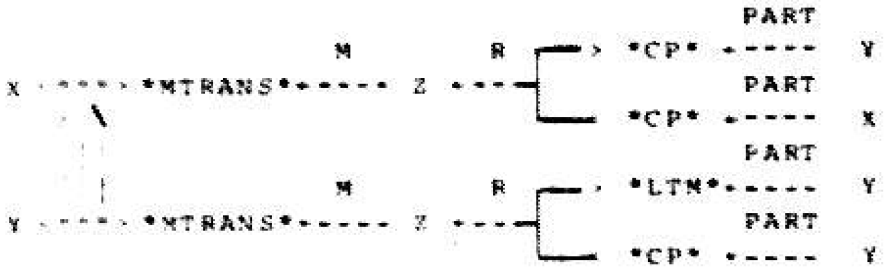
The key to producing a syntax net is realizing that once a verb has been chosen, an entire syntactic framework becomes known. For example, if 'convince' -- as in "The conspirators convinced Brutus that Caesar was dangerous" -- is the verb chosen, we would know that the sentence being generated must have

1) a Noun Phrase which will become the 'subject' (in most cases)

2) a Noun Phrase which will become the 'direct object'

3) an embedded sentence

But in fact even more is known. 'Convince' would only be chosen because a conceptual representation which satisfied its DCs was being realized. This would be a structure like:



Once it is known that 'convince' can be used to realize this structure, it is known immediately that

X must be made into the 'subject' noun phrase (the convincer)

Y will become the 'object' (the convincee)

Z will become the embedded sentence

In other words, once a verb has been chosen, the form of the syntax net to be created is known, and furthermore mappings between pieces of the syntax net and pieces of the conceptual representation are known as well. Thus the creation of the remainder of the net is a very strongly guided process, not a large search.

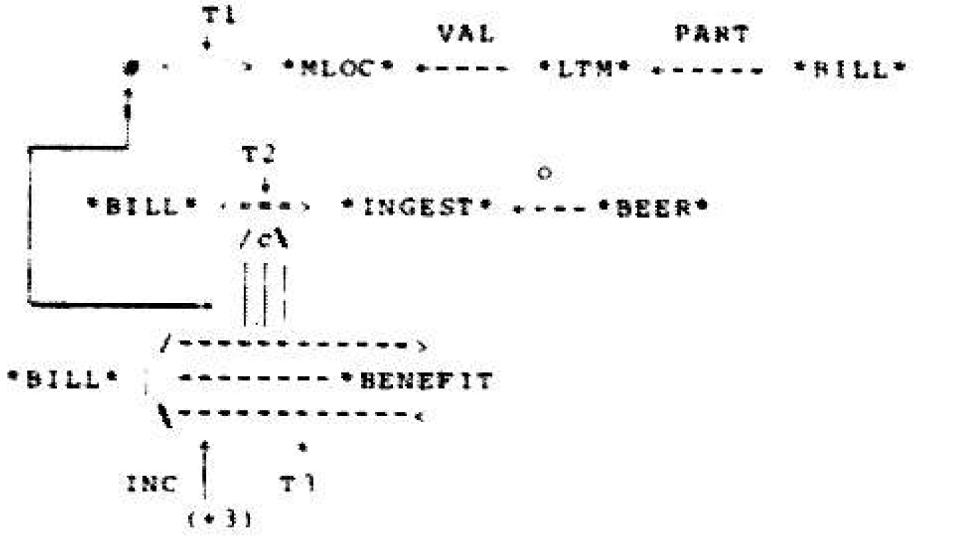
The process underlying BABEL, then, can be summarized as follows:

Choose an appropriate verb (sense)

Use the information associated with the verb to create a syntax network

Use the AFSTN to produce a surface string

For example, starting with the conceptual structure:



BABEL might choose the verb sense WANT1 and produce the syntax network:

N1:	LEX	WANT	N3:	LEX	DRINK
	ACTSBJ	N2		ACTSBJ	N2
	INF	N3		OBJ	N4
	MOOD	INDIC		MOOD	INDIC
	VOICE	ACT		VOICE	ACT
	FORM	SIM		FORM	SIM
	TENSE	PAST			
N2:	LEX	BILL	N4:	LEX	BEER
				DET	SOME

from which its AFSTN could generate the string:

"Bill wanted to drink some beer."

We have seen the sorts of considerations which must be accounted for in the different phases of the process. Several sorts of knowledge, some of it about language, and some of it purely non-linguistic, are needed. Let us move on to the question of how this knowledge can be represented and organized to effect computer generation from conceptual structures.

CHAPTER 5

THE STRUCTURE OF BABEL -- THE ORGANIZATION OF LINGUISTIC KNOWLEDGE

In any large computer program, whether it be a cognitive model, a compiler, or a payroll processor, it is important to maintain a design which distinguishes data from process. In many cases this is done for practical advantage -- the salaries used by a payroll program must be frequently changed while the process which operates on them remains relatively fixed. In other cases, proper design results in a program which is applicable to an entire class of problems rather than a specific instance of that class -- thus the transition from ad hoc compilers for individual languages to compilers embodying analyzers for particular language classes and on to compiler-compilers.

Both of these considerations have affected the design of BABEL. Certainly a component like vocabulary must be permitted to grow independently from the program which operates with it. Furthermore, it is desirable to have a process which provides a basis for the production of surface strings in many natural languages. Thus we have a class of tasks across which some paramaters -- namely, conceptual representation and memory organization -- remain constant, but another, linguistic knowledge, changes drastically.

for this reason, an effort has been made to treat linguistic knowledge as data wherever possible.

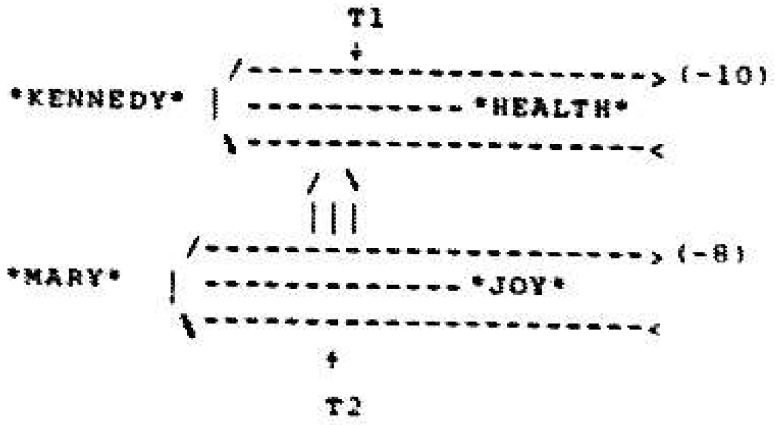
From another viewpoint, MT, Q-A, interviewing, and conversation also form a 'class' of generation tasks. The variable factor across these, however, seems to be what was earlier referred to as the WHAT-TO-SAY problem. There is no reason to view this as a fundamentally linguistic task and the current version of BABEL includes no component which will make the program's behaviour task-dependent across this class of tasks. The program is designed to be usable as part of a more sophisticated system which makes this WHAT-TO-SAY decision in a task dependent fashion.

In a cognitive model there is a third advantage to the separation of data and process which perhaps outweighs the other two. The separation makes theoretical claims about what kinds of knowledge must exist to perform a task and how this knowledge is organized in the human mind. Furthermore, it becomes clear what conceivable sorts of knowledge cannot exist within the framework provided. (For example, BABEL makes no provision for storing the correspondence between the English 'give' and the German 'geben'; nor the fact that 'give' is related to 'have' in any way). And when the processes which operate on these structures are understood, it becomes apparent what sorts of interaction between the various forms of knowledge are

possible and what sorts are NOT possible within the model. An explicit understanding of what sorts of knowledge are provided and the achievable interactions has proved to be a considerable aid in the development of this program.

BABEL can be seen as a collection of linguistic knowledge files accessible by a central generation routine, which is itself activated by and conversant with a combined memory-model and deduction device. Figure 5-1 sketches this organization.

A simple example will demonstrate how each component of the system enters into the generation process. Suppose BABEL is given the conceptualization



T1 T2 *NOW*

to realize. The DISCRIMINATION NETS are used to retrieve a CONCEXICON entry, which might be BECAUSE1 in this case. This entry puts the word "because" into the syntax net and guides BABEL into working separately on the <ANTECEDENT>

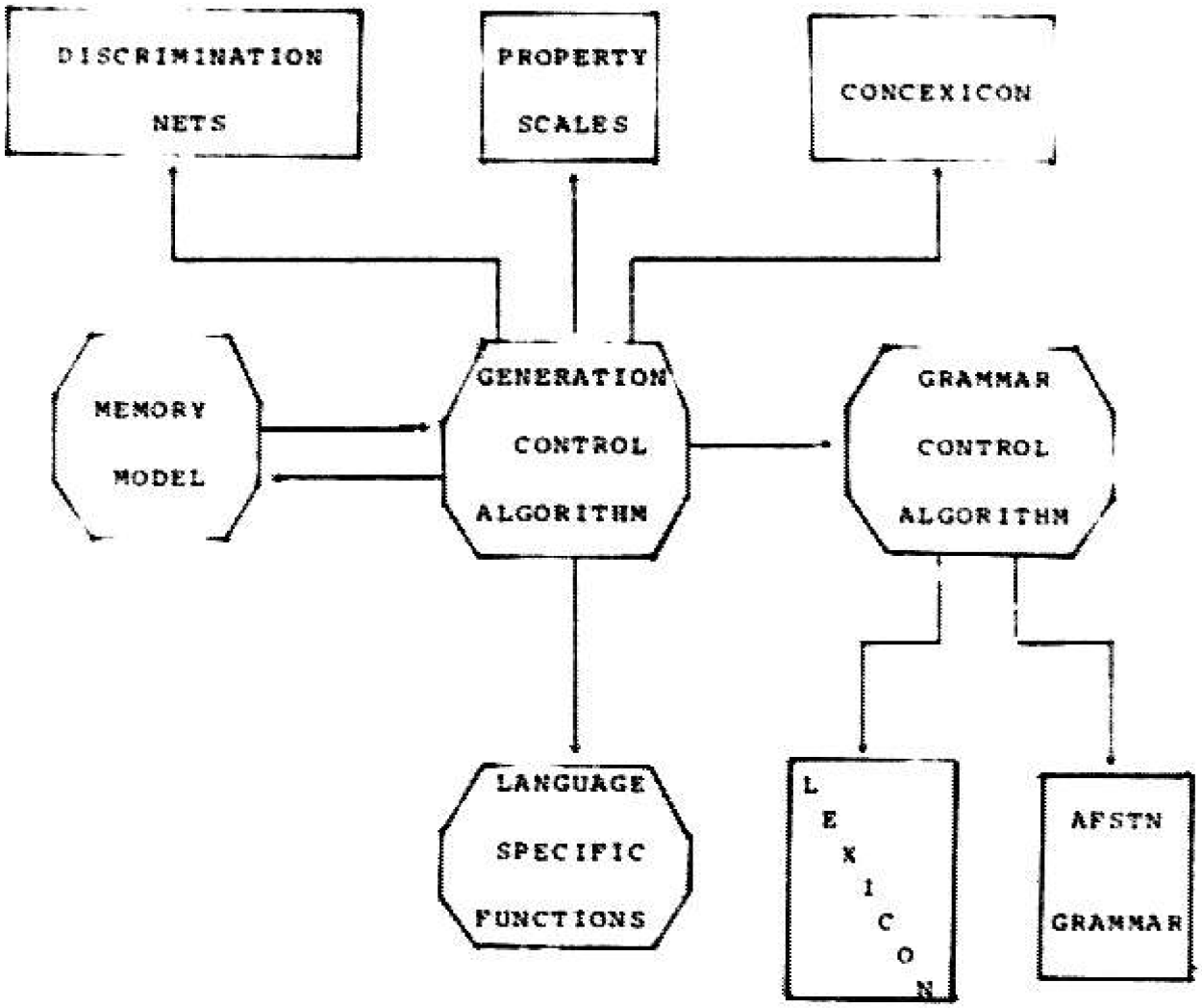

DISCRIMINATION NETS
PROPERTY SCALES
CONCEXICON
MEMORY MODEL
GENERATION CONTROL ALGORITHM
GRAMMAR CONTROL ALGORITHM
LANGUAGE SPECIFIC FUNCTIONS
LEXICON
AFSTN GRAMMAR

FIGURE 5-1

and <RESULT> conceptualizations. The DISCRIMINATION NETS retrieve CONCEXICON entries DIE1 and BECOME1 for the respective conceptualizations, which results in the verbs "die" and "become" being added to the syntax net and in LANGUAGE SPECIFIC FUNCTIONS being applied to determine tense and other information. In processing the CONCEXICON entry for BECOME1, the SCALES are consulted and the word "depressed" found from the elements in the <RESULT> conceptualization. Both the discrimination nets and the language specific functions may require action by the MEMORY MODEL. A complete syntax net is passed to the GRAMMAR CONTROL ALGORITHM, which forms the surface sentence, looking in the LEXICON for the past tense form of "become" in the process. Finally, the sentence "Mary became depressed because Kennedy died" is produced.

DISCRIMINATION NETWORKS

It was emphasized previously that one major linguistic task in generating from a conceptual base is that of selecting individual words to use in expressing the content of a given conceptualization. A word is chosen because the conceptualization satisfies the set of Defining Characteristics (DCs) for some sense of that word. A conceptual generator must therefore know the DCs for the words it deals with. The simplest way to organize such knowledge is to simply have a direct association, as on a LISP property list, between a word and its DCs.

If no further organization is placed on this knowledge, however, the program would be forced to choose words by an enumerative process -- i.e., look at each word and choose the first one whose DCs are satisfied. This approach must of course be immediately rejected on efficiency grounds alone, since it results in an expected retrieval time which (ignoring word use frequencies) increases linearly with vocabulary size.

A linear search has several characteristics, in addition to inefficiency, which make it <u>psychologically</u> undesirable:

1) There is a vast discrepency between retrieval times for various words. It would be desirable to have a scheme which made a word's retrieval time dependent on the

'inherent complexity' of its DCs. This predicts differences in retrieval times for words, perhaps considerable in some cases, but none approaching the linear search discrepencies[1].

3) The enumerative process makes no use of the information from 'failures'. When a predicate in the DC set of some word fails, it should be realized that all other words which have that predicate in their DC set will fail as well. And when a predicate succeeds, it should not be necessary to re-evaluate it later. Furthermore, successful predicates should help guide the search by directing it toward other words which have the same predicates in their DC set.

At least one method of information organization does have the characteristics we desire. It is called the 'discrimination network'.

Discrimination networks, or discrimination trees, have been widely used in models of verbal learning tasks <7,15>. Discrimination nets are generally implemented as binary trees. Each non-terminal node of the tree is associated with a predicate which must evaluate to either TRUE or FALSE. Each terminal node is associated with some 'response' information. In operation, a discrimination net is applied to a 'stimulus' -- in our case, a conceptualization. The predicates in the tree take the conceptualization as a

parameter. The algorithm for applying the discrimination net can be stated as follows:

1. Set CURRENT-NODE to the root node of the net.
2. If CURRENT-NODE is a terminal, go to step 6.
3. Evaluate the predicate at CURRENT-NODE.
4. If the value is TRUE, set CURRENT-NODE to its 'right hand' son and go to step 2.
5. If the value is FALSE, set CURRENT-NODE to its 'left hand' son and go to step 2.
6. Return the response associated with CURRENT-NODE.

The terminology used in connection with these trees has derived from the sorts of verbal learning tasks for which they have served as models. An example of this is the paired-associates nonsense syllable task. Figure 5-2 gives a list of nonsense syllable stimulus-response pairs and a discrimination net capable of finding the correct response for any of the stimuli. Notice that in order to find the correct response, the set of tests performed on the stimulus need only distinguish it from any stimulus requiring a different response, but not from any possible stimulus.

Previous use of discrimination nets has usually modelled the learning as well as the retrieval of information. BABEL contains no provision for acquiring new knowledge during operation; its discrimination trees are treated as data and are not modified by the program. The 'stimulus' presented to a tree is all or part of a conceptualization,

STIMULUS	RESPONSE
GAC	LOR
POB	JIV
GAX	ARG
HOB	NIT

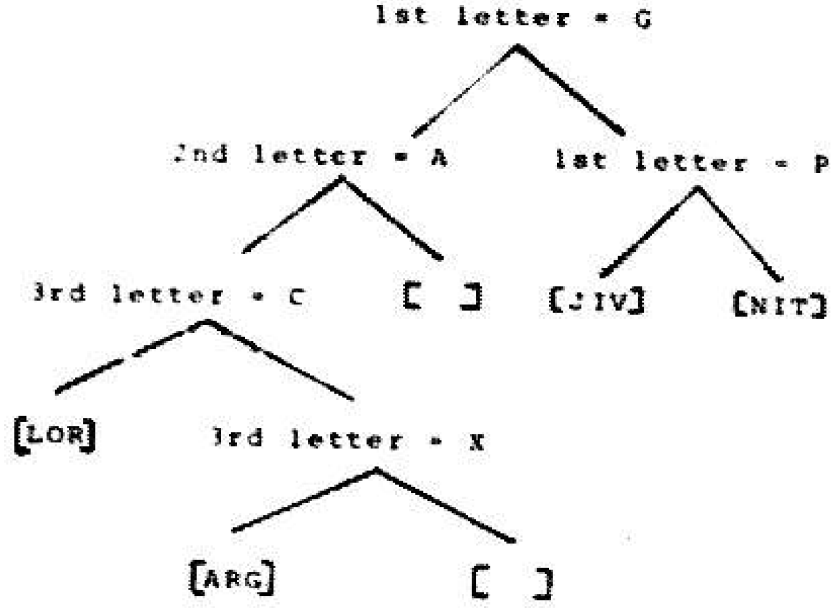


FIGURE 5-2

The responses found at the terminals are lists of 'concexicon' entries. A concexicon entry corresponds closely to the notion of word sense, since each is associated with a particular lexical entry and since ambiguous words will have separate concexicon entries for each sense. A concexicon entry is precisely defined by the attributes associated with it. Details of the concexicon are given in section 5.2.

Feigenbaum <7> learns nets which grow until any two distinguishable stimuli can be discriminated. Hunt's <16> work on concept learning requires nets which test only those features of stimuli which are relevant to the concept being learned. BABEL has nets of the latter sort: only those disctinctions needed for the purpose of generation need to be made. While there are potentially infinitely many patterns and relationships which could be detected, only a finite, and relatively small, subset of these will be interesting for the purposes of generation of a given language. Furthermore, as we saw in section 4.1, even the relationships which affect word choice in a particular language are important only in particular contexts.

One of the major advantages of BABEL's use of these nets is a CONTEXT DIRECTED FOCUS OF ATTENTION. The discrimination trees operate in an environment where responses are not associated with a finite set of known

stimuli. Context directed focus of attention is achieved by building into the trees the knowledge that certain features of stimuli are salient in certain contexts. Somewhat the same idea was used by Simon <37> in his model of human memory for chess positions. Here discrimination nets were used to find common configurations in a complete chess position; the notion of salient features keyed the search for these configurations.

In describing BABEL's discrimination nets, as well as in descriptions of other parts of the program, we shall need to refer to substructures within a conceptualization. Such a reference is called a FIELD SPECIFICATION and consists of a list of elements from the set

{ACTOR OBJECT MOBJECT TO FROM <*> < > <_T <_F
ANT Λ · · C <=D CON VAL PART TIME MODE }

These are the internal names used by the system to refer to roles in conceptual relations as indicated in Figure 5-1. The value of a FIELD SPECIFICATION (FS) applied to a conceptualization is computed as follows:

1) Set VALUE to the entire conceptualization.

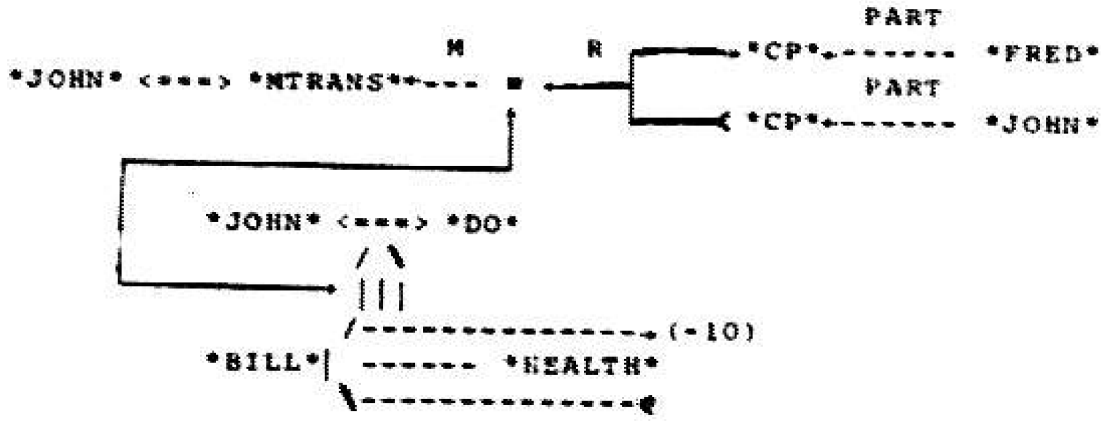
2) In the current VALUE, find the field referred to by the first element of the FS (CAR FS). Make the new VALUE the conceptual structure filling this field.

3) Remove the first element from the FS (FS ← CDR FS)

4) If the FS is exhausted (NULL FS) return the current VALUE; otherwise, go to step 2.

If at any point a field sought in step 2 is not present, NIL is returned as the VALUE.

The value of the FIELD SPECIFICATION (MOBJECT ANT ACTOR) applied to



is the PP *JOHN* .

(As a shorthand, the elements of the <ANTECEDENT> of a causal relation may be referenced without specifying ANT -- thus, the FS (MOBJECT ACTOR) would also reference *JOHN* in the above conceptualization. Of course, no ambiguity is introduced by this convention.) The predicates at the nodes of BABEL's discrimination trees contain FIELD SPECIFICATIONs which apply to the stimulus conceptualization

ROLE NAME	USE
ACTOR	refers to the <ACTOR> in EVENTS; the <CONCEPT> in STATEs and STATE-CHANGEs, unless this is an entire conceptualization
CON	refers to the <CONCEPT> in STATEs in which this field is an entire conceptualization (i.e., when the <ATTRIBUTE> is *NLOC*)
OBJECT	refers to the <OBJECT> in EVENTs
MOBJECT	refers to the <MOBJECT> in mental EVENTs
TO	refers to the <RECIPIENT> in the recipient case, <GOAL> in the directive case
FROM	refers to the <DONOR> in the recipient case, <SOURCE> in the directive case
<*>	refers to the <ACT> in EVENTs
<:>	refers to the <ATTRIBUTE> in STATEs
<:F, <:T	refer to the initial and terminal values of a statechange relation
ANT	refers to the <ANTECEDENT> in causal relations, the first conceptualization of conjunctive relations
∧	refers to the second conceptualization of a conjunctive relation
<, <:C, <:D	refers to the <RESULT> of the corresponding type of causal relationship
VAL	refers to the <VALUE> part of STATE relations

FIGURE 5-3

PART	refers to a PART modification of a structure -- PART i.e., *MARY* in *CP* ------*MARY*
TIME	refers to a TIME modification of a conceptualization
MODE	refers to a MODE modification of a conceptualization

FIGURE 5-3

being 'filtered' through the tree.

It was mentioned in section 4.1 that two basic types of predicates are necessary for distinguishing words. The first are basically pattern matching predicates, and come in nine flavors:

1. (EQU <Field_Specification> Token)

EQU tests whether a particular conceptual token fills a particular field. For instance, one of the defining characteristics for "breathe" is

(EQU (OBJECT) *AIR*)

2. (ID <Field_Specification> <Field_Specification>)

ID tests whether two field specifications reference the same conceptual structure. For example, one DC for "give" is

(ID (ACTOR) (FROM))

3. (DIF <Field_Specification> <Field_Specification>)

(DIF X Y) = ~(ID X Y)

4. (NMQ <Field_Specification> Token)

NMQ tests whether a particular conceptual token is a member of a field. The NODE modification of a conceptualization is represented as a list which may

contain elements like *NEG* (negation particle) and *CANNOT* ("cannot" (@) particle). For example, "prevent" has as one of its DCs

(MMQ (< MODE) *CANNOT*)

5. (MNLK <Field_Specification> "conceptual_link")

The "conceptual_link" is one of the symbols {<*> <:> <> <:C <:D}. MNLK tests whether a field contains a conceptualization with the specified "conceptual_link" as its "main link".

6. (MNLKC <Field_Specification> "link_code")

Each of the main connective links of conceptual dependency has been assigned a code, as follows:

LINK	CODE	Mnemonic
<***>	E	Event
< >	S	State
<:, <:C	K	Kausal
<:D	D	Double-cause
^	A	And
/---------, \---------<	C	stateChange

MNLKC tests whether the code for the main link of the contents of a field is that specified by "link_code".

7. (SKEL <Field_Specification> "skeleton_code")

A "skeleton" code is defined for every conceptualization. It is identical to the link_code for those conceptualizations whose main links have codes E, S, A, or C. For causal structures the skeleton code is xKy, where x is the link code for the main link of the <ANTECEDENT> and y the code for the main link of the <RESULT>. SKEL tests whether the skeleton code for the contents of a field is that specified by "skeleton_code".

8. (LESSS <Field_Specification> <number>)

<Field_Specification> will reference a field which marks a pointer on one of the scales or an INCrement on a scale. It will thus have some numerical value X. LESSS tests for X < <number>

9. (GRREAT <Field_Specification> <number>)

GRREAT is analogous to LESSS, testing for X > <number> .

The second basic class of predictions consists of those which interact with the memory. There are now four of these predicates:

1. (PROP <Field_Specification> <Property>)

The specified field should contain a conceptual nominal (PP), such as *JOHN* or *MILK*. <Property> must be one of a set of conceptual properties, like HUMAN or FLUID. PROP test whether the PP has the property specified. For example, one of the DCs for "drink" is

(PROP (OBJECT) FLUID)

These properties are like semantic markers <11>, but are associated with concepts rather than words.

2. (TIME_REL <Field_Spec_list>
(BEFORE/AFTER <Time_spec> <Time_spec>))

The <Field_Spec_list> consists of one or two Field_ Specifications, which must evaluate to time references. A <Time_spec> is either the atom *T*, which represents 'now' (time of utterance), or is of the form (* n) for n=1 or n=2. In the latter case the <Time_spec> represents the value of the nth element of the <Field_Spec_list> . TIME_REL calls on memory to attempt a proof of the specified time relationship. For example, one of the DCs for "want" is

(TIME_REL ((CON TIME) (TIME)) (AFTER (* 1) (* 2)))

3. (MEM_QUERY <Field_Spec_list>
<Conceptualization> <Restrictions>)

<Field_Spec_list> is a list of field specifications. <Conceptualization> is an arbitrary conceptualization. Some of its fields may be filled with the pattern ($ n), in which case that field is replaced by the contents of the field specified by the nth element of the <Field_Spec_list>. In addition, fields of the conceptualization may be filled by the pattern (∃ x) where x is any atom. In this case x will be considered a variable and Restrictions may further specify x, such as requiring (PROP x HUMAN) or (BEFORE x *T*).

MEM_QUERY asks memory to verify a conceptualization C formed by the substitutions from the <Field_Spec_list> values into <Conceptualization>, by finding or inferring a conceptualization C' which matches C in all non-variable positions and contains elements in all variable positions which satisfy the <Restrictions>[2]. The <Restrictions> may also use values computed by the <Field_Spec_list>. This is indicated by the (↑ n) pattern as used in the TIME_REL predicate. The predicate which tests whether an *ATRANS* event can be realized using "return" is

```
(MEM_QUERY    (  (OBJECT) (TO) (TIME)  )
              ((ACTOR ($ 1) <=> (*POSS*  VAL ($ 2)))
                                TIME (∃ Z))
              (  (BEFORE  Z  (↑ 3)) )
)
```

The second line of this predicate shows a LISP version of a conceptual dependency structure. This form consists of alternating field names and field values. The 'top-level' field is the entire conceptualization and it has no name. Since this form is difficult to read, particularly for non-trivial conceptualizations, our discussions will generally stick to the diagram format we have been using or some more 'Englishy' version such as

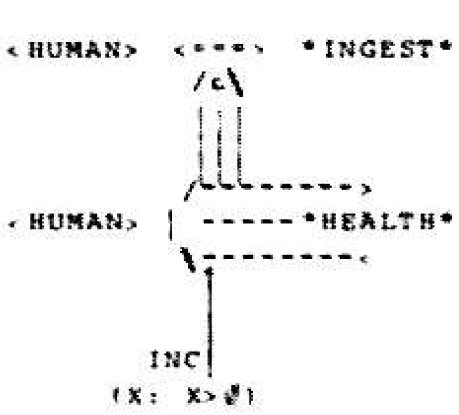
"was the (OBJECT) possessed by (TO) at some time Z prior to (TIME) ?"

We mentioned in Chapter 4 that it would probably be desirable to add an additional parameter, an effort coefficient, to such a predicate. Since we don't yet have a deductive model capable of performing the sorts of verifications needed, however, the value of such a coefficient would have to be chosen arbitrarily. We have thus chosen not to incorporate one at all.

4. (FUNC_OF <Field_Specification> <Conceptualization> <Restrictions>)

<Field_Specification> must evaluate to a conceptual nominal. <Conceptualization> is any conceptualization, which may contain fields filled by '*C'. Such fields are replaced by the value of <Field_Specification>.

Conceptual memory knows about the functions of objects. FUNC_OF asks memory whether the concept specified by <Field_Specification> has as one of its functions that specified by <Conceptualization>. For example, the specified function might be:



representing "a human ingesting #C could cause that human's health to improve". If #C were replaced by the concept *ASPIRIN*, memory would verify that this was one of the functions of *ASPIRIN*, which might ultimately lead to generation of the sentence 'John took an aspirin'.

A simple example shows why this cannot be handled by one of the MEM_QUERY predicates described above. Suppose #C referred to *MILK*. Certainly ingesting *MILK* can-cause a positive increment in *HEALTH*. This is not, however, noted as a function of *MILK*

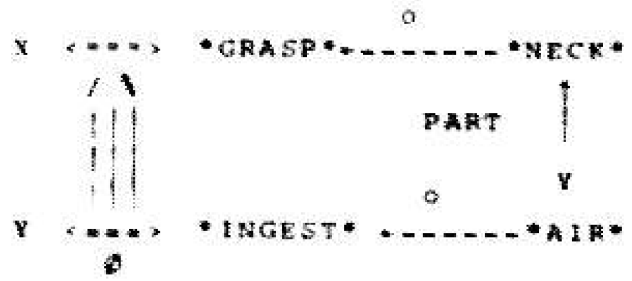
in the memory[3]. By separating FUNC_OF from MEM_QUERY, BABEL avoids expressing '*INGEST*<--o--- *MILK*' as "take milk". One might consider having a conceptual classification "substance-ingested-to-cause-better-health" just as we postulate a classification FLUID. If such a classification existed, the PROP predicate could be used rather than FUNC_OF, at least in the example we are discussing. Such a classification should exist, however, only if non-linguistic justification for it can be found; creating such markers to simplify the job of generation will lead to a language-dependent representation in the memory.

Predicates of types I and II are sufficient to make all the distinctions between conceptualizations which BABEL is capable of making. Experience in writing 'grammars' to generate from conceptual representations has shown that a third type of DC, while logically redundant, is of practical use.

This third type allows a single 'super' DC to specify an entire set of predicates. An example will clarify the idea behind this. The English verb "to breathe", in its most common sense, is represented conceptually as

X <---> *INGEST*<--o---*AIR*

while "to choke (someone)" is represented as



The <RESULT> in this representation of "choke" is just the representation of "breathe" modified by Ø. Rather than repeat the DCs necessary for defining 'BREATHE1' in the definition of 'CHOKE1', the characteristic

POT_HEAD * BREATHE1

can be used.

Evaluation of the predicate

(POT_HEAD <Field_Specification> <word-sense>)

consists of testing whether the DCs for <word sense> found on its property list) are satisfied by the conceptualization found in the specified field (i.e., whether <word sense> is a POTential 'HEAD' of a syntax net for this structure.) The idea is a conceptual analog of the definition of words by relations between other words in a semantic memory like that of Quillian <23>.

We now have sufficient background for investigating in some detail the discrimination nets used by BABEL. Each tree is designed to enable discrimination to be made between

a class of word senses which are in some sense 'similar'.

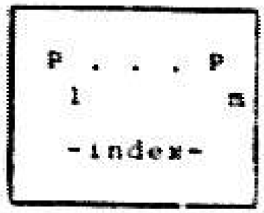
All discrimination tree nodes are indexed as follows:

1) the root node receives index 1

2) the 'left-hand-son' ('false' subtree) of a node with index N is assigned index 2N.

3) the 'right-hand-son' ('true' subtree) of a node with index N is assigned index 2N+1.

In diagramming the trees, each non-terminal node will be represented by a box:



-index- is the index of the node, determined by the indexing system just presented. The P_i are predicates of the sort we have just defined. The predicate evaluated at a node is the conjunction of the predicates P_i.

It sometimes turns out that several predicates will test true leading the program to 'believe' it is on the right path to a response. But it may be that one (or more) of these true results was merely fortuitous and it would have been better to have 'ignored' the fortuitous relation and followed a 'false' branch. The natural thing to do when this is discovered is to 'back up' to the node which would have been reached had the original fortuitous relation not misled us. This will be indicated by a (↑ <integer>) at the end of a branch

in the trees. In the implementation, a branch of this type is actually a pointer to the node with index <integer>. This means that BABEL's discrimination trees are not truly tree structures, but networks. The procedure for applying discrimination nets given above remains applicable, and, since care is taken to avoid any cycles in the nets, the process is still guaranteed to terminate.

For example, consider the sentence

"Bob told Jim that Mary would like it if Jim took her to the prom".

The meaning of this sentence would be represented in C.D. as an MTRANS event, the MOBJECT of which is a can-cause relation (something done by Jim could cause Mary some benefit). This representation bears a great deal of structural similarity to those which result in the choice of verbs like "advise", "recommend", and "ask-to". The net used to select a verb to express this meaning does not 'recognize' the crucial differences which prohibit the use of these verbs until the (incorrect) decision has been made that the "could cause benefit" structure of the MOBJECT is significant. When the mistake is realized, one of the 'pointer' nodes leads the process back to the node which would have been reached had the MOBJECT failed the test for a "could cause benefit" structure. From this node there is no path back to any node already passed; thus looping is avoided.

At each terminal node of a tree [illegible] of responses (the exact nature of these responses is explained in section 5.2; they may be considered word senses for the time being). The (T <integer>) form is also present at some of the terminal 'response' nodes in the trees. These pointers are used only in paraphrasing and will be explained in Chapter 6.

BABEL currently contains 15 different discrimination nets. We shall now look at a few of them in detail.

The first tree we shall look at organizes knowledge about verb senses which are encoded conceptually as EVENTs using the ACT *INGEST*. Figure 5-5 depicts this tree. Node 1 tests whether the OBJECT of the ingesting has as one of its functions the "causing of a positive increment in the *HEALTH* of one who ingests it." (In our descriptions and drawings of the nets we shall use 'anglicized' conceptual dependency rather than the more formal diagrams or internal LISP notation.) If this functional relationship holds, node 3 is reached with the response TAKE2, the "take medecine" sense of the verb.

In general, English 'ingesting' verbs distinguish between the ingesting of solids, liquids, and gases. This knowledge of English, in BABEL's terms, bears [illegible] tests on the physical properties of the OBJECT. Node 7 checks to see if it is a GAS. If so, BABEL has three possible

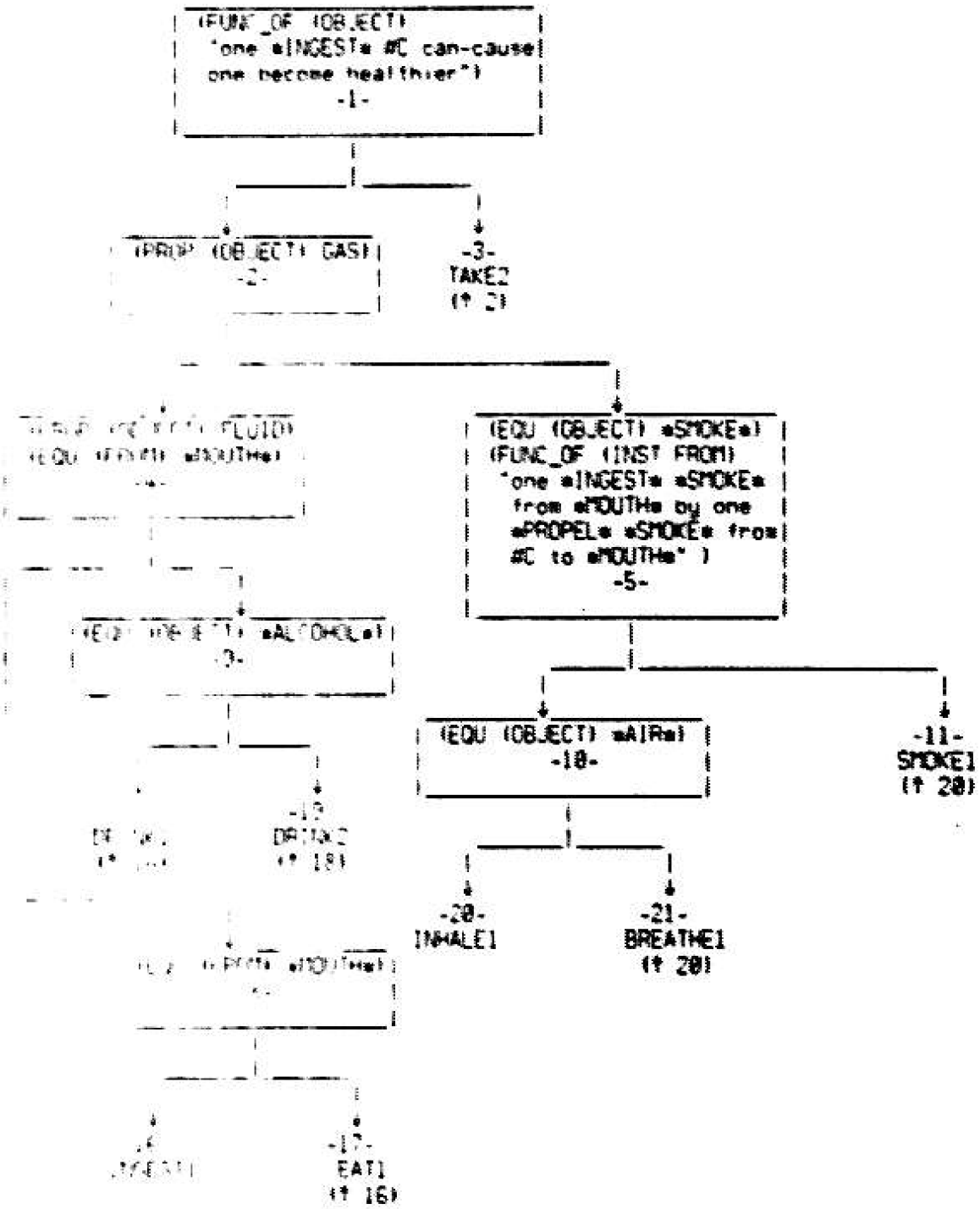

(FUNC_OF (OBJECT)
"one *INGEST* #C can-cause
one become healthier")
-1-
(PROP (OBJECT) GAS)
-2-
-3-
TAKE2
(↑ 2)
(EQU (OBJECT) *SMOKE*)
(FUNC_OF (INST FROM)
"one *INGEST* *SMOKE*
from *MOUTH* by one
PROPEL *SMOKE* from
#C to *MOUTH*")
-5-
(EQU (OBJECT) *AIR*)
-18-
-11-
SMOKE1
(↑ 20)
-20-
INHALE1
-21-
BREATHE1
(↑ 20)
-17-
EAT1
(↑ 16)

FIGURE 5-5

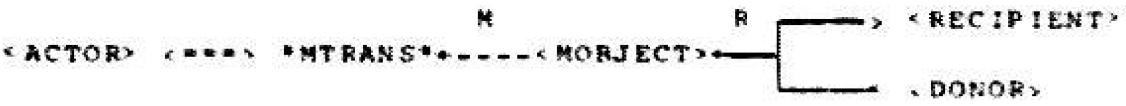
verbs for expressing the event. If the OBJECT is *SMOKE*, the verb 'to smoke' may be appropriate But ingesting smoke in a forest fire does not constitute 'smoking'[4], so further tests are needed. Node 5 makes these tests. The most important of these is one which tests the function of the object which is the source of the smoke. Memory must know that one of it's functions is for someone to ingest smoke from that object in order for node 11 with the response SMOKE1 to be reached. If any of the tests at node 5 fail a test is made to see if the OBJECT is *AIR* (node 10). If so, the response BREATHE1 is found; otherwise, INHALE1 is returned.

If the OBJECT is not a gas, but a FLUID (node 4), and it is ingested through the *MOUTH*[5], some sense of 'drink' will be found. If the OBJECT is *ALCOHOL* (node 9) the response DRINK2 is found; otherwise the response will be DRINK1.

Finally, for OBJECTs which are neither GASes nor FLUIDs, a test is made (node 8) to see if the ingesting is through the *MOUTH* of the ACTOR. For our examples this is always true, so EAT1 (node 17) is the response selected. For this reason, INGEST1 (node 16) is never found as a primary reading for any of our conceptualizations. It is however, found when generating paraphrases, as will be seen

when the multiple realization process is described in Chapter 6.

The next tree we shall look at deals with conceptualizations of the form:



which is used for 'mental information transfer' events. The tree is depicted in Figure 5-6.

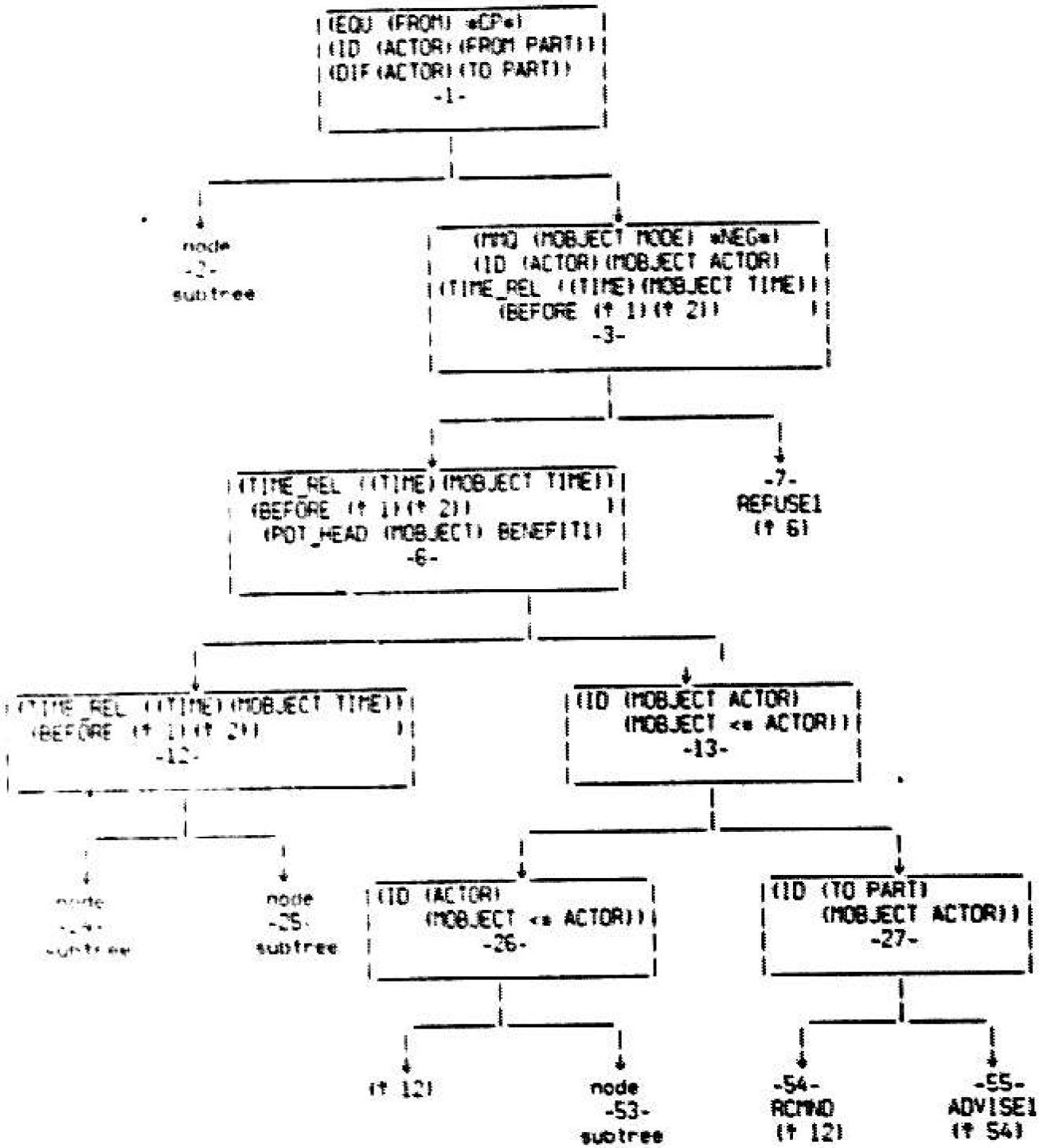

(EQU (FROM) *CP*)
(ID (ACTOR) (FROM PART))
(DIF (ACTOR) (TO PART))
-1-
node
-2-
subtree
(MNQ (MOBJECT MODE) *NEG*)
(ID (ACTOR) (MOBJECT ACTOR))
(TIME_REL ((TIME) (MOBJECT TIME))
(BEFORE (↑ 1) (↑ 2))
-3-
(TIME_REL ((TIME) (MOBJECT TIME))
(BEFORE (↑ 1) (↑ 2))
(POT_HEAD (MOBJECT) BENEFIT))
-6-
-7-
REFUSE1
(↑ 6)
(TIME_REL ((TIME) (MOBJECT TIME))
(BEFORE (↑ 1) (↑ 2))
-12-
(ID (MOBJECT ACTOR)
(MOBJECT << ACTOR))
-13-
node
-25-
subtree
(ID (ACTOR)
(MOBJECT << ACTOR))
-26-
(ID (TO PART)
(MOBJECT ACTOR))
-27-
(↑ 12)
node
-53-
subtree
-54-
RCMND
(↑ 12)
-55-
ADVISE1
(↑ 54)

FIGURE 5-6

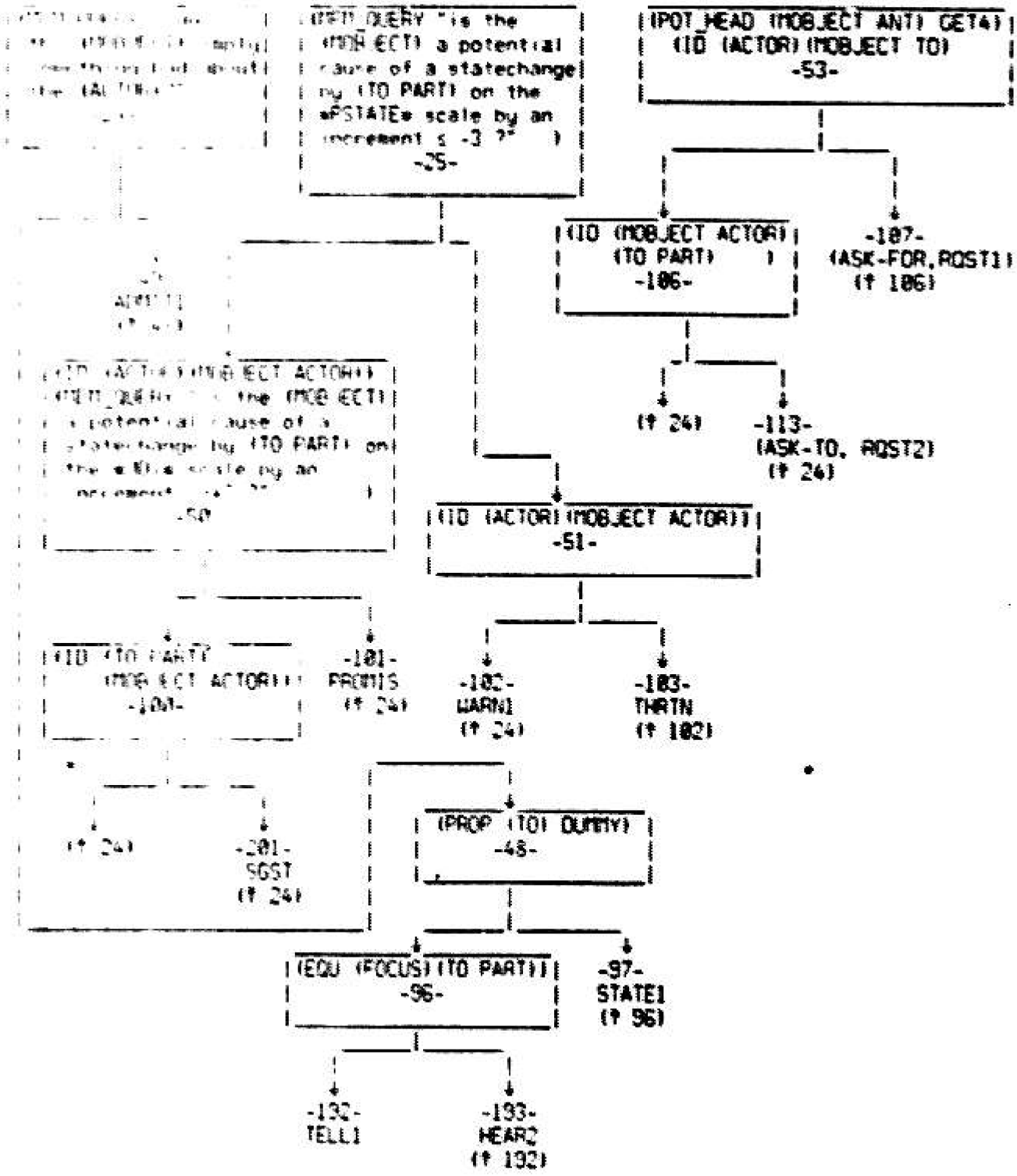

(POT_HEAD (MOBJECT ANT) GET4)
(ID (ACTOR) (MOBJECT TO)
-53-
(ID (MOBJECT ACTOR)
(TO PART)
-186-
-187-
(ASK-FOR, ROST1)
(↑ 186)
(↑ 24)
-113-
(ASK-TO, ROST2)
(↑ 24)
(ID (ACTOR) (MOBJECT ACTOR))
-51-
-101-
PROMIS
(↑ 24)
-102-
WARN1
(↑ 24)
-103-
THRTN
(↑ 102)
(↑ 24)
-201-
ROST
(↑ 24)
(PROP (TO) DUMMY)
-48-
(EQU (FOCUS) (TO PART))
-96-
-97-
STATE1
(↑ 96)
-132-
TELL1
-133-
HEAR2
(↑ 132)

FIGURE 5-6

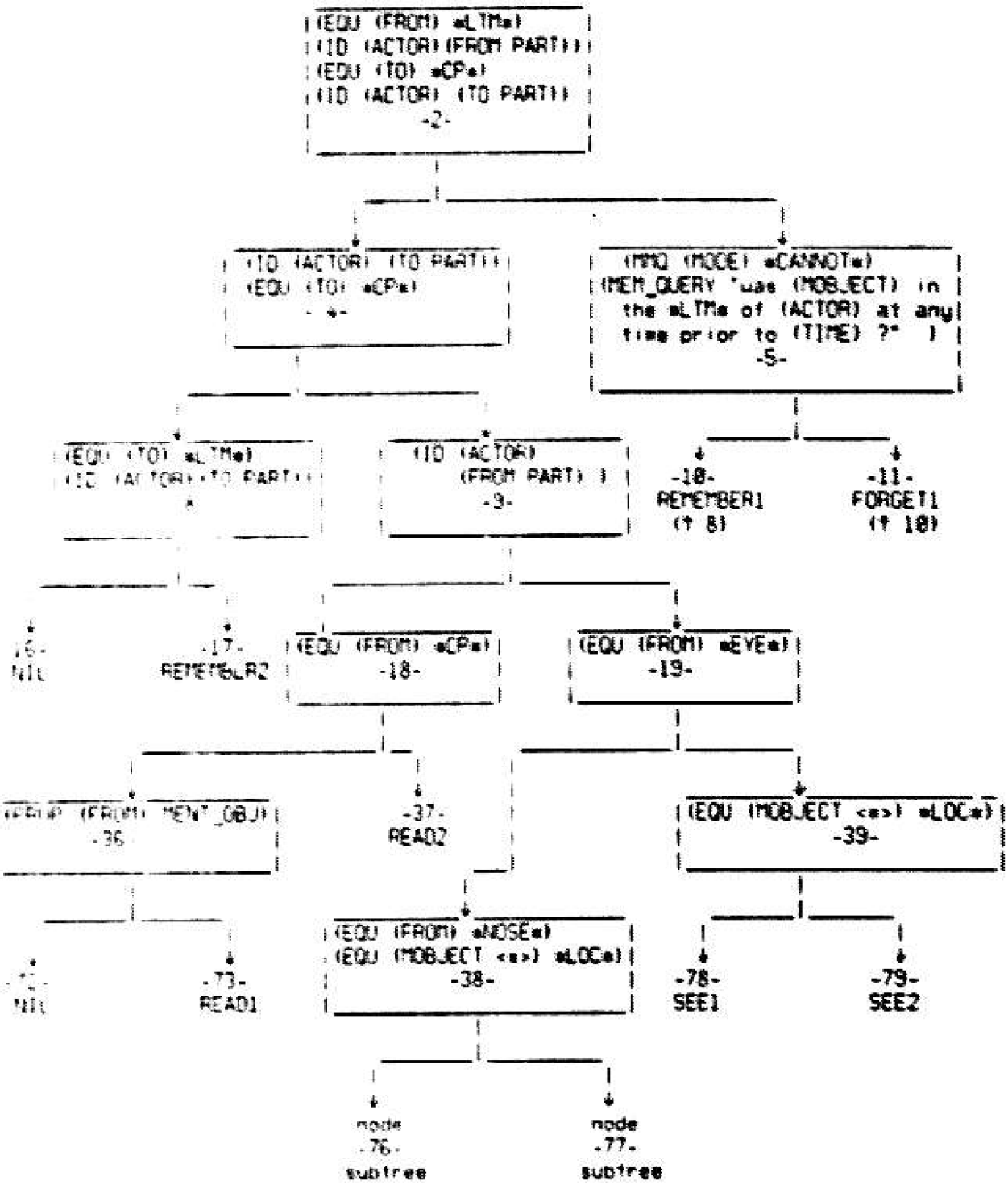

(EQU (FROM) *LTM*)
(ID (ACTOR) (FROM PART))
(EQU (TO) *CP*)
(ID (ACTOR) (TO PART))
-2-
(ID (ACTOR) (TO PART))
(EQU (TO) *CP*)
- *-
(MMQ (MODE) *CANNOT*)
(MEM_QUERY 'use (MOBJECT) in
the *LTM* of (ACTOR) at any
time prior to (TIME) ?")
-5-
-18-
REMEMBER1
(↑ 8)
-11-
FORGET1
(↑ 10)
-19-
-37-
READ2
-39-
-78-
SEE1
-79-
SEE2
node
-76-
subtree
node
-77-
subtree

FIGURE 5-6

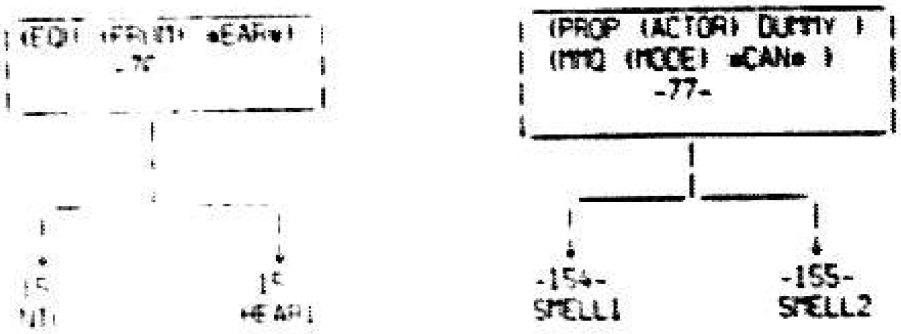


FIGURE 5-6

We have chosen to group the 'MTRANS' verbs into two general classes. Those which represent communication between individuals are found in the subtree rooted at node 1. The others, which mainly involve perception, are found in the subtree rooted at node 2. The predicate at node 1 distinguishes the two groups by checking that the =DONOR is the *CP* of the =ACTOR, and that the =DONOR and =RECIPIENT= mental locations are PARTs of different PPs.

Node 1 checks the remaining DCs for REFUSE. If any of these tests fail, control passes to node 4, where a check is made to see if the conceptualization indicates communication of the fact that "something would benefit someone". A group of verbs which express variations of this meaning is found in the subtree rooted at node 13. If the person benefitted is the =RECIPIENT=, either ADVISE (node 54) or RECOMMEND (node 55) will be chosen. If the benefitted person is the =ACTOR=, and the event causing the benefit has the =RECIPIENT= as its ACTOR, then ASK-TO or, if the event is the "giving of an object", ASK-FOR will be selected. Each of these has a synonym, a form of "request", in its response set (nodes 107, 113).

If the =MOBJECT= is not of the "could cause benefit" type, the subtree rooted at node 2 will be entered. A check is made to see if the time of the =MOBJECT= is in the future of the time of the *MTRANS* event. If so,

several verbs are candidates: WARN, THREATEN, PROMISE, SUGGEST. Either WARN or THREATEN is chosen if a check with memory indicates the <MOBJECT> is potentially harmful to the <RECIPIENT>. They are distinguished by the fact that THREATEN required the <MOBJECT>'s ACTOR to be identical to the ACTOR of the MTRANS. PROMISE is chosen if the <MOBJECT> could cause a positive increment in the position of the <RECIPIENT> on the *JOY* scale.

In the case that none of these verbs is applicable, a check with memory is made to see if ADMIT can be used (node 24 -- does the <MOBJECT> imply something bad about the <ACTOR>?). Finally, if the <RECIPIENT> is specified only by a DUMMY (a PP representing "someone"), the verb STATE is selected. Otherwise a choice between TELL and HEAR-from is made, based on the FOCUS marking of the conceptualization.

The subtree rooted at node 2 is considerably less complex. It distinguishes several sensory perception verbs, which are represented as "MTRANSing a conceptualization from a sense organ (EYE, EAR, NOSE) to the *CP*". Two types of "see" are accounted for: SEE1, "to see an object", and SEE2, "to see an event". Two types of "smell" are also taken care of: SMELL1, "to smell an object", and SMELL2, "an object smells (= "has an odor", "can be smelled by someone").

A few non-perception verbs are also part of this subtree. These include two types of "remember": REMEMBER1,

"to retrieve information from the *LTM*", and REMEMBER., "to store information into the *LTM*". FORGET is found in response to any conceptualization which satisfies REMEMBER1 but is modified by a '*CANNOT*', with the further condition, verified by a check with memory, that the <MOBJECT> was previously stored in the *LTM* (i.e., "forget" = "<ACTOR> cannot recall X: X previously located in *LTM* of <ACTOR>").

Finally, this subtree distinguishes the standard sense of "read" (READ1) and, just for fun, a "mind-reading" sense, READ2, which has the concise C.D. representation:

In nets like the ones just described a response R_i is appropriate for any conceptualization which satisfies a set of DCs D_i. When there exist i, j such that $D_i \supset D_j$, then, for a conceptualization which satisfies the conditions D_j, either R_i or R_j could be used as a response. It is very important that the trees be organized, as they always can be, so that response R_i, a response which expresses 'more' of the conceptualization, is found in such cases. Otherwise sentences like "John told me it would be good for me to take the course" would be generated from

conceptualizations which could be expressed more simply: "John advised me to take the course."

By this time the reader has undoubtedly found several occasions to look askance at some of the representations being assumed for verbs. We make no claims that these trees fully characterize the verb senses they are designed to deal with. In some cases it is clear that our 'under-representations' would be unsatisfactory in an operating model. In many others it is not obvious that situations would arise where the simplified representations would cause trouble. For instance, a true characterization of "ask-to" should probably include the fact that the intention of the 'asker' is that the 'asked' do the action requested. But it would be very rare for our lack of checking intention to result in the use of "ask-to" when it was inappropriate.

In order to write a conceptual generator, it was necessary to choose a particular conceptual representation. Conceptual Dependency was chosen because it is currently better developed than any other conceptual representation available. No claim is made that it is yet complete, in the sense of satisfactorily representing all natural language 'meanings' or even those of the vocabulary used by BABEL. A more complete representation will certainly result in larger trees and therefore more searching. There is no reason to believe it will alter the fundamental

nature of the generation process, which is the central issue.

5.2 Concexicon

The response found at the terminals of the discrimination nets are pointers to entries in a linguistic knowledge file called the CONCEXICON. This file is the major source of knowledge about the syntactic realization of conceptual relations. This information is organized by 'word senses'. An entry in this file has three fields:

CONCEXICON ENTRY

LEXICAL POINTER	FRAMEWORK	SPECIAL ACTIONS

The lexical pointer is a reference to an entry in the lexicon; the pointer for GIVE1 is to the lexical entry GIVE. Concexicon entries correspond closely to the usual notion of word senses, so many concexicon entries may refer to a single lexical entry. The concexicon entries FLY1 ("to pilot an aircraft"), FLY2 ("to travel by plane"), and FLY3 ("to move through the air") all point to the lexical entry FLY. This lexical pointer is actually the infiritive form of a verb. The lexicon itself, which includes other information about the verb, is described in section 5.6.

The FRAMEWORK of the concexicon entry contains the really significant information about the word sense to which it corresponds; namely, the syntactic environment which must be placed around it in the final syntax network. This FRAMEWORK consists of a list of FRAMEs, where each FRAME has three fields:

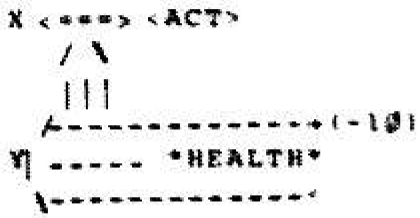
FRAME

SYNTAX RELATION	FIELD SPECIFICATION	SPECIAL REQUIREMENTS

The SYNTAX RELATION is a member of a fixed set of relations which can occur in the syntax nets. These include ACTSBJ, OBJ, and IOBJ mentioned earlier. Each SYNTAX RELATION is known to the surface grammar; most have specialized functions associated with them. The syntax relations provide the information necessary for the grammar to string a sentence together in proper left to right order from its components and to perform necessary morphology while doing so.

FIELD SPECIFICATIONs (FSs) were described in detail in the previous section. In a FRAME, the FS indicates where in the conceptualization the information which will be used to generate the value of the syntax relation will be found. For example, one of the FRAMEs for the concexicon entry KILL1 associates the syntax relation 'OBJ' with the FS

(< ACTOR). Since KILL1 will be found as a response to



the syntactic realization of Y will be put in an OBJ relation to KILL in the syntax net, and ultimately a sentence like "X kill Y (by ...)" will be genera*ed.

The SPECIAL REQUIREMENTs (SRs) of a FRAME are mainly used to introduce prepositions. One FRAME of the entry for ARRIVE1 indicates that it requires a syntax relation 'LOC' with SR (MAKPREP AT). This will cause the syntax net to have *he form:

G1:	LEX	ARRIVE	G_n:	PREP	AT
	.	.		POBJ	G
	.	.			n+1
	LOC	G_n			

The only other SR used is (QTHD X). This causes a specified lexical unit X to be inserted directly into the syntax net as the value of the syntax relation with which the SR is associated, rather than having the value generated from a part of the conceptualization as is usually done. For instance,

can be realized as "X be unable to . . ." The discrimination nets will find the concexicon entry UNABLE1, which has a lexical pointer BE. One of the FRAMEs for UNABLE1 has a syntax relation 'P_ADJ' (predicate adjective) which has a SR (QTHD UNABLE). Another use for QTHD is to enter 'particles' of verbs like 'pick up' and 'give back' into the syntax nets.

The third field of the concexicon entry is the SPECIAL ACTIONS (SAs). SAs, like SRs, are specialized functions. But rather than effecting changes in the syntax net being created, SAs modify the conceptual representation controlling the generation. The only SA now provided for is one which deletes elements of the conceptualization. Consider the case of UNABLE1 just mentioned. One of the SRs needed is INF2, a type of embedded sentence, to account for the "to . . ." in "X be unable to . . .". The information for 'INF2' must come from the entire conceptualization (C5-1), minus the MODE marker "B". (If the "B" were not ignored, the program could get into infinite recursion, generating a syntax net for "X be unable to be unable to be unable to . . .."). A FS does not permit specifying "all of a conceptualization minus . . .". The solution is to have the FS for the INF2 relation specify

ALL, and have a SPECIAL ACTION 'DELETION. Mode ' delete the "@" associated with UNABLE1 and to activate [illegible] t. FRAMEWORK for UNABLE1 is processed.

Certain syntax relations tend to occur with great frequency in particular conceptual roles. For example, ACTSBJs are frequently found as conceptual ACTORs. For this reason, 'default' field specifications have been associated with several of the relations. When the information to fill a syntax relation is indeed found in its default location, the FIELD SPECIFICATION may be omitted from the FRAME.

SYNTAX RELATION	DEFAULT FIELD SPECIFICATION
ACTSBJ	(ACTOR)
OBJ	(OBJECT)
OBJ2	(TO)
LOC	(TO)
INST	(CON)
P_ADJ	(< ^ >)
PIRS	(CON)
SECS	(A)
IOBJ	(TO)

The information specified by the FRAMEWORK must be associated with the concexicon entry (i.e., at the word sense level) rather than with the verb itself in the lexicon. This can be seen from our simple example of "drink" mentioned in section 4.1. One sense of drink requires an OBJ (a direct object), while the other sense is

realized in an intransitive form. As another example, there exist three principal senses of "want":

"John wants an apple".
"John wants his mother".
"John wants to play baseball".

each with its own syntactic environment. The first two senses have an OBJ relation, while the third sense requires an INF relation. Furthermore, even the first two senses differ with respect to the conceptual location of the OBJ. In the first sense, "apple" would be found as a conceptual OBJECT; in the second sense, "mother" would be a conceptual ACTOR.

BABEL's CONCEXICON, in its LISP format, is shown in Figure 5-7.

5.3 Scales

English adjectives, while comprising a 'unified' syntactic category for some grammatical theories, do not lend themselves to any single conceptual treatment. There are participial forms -- a defeated player, a stolen bicycle -- which are derived from verbs and relate conceptually to the underlying representations of those verbs. Other adjectives name more or less complicated physical properties possessed by some objects -- a spotted horse, a louvered window.

```
FORM OF EACH ENTRY IS
(<WORD-SENSE HEAD> (<LEXICON POINTER>) (<FRAMEWORK>) <SPECIAL-ACTIONS>)
<FRAMEWORK> ::= <FRAME> | <FRAME> <FRAMEWORK>
<FRAME> ::= (<CASE>  <FIELD-SPEC>   <SPECIAL-REQUIREMENTS> )


(ABOUT1 BE ((ACTSBJ) (LOC (<≡> VAL) (MAKPREP ABOUT))))
(ACCEPT1 ACCEPT ((ACTSBJ (<≡> VAL PART)) (OBJ (CON))))
(ADMIT1 ADMIT ((ACTSBJ) (S2) (PP1 (TO PART)(MAKPREP TO))))
(ADVISE1 ADVISE ((ACTSBJ) (OBJ2 (TO PART)) (INF2 (MOBJECT CON))))
(AND1 AND ((FIRS) (SECS)) )
(AND2 AND ((FIRS) (SECS (<≡>))) )
(ANGRY1 BE ((ACTSBJ) (IOBJ (MAKPREP AT)) (P_ADJ (QTHD ANGRY))))
(ARRIVE1 ARRIVE ((ACTSBJ) (LOC (MAKPREP AT))))
(ARRIVE2 ARRIVE ((ACTSBJ (OBJECT)) (LOC (MAKPREP AT))))
(ASK-FOR ASK ((ACTSBJ) (OBJ2 (TO PART))
              (IOBJ (MOBJECT OBJECT)(MAKPREP FOR))))
(ASK-TO  ASK ((ACTSBJ) (OBJ2 (TO PART))(INF2 (MOBJECT CON))))
(BE1 BE ((ACTSBJ) (P_ADJ)))
(BE3 BE ((ACTSBJ) (POSS (<≡> VAL))))
(BEAT1  BEAT  ((ACTSBJ) (OBJ (TO)) (INST (OBJECT)(MAKPREP WITH))))
(BECAUSE1 BECAUSE ((FIRS (<≡)) (SECS (CON))))
(BECOME1 BECOME ((ACTSBJ) (P_ADJ (ADDINC))))
(BELIEVE1 BELIEVE ((ACTSBJ (<≡> VAL PART)) (S2 (CON))))
(BELOC1 BE ((ACTSBJ) (LOC (<≡> VAL) (MAKPREP NEAR))))
(BELONG1 BELONG ((ACTSBJ) (PP1 (<≡> VAL) (MAKPREP TO))))
(BREATHE1 BREATHE ((ACTSBJ)))
(BUY1 BUY ((ACTSBJ (CON ACTOR)) (OBJ (<≡≡> OBJECT))
          (IOBJ (<≡≡> ACTOR)(MAKPREP FROM))
          (INST (CON OBJECT)(MAKPREP FOR))))
(CEASE1 CEASE ((INF3 ALL)) ADDITIONS ((TIME (TF))) DELETIONS ((TF)))
(CHOKE1 CHOKE ((ACTSBJ (CON ACTOR)) (OBJ (<≡ ACTOR))))
(CHOKE2 CHOKE ((ACTSBJ (<≡ ACTOR)) (INST (CON OBJECT)(MAKPREP ON))))
(COME1 COME ((ACTSBJ) (LOC (MAKPREP TO))))
(COME2 COME ((ACTSBJ (OBJECT)) (LOC (MAKPREP TO))))
(COME3 COME ((INF3 ALL)) ADDITIONS ((TIME (TS)))  DELETIONS ((TS)))
(COMPLAIN1 COMPLAIN ((ACTSBJ) (OBJ (MOBJECT CON))))
(CONSIDER2 CONSIDER ((ACTSBJ (<≡> VAL PART)) (OBJ2 (CON ACTOR))
                     (P_ADJ (CON <≡>))))
(DIE1 DIE ((ACTSBJ)))
(DISLIKE1 DISLIKE ((ACTSBJ (<≡ ACTOR)) (OBJ (OBJECT))))
(DISLIKE3 DISLIKE ((ACTSBJ) (OBJ (TO))))
(DISLIKE4 DISLIKE ((ACTSBJ (<≡ ACTOR)) (S2 (CON))))
(DO1   DO ((ACTSBJ) (OBJ (QTHD SOMETHING)))   )
(DRINK1 DRINK ((ACTSBJ) (OBJ)))
(DRINK2 DRINK ((ACTSBJ)))
```

FIGURE 5-7

```
(EAT1 EAT ((ACTSBJ) (OBJ)))
(ENABLE1 ENABLE ((ACTSBJ ) (INF (<E)) (INST2 (CON)))
        DELETIONS ((E VOICE)))
(ENJOY1 ENJOY ((ACTSBJ (<E ACTOR)) (S2 (CON))))
(EXPECT1 EXPECT ((ACTSBJ (<E> VAL PART)) (INF (CON))))
(EXPECT2 EXPECT ((ACTSBJ (<E> VAL PART)) (OBJ (CON ACTOR))))
(EXPECT3 EXPECT ((ACTSBJ (<E> VAL PART)) (OBJ (CON OBJECT))))
(FEAR1 FEAR ((ACTSBJ (<E> VAL PART)) (S2 (CON CON))))
(FEAR2 FEAR ((ACTSBJ (<E> VAL PART)) (OBJ (CON CON ACTOR))))
(FEAR3 FEAR ((ACTSBJ (<E> VAL PART)) (OBJ (CON CON OBJECT))))
(FEED1 FEED ((ACTSBJ) (OBJ (<E OBJECT)) (OBJ2 (<E ACTOR))))
(FLY1 FLY ((ACTSBJ) (LOC)))
(FLY3 FLY ((ACTSBJ (INST ACTOR)) (OBJ) (LOC)))
(FLY4 FLY ((ACTSBJ) (LOC)))
(FORGET1 FORGET ((ACTSBJ) (S2 (*OBJECT))))
(GET1 GET ((OBJ) (ACTSBJ (TO)) (IOBJ (ACTOR) (*AKPREP FROM))))
(GET2 GET ((OBJ) (ACTSBJ (TO)) (IOBJ (ACTOR) (*AKPREP FROM))))
(GIVE1 GIVE ((ACTSBJ) (OBJ) (OBJ2)))
(GIVE11 GIVE ((ACTSBJ) (OBJ) (IOBJ (*AKPREP TO))))
(GIVE8  GIVE ((ACTSBJ) (OBJ) (IOBJ (*AKPREP TO))
              (PART2 (GIVE BACK))))
(GO1 GO ((ACTSBJ) (LOC)))
(GRAB1 GRAB ((ACTSBJ) (OBJ)))
(HATE1 HATE ((ACTSBJ (<E ACTOR)) (OBJ (OBJECT))))
(HATE4 HATE ((ACTSBJ (<E ACTOR)) (IOBJ (CON))))
(HAVE1 HAVE ((ACTSBJ (<E> VAL)) (OBJ (ACTOR))))
(HEAR1 HEAR ((ACTSBJ) (PAST (*OBJECT))))
(HEAR2 HEAR ((ACTSBJ (TO PART))(IOBJ (ACTOR)(*AKPREP FROM))(S2)))
(HIT1  HIT  ((ACTSBJ) (OBJ (TO))))
(HIT2  HIT  ((ACTSBJ) (OBJ (TO)) (INST (OBJECT)(*AKPREP WITH))))
(HOPE1 HOPE ((ACTSBJ (<E> VAL PART)) (S2 (CON CON))))
(HURRY1 HURRY ((ACTSBJ) (LOC (*AKPREP TO))))
(HURT1 HURT ((ACTSBJ) (OBJ) (INST2 (CON))))
(IFTHEN1 IF ((FIRS (<E)) (SECS (CON))))
(INCEST1 INCEST ((ACTSBJ) (OBJ)))
(INHALE1 INHALE ((ACTSBJ) (OBJ)))
(INTEREST INTEREST ((ACTSBJ (CON OBJECT INVOLV)) (OBJ)))
(KEEP2 KEEP ((ACTSBJ) (OBJ) (IOBJ (*AKPREP FROM))))
(KEEP3 KEEP ((ACTSBJ) (OBJ) (LOC (FROM))))
(KILL1 KILL ((ACTSBJ) (OBJ (<E ACTOR)) (INST2 (CON))))
(KNOW1 KNOW ((ACTSBJ (<E> VAL PART)) (S2 (CON))))
(LEAVE1 LEAVE ((ACTSBJ) (LOC (FROM)) (*AKPREP FROM)))
(LIKE1 LIKE ((ACTSBJ (<E ACTOR)) (OBJ (OBJECT))))
(LIKE3 LIKE ((ACTSBJ) (OBJ (TO))))
(LIKE4 LIKE ((ACTSBJ (<E ACTOR)) (INF (CON))))
(LOVE1 LOVE ((ACTSBJ (<E ACTOR)) (OBJ (OBJECT))))
(LOVE4 LOVE ((ACTSBJ (<E ACTOR)) (INF (CON))))
```

FIGURE 5-7

```
(LOWER1 LOWER ((ACTSBJ) (OBJ) (LOC)))
(MAKE1 MAKE ((ACTSBJ) (PRSNT (<=))))
(MOVE1 MOVE ((ACTSBJ) (OBJ) (LOC)))
(OBJECT1 OBJECT ((ACTSBJ (<=> VAL PART)) (COBJ (CON CON))
                 (PART1 (OTHO TO))))
(ON1 1 ON ((ACTSBJ (<=> VAL)) (OBJ (ACTOR))))
(PAY-FOR PAY ((ACTSBJ (CON ACTOR))(OBJ (CON OBJECT))
              (OBJ2 (<=> ACTOR))
              (IOBJ (<=> OBJECT) (MAKPREP FOR))))
(PAY-TO PAY ((ACTSBJ (CON ACTOR))(OBJ (CON OBJECT))
             (OBJ2 (<=> ACTOR))
             (INF2 (<=>))))
(PLEASE4 PLEASE ((OBJ (CON)) (OBJ (<= ACTOR))))
(POISON1 POISON ((ACTSBJ) (OBJ (<= ACTOR))
                 (INST (<= OBJECT)(MAKPREP WITH))))
(POUR1 POUR ((ACTSBJ) (OBJ2 (OBJECT)) (LOC(MAKPREP INTO))
             (IOBJ (FROM)(MAKPREP FROM))))
(PREVENT1 PREVENT ((ACTSBJ) (SPRO (<=))) DELETIONS ((<= MODE)))
(PREVENT2 PREVENT ((ACTSBJ ) (SPRO (<=)) (INST2 (CON)))
(PROMISE1 PROMISE ((ACTSBJ) (OBJ2 (TO PART)) (S2)))
(RAISE1 RAISE ((ACTSBJ) (OBJ) (LOC)))
(READ1 READ ((ACTSBJ) (OBJ (FROM))))
(READ2 READ ((ACTSBJ) (OBJ (FROM))))
(RECEIVE1 RECEIVE ((OBJ) (ACTSBJ (TO))
                   (IOBJ (ACTOR)(MAKPREP FROM))))
(RECOMMEND1 RECOMMEND ((ACTSBJ) (PP1 (TO PART)(MAKPREP TO))
                       (S2 (MOBJECT CON))))
(REFUSE1 REFUSE ((ACTSBJ) (INF (MOBJECT))))
(RELIEVE1 RELIEVE ((ACTSBJ) (OBJ) (INST2 (CON))))
(REMEMBER2 REMEMBER ((ACTSBJ) (S2)))
(REMEMBER3 REMEMBER ((ACTSBJ) (S2)))
(REQUEST1 REQUEST ((ACTSBJ) (OBJ (MOBJECT OBJECT))))
(REQUEST2 REQUEST ((ACTSBJ) (INF2 (MOBJECT CON))))
(RETURN2 RETURN ((ACTSBJ) (OBJ) (IOBJ (MAKPREP TO))))
(RIDE1 RIDE ((ACTSBJ) (OBJ (INST ACTOR)) (LOC)))
(RUN1 RUN ((ACTSBJ) (LOC)))
(SEE2 SEE ((ACTSBJ) (OBJ (MOBJECT))))
(SEE3 SEE ((ACTSBJ) (OBJ (MOBJECT ACTOR))))
```

FIGURE 5-7

```
(SELL1 SELL ((ACTSBJ (<EE> ACTOR))(OBJ (<EE> OBJECT))
            (OBJ2 (CON ACTOR))
            (INST (CON OBJECT) (MAKPREP FOR))))
(SEND1 SEND ((ACTSBJ) (OBJ) (OBJ2)))
(SEND1 SEND1 ((ACTSBJ) (OBJ) (IOBJ (MAKPREP TO))))
(SHOW_UP1 SHOW_UP ((ACTSBJ) (LOC (MAKPREP AT))))
(SMELL1 SMELL ((ACTSBJ) (OBJ (MOBJECT ACTOR))))
(SMELL2 SMEL  ((ACTSBJ (MOBJECT ACTOR))))
(SMOKE1 SMOKE ((ACTSBJ) (OBJ (INST FROM))))
(STAB1 STAB ((ACTSBJ) (OBJ (TO PART))
            (INST (OBJECT (MAKPREP WITH)))))
(START2 START ((INF? ALL)) ADDITIONS ((TIME (TS)))
                           DELETIONS ((TS)))
(STATE1 STATE ((ACTSBJ) (OBJ (MOBJECT))))
(STRANGLE1 STRANGLE ((ACTSBJ (CO* CON ACTOR)) (OBJ (CON ~ ACTOR))))
(SUGGEST1 SUGGEST ((ACTSBJ) (S2) (PP1 (TO PART) (MAKPREP TO))))
(TAKE1 TAKE ((ACTSBJ) (OBJ) (IOBJ (FROM)(MAKPREP FROM))))
(TAKE2 TAKE ((ACTSBJ) (OBJ)))
(TAKE3 TAKE ((ACTSBJ) (OBJ) (OBJ2)))
(TELL1 TELL ((ACTSBJ) (S2) (OBJ? (TO PART))))
(THINK1 THINK ((ACTSBJ (<E> VAL PART)) (S2 (CON))))
(THREATEN1 THREATE* ((ACTSBJ) (INF (MOBJECT))))
(TOUCH1 BE ((ACTSBJ) (PP1 (MAKPREP IN)(OTHO CONTACT))
           (IOBJ (<E> VAL)(MAKPREP WITH)))      )
(TRADE1 TRADE ((ACTSBJ (CON ACTOR)) (OBJ? (<EE> ACTOR))
               (OBJ (CON OBJECT))
               (IOBJ (<EE> OBJECT) (MAKPREP FOR))))
(UNABLE1 BE ((ACTSBJ) (INF? ALL) (P ADJ (OTHO UNABLE)))
        DELETIONS ((MODE)))
(WALK1 WALK ((ACTSBJ) (LOC)))
(WANT1 WANT ((ACTSBJ (<E> VAL PART)) (INF (CON CON))))
(WANT2 WANT ((ACTSBJ (<E> VAL PART)) (OBJ (CON CON ACTOR))))
(WANT3 WANT ((ACTSBJ (<E> VAL PART)) (OBJ (CON CON OBJECT))))
(WARN1 WARN ((ACTSBJ) (OBJ? (TO PART)) (S?)))
```

FIGURE 5-7

Another 'class' of adjectives name locations along continuous dimensions. Many of these dimensions are physical. English provides an abundance of adjectives to describe the physical size of objects -- big, large, huge, vast, enormous, immense, tiny, miniscule, small, little, etc. We can find sets of adjectives for specific dimensions like height, mass, and even velocity listed in a standard thesaurus <27>.

Words in such groups are clearly related in meaning, and this relation must be explicit in a conceptual representation. These words are all relative; that is, "tiny" is not a measurable or perceptual quality like "3 cubic inches", but a relative quality. A "tiny" X is somewhere on the size dimension between a "small" X and a "minute" X. The words are not only relative to other words, but, more importantly, are relative to a norm for objects of a given class. The normal size of elephants and the normal size of rabbits are pieces of conceptual knowledge and are implicitly referenced by such phrases as "a big elephant" and "a big rabbit".

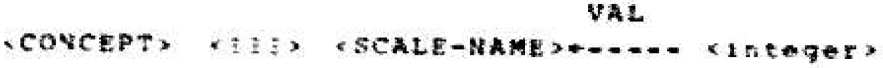
In Conceptual Dependency such relationships are represented with <u>scales</u>. A scale is a list of the form:

$$(n_0\ w_0\ n_1\ w_1\ n_2\ \ldots\ w_{m-1}\ n_m)$$

where each n_i is a real number and

$$-\infty = n_0 < n_1 < \ldots < n_m = +\infty$$

Each w_i is an adjective (or, more precisely, a pointer to the lexical entry for an adjective). The C.D. representation for the location of an object on a scale is



For objects whose location on a scale is at a point K,

$$n_i \leq K \leq n_{i+1}$$

BABEL uses w_i as the appropriate adjective to describe the relationship[6].

How are these relative scales to be related to the actual perceptual representation of the information? The question of how perceptual information is best encoded for computational operation is by no means solved and we do not intend to make new proposals for this here. For specificity, though, let us assume that we represented the perceptual information on a linear scale proportional to some measurable quantity; height, for instance, might be measured in units proportional to feet on an 'absolute' height scale. In order to decide the position of a building X feet high on the 'relative' height scale -- that is, adjectives "tall", "short", "towering", etc. -- we need two pieces of information about buildings:

E -- the average or expected position of a building on the absolute scale

D -- a relation, specific to buildings, between interval lengths on the absolute scale and those on the relative scale -- e.g., 5$ abs. units = 1 rel. unit

To determine the position R on the relative scale of a building at position X on the absolute scale we could use the relation:

$$R = \frac{X - E}{D}$$

The most important aspects of representation by scales are:

1) words correspond to ranges (not points) on relative scales

2) the relative properties have corresponding 'absolute' properties.

Whatever representations are used for these 'absolute' properties, a two-way mapping between the relative and absolute must be provided.

BABEL does not operate with any absolute representation but assumes conversion to relative scales has taken place prior to any request for generation.

The use of scales has been extended to cover certain 'emotional' or 'mental' states as well as physical attributes. This is not done to provide quantitative explanations for phrases like "double your pleasure, double your fun", but

to explain groups of adjectives which behave very much like the physical attribute adjectives. For example, English provides many words to express different degrees of 'excitation': excited, overwrought, agitated, raging, calm, placid, sober, tranquil, peaceful, halcyon.

The abstract scales of 'excitation', 'joy', 'health', etc., do not have absolute counterparts as do the physical scales. In fact, it is not obvious in most cases whether these scales should be thought of as absolute or relative. There do exist cases, however, where the notion of relative scales is clearly applicable. Even though there are no perceptual units for intelligence, we speak of 'smart dogs' and 'smart people' without implying that both possess the same amount of intelligence. Linguistically, at least, we seem to use an 'intelligence' scale in the same fashion as a size or weight scale. And while there exists no irrefutable evidence for the psychological reality of such scales, they have been found useful in psychological models <20> which have been implemented on computers.

Figure 5-8 lists the scales actually included in BABEL. Two points not mentioned in the explanation of scales above become apparent from these examples. First of all, there is on every scale an area about the 'norm' which English just provides no adjective to express. (This may be because of the scarcity of instances in which

```
~ each entry consists of
~ 1)   scale name (an atom)
~ 2)   lexical pointer for change in positive direction
~ 3)   lexical pointer for change in negative direction
~ 4)   list of alternating lexical pointers, numerical scale positions, beginn
ing and
~      ending with lexical pointers.  Assigns names to intervals on the scale.

*HEALTH*
HEALTHY
SICK
(DEAD -9.5 SICK 0 HEALTHY)

*JOY*
HAPPY
SAD
(DEPRESSED -7 SAD 0 HAPPY +8 OVERJOYED)

*ANGER*
CALM
ANGRY
(ANGRY -3.5 UPSET -1.5 NIL +.5 CALM)

*EXCITE*
CALM
EXCITED
(OVERWROUGHT -8.5 AGITATED -5.5 EXCITED -.5 NIL +.5 CALM +4.5 TRANQUIL)

*PSTATE*
OK
HURT
(DEAD -9.5 MAIMED -8. HURT 0 OK)

*SIZE*
BIG
SMALL
(MINUTE -7.5 TINY -4.5 SMALL -.5 NIL +.5 BIG +7.5 GIGANTIC)

CERTAINTY
NIL
NIL
(POSSIBLY +0.84 PROBABLY +0.96 CERTAINLY)
```

FIGURE 5-8

it is desirable to express such information.) In most instances, it is actually quite difficult to come up with an English sentence to express the notion, and we must resort to such expressions as "neither happy nor unhappy". BABEL fails to find a realization for these conceptual forms.

The second point concerns the actual choice of the n_i (breakpoints) on the scales. We have no evidence which leads to particular quantitative choices for positioning the adjectives on the scales. The relative positions of the ranges for "big" and "gigantic" are, of course, derived from their use in language. The actual values chosen for the different ranges are important for two reasons:

A) In translation, the ranges on corresponding scales for different languages must be such that words with corresponding meanings lie in the same range.

B) When inferences are made, they will change scale location values based on events which change such relationships. The intervals on the scales and the inference rules must correspond to the extent that inferences which are realized linguistically will be reasonable. That is, (unless we model characters in TV commercials), people don't get 'ecstatic' over a good cup of coffee, nor do they get 'suicidally depressed' over irregularity.

Since we have not had adequate experience in either creating scales for other languages or writing inference rules which manipulate these scales, the current n_i are purely ad hoc choices.

5.4 Language Specific Functions

The information in the concexicon is sufficient to produce a 'core' syntax network once a verb sense has been chosen. The net thus produced, however, will only express those parts of the conceptual structure being realized which can be predicted from the verb sense chosen. That is, only those parts of a conceptualization which fulfill syntactic relations required by the verb sense are processed in the course of interpreting the concexicon entry. Two other sorts of information must be added to the syntax net to complete it:

1) The conceptualization may express more than simply the required information. It may, for instance, specify the time or location of an event, or some 'parenthetical' information about an event -- e.g., the fact that it ultimately had 'good' results, which might lead to the inclusion of the adverb 'fortunately' in the syntax net.

2) The target language may require the inclusion of certain relationships in the syntax net in order to generate correct surface structures. 'Tense' in English is such a relation, in that main verbs of English sentences must be inflected to indicate one of a fixed set of tenses.

The functions which add such additional information to the syntax nets we call 'Language Specific' (LS) functions. Not all processes in BABEL which are specific to a particular language are included among those we refer to as LS functions. We shall see in section 5.5 that the functions which make up the surface grammar are Eng.ish specific. There are two properties which distinguish LS functions from others. First, they must incorporate knowledge of a particular language. Second, they must require access to conceptual knowledge or to the conceptual structures being realized. It is the latter requirement which separates LS functions from the functions of the surface grammar.

Let us proceed to look in detail at the individual LS functions employed by BABEL to produce English realizations.

5.4.1 Determiners

The conceptual nominals (PPs) handled by BABEL may have REFerence modification. Such modification is currently limited to two values, 'DEF' and 'INDEF'. In the generation of English syntax nets such a modification results in the incorporation of a new relation in the net.

The PP, when it is being realized, causes a node to be created whose LEX value is the English noun which names

that concept. This noun is the name found stored in the relation:

(ENGLISH-NAME <CONCEPT> <LEXICAL UNIT>)

e.g., (ENGLISH-NAME *DOG* DOG)

The function which handles REF modifications attaches a syntax relation 'DET' to this node. The value of this relation is 'THE' if the REF value is DEF. If the value of REF is INDEF, a check is made to see if the concept has the property ENTITY. If so, the value 'A' is chosen; otherwise 'SOME' is selected. Thus:

(*BALL* REF (DEF)) -->	N1:	LEX	BALL
		DET	THE
(*BALL* REF (INDEF))-->	N1:	LEX	BALL
		DET	A
(*BEER* REF (INDEF))-->	N1:	LEX	BEER
		DET	SOME

Selection of determiner is more complicated in German than English because determiners are inflected to show gender. This can be handled by including gender in the 'name' predicate:

(GERMAN-NAME <CONCEPT> <GENDER . LEXICAL UNIT>)

e.g., (GERMAN-NAME *DOG* (MASC . HUND))

5.4.2 PART, POSS, and OWN

PPs may also be modified by the relation PART, as in (*HAND* PART *JOHN*) which specifies a hand which is a bodypart of John. The effect of such a modification is to add a relation POSS to the node created for the PP. The value of this relation is a new node which is expanded to the syntactic representation of the value of the PART relation. Thus:

```
(*HAND* PART (*MAN* REF (DEF))) --> N1:  LEX    HAND
                                         POSS   N2
                                    N2:  LEX    MAN
                                         DET    THE
```

The syntax relation POSS causes a 'possessive' form to be produced by the surface grammar. The above piece of network might eventually be linearized to "the man's hand".

A PP may also be modified by the conceptual relation POSS (indicating the possessor of the object) or OWN (indicating the owner of the object). In BABEL, each of these modifications has precisely the same effect on the syntax net as the PART modification.

Although we haven't implemented functions to deal with these relations in producing German realizations, we note that, while possession and ownership are expressed with genetive (possessive) syntactic structures in German, the PART relationship cannot be handled this way.

In general, German expresses the notion of 'bodypart' with the use of definite determiners ("Norton broke Ali the jaw"). Thus the processes which handle these relationships (or at least the PART relation) must be LS functions.

It has been noted <9> that 'bodypart' relations in English are not always expressed with possessive forms. For example, we say (i) "Ken hit Ali in the jaw" rather than (ii) "Ken hit Ali's jaw". But while we say (iii) "Joe hit Ali's trainer" we cannot express this meaning as (iv) "Joe hit Ali in the trainer".

Several ways to deal with these facts might be considered. We could adopt a transformational component to operate on the syntax nets, essentially deriving (i) from (ii). But such a transformation could not be guaranteed to preserve meaning, because our syntax nets are (potentially) ambiguous. "The dealer hit Hank's hand" would be generated from the same syntax network whether it were in the context "breaking three fingers" or "giving him twenty-one". Only the former is a meaning paraphrasable as "the dealer hit Hank in the hand".

A workable alternative would be to allow the LS function which handles PART relations to hunt around the conceptualization being expressed and decide whether it is appropriate to transform the net. But rather than looking back and possibly changing the net, it is far simpler to look ahead when the concexicon entry HIT1 (the "forceful physical contact" sense) is selected. If the conceptual

object of the 'hitting' is a bodypart, a framework which directly produces the "hit _ in the _" net would be chosen. If not, the standard "hit <OBJ>" framework would be the one used.

5.4.3 TENSE

To every node which has a LEX value which is a verb (henceforth called a verbal node) BABEL adds a syntax relation TENSE. The value of TENSE is chosen from the set:

PAST	PASTPAST	PASTFUT
PRES	PRESPAST	PRESFUT
FUT	FUTPAST	FUTFUT

As the syntax net is being built, two variables, BASETIME and BASETENSE, are maintened. Initially, BASETIME←"NOW", BASETENSE←PRES. In order to choose the TENSE for a verb, a variable NEWTIME is set to the TIME of the conceptualization from which the verb was derived. NEWTENSE is chosen as PAST, PRES, or FUT according to whether NEWTIME is before, the same as, or after BASETIME. If BASETIME is PRES, TENSE is chosen to be NEWTENSE. Otherwise, BASETENSE is PAST or FUT, and TENSE is chosen to be NEWTENSE@BASETENSE (@ ←concatenate). Finally, BASETENSE is updated to the value of NEWTENSE and BASETIME is updated to the value of NEWTIME.

The precise effect of each of these nine tenses on surface realizations is described in section 5.5. We note here that this set of tenses handles only a small part of the English verbal tensing system, although our nine tenses are among the most frequently used. Bruce <3> describes a formal model for dealing with TENSE in natural language which employs both points and time intervals. His model specifies how to relate English tenses to chains of reference times. In order to use this formalism in a conceptual model, it is necessary to choose a chain of reference times to use. Insofar as this question can be treated on a language-free plain -- that is, as a subproblem of WHAT-TO-SAY -- BABEL is not designed to solve it. BABEL's tensing algorithm essentially employs the following heuristic for English:

1) All sentences begin with only time of utterance as a reference point.

2) A sentence embedded in a past or future sentence uses the time of the embedding sentence as a reference point.

Languages differ drastically in the set of time relationships which can be expressed within their tensing systems, and in the methods used for expressing those relationships. For example, the relationship expressed by the simple past tense in English may be expressed as a past perfect in German (in conversation) or as a simple past (in narrative). Tensing thus falls in the domain

of LS functions.

5.4.4 FORM

English sometimes places verbs in a progressive ('ing') form. In general, this form is used to express events taking place during some interval of time rather than the occurrence of an event at a point in time. Since BABEL does not know about time intervals, we have no conceptual source for the generation of such progressive forms. However, English also uses progressive forms in the present tense for most verbs, since simple present denotes habitual action or ability, rather than ongoing action; e.g.,

"John plays baseball" (habitual action)
"John is playing baseball" (ongoing action)

Exceptions to this rule seem to be verbs which express stative, rather than active, relationships: e.g.,

"John knows Bill went home"
"Dave wants to become a doctor"

When a verb is added to a syntax net, BABEL also adds a FORM relation. The value of this relation is 'SIM' (simple) except when two conditions hold: (i) the TENSE is PRES, PRESPAST, or PRESFUT, (ii) the conceptual structure from which the verb was derived is not a STATE. When both these conditions hold, the value of FORM is chosen as 'PROG'.

This results in verbs like 'hit', 'throw', 'give', 'tell', etc. being put in progressive form when used in present tense, but leaves verbs like 'hope', 'want', 'know', 'believe', etc. in simple form regardless of tense. This heuristic correctly produces sentences like the examples above, but fails for another class of English verbs which use simple present tense. These are the perception verbs; e.g.,

"I hear the dog barking"
"Bill sees the red block"

Since these verbs are represented as events (using the ACT *MTRANS*) in C.D., they are generated in progressive form in the present tense. Whether the use of progressives in English is best treated as a set of special cases -- verbs derived from STATES, perception verbs, ??? -- or whether some generalization can better explain their use is an open question.

Form must, of course, be treated as a LS function since it is English specific. German, for example, does not make a progressive - non-progressive distinction in any of its tenses.

5.4.5 MOOD and VOICE

Besides TENSE and FORM, every English sentence exhibits a characteristic MOOD. To every verbal node PABEL adds the syntax relation MOOD. The value of MOOD is chosen from the set {INDIC, INTERROG, COND, SUBJUNC}. INDICative mood is that exhibited by 'information-giving' sentences, such as:

"He expected to fail the exam."

INTERROGative mood is seen in sentences which 'question' information:

"Did he expect to fail the exam?"
"Who expected to fail the exam?"

Interrogative mood is reflected by (i) word order, and, sometimes, (ii) by the introduction of the auxiliary verb 'do'.

SUBJUNCtive and CONDitional mood are used in conjunction to relate counterfactual information:

"If he had come to the game, then we would have won."

The subjunctive posits an 'unreal' situation: "if he had come". This is effected through a change in the tensing of the sentence. The 'conditional' relates an 'unreal' result of such a situation: "we would have won". This is effected through the use of "would" in the verb string.

When choosing the value of MOOD, the program first checks to see if INTERROGative is appropriate.

There are two conditions under which it will be chosen. The first is when a MODE = ? modifies the main link of a conceptualization (e.g., <===> ?). This indicates that the truth value of the conceptualization is to be questioned. The second situation in which INTERROGative is selected is when the marker '?' fills a conceptual role. This role may be one of the cases of an ACT, e.g., ACTOR (usually resulting in a 'who' question), OBJECT ('what'), RECIPIENT ('whom'), or SOURCE or GOAL ('where'). It may be one of the slots of a causal relation. English provides the word 'why' for questioning the ANTECEDENT of most causals, but no special question word for RESULTs. The '?' may also occur in a modifying role, such as TIME ('when') or LOC ('where').

SUBJUNCtive and CONDitional moods are selected by BABEL when it realizes a <:C (can-cause) relation. When realizing this as an 'if-then' syntactic construction, the antecedent is realized as a sentence with SUBJUNCtive mood, and the RESULT as a sentence with CONDitional mood. This produces sentences like:

(1) "John would have died if Mary had stabbed him with the knife"

In some cases <:C relations are realized with a single verb. When this occurs, BABEL places the sentence in CONDitional mood:

(ii) "Bill would like the movie"

In all other cases, INDICative mood is chosen.

Unfortunately our definition of the <:C relation does not justify the simple algorithm which produces (i) above. The <:C relation may indicate a counterfactual, or may simply express an 'open' condition, without placing a truth value on its components. In the latter case, English use indicative mood in expressing both condition and result:

(iii) "If Mary stabbed him with the knife, then John died"

There is no way to know if (i) or (iii) is the appropriate realization from the information in a <:C relation itself.

Two remedies to this problem might be considered. In expressing a <:C, BABEL could ask the memory whether a counterfactual is being expressed -- that is, whether memory believes the corresponding <: relation actually does not hold. MOOD would then be chosen based on the outcome of this decision. Alternatively, we could modify our representation in some way so that the open -- counterfactual 'ambiguity' of <:C did not exist.

More study of the use of subjunctives and conditionals in both English and other languages is needed before a satisfactory treatment of these notions on a conceptual level will be possible. While their use certainly is related to conceptual relationships, it is clear that the English subjunctive cannot itself be considered a conceptual

relationship (which is unfortunate, at least from the point of view of generation). The German subjunctive can be used in the same situations as the English, but can be used in others as well. In English, the sentence:

"The report indicates he is very bright"

states nothing about the speaker's belief of what the report indicates. Nor can this be done except with the addition of an 'and' or a 'but'. In German, however, the embedded sentence "he is very bright" may be realized with SUBJUNCtive mood to indicate disbelief on the speaker's part. We claim that this use should have the same conceptual source (disbelief by the speaker) as the counterfactual use. This of course refutes any suggestion that the English subjunctive is co-occurrent with this conceptual relation.

A note of warning to the reader is in order here. Although throughout this discussion we have exemplified the various moods with sentences, the choice of mood by BABEL, with which we have been concerned, consists solely of attaching the syntax relation MOOD with an appropriate value to a syntax net. The extent to which the program is able to perform the correct syntactic manipulations to express this mood with word order and tensing will be indicated in the surface grammar description in section 5.5.

VOICE is a feature of English syntax distinguished by both word order and verbal form. Traditionally two voices are posited. ACTIVE voice is that seen in sentences in which an 'agent' is the subject:

"John threw the ball"

In PASSIVE voice the 'agent' is no longer the subject, an auxiliary 'be' is added to the verb string, and the participle of the verb is used:

"The ball was thrown by John"

Since we are uncertain as to the conceptual underpinnings of VOICE (is it more than the simple notion of FOCUS we use?) BABEL perfunctorily places the relation-value pair VOICE-ACTIVE on every verbal node and completely ignores the real problem of choosing VOICE.

5.5 Transition Network Grammar

The knowledge needed to produce a sentence from a syntax net resides in an AFSTN grammar, depicted in Figure 5-9. The control algorithm for the grammar is very close to that described by Simmons in <34>. Its function is to take a syntax net node (which we shall call the current node) and a state of the grammar and perform all actions necessary to reach a terminal state of the grammar. (Terminal states are those labeled T in Figure 5-9).

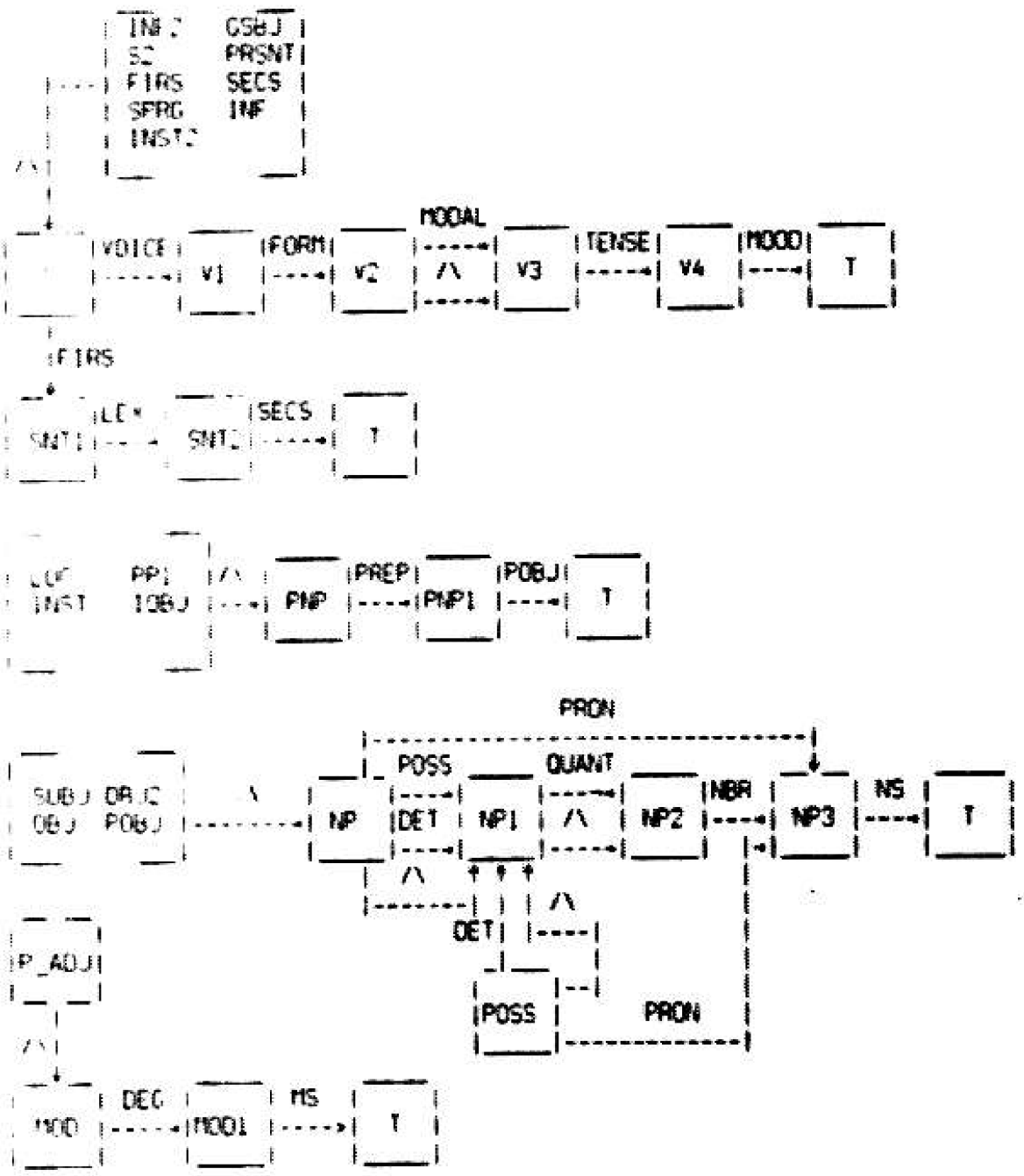

OBJ
PRSNT
FIRS
SECS
INF
MODAL
VOICE
V1
FORM
V2
V3
TENSE
V4
MOOD
T
FIRS
SECS
T
PREP
PNP
POBJ
T
PRON
POSS
QUANT
NP
DET
NP1
NP2
NBR
NP3
NS
T
DET
POSS
PRON
DEC
MOD1
NS
T

FIGURE 5-9

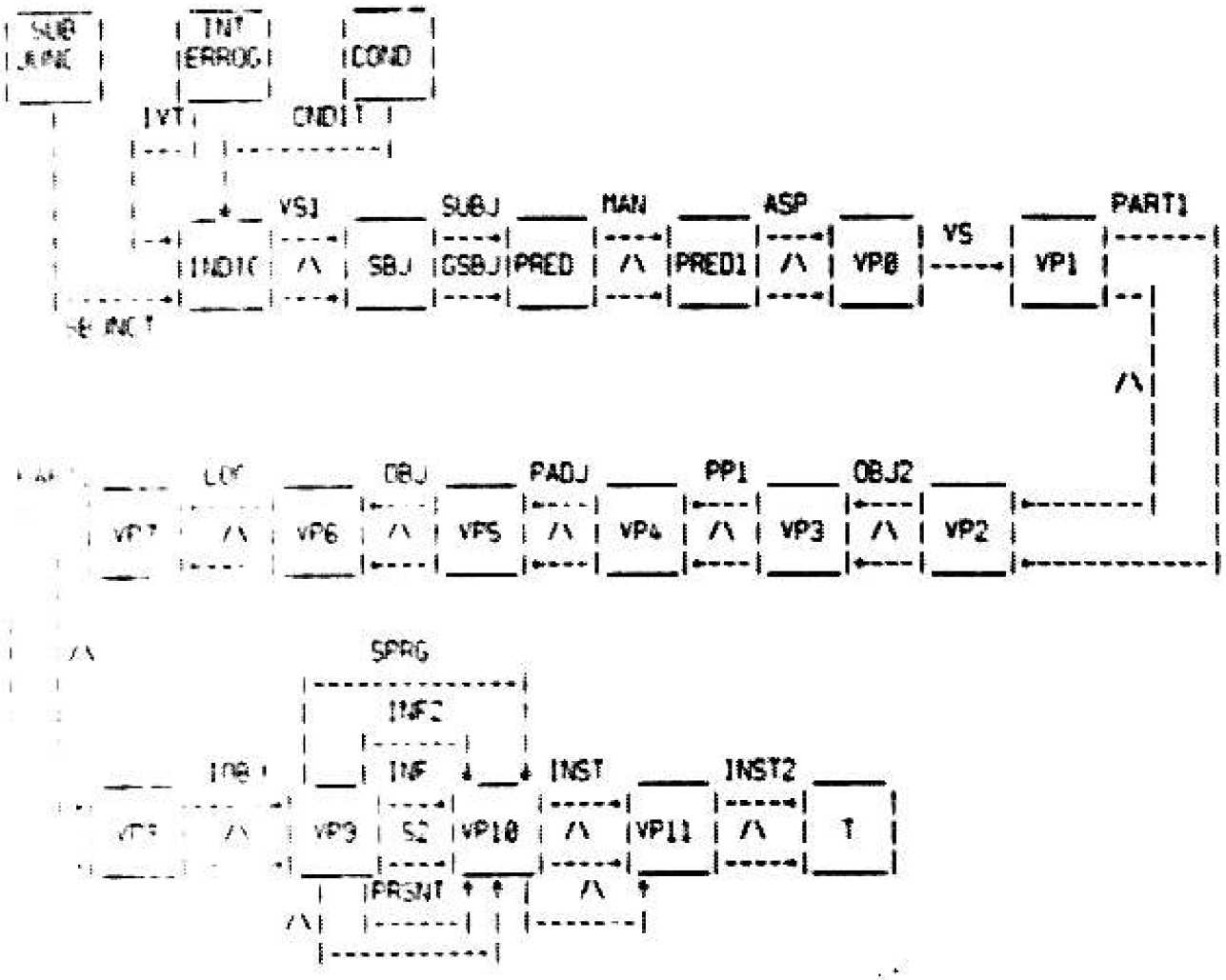

SUB
JUNC
INT
ERROG
COND
IVT
CND
VS1
INDIC
SUBJ
SBJ
MAN
GSBJ
PRED
ASP
PRED1
VP0
VS
VP1
PART1
OBJ
VP6
PADJ
VP5
PP1
VP4
OBJ2
VP3
VP2
SPRG
INF2
INF
VP9
S2
VP10
INST
VP11
INST2
T
PRSNT

FIGURE 5-9

Which actions are necessary depends on the relations attached to the node in the syntax net. The set of relations, and the functions associated with them, are quite different from those used by Simmons, and will be detailed in this section.

The syntax relations of the syntax net occur as arc labels (and, sometimes, as state names) in the grammar. Each arc connects a source, or 'tail', state to a goal, or 'head', state. There are three sources for these relations in the network: (i) the 'syntax relation' field of a FRAME in a concexicon entry, (ii) the relations added by the LS functions, and (iii) certain relations added by the surface grammar itself. Each relation belongs to one of the following classes:

TE -- 'Terminal Element' -- An arc labeled with a TE relation can be traversed if that relation occurs in the network attached to the current node when the arc is reached. In traversing the arc, the value of the relation is concatenated onto the end of the output string being built. Generation then continues from the head state of the arc.

SF -- 'Simple Function' -- An arc labeled with a SF relation can be traversed only if that relation occurs in the syntax net. In traversing the arc, the function with the same name as the arc label must be executed. Generation then continues from the head state of the arc.

EF -- 'Embedding Function' -- An arc labeled with an EF relation also requires the presence of that relation in the syntax net for its traversal. Three things are done:

(1) The function with the same name as the EF relation is executed.

(2) The value of the relation will always be another node of the syntax net. This node and the grammar state having the same name as the EF relation are used as arguments to the generation control algorithm.

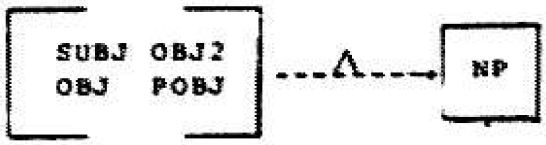
(3) Generation resumes from the head state of the arc.

Since execution of (1) for the EF relations may result in further encounters with EF relations, the generation algorithm must be recursive, stacking 'continuation points' (those specified in step (3)).

DF -- 'Default function' -- differs from SF only in the condition for following the arc. An arc labeled by a DF relation may always be traversed. The associated function specifies a default value for the relation if it is not actually present in the network. In generating noun phrases, for example, an arc labeled NBR is followed. The function NBR assumes the value SINGular if the relation is not actually present in the net.

Many of the arcs in the grammar bear the label '^'. We shall call these 'free' arcs. These may always be traversed, just as those labeled with DFs; in traversing a free arc, however, there is no function to be executed.

There are several instances in our grammar where several different states all have a common successor state, reachable via a free arc, and no other successor state. In Figure 5-9 a set of such states is lumped together into a single, partially closed box.



This represents four different states, {SUBJ, OBJ, OBJ2, POBJ}, each having but one outward path, which is a free arc to the state NP.

Following Simmons, we have a grammar composed of three basic sections: a verb string constructor, a noun-phrase constructor, and a sentence constructor. We now describe the relations and associated functions comprising each.

5.5.1 Verb String Constrution

This portion of the grammar operates first whenever a sentence is to be generated. It begins at state S, which may be reached either as the starting point for generation from a net or recursively through one of the states {FIRS, SECS, S2, PRSNT, INF, INF2, INST2, SPRG, GSBJ}. The node of the net being operated on must be a verbal node. Using the relations TENSE, FORM, VOICE, MODAL, and MOOD associated with the node (all put on by LS functions) a verb string is created and attached to the node as the value of a new syntax relation, VS.

VOICE -- class - SF

This function performs two actions. It creates an initial value for VS, and chooses the node which will eventually become the 'subject' of the sentence. Since we only have one possible value for VOICE (ACTive), this is accomplished very simply. The verb which is the value of the node's LEX relation is made the initial VS. The node which is the value of the node's ACTSBJ relation, if that relation is present, is chosen as the subject. This choice is recorded by attaching the relation SUBJ to the verbal node with the chosen node as its value. When this is done, the relation TYP is also attached to the verbal node, to indicate the 'type' (person, singular or plural) of the subject, since the final verb string must be inflected to reflect this. The only verbs in BABEL's vocabulary which do not have an ACTSBJ relation are those like 'annoy' which which have gerund phrases as subject. For these, no SUBJ relation is formed, but TYP is labeled as SING3, since English uses 3rd person singular inflection for these:

"Writing this paper annoys me."

A more complicated function would be needed to handle passive voice, but no theoretical problems are posed, since none of the necessary manipulations involves the use of conceptual knowledge, or of any other information not present in the syntax net.

FORM -- class = SF

If the value of the FORM relation in the syntax net is SIMple, this function does nothing. If it is PROGressive, the function changes VS to BE+progressive form(VS). Thus if VS = HIT and FORM were PROG, VS would be transformed to BE+HITTING.

MODAL -- class = SF

A verbal node may or may not have a MODAL relation associated with it. In the current program, it will be present only if the verb is a realization of a

C
<===> structure, in which case it will have the value CAN. The SF MODAL simply concatenates the value of the relation onto the front of the VS.

TENSE -- class = SF

This function gets the value associated with TENSE and applies another function (whose name is the same as the TENSE value) to the first word of the current VS. The result is then concatenated onto the front of the remainder of the VS. These tense functions are:

```
PRES (V) = the present tense form of V
PAST (V) = the past tense form of V
FUT  (V) = PRES(BE)+GOING+TO+INFIN(V)
PASTPAST (V) = HAD+past-participle(V)
PRESPAST (V) = PAST(V)
FUTPAST  (V) = PAST(BE) +GOINT+TO+INFIN(V)
FUTFUT   (V) = FUT (V)
PRESFUT  (V) = PRES(V)
PASTFUT  (V) = PAST(V)
```

In general, the determination of present, past, or future form of a verb must take into account the value of TYP, originally set by VOICE. That is, PRES(BE) may be IS, AM, or ARE, depending on TYP. The two cases where INFIN(V) appears are needed to handle cases in which CAN is present in the VS. We define CAN to have the INFINitive BE+ABLE+TO. For all other forms, the INFINitive is identical to the value of the LEX relation in the syntax net.

MOOD -- class = SF

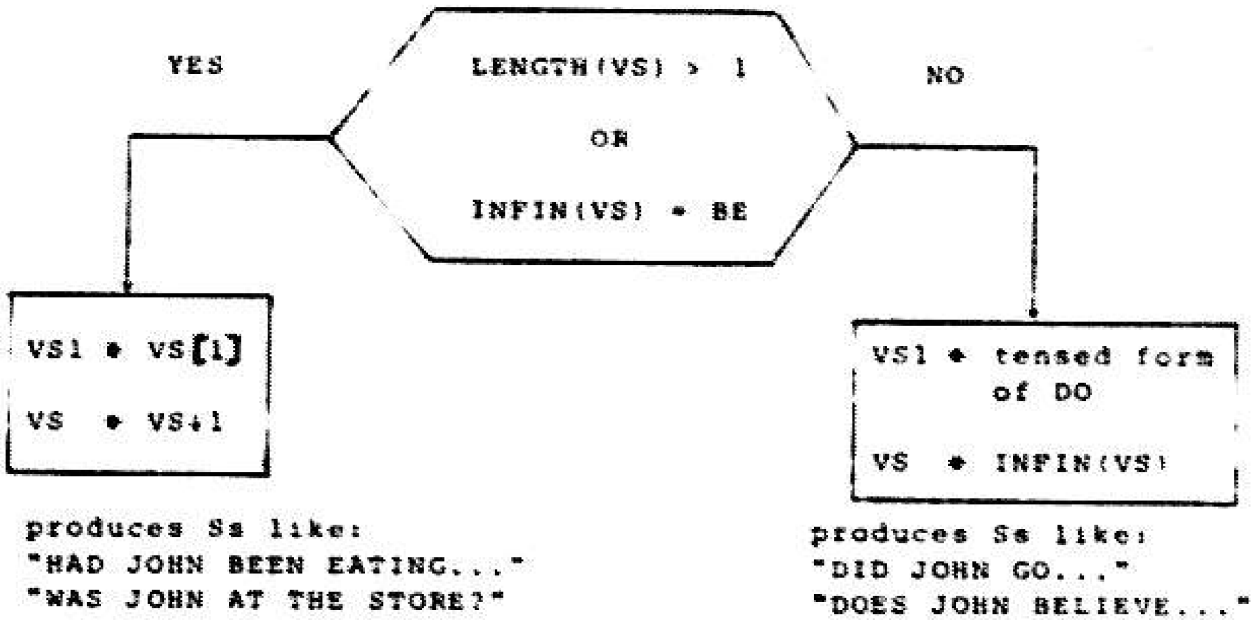
MOOD is a function which serves to change the current state in the grammar. The new state reached is the one whose name is the same as the value of the MOOD relation, which must be one of {INDIC, INTERROG, COND, SUBJUNC}. MOOD is the only SF in the grammar which changes the state to one other than that at the head of its arc.

CNDIT -- class = DF

The function CNDIT is performed for sentences with CONDitional MOOD. No CNDIT relation is ever actually present in the syntax net; the function is performed because it is in the DF class and is the only path out of the state COND. The function converts VS to WOULD+INFIN(VS[1])+VS+1, where VS[1] indicates the first element of VS and VS+1 indicates the remainder of VS after the first element has been removed. Thus the VS "WAS+GOING+TO+RUN" is converted to "WOULD+BE+GOING+TO+RUN"

IVT -- class = DF

IVT is the analog of CNDIT for INTERROGatives. This function does the correct thing only for 'yes-no' questions; BABEL does not have a general English question syntax. IVT creates a new syntax relation VS1, whose value will be a verb string which ultimately precedes the sentence subject. IVT also alters VS, the verb string which will follow the sentence subject. This is accomplished as shown in the following flow chart:



SBJNCT -- class = DF

No function has been implemented to handle the syntax of subjunctive mood. The surface grammar generates sentences from nets marked with SUBJUNCtive MOOD exactly as though they had been marked INDICative. Thus we get:

"If John went to the store ...", instead of
"If John had gone to the store ..."

5.5.2 Noun Phrase Construction

Noun phrases are constructed by the grammar segment beginning with state NP. This state is reached by traversing the free arc from one of the states (SUBJ, OBJ, OBJ2, POBJ). The current syntax net node at the time state NP is reached will always have a LEX value which is a noun.

We shall refer to such a node as a nominal node. In addition, a relation CASE with value NOMinative, OBJective, or POSSessive will have been attached to the node.

A new syntax relation NS, the analog of VS in the verb string constructor, is added to the node. The value of this relation is then transformed into a complete noun phrase, which is concatenated onto the output string. We now describe each of the functions involved in this process.

PRON -- class = SF

If the relation PRON is associated with the nominal node, its value will be a pronoun in nominative case. This pronoun, in the case form specified by the case relation, is made the value of NS. Thus, if PRON has the value HE, NS may be set to HE, HIM, or HIS.

POSS -- class = EF

If the relation POSS is associated with the nominal node, its value will be another nominal node. (This relation may have come from a conceptual POSS OWN, or PART relation, or, if the conceptual representation were extended from that permitted by BABEL, from meanings like "John's uncle", "John's responsibility", etc.)

The function POSS attaches the relation CASE with value POSS to the specified node. Since POSS is an EF, the generator algorithm is then applied to this node starting from the state POSS. This results in the formation of a string like "MARY'S" or "THE DOG'S", which is concatenated onto the end of the output stream.

DET -- class = SF

The value of DET, which will be A, THE, or SOME, is made the first element of NS.

QUANT -- class = SF

The value of QUANT, which is always an integer, is concatenated onto the end of NS. In addition, if the integer is greater than 1, the relation NBR with value PL is attached to the nominal node.

NBR -- class = DF

NBR takes the noun of the nominal node, places it in plural if the relation NBR with value PL is associated with the node, and then puts the noun in the correct CASE. The result of this process is then concatenated onto the end

of NS. In addition, the relation PRON is attached to the node, with its value being the nominative case pronoun appropriate for the node's noun. If this node is ever again used for NP generation, the PRON arc will be followed. This is the only way pronouns become part of the output string.

NS -- class = TE

The noun phrase built up as the value of NS is concatenated onto the end of the output string.

Prepositional phrases are generated from the grammar segment beginning with state PNP. This state can be reached only from one of the states {LOC, PP1, INST, IOBJ}. When the state PNP is reached, the current syntax net node will have two relations: (i) PREP, whose value is a preposition, and (ii) POBJ, whose value is a nominal node. The effects of these relations can be simply described.

PREP -- class = TE

The value of the relation PREP, an English proposition, is concatenated onto the output string.

POBJ -- class = EF

The function POBJ attaches the relation-value pair CASE-OBJ to the nominal node which is the value of the relation POBJ. Since POBJ is an EF, the generator proceeds to generate from the state POBJ using this nominal node as the current node. The state POBJ leads via a free arc to the state NP, resulting in the production of a noun phrase object for the preposition.

5.5.3 Sentence Construction

Production of the complete sentence begins at the state INDIC, which is reached either because the relation MOOD had value INDIC, or because a path to INDIC from one of the other 'mood' states was traversed. The current node will always be a verbal node at this point. The grammar now combines the verb strings, VS and VS1, with generated noun phrases and other elements to produce the final sentence in a left-to-right fashion. The functions which accomplish this are:

VS1 -- class * TE

If the relation VS1 is present (which will be the case if and only if the sentence is in INTERROGative MOOD), the value of VS1 becomes the first element of the output string.

SUBJ -- class = EF

If the relation SUBJ is present (it will have been attached by the VOICE funtion) its value will be a nominal node. The function SUBJ marks this node as being NOMinative CASE, and, since SUBJ is an EF, control passes to the state SUBJ with the nominal node as current node. This leads to the production of a noun phrase as the next element of the output string.

GSBJ -- class = EF

In no SUBJ relation exists, a GSBJ (Gerund SUBJect) relation will. This relation is part of the framework in BABEL's concexicon for every verb sense which has no ACTSBJ frame. The value of GSBJ is always a verbal node N. The function GSBJ affects this node. It attaches a relation VS to the node. The value of VS is the progressive form of the verb associated with N (this verb is the value of N's LEX relation.) GSBJ also attaches two flags to N, VS-MADE and DEL-SUBJ. GSBJ, being an EF, causes a transfer to the state GSBJ with N as current node. The state GSBJ leads to state S, and a sentence is generated from N and placed in the output string. The flags affect the generation of this sentence. VS-MADE inhibits all the

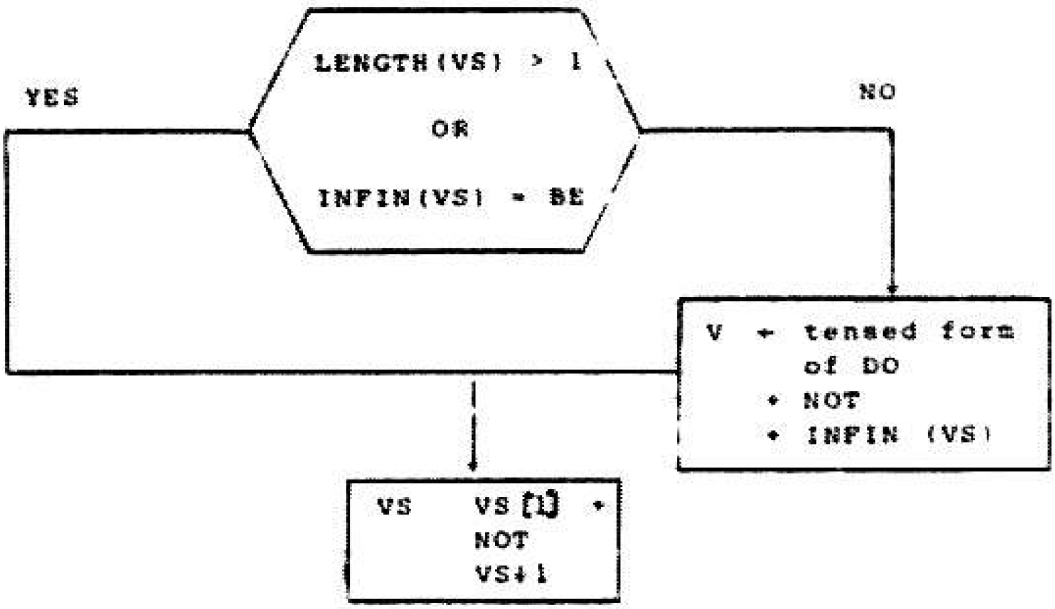
regular VS building actions of VOICE, TENSE, FORM, etc. And DEL-SUBJ inhibits expansion of SUBJ in the generation of this sentence -- that is, no subject NP will be produced. Thus, if the network attached to N might normally produce "John sold Bill a bike for 50 dollars", its generation as a GSBJ within another sentence would produce "selling Bill a bike for 50 dollars".

MAN -- class = TF

MAN is the relation added by the LS function which handles CERTAINTY. Its value is always an adverb, which is placed directly in the output string.

NGT -- class = SF

If this relation is present, it will have the value NOT. The function NGT inserts NOT into the VS as shown in the following flow chart:



VS -- class = TE

The verb string which is the value of the relation VS is placed in the output string.

PART1 -- class = TE

Certain verb senses are expressed in English with 'PARTicles' attached to the verb string: "He sat in at the administration building." These particles are placed in the output immediately after VS.

OBJ2 -- class = EF

Certain verbs of English can take two objects -- that is, a declarative, active voice sentence using these

verbs will have two noun phrases following the verb. OBJ2 is the relation for the 'leftmost' of these: "I loaned John the screwdriver." The function OBJ. merely marks the nominal node which is the value of the relation OBJ2 as being 'objective' case. The NP grammar segment comes into play just as it does with SUBJ to place a noun phrase in the output string.

PP1 -- class - EF

PP1 is a relation for the leftmost prepositional phrase of the sentence, if it precedes the object of the verb: "The witness admitted to the judge he had been at the meeting." The state PP1 has a free arc to state PNP from which prepositional phrases are produced. No change is made to the syntax net.

P_ADJ -- class - EF

This is the Predicate ADJective slot: "John is sick". The value of the relation P_ADJ is a node whose LEX value is an adjective. The state P_ADJ leads via a free arc to the state MOD, from which adjectival modifying strings are produced. The grammar provides for the node having a relation DEG with value COMParative or SUPerlative, but

there are no cases where BABEL actually generates a DEG relation. The adjective value of the LEX relation becomes the modifying string.

OBJ -- class = EF

The OBJ relation is for the direct object of a verb: "John hit Mary because he disliked her." The state OBJ leads to NP via a free arc . The function OBJ marks the nominal node which is the value of the OBJ relation as being in OBJective CASE.

LOC -- class = EF

LOC is another relation which leads to the insertion of a prepositional phrase in the output string. It is generally provided for 'locative' phrases: "He read the book in his room". It could be used for any prepositional phrase which fits at this spot in the sentence, however. The state LOC leads via a free arc to PNP.

PART -- class = TE

This is for verb 'particles' which do not 'stick to' their verbs (see PART1 above). "The prosecutor handed the document back to the witness". The word which is the

value of PART2 is concatenated onto the output string.

IOBJ -- class = EF

This is another slot for prepositional phrases, in particular, for those which follow the direct object of the verb. "The President asked his staff for a report." The state IOBJ leads via a free arc to PNP.

The transition from state VP9 to VP10 provides for the insertion of several different types of embedded sentence in the output string. Each of these types has its own relation, belonging to the EF class. The value of these relations is always a verbal node N. The embedded sentence is generated by passing this node back to state S of the grammar, just as was done with the GSBJ relation. As in that case, the exact form of the embedded sentence is determined in part by flag settings.

S2 -- class = EF

This is the simplest form of embedded sentence, being generated just as though it were not embedded. S2 performs no special actions and sets no flags. "John told the librarian Bill had taken the book."

INF2 -- class = EF

The function of INF2 adds the relation VS to N. The value of VS is TO+value of N's LEX relation. INF2 also puts a VS-MADE flag and a DEL-SUBJ flag on N. The affect of these flags was described with the GSBJ function above. 'ADVISE1', for example, has a frame with an INF2 relation "The colonel advised the general to order a retreat."

PRSNT -- class = EF

Certain embedded Ss have verb strings which utilize the infinitive form of the verb without the preposition TO: "We watched the Giants lose the game" "His mother made him stay at home". The function PRSNT attaches a VS relation N, its value being N's LEX value. PRSNT then sets the VS-MADE flag.

INF -- class = EF

The function INF is identical to INF2 with one exception. INF does not put a DEL-SUBJ flag on N. Instead, INF adds the relation INFOF to N, its value being the verbal node governing the INF relation (i.e., the embedding

sentence.) SUBJ checks for this relation. If it is present, and the SUBJect relations of both embedding and embedded sentences have the same node as their value, the subject NP is not generated, exactly as if DEL_SUBJ had been set.

"John wants his father to take him to the ball game." (subjects don't match)

"John wants to go to the ball game." (subjects match)

SPRG -- class = EF

This handles embedded Ss which use progressive verb forms preceded by "from": "He tried to prevent the senator from making a big mistake". The function SPRG adds a VS relation to N, its value being FROM+progressive of N's LEX value. SPRG sets the VS-MADE flag.

INST -- class = EF

This is another prepositional phrase relation. It is frequently useful to enter those prepositional phrases sometimes termed 'instrumental' in English: "He tightened the bolt with a wrench." We do not restrict its use to such cases, however. It is merely a slot for the insertion of a prepositional phrase which may come after an embedded

sentence: "We advised the owner to have his brakes checked in the letter."

INST2 -- class = EF

Many English sentences have 'instrumental' phrases which have the form BY + progressive form of verb + predicate: "The doctor requested that his patient pay the bill by sending the patient a letter." These are handled by the INST2 relation in BABEL and are placed as the 'rightmost' construction in any sentence. INST2 is identical to SPRG except for two features: (i) INST2 uses BY rather than FROM in the VS, (ii) INST2 sets a DEL_SUBJ flag on N.

Finally, sentences which are conjunctions are generated by the path:

FIRS and SECS are EFs which lead to state S. Each of them thereby causes a sentence to be produced from the verbal nodes which are their values. LEX is a TE which simply inserts its value ('AND' or 'BECAUSE') in the output between these two sentences.

5.5.4 Remarks

Certainly BABEL's surface generator contains neither descriptive formalisms nor implementation mechanisms drastically different from other currently popular systems -- e.g., transformational grammar, semantic nets, or systemic grammar. Philosophically, however, we have taken positions which differ markedly from those generally ascribed to other grammars:

1) BABEL's surface grammar is not designed to relate meanings to strings. It is concerned solely with constituent structure and constituent ordering. This is not to deny that these features are often related to meaning. BABEL employs such knowledge, however, in creating its syntax nets: the process of generating a sentence from such nets, which is logically distinct, makes no use of this knowledge.

2) BABEL's grammar is one-directional; it is not intended to be useful for language analysis. Riesbeck <25> discusses why it is neither necessary nor desirable to produce a syntactic description of a sentence (such as is embodied in our syntax nets) in the process of conceptual analysis.

3) The surface grammar is definitely a performance and not a competence grammar. No claim is made that either BABEL or its surface grammar should generate all 'grammatical' English sentences. There will exist possible syntax nets which would lead the grammar to generate ill-formed sentences: theoretically such nets should not be created by the conceptual → syntax mapping. As far as meaning is concerned, there is absolutely no check anywhere in the BABEL system that the information being expressed makes sense, NOR SHOULD THERE BE ANY SUCH CHECK. It is the task of underlying conceptual mechanisms to see that a conceptualization 'makes sense'. If that mechanism decides that the idea of "colorless green ideas sleep furiously" makes sense, well, so be it. There is no reason for

> the generator to decide an idea shouldn't be expressed. Its sole job is to find the right sequence of words to express the meaning it is given. If it is given semantic nonsense the responsibility (and the blame) must lie with the process which produced that nonsense.

The grammar described above is extremely limited in the range of English constructions it can produce, even by current computer standards. And even those produced are sometimes handled in overly specific ways. For instance, our notion of SPRG -- embedded Ss with progressive VSs introduced by FROM, as in "prevent him from falling" -- should be generalized to make the preposition a parameter, thereby handling "talk him into selling", ask about buying", etc. The main reason this has not been done is that we have tried to focus on those aspects of language production which involve conceptual knowledge rather than pure syntactic knowledge. We believe the format in which syntax is handled is adequate to incorporate the sorts of syntactic knowledge used by more advanced grammars.

One interesting feature of BABEL's surface grammar is its categorization of embedded Ss into syntactic classes (INF, INF2, SPRG, etc.) based not on the content of the embedded sentence, but on the main verb of the embedding sentence. We make no claim to having exhausted the types of embedding found in English, but the work done to date would indicate that the number is not large (probably no

more than two to three times the number discriminated by BABEL1.

5.6 Lexicon

The lexicon used for surface generation by BABEL is trivial. This lexicon is in no way like the concept of a lexicon postulated by transformational grammarians which would include things like 'complex symbols' and syntactic environment frameworks. It is not intended to be useful for language analysis in any way, but only to serve a few simple morphological requirements of the surface generator. It consists of a set of properties and a list of 'object - property value' pairs associated with each. The entire lexicon is shown in Figure 5-10. The properties used are:

PAST -- irregular past tense forms are given explicitly in the lexicon. All others are computed by appending "d" or "ed" to the infinitive form of the verb. The infinitive is the printname of the lisp atom used as the name for a lexicon entry and found in the first field of a concexicon entry.

|EN -- irregular past participle forms are given explicitly. All others are formed by appending "d" or "ed" to the infinitive form.

```
        SING3
(
(BE IS)(GO GOES)(HAVE HAS)(DO DOES)(CAN CAN)
)

        PAST
(
(BE WERE)(BECOME BECAME)(BUY BOUGHT)
(CAN COULD)(COME CAME)
(DO DID)(DRINK DRANK)(EAT ATE)
(GET GOT)(GIVE GAVE)(GO WENT)(GRAB GRABBED)
(HEAR HEARD)(HAVE HAD)(HIT HIT)(KNOW KNEW)(MAKE MADE)(READ READ)
(SEE SAW)(SELL SOLD)(TAKE TOOK)(TELL TOLD)(THINK THOUGHT)

)
        ING
(
(BE BEING) (HAVE HAVING) (GRAB GRABBING) (STAB STABBING)
)
        EN
(
(BE BEEN)(BUY BOUGHT)(COME COME)(CAN BEEN-ABLE-TO)
(DO DONE)(DRINK DRUNK)(EAT EATEN)
(GET GOTTEN)(GIVE GIVEN)(GO GONE)
(HAVE HAD)(HIT HIT)(HEAR HEARD)
(KNOW KNOWN)(MAKE MADE)(READ READ)
(SEE SEEN)(SELL SOLD)(TAKE TAKEN)(TELL TOLD)(THINK THOUGHT)
)

        PRON
(
(JOHN HE) (BILL HE) (MARY SHE) (FRED HE)
(HAMLET HE) (HAMLET HE) (OTHELLO HE) (IAGO HE) (CASSIO HE)
(DESDEMONA SHE)
(FALSTAFF HE)(SOMEONE HE)
)
        CONJ
(
(AND T) (BECAUSE T)
)

        OBJ
(
(HE HIM) (SHE HER) (IT IT) (THEY THEM) (I ME) (YOU YOU) (WE US)
)

        POSS
(
(HE HIS) (SHE HER) (IT ITS) (THEY THEIR) (I MY) (YOU YOUR) (WE OUR)
)

        INF
(
(CAN BE-ABLE-TO) (IS BE)(HAS BE)(HAD HAVE)(HAS HAVE)
)
```

FIGURE 5-10

SING3 -- Irregular third person singular forms for verbs are given explicitly. All others are formed by appending "s" to the infinitive form.

PRON -- All nouns which require the pronouns 'he' or 'she' in the third person nominative singular are explicitly given. All others are assumed to use 'it', and all nouns are assumed to use 'they' in the nominative plural.

OBJ -- The objective case form for pronouns is listed. All nouns are assumed to have identical objective and nominative forms.

POSS -- The possessive case for pronouns is listed. Possessive case for nouns is formed by appending "s" to the nominative (singular or plural) forms.

CHAPTER 6

THE PROCESS OF GENERATION -- THE USE OF LINGUISTIC KNOWLEDGE

The task of converting a conceptualization into an English sentence, referred to as realizing that conceptualization, requires more than the static linguistic and conceptual information detailed in Chapter 5. There must exist a process which utilizes this knowledge to produce the realization. The comprehensiveness of this process -- that is, the domain of meanings for which it is capable of producing 'acceptable' realizations -- provides a measure of the adequacy of the knowledge base. The efficiency of this process provides an indication of how well organized this knowledge is -- e.g., how well it captures linguistic generalities. We have presented the knowledge base of BABEL with but few arguments to support our organization over other possibilities. BABEL's discrimination nets, for example, are organized around conceptual ACTs. One could organize the nets according to time of events (past, present, or future with respect to time of generation) or some even less meaning oriented feature such as the number of conceptual cases present in a stimulus. Alternatively, one might encode this same knowledge in a format quite different from a discrimination net. There are countless organizational possibilities which will provide equivalent input/output

behavior. The differences lie in processing efficiency. No attempt will be made to justify our particular organizational decisions for two reasons:

A) While numerous other possibilities exist, none has been seriously proposed. Some could be eliminated for obvious, but uninteresting reasons. In other cases we could give only intuitive, but perhaps unconvincing, arguments.

B) Any argument favoring a particular organization will be based primarily on efficiency considerations. But there is no accepted standard for measuring efficiency in the task of conceptual generation. Should 'expected' generation time, or 'maximum' generation time, or some function of time and memory requirements be minimized? More importantly, are we interested in changes with respect to vocabulary size, conceptual domain, or what? While these are interesting questions, and ones which must be pursued eventually, a discussion of them would add little to an understanding of BABEL.

In this chapter we will describe the process by which an English realization of a C.D. structure is produced. This process takes an arbitrary conceptualization as input and creates a syntax net. The functions which accomplish this have access to the conceptual memory model, and thus to conceptual knowledge. Once the syntax net is completed, the conceptualization is discarded and a second set of functions produces an English sentence from the net. These latter functions have no access to conceptual knowledge, but are concerned solely with what we consider surface syntax of English.

Once the method of producing single realizations of conceptualizations is understood, it requires but slight

expansion to understand the production of multiple realizations from a single conceptualization, or paraphrase production.

6.1 Initialization

Before realizations of conceptual structures can be produced, an initialization process must be carried out. This process associates various linguistic knowledge data files described in Chapter 5 with LISP atoms and sets up internal pointers to enable the program to access the information. For the most part this operation is quite straight-forward, being accomplished by storing information on LISP property lists (P-lists). We shall use the notation

<property-name> (<atom>) = <value>

to represent the association of <value> with <atom> under <property-name>. Such operations will be mentioned only briefly. In a few cases more complex processing of the files takes place, involving manipulation and addition of pointers in list structures. These operations, and their purposes, will be described in more detail.

For each syntax relation (<SR>) with a default field-specification (Section 5.2), the default value is placed on the P-list of the <SR> under the property name FRAM-STDS -- e.g., FRAM-STDS (ACTSBJ) = ACTOR .

Each conceptual dependency link is represented by a LISP atom. The code (section 5.1) associated with each link

is placed on the P-list of this atom under the property name LNKCODE -- e.g., LNKCODE (<*>) * E .

Each of the conceptual relations which has a language specific function associated with it (Section 5.4) has the name of that function placed on its P-list under the property name LSF -- e.g., LSF (REF) * CHOOSE-DETERMINER .

Each of the syntax relations which requires a new syntax net node as its value (these are indicated below) has the flag NSTRUC placed on its P-list -- e.g., NSTRUC (ACTSBJ) * TRUE .

Each property scale (figure 5-8) is placed on the P-list of its scale-name under the property SCALE -- e.g., SCALE (*JOY*) * (-10 DEPRESSED . . .) .

Each entry in the Concexicon file (figure 5-7) is a list of the form:

(<entry-name> <lexicon-pointer> <framework> <special-actions>)

<entry-name> is a unique atom for each entry (e.g., BUY1), and <lexicon-pointer> is another atom, whose print name is an English word (usually the infinitive form of a verb). <framework> and <special-actions> were described in section 5.2. The initialization process places the <lexicon-pointer> onto the P-list of <entry-name> under the property LEX -- e.g., LEX HAVE1) * HAVE -- the <framework> onto the same P-list under the property FRAMES, and the <special-actions> onto the P-list under the property SPECACT.

The file of predicates used in the discrimination nets is

initially stored in a LISP array ALLPS. We denote the Ith predicate in this file as $PRED_I$ (it must be one of the forms described in section 5.1). Each element of ALLPS consists of a predicate and a flag; all these flags are set to NIL during initialization. Following initialization:

$$ALLPS(I) = (PRED_I . NIL) \quad I=1,2, \ldots N$$

where N is the total number of distinct predicates. The purpose of the flag will be described shortly.

The next phase of initialization is the construction of the discrimination nets (D-nets). The external format of a D-net -- that is, the form in which one is constructed by the user of BABEL -- is a binary tree structure conforming to the following syntax:

<D-net> ::= (<non-term node> <l-subnet> <r-subnet>) |
(<response node> <back pointer>)

<l-subnet> ::= <subnet>

<r-subnet> ::= <subnet>

<subnet> ::= <D-net> | <back pointer>

<back pointer> ::= <positive integer> | NIL

<non-term node> ::= <positive integer list>

<response node> ::= <concexicon-entry list>

A <non-term node> is a list of integers which are indices of predicates in ALLPS. An integer occuring as a <backpointer> in a D-net must be the index of some node of that D-net. These indices are determined by assigning the root node index 1 and the left and right subnets of a D-net whose root node has index M the indices 2M, 2M+1, respectively.

A <conceptual-entry list> is a list of atoms which occur as <entry-name>s in the conceptual file.

The initialization process converts this external format to an internal one. In doing so, the tree structure is converted to a network. Each integer I in the list comprising a <non-term node> is replaced by a pointer to ALLPS[I]. Each branch to a non-null <back pointer> is replaced by a pointer to the node which that <back pointer> references. As a result of these replacements none of the nets contain indirect references to either predicates or D-net nodes. Each initialized D-net is stored as the value of a distinct LISP atom. We shall refer to the D-nets by the names of these atoms. BABEL includes 15 different D-nets:

NET NAME	APPLICABILITY
AND	<CONJUNCTION>s
ATRANS	<EVENT>s with the ACT *ATRANS*
BELIEV	<STATE>s with the <ATTRIBUTE> *MLOC*
DK	mutual causation
EKC	<EVENT> cause <STATE-CHANGE> relations
EKE	<EVENT> cause <EVENT> relations
EKS	<EVENT> cause <STATE> relations
EVT	<EVENT>s
GRASP	<EVENT>s with the ACT *GRASP*
INGEST	<EVENT>s with the ACT *INGEST*
KAUS	<CAUSAL> relationships
MTRANS	<EVENT>s with the ACT *MTRANS*
PTRANS	<EVENT>s with the ACT *PTRANS*
SC	<STATE-CHANGE>s
STAT	all <STATE>s

The AFSTN grammar (section 5.5) is stored entirely on P-lists. Every non-terminal node in the grammar has a list

of <PATH>s stored on its P-list under the property name AFSTN. Each <PATH> is either a pair of atoms (<arc label> <goal node>), or in the case of a 'free arc', simply the atomic <goal node>-- AFSTN (NP) = ((PRON NP3) (POSS NP1) (DET NP1) NP1) .

Finally, the lexicon (figure 5-10) is set up. For each property type (e.g., SING3), each of the atom-value pairs listed with it is processed by placing on the P-list of the atom under the property named by the type the value indicated -- SING3 (BE) = IS .

The initialization process is performed only once; it is not necessary to reinitialize the system for each realization.

6.2 Selection and Application of Discrimination Nets

When BABEL receives a conceptual representation (C.R.) to be realized, its first action is to select a set of one or more D-nets to use in an attempt to discover a main verb for the output sentence. The choice is made by means of a quick structural matching of the C.R. against known patterns. The skeleton of the C.R. (section 5.1) is formed.

if the SKELETON is:	the set of D-nets used is:
A	AND
C	SC
EKC	EKC,KAUS
EKE	EKE,KAUS
EKS	EKS,KAUS
xKy (other than above three)	KAUS
D	DK

If the skeleton is S, and the <ATTRIBUTE> of the conceptualization is *MLOC*, the D-net BELIEV is used; for conceptualizations with skeleton S and other <ATTRIBUTE> fields, the D-net STAT is used. If the skeleton is simply E (an EVENT), the D-net EVT will be in the set used. In addition, if the ACT of the EVENT is one of (GRASP, INGEST, PTRANS, ATRANS, MTRANS), the D-net of the same name will also be placed in the set.

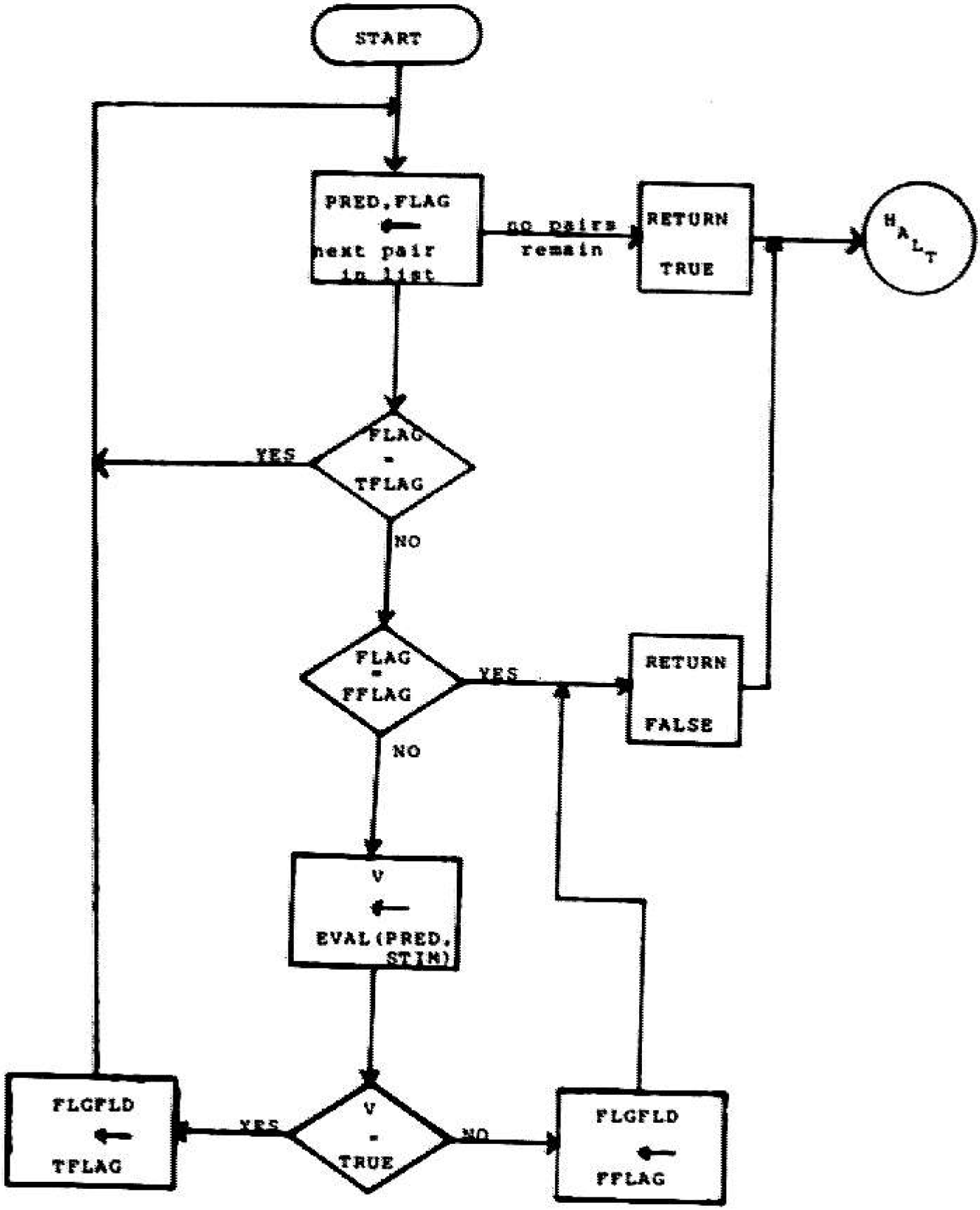
The D-nets in the set thus selected are sequentially applied to the conceptualization until a response is found. The algorithm for applying a D-net is basically that given in section 5.1. Certain details were omitted from that algorithm and will now be presented. It should be borne in mind that this discussion applies to the application of D-nets to embedded conceptualizations as well as those passed to BABEL by the memory for realization.

Each time a D-net is entered, the variables TFLAG and FFLAG are assigned unique values. Each non-terminal node of a D-net, after initialization, is composed of a list of entries in the array ALLPS -- that is, a list of predicate-flag pairs. The value of a predicate at a non-terminal node is taken to be the value of the conjunction of the predicate parts of those pairs. The evaluation takes place from left to right, stopping as soon as one of the predicates evaluates false. Before the predicate part of any pair is actually evaluated, however, a check is made to see if the flag part matches the value of either TFLAG or FFLAG. If a match is found, evaluation of the predicate is

inhibited and the value TRUE or FALSE is assumed, depending whether the match was with TFLAG or FFLAG, respectively. If no match is found, evaluation of the predicate takes place and the result is saved by storing TFLAG or FFLAG in the flag part of the pair, depending on whether the value obtained was TRUE or FALSE. This use of the flag associated with each predicate ensures that no predicate will be evaluated more than once in the application of a D-net to a conceptualization, regardless of how many nodes of the net reference a given predicate. Figure 6-1 depicts in flowchart form the process which evaluates a predicate at a non-terminal node for a conceptual stimulus STIM.

The overall net application process is depicted in Figure 6-2. Initially the variable NET is set to the entire network structure of the D-net (as defined by the syntax in the preceding section and modified by the initialization process). It is unnecessary to make special checks for <back pointer>s in this algorithm, because the initialization has altered the 'parent' node of each <back pointer> by replacing one of its <subnet>s with a pointer to the subnet indicated by <back pointer>.

The result of this process is either a list of response items found at a <response node>, or failure, indicated by following a branch to a null <back pointer>. In the case of failure, the next D-net from the list is tried. If all nets fail, the program is forced to give up; it cannot produce a realization for the stimulus. If a response list is found, the



FLGFLD = contents of the FLAG field of the current predicate-flag pair

FIGURE 6-1

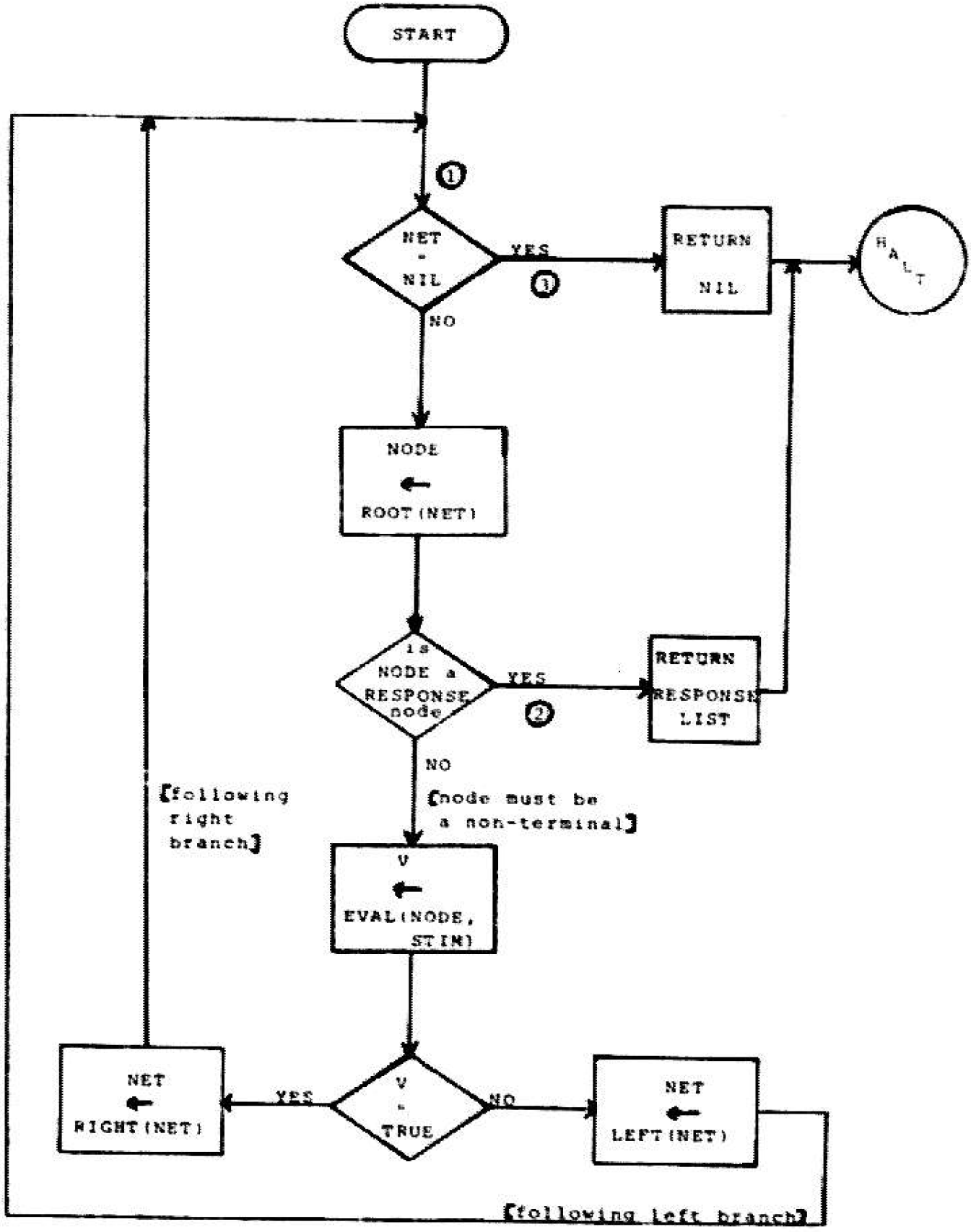

START
①
NET
←
NIL
YES
③
RETURN
NIL
HALT
NO
NODE
←
ROOT(NET)
is
NODE a
RESPONSE
node
YES
②
RETURN
RESPONSE
LIST
NO
{node must be
a non-terminal}
{following
right
branch}
V
←
EVAL(NODE,
ST IN)
NET
←
RIGHT(NET)
YES
V
←
TRUE
NO
NET
←
LEFT(NET)
{following left branch}

FIGURE 6-2

process of syntax net construction can take place.

6.3 Syntax Net Construction -- Sentence Production

A syntax net (SN) consists of NODEs, directed labeled ARCs, and TERMINAL UNITs. Each ARC has a NODE at its source and a syntax relation (section 5.5) as its label. An ARC may have either a NODE or a TERMINAL UNIT at is goal. The syntax nets are built as LISP P-list structures. The NODEs and TERMINAL UNITs are atoms. An ARC labeled "L" from X to Y is created by adding to the P-list of X the pair (L Y).

The syntax relations are of two classes, NODE→NODE and NODE→TERMINAL UNIT.

NODE → NODE relations:

ACTSBJ	GSBJ	INF3
OBJ	FIRS	S
OBJ2	SECS	S2
POBJ	POSS	INST2
PP1	P_ADJ	SPRG
LOC	INF	PRSNT
IOBJ	INF2	

NODE → TERMINAL relations:

VOICE	DET
FORM	QUANT
MODAL	MAN
TENSE	ASP
MOOD	PART1
LEX	PART2
PREP	

An ARC with a NODE→NODE class label can only connect two NODEs of the SN. An arc with a NODE→TERMINAL class label can only lead from a NODE to a TERMINAL UNIT.

At all times during the generative process there is a node marked as the active node (<AN>), a syntax relation marked as the active syntax relation (<ASR>), a conceptual structure marked as the active conceptual structure (<ACS>), and a push-down stack (<PDS>) of triples: (frames, conceptual structure, SN-node).

Initially the SN consists of an isolated node TOPNODE. This node is markes as the <AN>. The initial <ASR> is S (Sentence) and the initial <ACS> is the conceptualization given to BABEL for realization. The <PDS> is initially empty. We can now give a concise explanation of the process by which the SN is constructed.

BASIC GENERATION ALGORITHM

(1) If the ACS is a conceptualization, a set of D-nets is selected and applied. This results either in failure, in which case the realization also fails, or the discovery of a list of concexicon entries. Assume for the present that this list contains only one element. (If it contains more than one, we take the first). This entry will have the form:

<LEXICAL POINTER>	<FRAMEWORK>	<SPECIAL ACTIONS>

If, on the other hand, the <ACS> is not a conceptualization, it will be a list with a PP as its first element -- e.g., (*HAND* PART (*JOHN*)). In this case, BABEL retrieves the ENGLISH-NAME (section 3.2.6) for this PP.

(2) A new ARC is added to the network, labeled with the <ASR> and leading to a new NODE created at this time. The newly created ARC has the <AN> as its source. The new NODE then becomes the <AN>.

(3) An ARC labeled LEX is added to the network. This arc leads from the <AN> to either the LEXICAL POINTER of the concexicon entry or the lexical unit retrieved from the ENGLISH-NAME relation, depending on the result of step (1).

For example, if the D-net application returned the concexicon entry ASK-TO (which has <LEXICAL POINTER> ASK) in response to a stimulus given to BABEL, steps (1)-(3) would produce:

```
TOPNODE:      S      G1
G1:           LEX    ASK
        <AN>  = G1
```

(4) If a concexicon entry was found in step (1), any <SPECIAL ACTIONS> (section 5.2) are now taken. <SPECIAL ACTIONS> can only affect the <ACS>; they have no direct effect on the syntax net.

(5) The modifying fields of the <ACS> (section 3.2.5) are now processed by language specific functions (section 5.4). These functions add only NODE-TERMINAL arcs to the network, all of which lead out from the <AN>. This process might expand the network depicted in step (3) into:

```
TOPNODE:      S      G1
G1:           LEX    ASK
              TENSE  PAST
              FORM   SIM
              VOICE  ACT
              MOOD   INDIC
```

If a concexicon entry was found in (1), the variable FRAMES is set to the <FRAMEWORK> of that entry. Otherwise, FRAMES is set to NIL.

(6) If FRAMES is NIL, there are no further ARCs to be connected to the <AN> and control passes to step (7). Otherwise, the first frame of FRAMES is picked off and the triple [REMAINING-FRAMES <ACS> <AN>] is put onto the <PDS>. The

frame to be processed will have the form:

‹SYNTAX RELATION›	‹FIELD SPECIFICATION›	‹SPECIAL REQUIREMENTS›

‹SYNTAX RELATION› is made the ‹ASR›. If the ‹FIELD SPECIFICATION› is omitted, the default field-spec for the given ‹SYNTAX RELATION› is filled in. The ‹SPECIAL REQUIREMENTS› (section 5.2) are disposed of next. Of major interest here is the preposition inserting requirement, (MAKPREP ‹PREPOSITION›). MAKPREP generates a new node and an arc labeled with ‹ASR› from the ‹AN› to the new node. An arc labeled PREP is generated from the new node to the ‹PREPOSITION›, the new node is made the ‹AN›, and the ‹ASR› becomes POBJ (Prepositional OBJect). Thus, for example:

BEFORE: ‹AN› = G75 ‹SR› = LOC ‹SPECIAL REQUIREMENTS› = (MAKPREP TO)

AFTER processing the special requirements the network contains:

G75: LOC GB3

GB3: PREP TO (GB3 is a newly generated node)

‹AN› = GB3 ‹ASR› = POBJ

Finally, the ‹FIELD SPECIFICATION› is applied to the ‹ACS›, the result becoming the new ‹ACS›. This may be a conceptualization -- e.g., if the field spec is MOBJECT -- or a structure headed by a PP -- e.g., if the field spec is ACTOR. In either case, the syntax net creation algorithm is entered recursively at step (1) to expand the net, rather than proceeding to step (7).

(7) If the <PDS> is empty, it indicates that all required processing has been done and production of the syntax net is complete. Otherwise, FRAMES, <ACS>, and <AN> are restored from the top triple of <PDS>, the triple is removed, and step (6) is re-entered.

One detail left out of the above algorithm is necessary to complete the description of network construction. As stated, the algorithm would produce only tree structures. This is because arcs added to the net always point to newly created nodes (except for those which point to TERMINAL UNITs). In actuality, there is a 'memory' which recognizes cases where a <ACS> reoccurs and generates an arc back to an existing node of the syntax net. For example, given the conceptual representation of

"The woman begged the tourist to give her some money"

BABEL will first find a concexicon entry, say BEG1, corresponding to this sense of "beg". The associated <FRAMEWORK> will include an ACTSBJ syntax relation with a <FIELD SPECIFICATION> which retrieves some conceptual node (C1) representing "the woman". The same memory node C1 will eventually be referenced in processing the <FRAMEWORK> of the concexicon entry GIVE1.[1] Rather than regenerate the syntax net corresponding to this conceptual node, a second arc to the syntax net node already generated for "the woman" is added. Pictorially:

after processing the ACTSBJ frame of BEG1:

TOPNODE:	S	G1	G2:	LEX	WOMAN
				DET	THE
G1:	ACTSBJ	G2			
	LEX	BEG			

after processing the OBJ2 frame of GIVE1:

TOPNODE:	S	G1	G2:	LEX	WOMAN
				DET	THE
G1:	ACTSBJ	G2			
	INF2	G3	G3:	LEX	GIVE
	LEX	BEG		OBJ2	G2

This "reuse" of portions of the syntax net is accounted for by the following modification of the algorithm described above. An association list, initially empty, of conceptual structure - syntax_net_node pairs is maintained. Each time step (1) is entered, a check is made to see if the <ACS> is associated with a node in this list. If so, the arc created in step (2) leads not to a new node, but to the node associated with the <ACS>, and control passes immediately to step (7), bypassing all processing of the <ACS>. The association list is built up by adding the pair (<ACS> <AN>) to it each time step (3) of the algorithm is entered.

The linearization of the syntax net is accomplished by the AFSTN grammar. The details of BABEL's particular grammar were given in section 5.5. The grammar is entered at grammar_node S and syntax net node G, where G is the node related by syntax relation S to TOPNODE. The algorithm which controls flow through the grammar and syntax net is that described in connection with Simmons' work (section 2.4). The reader is referred to <34> for a more detailed specification of this algorithm.

Appendix 1 shows an example of BABEL realizing a stimulus conceptualization with output from the program augmented to depict the syntax net construction process in detail.

6.4 Paraphrase Production

The algorithm of the preceding section demonstrates how a conceptualization can be used to produce a syntax net which in turn can be used to produce an English sentence. Since each step of the process is deterministic, some additional mechanism is needed to produce paraphrases, or multiple realizations, from a single conceptualization.

One way to do this would be to define meaning-preserving transformations on the syntax nets -- changing VOICE from ACTive to PASsive would yield a different surface string. But such syntactic paraphrasing is clearly not the source of the paraphrases produced by BABEL. In fact, we have not incorporated any form of syntactic transformational component. Rather, BABEL's paraphrases are obtained by allowing the D-net application algorithm to find more than one response. Distinct syntax nets are then produced for each response.

It was pointed out earlier that there may be more than one D-net applicable to a given stimulus. By allowing all the nets applicable to a given stimulus to be tried, rather than stopping as soon as one produces a response, multiple responses can be found. Since the nets are organized to group 'related'

meanings into a single net, however, it is often the case that more than one appropriate response exists within a single net. There are two ways in which such responses may be found.

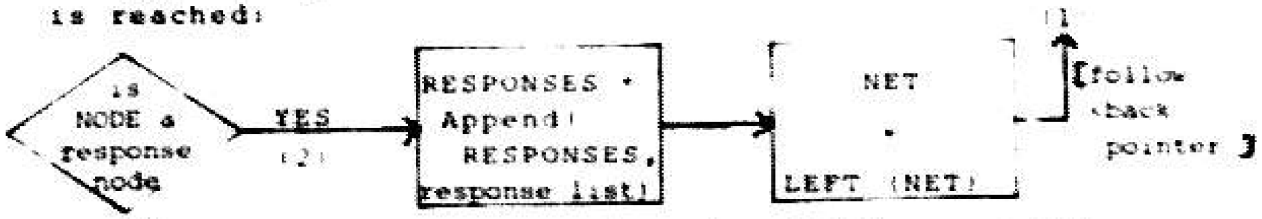
First, a <response node> consists of a list of concexicon entries. A stimulus which finds the response RETURN1 may simultaneously find GIVEB ("give back"), because both are part of the same <response node>. This handles cases of what we term 'conceptual synonymy'. Such cases do not explain a great deal of paraphrase, however, and become rarer as conceptual representations are refined to represent meanings more precisely. The most important source of paraphrase generation is the <back pointer> field associated with each <response node>. (We represented these fields by (* <INTEGER>) at the <response node>s of the nets in Chapter 5.) These <back pointer>s were ignored in the basic generation algorithm stated above, since that algorithm always halted when a <response node> was reached. It is possible, however, to save the concexicon entry list at a <response node> and follow the <back pointer> associated with it. This process can continue until a null <back pointer> is reached. In this way, a list of conceptually distinct responses (e.g., THRTN, WARN1, TELL1), can be found within a single D-net for a single stimulus. Intuitively, following the <back pointer> from a <response node> corresponds to 'ignoring' some feature(s) of the stimulus which English provides a special way of expressing

and finding a more general way to express that information.

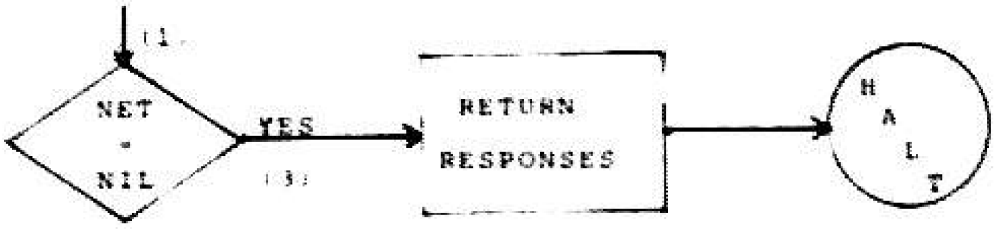
More precisely, we must modify the flow chart of Figure 6-1 as follows:

(A) Initialize a variable RESPONSES to NIL

(B) At (2), modify the action taken when a ‹response node› is reached:



(C) At (3), modify the action taken when NET becomes NIL (no more paths can be followed in the net):



RESPONSES will now be a <u>list</u> of concexicon entries. Some of these may be conceptually synonymous, others conceptually distinct. But each of them in turn may be treated as though it were the result of step (1) of the basic generation algorithm, resulting in a distinct syntax net for each.

The control structure required to produce all paraphrases of a given conceptualization is that of an AND-OR tree search. Roughly, this is because each application of a set of D-nets yields a set of responses R_i. Since <u>each</u> response can be used to generate a different solution (paraphrase), the set of responses yields a <u>disjunctive</u> node in the solution space. Each response R_i has a set of frames $F_{i,j}$ in its concexicon entry, <u>all</u> of which

must be processed in order to generate a single solution. The $F_{i,j}$ therefore result in a conjunctive node in the solution space. By finding all solution paths in this AND-OR tree, paraphrase is accomplished.

CHAPTER 7

GENERATION OF EVENT NOMINALS

It is apparent that the implemented conceptual generator, BABEL, leaves unexplained many quite significant aspects of the problem commonly referred to as "natural language generation". Even within the narrower area of 'output vocabulary' the program deals with only a subset, albeit a significant one, of English word categories. In English grammar books it is common to find such categorizations as noun, verb, adjective, conjunction, preposition, and article. Insofar as these notions are useful in dealing with English syntax, an English generator can make use of them. But since we are dealing with conceptually-based generation, such categorizations are far too crude as a means of dividing up the task. That is, just because two words fall under a single abstract class like 'noun' or 'conjunction', it is not true that they may be treated in a uniform manner. The conceptual content expressed by a word is as important as its grammatical class. In this chapter we outline how BABEL could be extended to generate sentences containing a class of nouns which describe <u>events</u>.

Our model shows how a sentence can be constructed by first choosing a verb which conveys some of the information content to be expressed and then using information associated with that verb to guide sentence formation. The emphasis has been on verb selection; nouns have been restricted to those which name either people or classes of physical objects. For these we have used a simple associative mechanism (the ENGLISH-NAME relation, section 3.2.6) to retrieve the nouns.

For example, we have assumed that there exists a memory node SCISSORS which represents an abstract 'concept'. Associated with this node are:

a) the ENGLISH-NAME relation

b) a physical description of the 'abstract' scissors

c) a functional description of the 'abstract' scissors

d) information about "scissors" -- e.g., found in desk drawers, purchased at department stores, cost about $2.00 etc.

e) 'idiosyncratic' beliefs about scissors -- e.g., "scissors" were given to man by the devil and all of them should be destroyed.

f) tokens (instances) of SCISSORS -- these are the nodes which represent particular physical objects (like the scissors on my desk) and which appear in the kinds of conceptualizations we have dealt with.

Now there is nothing novel here; computer models have commonly dealt with such archetype nodes and instance nodes, and natural language programs (like Winograd's) have always associated English words with these archetype nodes. This is a perfectly natural way to deal with common nouns which name classes of physical objects and proper nouns which name tokens of such classes. But there are many common English nouns -- e.g., "collision", "murder", "destruction", "death" -- which do not name physical objects. Such words express meanings which C.D. represents with conceptualizations and which BABEL now expresses only with verbs. More specifically, we posit that for any of the verb-noun pairs below, the verb and the noun must be generated from identical underlying conceptual representations:

(collide collision) (murder murder) (destroy destruction)
(die death)

Given that these nouns have conceptual representations identical to those of the verbs, it is a simple matter to extend BABEL to find these nouns from their meaning structures. It is necessary only for the noun to be associated with the same discrimination net terminal node as is the corresponding verb.

In order to use the nouns in English sentences, though, it is necessary to specify their syntactic environments. By the syntactic environment of a word W we mean the syntax

net relations involving a node N which has an arc labeled LEX pointing to the word W. There are two aspects to this environment:

i) What arc types can lead out from node N? We shall refer to these arc types as syntactic predictions of N. If W is the verb "murder", for example, there exist syntactic predictions for TENSE, ACTSBJ, and OBJECT arcs, among others. The noun "murder", however, makes different predictions, as we shall see.

ii) What arc types can lead into node N? We shall refer to these arc types as those with which N has syntactic compatibility. If W is the verb "die", it is compatible with the relation INF but not with the relation OBJ2. The notion of compatibility will be more precisely defined shortly.

"Murder" (noun) and "murder" (verb), "death" and "die" may look the same in a meaning representation, but they are not mutually substitutable in English sentences or English syntax nets. For this reason it is not possible to have a single concexicon entry DIE1 with its LEXICAL POINTER field containing pointers to both "die" and "death".

BABEL already deals with the question of syntactic predictions for verbs. The program contains two sources for such predictions. Language Specific functions attach relations like TENSE and MOOD to verbal nodes. The FRAMEWORK field of a concexicon entry makes predictions specific to the particular verb to which that entry corresponds. The concexicon entry COLL1V for the verb "collide" would have a FRAMEWORK which specified an ACTSBJ

and a PP1 requiring the preposition "with". From this FRAMEWORK sentences of the form "X collided with Y" would be generated.

The syntactic prediction problem for nouns could be handled analogously -- that is, by having concexicon entries for nouns as well as for verbs. For "collision" we want to generate a noun phrase of the form "collision between X and Y". We invent a new syntax relation NPP to handle prepositional phrases placed after their governing noun phrases, as in "dog in the back yard". We then create a concexicon entry COLLIN, with "collision" in its LEXICAL POINTER field, and a FRAMEWORK which would generate a piece of syntax net like:

```
N57:    LEX     COLLISION          N59:    LEX     AND
        NPP     N58                        NPA     N60
                                           NPB     N61
        PREP    BETWEEN
        POBJ    N59                N60:    LEX     CHEVY

                                   N61:    LEX     FORD
```

Pointers to COLLIV and COLLIN would exist in the response list of the same terminal node of some discrimination net.

If the AFSTN grammar is modified to handle the NPP relation and 'conjunctive' noun phrases, the above piece of net will produce "the collision between the Ford

and the Chevy" (we have omitted determiners in the syntax net above). An alternative FRAMEWORK could be used to produce "the collision of the Ford with the Chevy". Similarly, "death" would have two FRAMEWORKS, one of which would produce "X's death", and the other "death of X". "Anxiety", on the other hand, would have only a FRAMEWORK for producing "X's anxiety" (at least if our model was based on the author's dialect).

The use of Language Specific functions to make syntactic predictions can be extended to nouns as well. Just as a TENSE is produced whenever a verbal node is put into the syntax net, so a (possibly null) DETerminer should be chosen whenever a nominal node is created. The program currently (see section 5.4.1) chooses determiners not by prediction, but from the existence of a REF link in the conceptual structure. This is theoretically unsound, since it leaves what is really linguistic information in the conceptual representation. What is needed is a sophisticated process, activated upon creation of a nominal node, which decides on an appropriate determiner.

In discussing syntactic prediction, we have made no distinction between the notions of 'noun' and 'verb'. We have, rather, been treating them symmetrically:

(i) both make syntactic predictions
(ii) both may be generated through the D-net generation process or by association with a memory concept.

There comes a point, however, where the generator can no longer ignore the grammatical category of the word it chooses. Once generation begins and choices are being made, these choices must not only direct the generation process through the conceptual network, but must also place syntactic constraints on later choices. In particular, whenever a choice must be made between a conceptually synonymous noun-verb pair, the one chosen must be syntactically compatible with the prediction which led to that choice. This can be illustrated with an example.

For simplicity, let X1 be the conceptual representation of "the street was wet" and let X2 be the conceptual representation of "the Ford collided with the Chevy". Suppose BABEL were given the task of realizing:

(C7-1)

Consider the following realizations of (C7-1):

a) The Ford collided with the Chevy because the street was wet.

b) The Ford collided with the Chevy because of the wet street.

c) The wet street resulted in the collision between the Ford and the Chevy.

d) The collision between the Ford and the Chevy resulted from the street's being wet.

e) The collision between the Ford and the Chevy was due to the wet street.

f) The wet street caused the collision between the Ford and the Chevy.

The point of these examples is that the underlined relations, some of them verbs and others conjunctions but all possibly found as lexical pointers of entries used in realizing a conceptual causal relation, set up syntactic constraints on the manner of realization of X2. (They do the same for X1, of course).

Now suppose that our model had only one concexicon entry in its vocabulary suitable for expressing the causal relationship in (C7-1). Assume this was the entry RESF1 corresponding to the use of "result from" in (d) above. The entry would have

(i) a lexical pointer to the verb "result"

(ii) a FRAMEWORK which included a FRAME with ACTSBJ as its Syntax Relation and (RESULT) -- which in this case specifies X2 -- as the Field Specification.

In realizing (C7-1), BABEL would first find this entry RESF1. In processing the ACTSBJ frame, BABEL should see that X2 could be expressed with either of the concexicon entries COLLIV or COLLIN mentioned earlier. But only the latter (the "collision" entry) is compatible with the decision already made to realize (C7-1) with RESF1. How is the program to realize this?

We can solve the problem by defining "classes" of syntax relations. Let us define the following two classes:

V-PREDICTIVE		N-PREDICTIVE
INF	FIRS	SUBJ
INF2	SECS	OBJ
S	GSBJ	OBJ2
S2	PRSNT	POBJ
INSTJ	SPRG	ACTSBJ

As can be seen from figure 5-9, the V-predictive relations are those which are also states leading directly to the state S in the AFSTN grammar. N-predictive relations bear this same relation to the NP state of the grammar. That is, the "embedded sentence" relations are S-predictive, while the "noun phrase" relations are N-predictive.

Next we mark every concexicon entry as S-compatible or N-compatible, according to whether its lexical pointer corresponds to a verb or a noun. When X2 is to be expressed in our example -- that is (in the terminology of chapter 6) when X2 becomes the ACTIVE CONCEPTUAL STRUCTURE (<ACS>) -- there is available a piece of information ready-made to help make the choice between COLL1V and COLL1N. This is the ACTIVE SYNTAX RELATION (<ASR>). The value of <ASR> would be ACTSBJ, which is N-predictive. Since, of the options open to us, only COLL1N is N-compatible, it would be chosen and the syntax net for the noun-phrase "the collision between the Ford and the Chevy" produced from X2. Finally, sentence (d) would be generated by the AFSTN grammar.

If, on the other hand, BABEL had only the entry BECAUS1 to express this causal relation, the <ASR> would be FIRS when X2 became the <ACS>. Since FIRS is S-predictive, BABEL would choose COLLIV which is S-compatible and generate sentence (a).

To handle syntactic compatibility, we have proposed three additions to BABEL:

(i) classifying the syntax relations as S-predictive or N-predictive

(ii) marking the concexicon entries as S-compatible or N-compatible

(iii) choosing a concexicon entry only if it is compatible with the <ASR>.

These modifications will enable the program to choose a form which is compatible with syntactic predictions of previous choices. Since no syntax relation is simultaneously S and N-predictive, the problem of choosing between noun and verb forms is handled simultaneously, provided the choice can be postponed until after a syntactic prediction for an S or V compatible concexicon entry has been made.

The validity of this provision depends on an assumption which has been made implicitly throughout BABEL. This assumption was that, when a response list in a discrimination net had multiple entries, it was sufficient to make a random choice among them. That is, the program makes no usage distinctions between conceptual synonyms.

Unfortunately, this assumption (actually a practice rather than an assumption) may limit the quality, or at least the naturalness, of the language produced by the program. It has been our feeling that whenever random selections are made in generation, it is probable that a real problem is being bypassed. The discussion of noun-verb choice points out the problem in this case. Our proposed solution to noun-verb choice was not itself random selection, but in certain cases our solution makes this decision the direct result of earlier random selection.

Returning to our example (C7-1), consider the case when a model is allowed to have in its vocabulary several conceptually synonymous ways of expressing the causal relation, e.g. "because", "because of", "result in", "result from", "due to", . . . These concexicon entries cannot be distinguished on the basis of syntactic compatibility with previous choices, because all are S-compatible. We can, of course, make a random choice as is done currently. Once this choice is made, a prediction is set up which determines the selection of "collide" or "collision" in realizing X2. Should not the fact that "result in" leads to the use of "collision" rather than "collide" help determine whether "result in" is chosen?

The answer is "yes". This is only a claim that the effects of particular choices should be part of the criteria

used in making a choice. (The reader may well wonder whether there is any reason for the program to discover the choice of causal relations prior to considering the noun-verb alternative. This issue, which has implications far beyond noun-verb selection, is considered in the final chapter. BABEL, as developed so far in this thesis, produces such situations in the course of generation. Let us see how they might be dealt with).

Now given two conceptually synonymous responses R_1 and R_2 such that R_1 predicts a nominal realization N for some structure C while R_2 predicts a verbal realization V for C, how can the program use this information in an intelligent fashion to decide between R_1 and R_2?

The key must be in the different effects of N and V. We are now considering the effects "once removed" of choosing R_1 as opposed to R_2. Great care must be taken here; like making a move in a chess game, the program cannot look at all effects of a decision indefinitely far into the future. No proposal for a general cut-off heuristic will be given here; we do not know what all the effects of choosing a noun vs. a verb, or of choosing between conceptually synonymous verbs, are. It can be seen that these effects extend even beyond sentence boundaries, however. Consider:

(A) "The news service reported the death of another Naples' citizen in the cholera epidemic. It was the third one this week."

(B) "The news service reported that another Naples' citizen died in the cholera epidemic. It was the third one this week."

(A) is quite acceptable (in this author's dialect), while the second sentence of (B) would not be produced as a follow-up to the first, but rather something like "It was the third such death this week." The difference seems to be the existence of the noun "death" in the first sentence of (A) to serve as a referent when the second sentence is encountered. Since RABEL does not presently deal with connected discourse, it is unlikely that any minor additions to the program will enable it to take such consequences into account.

There are, however, some intra-sentential effects of noun-vert choice which could be considered. The most obvious of these is "information prediction". Concepticon entries not only make syntactic predictions (through the Syntax Relations in their FRAMEWORKs) but make information predictions as well. This is done through the FIELD-SPECIFICATIONs of the FRAMEWORKs, which indicate which portions of the conceptual stimulus will be realized in the syntax net and thus the surface structure.

It happens that nouns and verbs which are conceptually synonymous may nevertheless make different information

predictions. "Collide" requires some mention of the objects involved -- "The Ford collided with the Chevy", "two cars collided" -- but "collision" can be used to describe an event without mention of these objects:

> "I saw a collision at Grant and Broadway."

The reason for leaving out information may be that the information is unknown or irrelevant, or it may be that context makes clear the event being referred to, as in

> "The guerillas captured the city and put the mayor to death. While world opinion was sympathethic to their cause, the assassination was a great mistake."

There is no way to refer to the event with a pronoun in this context, so either "assassination" or "assassinate" must be used. Using the verb, though, would require repeating (at least) the information about the 'victim':

> " . . . it was a great mistake (for the guerillas) to assassinate the mayor."

It is sometimes the case that information which is easy to express in conjunction with a verb form is difficult to express if the nominal form is used:

> "The history book describes how Archimedes destroyed the enemy fleet by concentrating the sun's rays on their ships."

is fine, but if "destruction" is used:

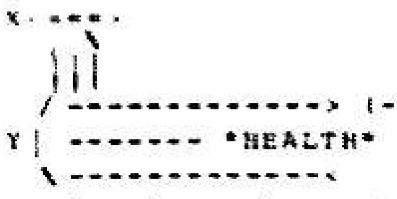
> "The history book describes the destruction of the enemy fleet by Archimedes by concentrating the sun's rays on their ships."

is difficult to complete in a natural sounding manner.

For simplicity, suppose we imagine giving BABEL a different concexicon entry for each of the possible information prediction combinations which can occur with a verb like "assassinate" or a noun like "destruction". (In practice we would want to deal with the notion of required and optional information, as does Fillmore <9>. For this discussion the distinction between (i) a single entry with optional elements, and (ii) separate entries for different combinations, does not matter). These concexicon entries are formal objects, not pieces of computer code, and can be manipulated by the program. In particular, the generator could compare the information predictions of two candidate entries and determine, in collaboration with the memory, which made 'preferable' predictions.

For example, let the conceptual stimulus:

(C7-2)



be the one on which the generator is working (embedded, let us say, as the *OBJECT of an MTRANS). For the moment let the remainder of the <ANTECEDENT> remain unspecified. Suppose that, by checking some conditions used to distinguish "murder", "assassinate", and "kill", BABEL reached a response node with pointers to concexicon entries MURDERIV (the verb

"murder") and MURDERIN (the noun "murder"). An entry for the verbal form which predicted realization of the 'murderer', the 'victim', and the 'method of murder' would look like:

(F7-1)

SYN. REL.	FIELD SPEC.	SPEC.ACT.
ACTSBJ	(CON ACTOR)	NIL
OBJ	(<= ACTOR)	NIL
INST2	(CON)	NIL

A FRAMEWORK for MURDERIN, predicting mention of the 'victim' only, would appear as:

(F7-2)

SYN.REL.	FIELD SPEC.	SPEC. ACT.
NPP	(<= ACTOR)	MAKPREP OF

Information predictions can be compared by comparing the sets of FIELD SPECIFICATIONs. In this case, it is easy to see that the predictions of F7-2 are a proper subset of those of F7-1. In particular, F7-1 predicts realization of the (CON ACTOR) (="murderer") and (CON) (="method"). Now it may be that the memory model would have information that these things were known from context, or unknown altogether (e.g., <ANTECEDENT> = *ONE*<=>*DO*). In either case there would be no desire to express them and F7-2 could be selected, generating a sentence like

"The newspaper reported the murder of a foreign diplomat."

On the other hand, both might be important pieces of information to communicate, in which case FT-1 could be chosen and the surface sentence produced might be

"The newspaper reported that a local resident murdered a foreign diplomat by placing a bomb in his car."

In general there is no guarantee that a concexicon entry will exist which predicts all relevant and no superfluous information. In such cases an evaluation has to be made (how important is it to express, or omit, certain information?) Thus the program may have no alternative but to generate either

"The newspaper reported that <u>someone</u> murdered a foreign diplomat by planting a bomb in his car."

or

"The newspaper reported the murder of a foreign diplomat."

even though its <u>preference</u> might have been to mention only the "victim" and the "method" leaving the "murderer" unmentioned.

In summary, several proposals have now been made for dealing with the problem of utilizing 'event' nominals. A notion of syntactic prediction for nouns was introduced, analogous to that which already exists for verbs. Concexicon entries for nouns, which guide the encoding of conceptual information into a noun phrase description of a

conceptualization, take care of such predictions.

The next problem faced is that of selecting a noun or a verb, when either is capable of expressing a given meaning. Here the notion of syntactic compatibility is introduced, to be implemented by creating "classes" of syntax relations and marking concexicon entries as compatible with relations of a given class. A requirement is made that an entry be selected only if it is syntactically compatible with the relation it is to participate in.

These mechanisms guarantee that the nominal and verbal forms will be used grammatically in the sentences generated. This does not solve the selection problem in all cases, however. Whenever a choice occurs between entries which are of the same compatibility class it may be necessary to look at the effects of the individual choices. "Learn of" and "learn that" both fall into the S-compatibility class, but the former predicts a noun phrase to express information which the latter expresses with a clause. Provided both predictions are fulfillable (how often this occurs is a function of the vocabulary and syntactic knowledge of a particular model) it was proposed that the selection process consider information predictions of the noun-verb alternatives. This could be done by allowing the program to compare the alternative concexicon entries and determine which most effectively expresses the information

which the memory model desires to express.

Beyond the notion of syntactic and information considerations are other, still less understood, effects of noun-verb choice. For one thing, there is the notion of 'focus';

"The car collided with the truck"

and

"The collision between the car and the truck"

both impart the same information, but the former includes a focus on the "car" which the latter does not. The notion of "topic of conversation" may also be a factor. In a discussion of the "effect of losing a war on a society" is it preferable, in describing a particular instance of a nation losing a war, to use the noun "defeat" rather than the verb "defeat"? Such psychological and stylistic aspects of word choice remain unaccounted for by the modifications proposed in this chapter.

From a program structure standpoint, the entire problem of word choice in the generator could be dealt with by allowing each response node of a discrimination net to have an arbitrary program associated with it. This program could perform whatever actions were needed to choose one of the entries in that node's response list.

Whether such a non-restrictive, and inelegant, solution can be entirely avoided remains an open question.

We have shown in this chapter that there are considerations common to many word choice problems. By allowing BABEL to make use of its own linguistic - conceptual knowledge as data rather than as program, the generator could effectively take these considerations into account in a manner common to a large class of response sets.

CHAPTER 8

WHERE FROM, WHERE AT, WHERE TO

Artificial Intelligence, like any other academic discipline, has its own set of well recognized research areas. Some of these are in the realm of practical problems like "designing an intelligent question answering system" or "developing automatic scene analysis routines". Others are the more esoteric problems which seem to be the current limiting factors in the field -- questions like "how to use context to direct and limit search." Now it is certainly well recognized that there exists some problem which could appropriately be termed "language generation" -- one which is a subpart of the practical problems of machine translation, question answering, etc. But it is by no means accepted that that problem is "producing English sentences from conceptual representations" which is what this thesis has been concerned with.

8.1 The WHY and WHEN of Conceptual Representation

What is most likely to be disputed about our definition of the problem of generation is the nature of the representations used as the source of generation -- that is, whether conceptual representations (whether or not similar

to Conceptual Dependency) are indeed the best ones for a computer to use for language processing tasks.

It would be very nice if we could give a formal definition of 'conceptual representation' and proceed to prove that, for some class of tasks, such representations were optimal. We can't. What we can, and will, do, however, is to compare such representations with the alternatives currently proposed, pointing out the advantages and disadvantages of each. This will, at the very least, give the reader an insight into the motivations which led to the model of generation presented in the preceding chapters.

One much debated issue in the representational question is the notion of procedural vs. static representation. A conceptual system could use either, none of the arguments which we make for conceptual representation will depend on which alternative is chosen. Another point of debate concerns the nature of the basic elements which make up the representations, and, in particular, how they relate to language. This is generally referred to as the question of "depth" of representation, and it is here that conceptual representations differ distinctly from others in ways which have important implications for language processing.

There are three basic questions which we can ask about proposed representations:

(i) Can they be readily derived from natural language strings?
(the ANALYSIS problem)

(ii) Can natural language strings be produced from them.
(the GENERATION problem)

(iii) Can they be manipulated in the ways necessary to perform useful tasks/
(the MEMORY problem)

We can compare proposed representations with respect to their ability to cope with these three problems. Not unsurprisingly, we are seldom if ever presented with two proposals of which one is superior in all three areas. We must look at the tradeoffs involved and ask which representation optimizes 'overall' performance for a given task.

It is not difficult to partition representations which have been proposed into three categories:

(a) word based (WB) -- In such a representation, the information element is the (English) word.

(b) sense based (SB) -- These representations consist of networks of relationships between senses of the words of a particular language.

(c) conceptually based (CB) -- Such a representation consists of networks of relationships between abstract concepts. There may be a direct correspondence between some concepts and word senses of a given language in some cases, but there is more generally a complex relationship of concepts corresponding to a particular word sense.

Now it would be possible to define a 'continuum' of representations for which such categorizations would be so imprecise as to be absurd. But actual proposals have produced no such continuum. There have been distinctly

WB systems (Klein <16>), distinctly SB systems (Simmons <8>, Sandewall <29>), and distinctly CB systems (Rumelhart et al. <26>, Schank et al, <3>), but little if anything which falls into the potential gray areas between categories. For this reason we shall discuss the relative merits of these classes of representation rather than the merits of particular proposals.

Word based representations are no longer proposed for work in computational linguistics. Perhaps the simplest possible such representation is the natural language sentence itself, in which the only units are words and the only relation a transitive "left of" which orders the words. More sophisticated WB systems would have trees or networks of dependency relations between words. The most fundamental problem with such representations is their retention of linguistic ambiguity. This simplifies the problems of analysis and generation, but leaves information encoded in a form requiring a great deal of processing before it can be used for most purposes. Having parsed:

(SB-1) "The newsboy delivered a paper to the journalism department"

(SB-2) "Jack Anderson delivered a paper to the journalism department"

a WB representation like Klein's no more represents the natural interpretation of these sentences than one in which the newsboy is a sought-after lecturer and Mr. Anderson's

enemies in Washington have succeeded beyond their wildest dreams. It is universally recognized that disambiguation of such words as 'deliver' and 'paper' above, at least to some word sense level, is necessary for tasks like Q-A and MT.

WB systems have the flaw that disambiguation must be redone every time the ambiguously represented information is needed. We can contrast this with a CB system which pays for its retention of an unambiguous representation by necessitating the re-encoding of the information into natural language whenever it is to be expressed.

A Sense Based system allows arbitrary word sense units in its representations. These units are unambiguous and thus permit unambiguous representation of sentences like (S8-1, 2) above. This of course complicates the process of analysis by forcing it to solve the disambiguation problem.

The generation problem, however, is not necessarily complicated by the transition from word to sense. One of the most salient features of a SB system, from a viewpoint of language generation, is that the mapping from a sense to a word is independent of the context in which the sense occurs. The word to express the sense can thus be found quite simply (this is generally done by means of a direct link(s) from the sense to one (or more) words.)

The major differences between SB and CB systems are those which concern the memory problem. One of the most important such differences is in paraphrase capacity. By paraphrase we don't mean the literal generation of paraphrase sentences, as was demonstrated by examples in Chapter 1. Rather, we mean the ability of a system to recognize different ways of saying something as conveying the same information. This is accepted as a necessary capability of an intelligent system; we shall give some examples of how this capacity is used (and what goes wrong if it isn't) momentarily. But first let us see how SB and CB approaches to paraphrase differ.

Suppose we classify paraphrases of individual sentences into three categories:

A) SYNONYMY-based -- a trivial form of paraphrase can be achieved by having a dictionary of word equivalence classes, e.g., the mutual substitutability of 'mare' and 'female horse', or, more properly, one sense of 'mare' and a semantic relationship between particular senses of the words 'female' and 'horse'.

B) SYNTAX-based -- more interesting paraphrases can be obtained by the application of syntactic transformations to the constituent structure of a sentence. Paraphrase using passive rather than active voice, or relative clauses rather than adjectives, are possible.

In a SB system, these transformations are distinguished by the fact that they do not alter word sense choices.

C) COMPONENT-based -- many of the paraphrases most natural for humans are of neither of the above mentioned types. Consider:

John gave Mary a book ===> Mary received a book from John

John wants to eat the ice cream ===>
John believes that he would enjoy eating the ice cream.

These are not paraphrases of type (B), since they are intimately related to the meanings of individual words rather than to the structure of the particular sentences. Neither are they paraphrases of type (A), since they do not involve word, or word sense, synonymy. Rather, they seem to involve the knowledge of word component identity.

Component based paraphrases can be performed from a SB representation. Simmons, for example, has adopted the reasonable solution of implicational rules. These rules map one word sense onto another and map the case relations of one onto case relations of the second. The BUY-SELL rule was illustrated in section 2.4. As further examples consider

P2:

```
GIVE                          RECEIVE
    AGENT (V1)                    AGENT (V3)
    OBJ   (V2)      ====>         OBJ   (V2)
    GOAL  (V3)                    SOURCE(V1)
```

P3:

```
LIKE                          PLEASE
    DAT   (V1)                    DAT   (V1)
    OBJ   (V2)      ====>         OBJ   (V2)
```

In this model, there is no distinction between paraphrase

rules and inference rules like:

```
IS-1: GIVE                                        HAVE
          AGENT (V1)          ==>                   DAT (V3)
          OBJ   (V2)                                OBJ (V2)
          GOAL  (V3)
```

The transformation of a structure A into a structure B according to one of the rules is a deduction; it is also a paraphrase in those cases where B could be transformed into A.

There exists a reasonsable argument for not distinguishing paraphrases from inferences in a SB system. If as part of a Q-A task the system is told:

(1) "Mary stole a boat".
(2) "John then bought the boat from Mary".

and is later asked:

(1') "Who sold John the boat?"
(2') "Is John the legal owner of the boat?"

it can answer (1') by application of the BUY-SELL paraphrase rule P1, but can answer (2') only through inference rules which do not produce paraphrases. But since the paraphrase rules have the same form as the inference rules, and since there is no way to know from the form or content of a question whether it can be answered by a paraphrase rule, there seems to be no reason to distinguish paraphrase from general inference rules (except possibly for performing the somewhat artificial task of sentence paraphrase).

To implement this method of paraphrase for a set of N mutually paraphrasable word senses requires $O(N^2)$ such rules. One could of course chain the rules - there would be no rule like P3 but a rule associating LIKE with ENJOY and another associating ENJOY with PLEASE - and only N rules would be required, but at a cost of performing, on the average, the application of N/2 rules to obtain a given paraphrase.

There is an obvious argument here in favor of a system with a single underlying representation for the N words, which is what a conceptual system provides. It is the same argument which has been used in favor of an interlingual representation for MT. The argument is that, for MT via an interlingua, one needs only N programs (rules) to translate N languages into the interlingua, and N additional programs to translate from the interlingua into the N languages. With 2N programs it is then possible to perform translation between an arbitrary pair of the N languages. Without the interlingua, the pair L, M of languages requires two programs: A1: L → M, and A2: M → L. The total requirement is then

$$2*C^{N}_{2} = N*(N-1) = O(N^2)$$

programs.

Besides theoretically requiring more mechanism, at least as measured by number of rules, for paraphrasing, the NS system poses additional memory organization and processing problems which do not exist in the conceptual model. The most obvious of these is the problem of multiple copies of information in memory. Imagine a psychiatric interviewing program, for instance, which might be told:

"My father likes to bet..."
"He enjoys gambling so much that..."

If the model stores both LIKE(FATHER, BET) and ENJOY (FATHER,GAMBLE) it will be, first of all, inefficient in memory storage, and, more importantly, lacking in reasonable memory organization.

The former problem (multiple storage) could result in aberrant behaviour:

PATIENT "What did I tell you about my father, other than that he enjoys gambling?"

PSYCH. MODEL "That he likes to bet."

Such replies could constitute a danger to the physical well being of the machine harboring the model as well as the mental well being of the patient.

To avoid this problem in a NS system, however, would seem to require making all inferences (or at least all paraphrases, if they were distinguished from inferences) at some point. This seems bad enough only considering a paraphrase set such as

(LIKE, ENJOY, PLEASE, BE FOND OF, . . .)

But in the above example it is not just paraphrases of 'like' which must be considered; those of 'gamble' (and 'my father', for that matter) must be looked at as well. The problem is combinatorial and the number of word-sense paraphrases possible for a sentence with even simple embedding can be very large, as has been seen in the description of BABEL's application to this task.

The organization problem is difficult to analyze without concrete proposals about the structure of memory. The problem is how to integrate new information into an existing memory model. To store every new fact as an isolated entity is clearly absurd. When words have common components, however, these can be used by a conceptually based system to aid integration. A SR system can only find the relationship through inference. To take a concrete example, the two facts:

(1) The Greeks poisoned Socrates.
(2) Socrates swallowed hemlock.

are related representationally in our conceptual system, for example, not just by virtue of being facts about Socrates, but by virtue of being facts about Socrates INGESTing something (that 'something' being an unspecified poisonous substance in one case, 'hemlock' in the other). The relationship between 'poison' and 'swallow' can be discovered only through (probably two or more levels of)

inference in a SB system.

We thus gain two very important advantages in a CB system over a SB system:

> CB systems DO NOT NEED TO FIND LINGUISTIC PARAPHRASES FOR STORED INFORMATION. The existence of a single underlying representation for paraphrases obviates the necessity of finding these paraphrases in order to recognize information with different linguistic encodings.
>
> CB systems AID IN MEMORY INTEGRATION. The deeper level of analysis makes explicit relationships between information in cases where the same relationships could be uncovered only by deduction from a 'shallower' representation.

Semantic nets (such as Simmons') do not claim to be language independent. A simple example from the realm of MT will demonstrate why no such claim can be made for any SB system. Suppose it is desired to translate into German the two English sentences:

> "The boy ate the berries."
> "The bear ate the berries."

In particular, 'ate' is to be translated by the verbs 'essen' and 'fressen' in the respective examples. If the semantic nets for the two sentences both use a single word sense for 'ate' the translation is impossible, since the word sense potentially maps onto (at least) two different German lexical entries, and the decision cannot be made on context-free grounds.

If, on the other hand, it is claimed that the semantic representations of the sentences involve two senses of 'ate':

EAT1 -- human ingesting
EAT2 -- animal ingesting

then the translation can be done. Of course, in analysis the choice of senses for 'ate' becomes sensitive to semantic context, but this adds no complexity to analysis since semantic ambiguity must be handled anyway. But it would seem that the list of senses for 'eat' must be expanded to include:

EAT3 -- to ingest a brittle substance
EAT4 -- the way arachnids get nourishment

etc.

since there may be languages which have words to distinguish these types of 'eating'.

It is obviously pointless to list all potentially distinguishable senses of 'eat'. There are two solutions to the problem. The most natural is to relax the context free mapping assumption for generation. This of course means that a SB system will be faced with the same basic generative problem as a conceptually based system.

The other solution is to hypothesize the existence of a (context sensitive) transformation from semantic networks of English to those of German. No one has yet developed such an algorithm, however, and there does not seem to be any reason to expect that such a solution could be simpler than the mechanisms needed for generation in CB model.

If generation for a SB system must be context sensitive to solve the MT problem, then it is reasonable to ask where the SB system differs from the conceptual system in this task domain. It is only natural to look at analysis and assume that, as is often claimed, it is simpler to perform an analysis to the word sense level than it is to the conceptual level. To make this discussion concrete consider the two examples:

A) Thousands of public employees are being overpaid because the city council gave in to labor demands.

B) Thousands of public employees are being overpaid because the city council purchased an IBM 360 for for the payroll department.

(Further research is needed on why 'city council' examples are inherently useful for MT related problems.)

It is reasonable to assume that many languages will require different means of expressing the two 'meanings' of 'overpaid'

OVERPAID1 -- paid more than the value of the work performed

OVERPAID2 -- paid more than specified by a contract

Clearly a great deal of well-directed deduction will be required of a system to make this distinction. If it is not made during analysis, then it must be done by the generator. (Here the standard assumption is being made that everything done after analysis in MT can be termed 'generation'.) This of course introduces a whole new

complexity into the concept of generation. Furthermore, it means that analysis will have bogged off the truly difficult portion of the task, at least in this example. But a better argument can be given against considering this a task for the generator. If these two examples were given to a Q-A system, which was then asked:

> "Are the employees legally obliged to repay the extra money?"

the answer depends in part on making the same distinctions about 'overpaid' as were required for the MT task. The distinction is no longer needed for a linguistic purpose, however. Rather than claim that the disambiguation should be made by the generator in an MT task and a memory model in a Q-A system, it is most consistent to have it made as a part of analysis in both cases. Such an analysis requires that the SB system have the same sort of sophisticated communication between its analysis algorithm and general deductive system as is required in a conceptual framework.

In this discussion of the relative merits of WB, SB, and CB representations, no claim of unconditional superiority for one form of representation was made. Given any of the three alternatives, problems could be designed for which that alternative was in fact preferable and could even give the appearance of 'intelligent' processing. We have stressed the underlying components of tasks which make conceptual representation the preferable alternative.

Insofar as we want computers to perform these tasks, we must either (i) look for entirely new representations, or (ii) construct algorithms for analysis, inference, and generation based on conceptual representations. It is this latter approach which was taken by BABEL.

In order to attack the problem of natural language generation, it was first necessary to provide a definition of this problem. This was done by considering a design for a general purpose computer system for communication in human language and selecting a requisite subprocess which seemed to naturally fit one's intuitive feel for "generation". The subtask chosen was that of mapping meanings in contexts into single sentences.

No attempt has been made to give a hard and fast test for what constitutes a meaning, but certain stringent conditions were placed on the 'objects' which served as representations of these meanings. In particular, it was required that these representations be free of the syntax of any natural language, and free of the words of any particular natural language. More precisely, the meaning representations were composed of relations between elementary units. There was no general mapping from single units (or relations) into groups of words, nor was there any indication within the meaning representation of which units and relations were to be combined into single linguistic entities (words or syntactic features).

The contexts in which meanings are realized are embodied in a memory model. The generator has complete access to the work knowledge, behaviour beliefs, and

inference capacity of this model. A single meaning may be realized with different surface strings in differing contexts.

We have tried to separate the notions of language specific information, of use solely in the task of realizing a meaning, and non-linguistic information, which may be used in language generation but is also useful for processes such as inference or memory organization.

Language specific information and processes must be fully specified as part of the task we have called "generation". It would thus be unacceptable to have "black box" functions of the form MAINVERB (conceptualization), which would return a verb to be used in realizing conceptualization, or PLURAL (word) to return the plural form of a word. Non-linguistic processes were treated in "black box" fashion, however. We did not require our model to specify how the predicate (PROPERTY TABLE) PLUR) would be evaluated, or how information about time relationships is stored in memory. BABEL does, of course, specify the points in the generative process at which these black boxes are accessed. In other words, while the generator cannot be independent of memory capabilities, there is a conscious effort in the system developed in this thesis to make the generator as independent as possible of details of memory organization and processing.

This leaves open the **status** of a third sort of information a language generator might conceivably utilize. This is information which is used only for language generation, but is not specific to any single natural language. In a sense the data structures of BABEL (discrimination nets, concexicon entries, etc.) and the routines which manipulate them are candidates for the status of "generative universals". Such was our intent in designing the system. But as yet our attempts to generate languages other than English have not been sufficiently extensive to make a truly convincing claim in this area.

The actual process of generation takes place in two phases. The first phase, which is the one emphasized throughout the thesis, consists of constructing a syntax net from a meaning stimulus. To accomplish this it is necessary to choose words (senses) to express meanings, and to relate these words syntactically. The greatest part of the word selection is handled by discrimination nets. A sort of "synthesis by analysis" procedure inspects the stimulus, detecting patterns and meanings which are significant for word selection in the target language. A great deal of conceptual and contextual information must be accessible to distinguish between candidate words, and even inference capabilities must be invoked at times.

There are two main sources of the information from which BABEL establishes syntactic relationships between words. Word senses are associated with syntactic predictions found in the concexicon files. These predictions indicate the syntactic role to be played by as yet unrealized conceptual information. That is, conceptual relations in a meaning stimulus do not correspond directly to syntactic relations. But once it has been partially determined how that stimulus will be expressed -- e.g., what verb will be used -- some conceptual-syntactic correspondence can be made. The predictions set up by word senses include not only syntactic relations, but prepositions to be used. Thus the model treats prepositions, to a great extent, as words which English rather capriciously uses to relate other meanings once it has been decided how these meanings are to be expressed.

The second source of information for establishing syntactic relationships is the LANGUAGE SPECIFIC functions. These handle notions like TENSE, which are required in the target language but which are not expressed with words which realize a portion of the conceptual stimulus. They also handle conceptual information which may be present in the stimulus but not predictable from any word generated in the process of realizing that stimulus. The conceptual PART and PLUS relations, which are both realized with a

syntactic possessive in English, are examples of such information.

The second phase of generation linearizes the previously constructed syntax net. This is accomplished through the use of an Augmented Finite State Transition Network grammar. This grammar incorporates language specific, but meaning independent, knowledge -- e.g., how verbs must be inflected to express a particular tense, or the order of constituents in a noun phrase. This process accesses neither the conceptual stimulus nor the memory model.

The result of this design is a rigidly 'stratified' model. It assumes the existence of some language-free process (WHAT-TO-SAY) which decides on information to be expressed. The first phase of generation (syntax net construction) operates on this information and deals with all that information which relates meaning to language. A final process (net linearization) operates on the output of this phase and deals with meaning-free aspects of language. Such a sequential processing represents a 'first order' approximation to an ideal generator. A more sophisticated model would treat these aspects of generation as co-processes, permitting far more interaction between meaning based operations and the developing surface detail of the sentence.

In testing the model only one meaning representation, Conceptual Dependency, was used. We firmly believe, however, that neither the details of this representation nor even its most basic properties, (primitive ACTS, a conceptual case system) are essential parts of the generative theory developed here. The language specific information, which makes BABEL produce English, has been made quite visible and easy to change. It is representation specific information which makes BABEL work with Conceptual Dependency representations. This information, from a programming viewpoint, is more deeply embedded in the model. But even it could be altered to accommodate a radically different representation without altering the basic generative theory embodied in the program.

In retrospect, we feel that the specific task selected for study in this thesis was a reasonable one. It would have been possible to define "generation" in a broader sense, either by including "meaning selection" at the start of the task or by removing the single sentence restriction on the output. But in spite of ignoring these problems we feel that the resultant theory could be extended to deal with them without requiring drastic internal change. At the same time, the task chosen was broad enough to encompass the major language specific aspects of even a much more all encompassing view of generation. A narrower domain

might well have led to unwarranted assumptions about the capabilities of other processes and thereby to a non-extendible theory.

Psychological Considerations

Throughout this thesis BABEL has been referred to as a "generation model". In hindsight, the use of the term "model" was probably not a wise idea. Our goal was to express meanings in language, not to produce sentences by processes stepwise analogous to those used by humans.

Regardless of our intentions, it is difficult to avoid psychological speculation in looking at a program which performs an intrinsically human task. Because our implementation utilizes Conceptual Dependency, a representation for which some psychological validity has been claimed [5], such speculations are certain to be made by others.

One might ask what sorts of predictions BABEL, viewed as a psychological model, makes about observable human language generation. Simply looking at isolated sentences produced by the program is unenlightening. The sentences are 'grammatical' and 'meaningful', but clearly much more limited in syntactic variety and meaning domain than those of any human speaker. But no particular syntax or meaning limitation appears inherent in the methods used.

Of more interest is the fact that BABEL makes no decision about splitting information into sentences. Thus, we could devise conceptualizations which would yield very long or deeply nested sentences which would not be observed in human generation -- e.g.,

"John heard that Bill told Jim that Mary . . ."

It is conceivable that this is not an argument against the psychological validity of BABEL. If some conceptual process operating temporally prior to BABEL selected sentential size information chunks, then BABEL could remain unchanged and would not generate such awkward sentences. Intuitively it seems implausible that this fragmentation could take place prior to generation, but it is simply not a behaviour which is readily observable in humans.

BABEL also makes predictions about paraphrasing. Actual experiments <26> indicate that humans can produce the kinds of paraphrases produced by BABEL, as well as syntactic and "synonym substitution" paraphrases. So people at least have the knowledge that, for instance, "give" and "get" have the same, or closely related, meanings. Such experiments are subject to two sources of confusion, though. First, when subjects are given sentences to paraphrase, they are starting with linguistic matter. BABEL starts with meanings. There is no way to observe what part of the subjects' behaviour is a result of his knowledge about the

particular words and syntax of the stimulus sentence, and what is due to its meaning. Second, subjects will differ on their interpretation of instructions to "paraphrase" a sentence, or produce sentences which "mean the same thing". There is no correct answer in the task; each subject may have his own interpretation of just what he is to do.

The most interesting predictions made by BABEL are perhaps those which make word choice a function of context and world knowledge as well as meaning. Ability to test such a prediction is again limited by the impossibility of presenting 'pure meaning' as a stimulus to a human. We might try an experiment using pictures rather than sentences to avoid linguistic biases in the stimuli. Using sequences of movie scenes, we could set up differing contexts for some 'target' scene. BABEL would predict different descriptions of the target scene in differing contexts. While such an experiment might well confirm BABEL's predictions, it would leave in doubt the question of whether the context was affecting the generation of language to express meaning, or merely affecting the analysis of the scene.

Rather than look at predictions made by BABEL, one could merely inspect the psychological evidence available pertaining to language generation. Again, the inability to create a pure meaning stimulus severely limits experimentation. We can look at written and spoken

sentences, but this reveals virtually nothing of the means by which these sentences were produced, which is what we are interested in. More useful is temporal and introspective information:

(1) sentences are spoken from 'left to right'

(2) pauses between words are irregular; it sometimes appears that part of a sentence is spoken before the words, or perhaps the ideas, for the remainder of the sentence have even been determined.

(3) the 'tip of the tongue' phenomenon -- people are sometimes positive that they know the meaning they want to express, but just can't find the word for it.

Now (1) provides very little information; it deals with only the tip of the iceberg. The AFSTN grammar maps syntax nets into sentences from left to right; an algorithm which reads off terminals from a phrase marker does the same. (1) alone simply doesn't tell us anything about the order in which words and phrases are thought of. If one is willing to accept that pauses are indicative of time taken to choose words or ideas, then such pauses are evidence against BABEL. This is because BABEL produces an entire syntax net before beginning the linearization process. This process is carried out by the AFSTN grammar, and the only pauses it predicts are due to differences in complexity of syntactic processing -- that is, processing of TENSE and VOICE, construction of noun phrases from their elements, etc.

If the pauses represent word selection processing, then BABEL fails because the syntax net includes all words used in the sentence. Even in the creation of the syntax net, BABEL does not operate on information in the order in which that information appears in the surface. Rather, it chooses verbs before realizing the information which becomes subject, direct object, etc. This is done because the selection of a verb is used by the progress, as has been described, to guide the generation process. It does not seem that the selection of a subject first could be used to such computational advantage. Because of this non 'surface order' processing, the failure of BABEL to match apparent human performance in this area could not be dealt with by merely arguing that people speak words as they think of them, and that therefore the pauses are due to processing times, predicted by BABEL, used in finding words to express ideas.

If, however, the pauses are seen as time spent generating ideas, then it becomes necessary to make generation interact with the WHAT-TO-SAY process. Such a model would decide to talk about some memory node M185; while deciding how to express M185 (as a sentence subject, perhaps), it would be choosing conceptual information to express about M185. The processes of idea generation, word selection, and syntax become terribly intertwined. Even given the great

syntactic variety of natural language (of English, at any rate), it would be difficult to avoid blind alleys requiring back-up. From a psychological standpoint, however, such a model might be preferable to a computationally more efficient one.

The tip of the tongue phenomenon (1) is not predicted by BABEL. A prediction is made, however, that words which convey a great deal of conceptual information and which have to be discriminated from many 'similar' words will take longer to retrieve than less 'complex' words. That is, BABEL does treat word selection as a quite difficult task and not as a question of simply following pointers from 'concepts' to words. If the tip-of-the-tongue feeling is what it seems, then BABEL's treatment is, to some extent, psychologically correct.

In summary, BABEL does not provide anything like an intuitively adequate psychological model of the vast problem of language generation. Some inadequacies could be cured without major revision of the model (although possibly with loss of computational efficiency); others would require changes to some of the basic assumptions of the model. Since our goals in constructing this program were not in the area of psychological modelling, however, such revisions have not been attempted. It is entirely possible that further research, particularly in that area we have termed

WHAT-TO-SAY, will reveal a greater unity of process between good psychological models and good computational models.

The (Near or Distant) Future

Many problems remain to be solved before computers become useful devices for the production of human language. Open any book, choose any paragraph (almost any sentence, for that matter), is virtually certain to bring up some questions not even 'fudged over', much less solved, in this thesis. The following list includes some of (what currently appear to be) the most important issues:

(1) Many areas of meaning are not readily <u>represented</u> using only conceptual structures thus far presented.

(a) <u>Spatial relations</u> need a uniform treatment. A model based on physical reality is almost certainly desirable in the representation. But finding these relationships from the natural language constructions used to describe them will require considerable knowledge of normal spatial relationships in the world. "A fence around a yard," "chairs around a table", and "people around a fireplace" are quite different spatial "around" relations.

(b) <u>Quantification</u> has been thoroughly avoided in this thesis. What are good representations of the meanings of sentences which use words like "each", "all", "some", "few", etc.? Or sentences like "The statement was widely disbelieved" or "The war caused great misery?" What problems may arise in the generation of such sentences?

(c) Nouns which name neither physical object classes nor simple events are common in English. What should be done about words like "war" and "party", which represent large complexes of events and situations? Or relation naming nouns, like "father" or "brother"? Or nouns like "present", "decision", and "mistake"?

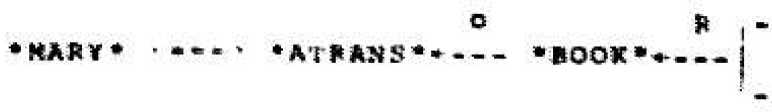
(d) Verbs and adjectives which depend on detailed physical features are difficult to represent in terms of the conceptual mechanisms used by BABEL. What constitutes "dancing" or "marching"? Should "spotted" and "striped" be represented in more primitive terms?

These are but a few of the representational questions yet to be solved. It is entirely possible that they are purely representational questions, and will present no particular difficulties for generation once solved. On the other hand, we may not be so lucky.

(2) How should a given object be expressed? The same person may be simultaneously "Bill", "one of George's cousins", and "the man standing on the corner". Of primary concern here are the questions: (i) When does an object need to be uniquely specified and when are only certain of its properties of interest? (ii) When an object must be uniquely specified, from what set of objects must it be explicitly distinguished by the language chosen to realize it? ("the spare tire" clearly refers to the one in the trunk of the car with a flat, not to any of the spare tires

in trunks of cars going by on the road).

(3) To what extent must syntax interact with word selection? The beginnings of such interaction are discussed in Chapter 7, where the notion of syntactic compatibility was introduced. Here we were dealing with noun-verb differences. But more complicated situations required greater sophistication. In generating "John asked Mary to S" the embedded sentence S must have "Mary" as a deleted logical subject. Thus if the meaning underlying S were



it is alright to choose "give", which makes "Mary" the subject, as the main verb of the embedded sentence, but not alright to choose "receive". It is simple enough to devise rules to block the generation of such sentences in the AFSTN stage of generation. To forsee the problem and avoid choosing "receive" is much more difficult.

(4) The conceptualization→sentence relationship should be eliminated. It should be possible to express a large conceptual network as a sequence of sentences. What are good rules for organizing the information into individual sentences? To what extent are such

rules dependent on the information content of the network, and to what extent dependent on linguistic considerations?

(5) At what point must the generator actually worry about potential ambiguity in its generated sentences? To what extent does this require the generator to incorporate, or have access to, a model of language analysis?

(6) At what points in the generation process should a model of the hearer's world be taken into consideration?

(7) The WHAT-TO-SAY problem remains a huge obstacle to many tasks involving natural language generation. A theory of information flow in conversations is needed, as well as better memory organization and search techniques than any now existing.

Perhaps the most fundemental problem to be faced by researchers in natural language processing is a methodological one. As we progress more deeply into the interactions between knowledge and language, we are forced to limit the domain of investigation. This can be done on a linguistic level, through limitations of vocubulary and syntax, on a conceptual level, by limiting the world of discourse, and on a task level, by designing algorithms specifically for machine translation or question answering.

The art of choosing a limited yet fruitful domain is still in its infancy.

We believe the work described in this thesis gives reason to be confident that machines will someday be able to converse with humans in natural language. But looking at the many problems which remain the area of generation alone, we see no reason to claim to have yet reached the proverbial corner "around which the solution is just".

NOTES

CHAPTER 1

1. The proper organization of a conceptual memory is still an unsolved problem. Associative network structures are commonly proposed for such memories. Rieger <24> describes a model based on conceptual representations.

CHAPTER 3

1. C.D. separates the 'mind' into three areas: Conscious Processor, Immediate Memory, and Long-Term Memory. The nature of these mental locations, and some psychological and linguistic ramifications of this division, are discussed in <32>.

2. The question of whether conceptual representations really can provide a canonical form for a given 'meaning' cannot be answered without some independent test of meaning identity. Of far greater importance is the property of similar representations for similar meanings. If the input had been "John told Mary that reading the book would pick her up" the conceptual analysis would not have been identical to (8), but would have been very similar. In particular, the 'Benefit' scale would have been replaced by some less general one, like *JOY*. (8) would be an immediate inference from this representation, and the remainder of the inference path described in the text could be followed.

4. We shall use the notation CR ("language string") to stand for the Conceptual Representation of the "language string".

CHAPTER 4

1. We shall discuss in Chapter 7 how many nominals and adverbials also convey a great deal of conceptual information. From the standpoint of word selection, we will show how these can also be handled by the mechanisms used for verbs. The nominals handled by the currently implemented BABEL, however, are simply names of objects and people and do not break up into large conceptual structures.

2. It may be claimed that condition (C4-5) is too weak for the choice of the word 'return'. Events which intervene between T0 and T1, such as Mary's selling the book, might make 'return' inapplicable. English speakers undoubtedly have differing requirements for the use of the word, and until the model is actually tested in a realistic application, it is impossible to say how complex a test will be necessary to obtain reasonable realizations. The point of the example is that, whatever conditions are chosen, they will necessitate accessing the memory's world model.

3. In a running model one would probably not want to make the absolute distinction made here between 'finding' the requested information and 'proving' it. Rather, the memory would probably be asked to allocate a given amount of effort to verifying the information. In no case would we want to actually turn a theorem prover loose spending arbitrarily large amounts of time on such a problem, since failure to utilize the 'best' word is certainly not disastrous. It should be noted that a model which permitted multiple representations for a given meaning would be much less likely to be able to find the information already stored and thus would generally have to attempt some sort of proof.

CHAPTER 5

1. There are two aspects to the retrieval complexity we are hypothesizing. One is the sort of complexity inherent in conceptual representations, which could be made explicit by defining a complexity on conceptual structures in much the same fashion as has been done for formal grammars <8>. The second aspect is 'evaluation' complexity of predicates used as DCs. Those which interact with memory presumably are more time consuming than the simple pattern matching ones.

2. BABEL currently deals only with existentially quantified variables in its conceptualizations. Little work has been done on the uses of quantification in conceptual structures, and it remains to be seen whether situations exist in which universally quantified variables would also be useful in word selection predicates.

3. Functional information is present basically for inferences about what a person will do with an object if he has it -- drink beer -- or why he might want it -- if someone wants an apple he may be hungry.

English bases linguistic distinctions on very few of the functional relationships which exist in a conceptual memory. See Rieger ('24') for further discussion of this topic.

4. The C.D. representation of this verb involves INGESTing *SMOKE* by PROPELLing it from some object to the *MOUTH*. This is an example of 'elaboration' of an ACT (in this case the ACT *INGEST*). C.D. represents this by relating an entire conceptualization (in this case one involving *PROPEL*) to the ACT as an INSTrumental. Schank ('19') describes the uses of this construction. Since it presents no new problems or insights in generation, it is not detailed in this thesis.

5. C.D. associates a DIRECTIVE case with *INGEST*. Since all our *INGEST* examples deal with the *MOUTH* of the ACTOR as SOURCE and the *INSIDE* of the ACTOR as GOAL, we simply omit them from the diagrams. They are present in the internal representations used by the program. Other verbs, such as 'absorb', would use different values in these slots.

6. By using integral values in conceptual representation and real valued 'breakpoints' on the scales, we avoid making choices between two applicable ranges.

CHAPTER 6

1. The order of processing of the two frames which reference C1 could be reversed, but this would have no effect on the final form of the syntax net.

CHAPTER 7

1. This is not totally true. The program does make a few distinctions based on <u>focus</u> -- e.g., (give - receive), (tell - hear from). In our implementation, this is accomplished by a predicate in the discrimination net which looks for FOCUS markers in the conceptual stimulus, branching to different response nodes depending on their presence. To be consistent with the suggestions which follow, we should really store these responses at a single response node. The generator would then inform memory of various focussing possibilities it had found and allow memory to indicate a preference, if it had one. The concexicon entry for GIVE1, for example, would include an indication of an ACTOR focus. The entry RECEIVE 1 would indicate RECIPIENT focus.

REFERENCES

<1> Abelson, R., and Carrol, J., "Computer Simulation of Individual Belief Systems", Amer. Behav. Sci., Vol 8, No. 9, 1965.

<2> Bobrow, D., "Natural Language Input for a Computer Problem Solving System", in Semantic Information Processing, M. Minsky (ed.), MIT Press, Cambridge, Mass., 1968.

<3> Bruce, B., "A Model for Temporal References and its Application in a Question Answering Program", Artificial Intelligence, Vol. 3, No. 1, 1972.

<4> Charniak, E., "Towards a Model of Children's Story Comprehension", AI TR-266, Massachusetts Institute of Technology, Cambridge, Massachusetts, December, 1972.

<5> Chomsky, N., Aspects of the Theory of Syntax, MIT Press, Cambridge, Mass., 1965.

<6> Colby, K., Weber, S., and Hilf, F., "Artificial Paranoia", AIM-125, Computer Science Department, Stanford University, Stanford, California, July, 1970.

<7> Feigenbaum, E., "The Simulation of Verbal Learning Behavior", in Computers and Thought, Feigenbaum and Feldman (eds.), McGraw Hill, New York, 1963.

<8> Feldman, J.A., "Some Decidability Results on Grammatical Inference and Complexity", Information and Control, 1972.

<9> Fillmore, C.J., "The Case for Case", in Universals in Linguistic Theory, Bach and Harms (eds.), Holt Rinehart, and Winston, New York, 1968.

<10> Fillmore, C.J., "Types of Lexical Information", Working Papers in Linguistics, No. 2, November, 1968.

<11> Fodor, J. and Katz, J., "The Structure of a Semantic Theory", in The Structure of Language, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.

<12> Friedman, J., "Directed Random Generation of Sentences", CACM, Vol. 12, No. 1, January, 1969.

<13> Friedman, J., "A Computer System for Transformational Grammar", CACM, Vol. 12, No. 6, June, 1969.

<14> Friedman, J., A Computer Model of Transformational Grammar, American Elsevier, New York, 1971.

<15> Hintzman, D., "Explorations with a Discrimination Net Model for Paired Associate Learning", Journal of Math. Psych., Vol. 5, 1968.

<16> Hunt, E., Concept Learning: an Information Processing Problem, Wiley, New York, 1968.

<17> Joos, M., The English Verb: Form and Meanings, Univ. of Wisconsin Press, Madison, Wis., 1968.

<18> Klein, S., "Automatic Paraphrasing in Essay Format", Mechanical Translation, Vol. 8, No. 4, June, 1965.

<19> Klein, S., "Control of Style with a Generative Grammar", Language, Vol. 41, No. 4, December, 1965.

<20> Loehlin, J., Computer Models of Personality, Random House, New York, 1968.

<21> McCarthy, J., "The Home Information Terminal", Man and Computer Proc. Int. Conf., Bordeaux, 1970, S. Karger, Basel, 1972.

<22> Quam, L. and Diffie, W., "Stanford Lisp 1.6 Manual", Stanford A.I. Lab. Operating Note 28.7, Stanford University, Stanford, California.

<23> Quillian, R., "Semantic Memory", in Semantic Information Processing, M. Minsky (ed.), MIT Press, Cambridge, Mass., 1968.

<24> Rieger, C., 'Conceptual Memory', Ph.D. Thesis, Computer Science Dept., Stanford University, Stanford, California, 1974.

<25> Riesbeck, C., 'Computational Understanding of Natural Language Using Context', Ph. D. thesis, Computer Science Dept., Stanford University, Stanford, California, (forthcoming).

<26> Roberts, K.H., "An Investigation of Paraphrasing: The Effects of Memory and Complexity", Tech. Report No. 30, Human Performance Center, University of Michigan, June, 1971.

<27> Roget's Thesaurus, St. Martin's Press, New York, 1965.

<28> Rumelhart, D., Lindsay, P., and Norman, D., "A Process Model for Long-Term Memory", in Organization and Memory, Tulving and Donaldson (eds.), Academic Press, New York, 1972.

<29> Sandewall, E., "Representing Natural Language Information in Predicate Calculus", Machine Intelligence 6, Metzler and Michie (eds.), 1971.

<30> Schank, R., "The Fourteen Primitive Actions and their Inferences", AIM-183, Computer Science Department, Stanford University, Stanford, California, March, 1973.

<31> Schank, R., "Conceptual Dependency: A Theory of Natural Language Understanding", Cognitive Psychology, Vol. 3, No. 4, October, 1972.

<32> Schank, R., Goldman, N., Rieger, C., and Riesbeck, C., "Primitive Concepts Underlying Verbs of Thought", Stanford Artificial Intelligence Project Memo 162, February, 1972.

<33> Schank, R., Goldman, N., Rieger, C., and Riesbeck, C., "MARGIE: Memory, Analysis, Response Generation, and Inference on English", Proceedings of the Third International Joint Conference on Artificial Intelligence, 1973.

<34> Simmons, R., and Slocum, J., "Generating English Discourse from Semantic Networks" CACM, Vol. 15, No. 10, October 1972.

<35> Simmons, R., "Semantic Networks: Their computation and use for understanding English Sentences", Computer Models of Thought and Language, Schank and Colby (eds.), San Francisco, 1973.

<36> Simmons, R., "Some Semantic Structures for Representing English Meanings", in Language Comprehension and the Acquisition of Knowledge, Freedle and Carrol (eds.), V.H. Winston and Sons, Washington, 1972.

<37> Simon, H. and Gilmartin, K., "A Simulation of Memory for Chess Positions".

<38> Smith, D., "Mlisp", AIM-135, Computer Science Dept., Stanford University, Stanford, California, October, 1970.

<39> Su, S. and Harper, K., "A Directed Random Paragraph Generator", 1969, Intertational Conference on Computational Linguistics, Sweden.

<40> Weizenbaum, J., "ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine", CACM, Vol. 9, No. 1, January, 1966.

<41> Winograd, T., "Procedures as a Representation for Data in a Computer Program for Understanding Natural Langauge", TR-84, M.I.T. Project MAC, February, 1971.

<42> Woods, W., "Transition Network Grammars for Natural Language Analysis", CACM, Vol. 13, No. 10, October, 1970.

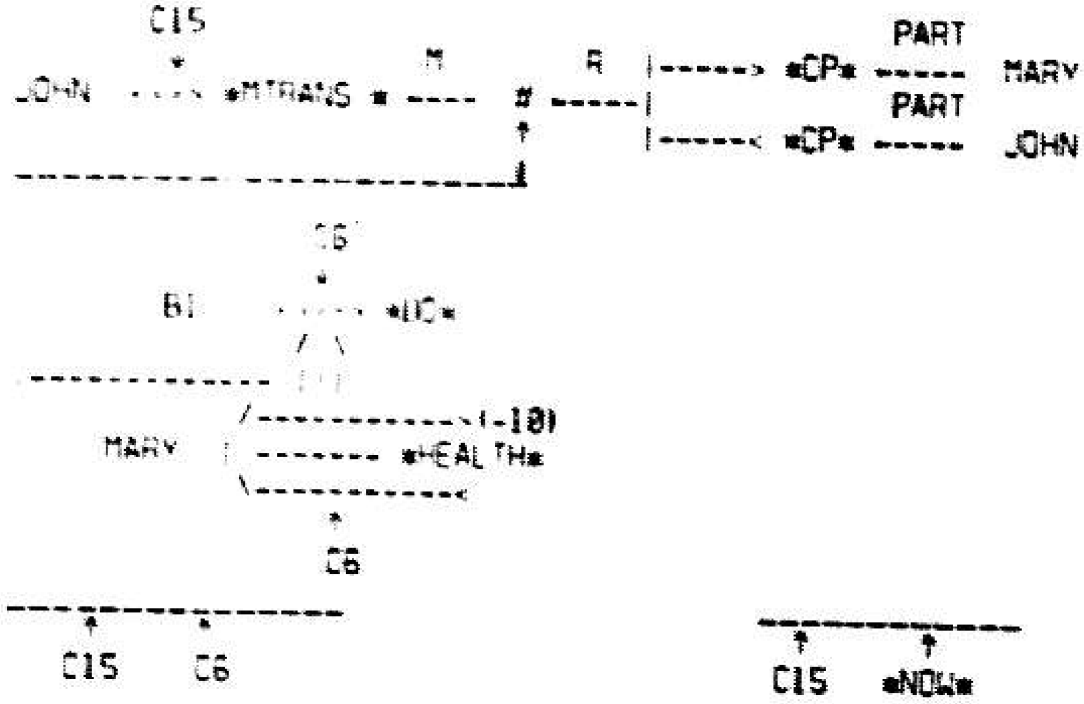
<43> Woods, W. and Kaplan, R., "The Lunar Sciences Natural Language Information System", BBN Report No. 2265, Bolt, Beranek, and Newman, September, 1971.

<44> Yngve, V., "Random Generation of English Sentences", 1961 International Conference on Machine Translation of Languages and Applied Language Analysis, Teddington, Her Majesty's Stationery Office, London, 1962.

<45> Yngve, V., "Implications of Mechanical Translation Research", Proceedings of the American Philosophical Society, Vol. 108, No. 4, 1964.

APPENDIX 1

Following is annotated output produced by BABEL in constructing a syntax net for the conceptual stimulus:



which corresponds to the internal LISP form:

```
((ACTOR (JOHN) ... (*MTRANS*) MOBJECT ((CON ((ACTOR (BILL) <*> (*DO*)~
 OBJECT (*CP* ...) TIME (C0005) FOCUS ((ACTOR))) <* ((ACTOR (MARY) <*>F~
 ...FACT ...) ... (*HEALTH* VAL (-10))) TIME (C0006))) FROM (*CP* REF~
 (*A* PART (JOHN))) TO (*CP* REF (*A* PART (MARY))) TIME (C0015) FOC~
US ((ACTOR)))
```

Because the skeleton of this stimulus is "E", and the ACT is "MTRANS", the "MTRANS" discrimination net (figure 5-6) is employed. The response "ARN1" is selected:

```
WORD SENSE SELECTED = WARN1
NO SPECIAL ACTIONS
PROCESSING MODIFIERS

NEW SYNTAX NET

N0006:  MOOD     (INDIC)
        VOICE    (ACT)
        FORM     (SIM)
        TENSE    (PAST)
        LEX      (WARN)
```

TENSE, FORM, VOICE, and MOOD are determined. Next the program begins to process the FRAMEWORK of WARN1:

```
PROCESSING FRAMEWORK

BEGINNING NEXT FRAME:
SYNTAX RELATION = ACTSBJ
CONCEPTUAL-SPECIFICATION = (ACTOR)
SPECIAL REQUIREMENTS = NIL

WORD SENSE SELECTED = JOHN
NO SPECIAL ACTIONS
PROCESSING MODIFIERS

NEW SYNTAX NET

N0005:  ACTSBJ   (N0007)
        MOOD     (INDIC)
        VOICE    (ACT)
        FORM     (SIM)
        TENSE    (PAST)
        LEX      (WARN)

N0007:  LEX      (JOHN)
```

The ACTOR of the MTRANS (JOHN) is made the ACTSBJ of "warn".

```
BEGINNING NEXT FRAME:
SYNTAX RELATION = OBJ2
FIELD-SPECIFICATION = (TO PART)
SPECIAL REQUIREMENTS = NIL

WORD SENSE SELECTED = MARY
NO SPECIAL ACTIONS
PROCESSING MODIFIERS

NEW SYNTAX NET

N0006:  OBJ2    (N0008)          N0007:  LEX     (JOHN)
        ACTSBJ  (N0007)
        MOOD    (INDIC)          N0008:  LEX     (MARY)
        VOICE   (ACT)
        FORM    (SIM)
        TENSE   (PAST)
        LEX     (WARN)
```

The RECIPIENT (TO PART) of the NTRANS, (MARY), becomes the OBJ2 of "warn".

```
BEGINNING NEXT FRAME:
SYNTAX RELATION = S2
FIELD-SPECIFICATION = (MOBJECT)
SPECIAL REQUIREMENTS = NIL
```

The MOBJECT is a conceptualization which has the skeleton "EKC" (event-cause-statechange). Thus the EKC discrimination net is tried first. The response KILL1 is selected:

```
WORD SENSE SELECTED = KILL1
NO SPECIAL ACTIONS
PROCESSING MODIFIERS

NEW SYNTAX NET

N0006:  S2      (N0013)          N0007:  LEX     (JOHN)
        OBJ2    (N0008)
        ACTSBJ  (N0007)          N0008:  LEX     (MARY)
        MODU    (INDIC)
        VOICE   (ACT)
        FORM    (SIM)
        TENSE   (PAST)
        LEX     (WARN)

N0013:  MODU    (INDIC)
        VOICE   (ACT)
        FORM    (SIM)
        TENSE   (FUTPAST)
        LEX     (KILL)
```

The FRAMEWORK of KILL1 must be processed next. Any remaining FRAMEs for the concexicon entry WARN1 (in this case there are none) are saved on the push-down list.

The first FRAME for KILL1 specifies an ACTSBJ to be found as the ACTOR of the ANTECEDENT of the active conceptual structure. This turns out to be the node for BILL:

```
PROCESSING FRAMEWORK

BEGINNING NEXT FRAME:
SYNTAX RELATION = ACTSBJ
FILLER-SPECIFICATION = (ACTOR)
SPECIAL REQUIREMENTS = NIL

WORD SENSE SELECTED = BILL
NO SPECIAL ACTIONS
PROCESSING MODIFIERS
```

NEW SYNTAX NET

```
N0006:  SC      (N0013)
        OBJC    (N0008)
        ACTSBJ  (N0007)
        MOOD    (INDIC)
        VOICE   (ACT)
        FORM    (SIM)
        TENSE   (PAST)
        LEX     (WARN)

N0013:  ACTSBJ  (N0017)
        MOOD    (INDIC)
        VOICE   (ACT)
        FORM    (SIM)
        TENSE   (FUTPAST)
        LEX     (KILL)

N0007:  LEX     (JOHN)

N0008:  LEX     (MARY)

N0017:  LEX     (BILL)
```

The next FRAME of KILL specifies an OBJ

(direct object):

```
BEGINNING NEXT FRAME:
SYNTAX RELATION = OBJ
FILLER-SPECIFICATION = (<= ACTOR)
SPECIAL REQUIREMENTS = NIL
```

The conceptual node for MARY was processed earlier, and syntax net node N0008 already exists for it.

```
SYNTAX NODE ALREADY EXISTS:  N0008
NEW SYNTAX NET

N0006:  SC      (N0013)
        OBJC    (N0008)
        ACTSBJ  (N0007)
        MOOD    (INDIC)
        VOICE   (ACT)
        FORM    (SIM)
        TENSE   (PAST)
        LEX     (WARN)

N0017:  OBJ     (N0008)
        ACTSBJ  (N0017)
        MOOD    (INDIC)
        VOICE   (ACT)
        FORM    (SIM)
        TENSE   (FUTPAST)
        LEX     (KILL)

N0007:  LEX     (JOHN)

N0008:  LEX     (MARY)

N0017:  LEX     (BILL)
```

```
BEGINNING NEXT FRAME:
SYNTAX RELATION = INST2
FIELD-SPECIFICATION = (CON)
SPECIAL REQUIREMENTS = NIL
```

BABEL doesn't know that the expression of the instrument (means) of "killing" is optional in English. In this case, since it is a 'dummy' (the ACT *DO*, which is serving merely as a place holder), it would be better not to express it. Nevertheless:

```
WORD SENSE SELECTED = D01
NO SPECIAL ACTIONS
PROCESSING MODIFIERS

NEW SYNTAX NET

N0006:  SC      (N0013)          N0007:  LEX     (JOHN)
        OBJC    (N0008)
        ACTSBJ  (N0007)          N0008:  LEX     (MARY)
        MOOD    (INDIC)
        VOICE   (ACT)            N0017:  LEX     (BILL)
        FORM    (SIM)
        TENSE   (PAST)
        LEX     (WARN)

N0013:  INST2   (N0022)          N0022:  MOOD    (INDIC)
        OBJ     (N0008)                  VOICE   (ACT)
        ACTSBJ  (N0017)                  FORM    (SIM)
        MOOD    (INDIC)                  TENSE   (PRESFUT)
        VOICE   (ACT)                    LEX     (DO)
        FORM    (SIM)
        TENSE   (FUTPAST)
        LEX     (KILL)
```

Next the FRAMEWORK for D01 is processed.

The first FRAME yields an ACTSBJ for "do":

PROCESSING FRAMEWORK

BEGINNING NEXT FRAME:
SYNTAX RELATION = ACTSBJ
FIELD-SPECIFICATION = (ACTOR)
SPECIAL REQUIREMENTS = NIL

SYNTAX NODE ALREADY EXISTS : **N0017**
NEW SYNTAX NET

The conceptual node for BILL was processed earlier. Syntax net node N0017 exists for it.

N0006:	SC	(N0013)
	OBJC	(N0003)
	ACTSBJ	(N0007)
	MOOD	(INDIC)
	VOICE	(ACT)
	FORM	(SIM)
	TENSE	(PAST)
	LEX	(WARN)
N0013:	INSTC	(N0022)
	OBJ	(N0006)
	ACTSBJ	(N0017)
	MOOD	(INDIC)
	VOICE	(ACT)
	FORM	(SIM)
	TENSE	(FUTPAST)
	LEX	(KILL)

N0007:	LEX	(JOHN)
N0008:	LEX	(MARY)
N0017:	LEX	(BILL)
N0022:	ACTSBJ	(N0017)
	MOOD	(INDIC)
	VOICE	(ACT)
	FORM	(SIM)
	TENSE	(PRESFUT)
	LEX	(DO)

Finally, the FRAME for the OBJ (direct object) of "do" specifies that, for this 'dummy' sense, English uses the noun "something". This information is encoded as a SPECIAL REQUIREMENT in the FRAME:

BEGINNING NEXT FRAME:
SYNTAX RELATION = OBJ
FIELD-SPECIFICATION = (OBJECT)
SPECIAL REQUIREMENTS = ((QTHD SOMETHING))

NEW SYNTAX NET

N0006:	SC	(N0013)
	OBJC	(N0008)
	ACTSBJ	(N0007)
	MOOD	(INDIC)
	VOICE	(ACT)
	FORM	(SIM)
	TENSE	(PAST)
	LEX	(WARN)
N0013:	INSTC	(N0022)
	OBJ	(N0008)
	ACTSBJ	(N0017)
	MOOD	(INDIC)
	VOICE	(ACT)
	FORM	(SIM)
	TENSE	(FUTPAST)
	LEX	(KILL)
N0007:	LEX	(JOHN)
N0008:	LEX	(MARY)
N0017:	LEX	(BILL)
N0022:	OBJ	(N0026)
	ACTSBJ	(N0017)
	MOOD	(INDIC)
	VOICE	(ACT)
	FORM	(SIM)
	TENSE	(PRESFUT)
	LEX	(DO)
N0026:	LEX	(SOMETHING)

No FRAMEs remain for DO1, and, in popping the stack, the program finds that none remain for KILL1 or WARN1, either. Thus the syntax net is complete and linearization produces the surface form:
(JOHN WARNED MARY BILL WAS GOING TO KILL HER BY DOING SOMETHING)