

AD-762 471

A CONCEPTUALLY BASED SENTENCE
PARAPHRASER

Neil M. Goldman, et al

Stanford University

Prepared for:

Advanced Research Projects Agency

May 1973

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

DISCLAIMER NOTICE

THIS DOCUMENT IS THE BEST
QUALITY AVAILABLE.

COPY FURNISHED CONTAINED
A SIGNIFICANT NUMBER OF
PAGES WHICH DO NOT
REPRODUCE LEGIBLY.

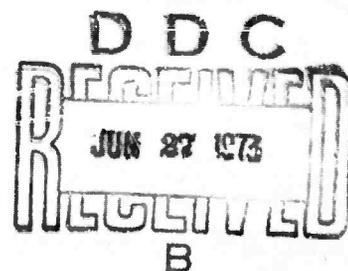
STAN-CS-73-357

AD 762471

A CONCEPTUALLY BASED SENTENCE PARAPHRASER

BY

NEIL M. GOLDMAN
CHRISTOPHER K. RIESBECK



SUPPORTED BY

ADVANCED RESEARCH PROJECTS AGENCY

ARPA ORDER NO. 457

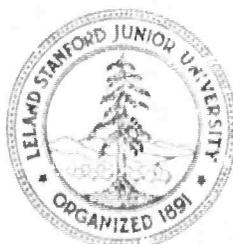
MAY 1973

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

COMPUTER SCIENCE DEPARTMENT

School of Humanities and Sciences

STANFORD UNIVERSITY



DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Stanford University Department of Computer Science Stanford, California 94305		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE A CONCEPTUALLY BASED SENTENCE PARAPHRASER			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) technical report, May 1973			
5. AUTHOR(S) (First name, middle initial, last name) Neil M. Goldman and Christopher K. Riesbeck			
6. REPORT DATE May 1973	7a. TOTAL NO. OF PAGES 90 92	7b. NO. OF REFS 12	
8a. CONTRACT OR GRANT NO. ARPA SD-183	9a. ORIGINATOR'S REPORT NUMBER(S) STAN-CS-73-357		
b. PROJECT NO. ARPA Order # 457	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) MEMO AIM-196		
c.			
d.			
10. DISTRIBUTION STATEMENT Releasable without limitations on dissemination.			
11. SUPPLEMENTARY NOTES Reproduced from best available copy. 		12. SPONSORING MILITARY ACTIVITY	
13. ABSTRACT This report describes a system of programs which performs natural language processing based on an underlying language free (conceptual) representation of meaning. This system is used to produce sentence paraphrases which demonstrate a form of understanding with respect to a given context. Particular emphasis has been placed on the major subtasks of language analysis (mapping natural language into conceptual structures) and language generation (mapping conceptual structures into natural language), and on the interaction between these processes and a conceptual memory model.			

MAY 1973

COMPUTER SCIENCE DEPARTMENT
REPORT NO. CS-357

A CONCEPTUALLY BASED SENTENCE PARAPHRASER

by

Neil M. Goldman
Christopher K. Riesbeck

ABSTRACT: This report describes a system of programs which performs natural language processing based on an underlying language free (conceptual) representation of meaning. This system is used to produce sentence paraphrases which demonstrate a form of understanding with respect to a given context. Particular emphasis has been placed on the major subtasks of language analysis (mapping natural language into conceptual structures) and language generation (mapping conceptual structures into natural language), and on the interaction between these processes and a conceptual memory model.

*The authors are indebted to Professor Roger Schank, whose work provided the theoretical foundation for the system described in this report, and who co-ordinated the efforts of several students in this research. The authors also gratefully acknowledge the contribution of Charles Rieger, who collaborated in the design of the paraphrase program and provided a programmed memory model compatible with its needs.

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract 3D-183.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

Reproduced in the USA. Available from the National Technical Information Service, Springfield, Virginia 22151.

A CONCEPTUALLY BASED SENTENCE PARAPHRASER

by

Neil M. Goldman
Christopher K. Riesbeck

CONTENTS

I.	INTRODUCTION	1
II.	CONCEPTUAL REPRESENTATION	10
III.	ANALYSIS OF NATURAL LANGUAGE SENTENCES	21
IV.	PRODUCTION OF NATURAL LANGUAGE SENTENCES FROM CONCEPTUAL REPRESENTATIONS	44
V.	CONCLUSION	73
VI.	EXAMPLES OF PROGRAM OUTPUT	76

I.

INTRODUCTION

MARGIE -- Meaning Analysis, Response Generation, and Inference in English -- is a model of natural language processing incorporated in a computer program now running at the Artificial Intelligence Laboratory at Stanford University. The program contains the core processes -- language analysis, memory model, and language generation -- necessary for several natural language tasks. Its operation in one task domain, sentence paraphrasing, is the topic of this report.

The task of paraphrasing English sentences can be stated simply as follows:

Given an English sentence, produce other sentences which English speakers interpret as having the same meaning.

Of course the notion of 'meaning' is a very vague one and the only test available for the acceptability of a proposed paraphrase of a sentence is to ask native speakers whether the two sentences 'mean' the same thing. Fortunately speakers seem to agree on the meaning of an isolated sentence, at least to a considerable amount of detail; the question of whether the sentences we generate are paraphrases according to some more formal definition of 'meaning' and 'paraphrase' will not concern us here.

The following examples, which the program produces, should give the reader a better feeling for this notion of sentence paraphrasing:

- Source: JOHN GAVE MARY A BICYCLE.
Paraphrase: MARY RECEIVED A BICYCLE FROM JOHN.
- Source: JOHN ADVISED MARY TO DRINK THE WINE.
Paraphrase: JOHN TOLD MARY SHE WOULD LIKE TO DRINK THE WINE.
- Source: MARY WANTS TO CHOKE FRED.
Paraphrase: MARY BELIEVES SHE WOULD ENJOY PREVENTING FRED FROM BREATHING BY GRABBING HIS NECK.
- Source: JOHN PREVENTED MARY FROM GIVING BILL THE BOOK BY GIVING THE BOOK TO FRED.
Paraphrase: BILL WAS UNABLE TO GET THE BOOK FROM MARY BECAUSE JOHN GAVE FRED THE BOOK.
- Source: JOHN KILLED MARY BY CHOKING HER.
Paraphrase: JOHN STRANGLED MARY.
Paraphrase: JOHN CHOKED MARY AND SHE DIED BECAUSE SHE WAS UNABLE TO BREATHE.
- Source: JOHN TOLD MARY HE WOULD HIT HER WITH HIS FOOT.
Paraphrase: JOHN THREATENED TO KICK MARY.
- Source: JOHN LOANED A BICYCLE TO MARY.
Paraphrase: JOHN GAVE MARY A BICYCLE AND HE EXPECTS SHE WILL RETURN IT TO HIM.

Each of these examples is handled by the current program, the 'Source' coming from a human user, the 'Paraphrase' being produced by the program. The program does not handle pronouns in the input; the fifth example would be typed in as "JOHN KILLED MARY BY CHOKING MARY". There are also some minor distinctions in form between the

output produced by MARGIE and the paraphrases in these examples, although in all cases the sentences produced are close to those shown and use the same words. A complete listing of the program's actual performance on several examples is provided at the end of this report.

The ability to paraphrase single sentences is not itself particularly interesting, primarily because of its artificiality. Certainly humans have the ability to create sentence paraphrases, as has been demonstrated in psycholinguistic research (4). It is an ability, however, which is seldom used outside of experimental contexts. (Trying to explain the meaning of a complicated sentence to a non-native speaker or a child is one natural use of this ability.) Furthermore, none of the commonly proposed computational tasks which deal with natural language processing directly involve sentence paraphrasing.

Nevertheless sentence paraphrasing, at least as defined and accomplished by the model described in this report, is an interesting research task for several reasons:

A) MARGIE's method of performing this task can be viewed as INTRA-LINGUAL MACHINE TRANSLATION. This is because the paraphrases are not produced by directly converting patterns in the source sentence into patterns in the paraphrase sentence. Rather, the inputs are first converted into a language-free (conceptual) representation of their meaning. This representation alone is then used to produce paraphrase sentence(s). Neither the words nor the

syntax of the input are considered in their generation. The same language-free representation produced by the analyzer for the paraphrase task could be used to generate German or French realizations as well as English. To perform translation from English to a second language it would not be necessary to alter the input-output behavior of the analysis algorithm. The generation algorithm would need to be provided with all the necessary linguistic data for the target language. Analogously, the generation algorithm used for English sentence paraphrasing could be used in conjunction with a German analysis routine to perform German-English translation.

Although neither the analysis, generation, or memory model of the current program is powerful enough to be used yet for the translation of interesting text, this is not due to a theoretical difference in the mechanisms involved in translation and paraphrase. MARGIE is designed to handle both these tasks and others (question answering, conversation) in three stages:

- i. produce a conceptual representation of the meaning of the input.
- ii. decide on the 'conceptual' content of the response.
- iii. produce a target language response which expresses this meaning.

The second of these stages is probably the least understood of these processes. Paraphrase and translation are closely related precisely because the same simple algorithm can be used for this step; namely, the representation produced by (i) can itself be used as the conceptual content passed to (iii).

B) Phrases can demonstrate UNDERSTANDING. It is possible to

obtain many paraphrases by syntactic manipulations. A sentence with a subject and a direct object can be put in active or passive voice, yielding such paraphrases as

"John threw the ball to Mary"

"The ball was thrown to Mary by John"

Such paraphrases could be produced without the conceptual analysis performed by MARGIE. A computer implementation of a transformational grammar could certainly do this. But no one would claim that such 'syntactic' paraphrases demonstrate understanding.

More interesting paraphrases result from situations in which two words may be used interchangeably but require a change in the syntax of the sentence:

"The university owns the land"

"The land belongs to the university"

It might appear that these transformations could be handled by 'word sensitive' transformational rules. But they actually require an analysis which finds 'semantic senses' of words, as is demonstrated by the paraphrase relation:

"I sold the Chevy to Fred"

"Fred purchased the Chevy from me"

but lack of paraphrase relation between

"I sold my idea to the management" and

"The management purchased my idea from me"

Such paraphrases thus require semantic disambiguation of words, a problem which, in much generality, is still beyond the capabilities

of current language processing programs. Since it is generally recognized that the solution to this problem requires some sort of understanding by the program, a system which produces these paraphrases in the appropriate contexts demonstrates some sort of understanding.

The need for disambiguation in paraphrasing can be seen even more clearly in the sort of paraphrase which breaks a word down into its 'components'. We might paraphrase

"Jerry dropped the lamp" with
"Jerry let go of the lamp which allowed the lamp to fall"

but we would not want

"Jerry dropped five dollars at the race track"

paraphrased analogously, at least for the primary reading of this sentence.

Even supposing the disambiguation problem were solved (or eliminated, by suitably restricting vocabulary and context), these component based paraphrases introduce a new problem. The same sort of mechanism which handled semantic synonymy might also handle the paraphrase:

Source: "My friend advised me to visit Spain"
Paraphrase: "My friend told me I would enjoy visiting Spain"

Suppose however it was desired to produce the above Source given the Paraphrase. Instead of simply recognizing the pattern 'advise' and applying a transformation, it is required that the pattern 'tell X (that) X would enjoy . . . ' be found in the analysis of the input.

To do this efficiently requires increased sophistication in a pattern matcher. Furthermore, if we wish to get 'advise' as a paraphrase of 'tell X (that) X would like to . . . ' and 'suggest to X (that) . . . would please X' it is apparent that matching syntactic patterns of word senses would rapidly run into problems from the quantity of patterns needed. This problem is avoided in MARGIE's method of paraphrasing.

MARGIE has no rules which specify explicit paraphrase relations between patterns of word senses. Given that MARGIE's paraphrases are produced from a language free representation, of course, no such patterns even exist. MARGIE searches instead for conceptual patterns. These are dependent on the meaning of the source sentence, but not on the particular words or syntax used. The patterns sought are no more complex than those which would be needed for the component based paraphrases above, and the number of patterns which must be discriminated is much smaller.

Finally, there is a form of paraphrase which is not even theoretically obtainable through word or word sense pattern matching, and which demonstrates even more clearly a sort of understanding.

For instance,

Source: "John told Fred he would bomb his office"
Paraphrase: "John threatened to bomb Fred's office"

cannot realistically be produced by finding a pattern involving 'tell' and 'bomb', since there are an infinite number of things which John could tell Fred that would constitute a threat. Although MARGIE

cannot perform all the functions necessary to produce such paraphrases, it does have the required linguistic mechanisms. This point will be discussed further in section IV.

In the absence of any clear notion of what 'understanding' is, it is pointless to claim that the production of a given paraphrase demonstrates a capacity for understanding. We will describe the processes by which MARGIE obtains such paraphrases and leave it for the reader to consider whether this meets his standards for classification as 'understanding'.

C) MARGIE's paraphrase production exhibits a use of CONTEXT in language processing. One of the most common criticisms of natural language research is the tendency to deal with example sentences outside of any context, whereas human language processing always occurs in complex social and linguistic contexts which affect both analysis and generation.

The paraphrasing of single sentences seems to share this fault. But the model described here performs all analysis and generation in the context of a memory model, comprising facts, beliefs, and rules which are actively used during the paraphrase process. Furthermore, the information contained in the natural language sentences being analyzed can be added to this memory model and affect the production of paraphrases of later sentences. Although MARGIE does not use linguistic context (the particular words or syntactic forms present in the input sentence) in its generative process, a limitation not shared by humans, it does use the non-linguistic context present in

the memory model is affected by the 'conceptual content' of the linguistic context. For instance, if MARGIE has been told

"Bill had the book", and
"Mary has the book",
and is then asked to paraphrase
"Mary will give the book to Bill"
it can produce
"Mary will return the book to Bill"

Since an understanding of the language analysis and generation processes described in this report requires an understanding of the nature of conceptual representation and, to a lesser degree, a knowledge of the particular representations used by MARGIE, the next section will be devoted to representational matters. It is suggested that readers already familiar with Schank's [6] work on Conceptual Dependency skip Section II and refer back to it for explanations of unfamiliar terms or notations

Many forms of representation of language content have been proposed by computational and theoretical linguists. Some are 'syntax', or form, based; others 'semantics', or meaning, based. 'Conceptual' representations may be distinguished from others in several ways:

- (A) A conceptual representation is 'language-free' -- that is, the same set of units and relations are used to describe meanings which may be encoded in any human language.
- (B) The representations 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'.
- (C) 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.
- (D) The representations are proposed as psychological models of human cognitive structures.

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 [6,7]. It

is not the purpose of this report to give arguments favoring conceptual systems in general or C.D. in particular. Those interested in such matters will find such material in the above references. The rest of this section is devoted to a brief survey of those aspects of C.D. pertinent to the remainder of the report.

(1) EVENTS

ACTS and ACTORS

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 '*ATRAN.*'. For 'John gives' we have the representation:

JOHN <====> *ATRANS*

C.D. CASES

The concepts of 'eating' and 'giving' involve more than just ACTORS and ACTs. One must eat or give 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.

and a SOURCE and GOAL location. The DIRECTIVE case provides slots for these locations, and is symbolized:

```

      O |-----> <GOAL>
      |
<-----|
      |-----< <SOURCE>

```

"John goes to the store"

```

      O                               O |-----> *STORE*
*JOHN* <=> *PTRANS* <----- *JOHN* <-----|
      |                               |-----<

```

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

(2) 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:

```

      VAL
<CONCEPT> <====> <ATTRIBUTE> <-----> <VALUE>

```

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

```

      VAL
*BOOK* <====> *POSS* <-----> *FRED*

```

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. The only SCALES referred to in this paper are *HEALTH* (physical health), *JOY* (mental pleasure), and *ANGER* (emotional anger).

"Socrates is dead"

```

      VAL
*SOCRATES* <====> *HEALTH* <-----> (-10)

```

"Bill is happy"

```

                                VAL
    *BILL* <===> *JOY* <----- (+3)
  
```

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

```

                                VAL
    <CONCEPT> <===| |-----> <ATTRIBUTE> <-----> <new-VALUE>
                                VAL
                                <-----> <old-VALUE>
  
```

or, alternatively,

```

                                VAL
    <CONCEPT> /-----| |-----> <ATTRIBUTE> <-----> <new-VALUE>
                \-----|
                                VAL
                                <-----> <old-VALUE>
  
```

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"

```

                                VAL
    *SOCRATES* <===| |-----> *HEALTH* <----- (-10)
                <----->
  
```

When the change of state is along a scale, it is common that neither the precise initial nor 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"

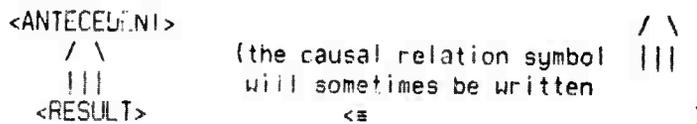
```

                                <-----> *HEALTH*
    *TRUMAN* <===|
                ↑ |-----< *HEALTH*
                INC|
                (-5)
  
```

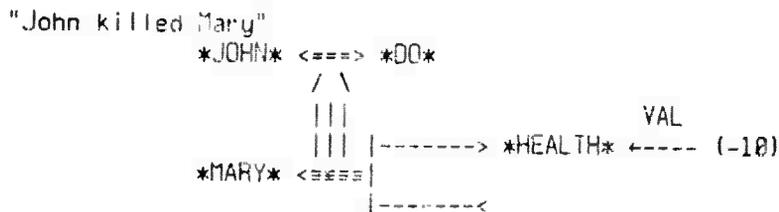
EVENTs, STATEs, and STATE-CHANGEs are all types of relationships which are termed 'conceptualizations'.

(3) CAUSALs and CONJUNCTIONs

Two types of causal relationship will be used in examples. The first is a relation in which the occurrence of an ANTECEDENT conceptualization causes a RESULT conceptualization:

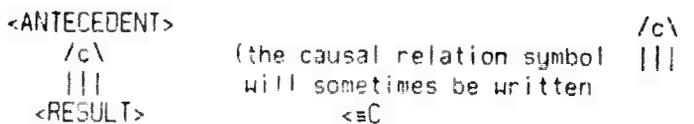


An example of the use of the causal is:



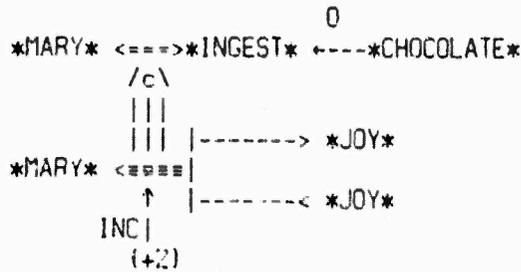
(*DOO* is a 'dummy' ACT used to hold the place of some actual, but unknown, ACT and its required cases.)

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

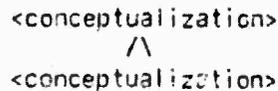


This relation indicates that the occurrence of the ANTECEDENT conceptualization could cause the RESULT conceptualization, but does not indicate the actual occurrence of either.

"Mary likes to eat chocolate"

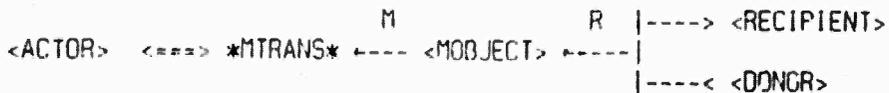


Both CAUSAL relationships are themselves conceptualizations. Furthermore, any two conceptualizations can be joined by the symbol '/\` to form a CONJUNCTION, which is also a conceptualization.



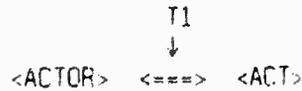
(4) Mental ACTs

Many English verbs -- tell, remember, teach, read -- involve the transfer of information. Conceptual primitives for representing these meanings are discussed in [8]. In this report we shall use only one 'mental' ACT, *MTRANS*. 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.' In this paper we shall limit mental locations to 'conscious processors' (*CP*) and 'long-term memories' (*LTM*) of human beings, and physical objects which in some sense serve as information stores (books, televisions, . . .). The notation for an EVENT using *MTRANS* is:

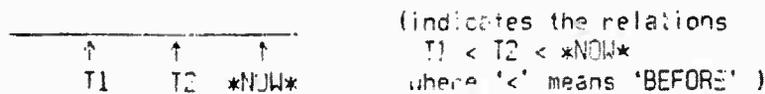


(5) TIMEs and other modifications

Still to be accounted for is the concept of the time of occurrence of an event, which usually is reflected by verbal tensing in language. MARGIE 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:

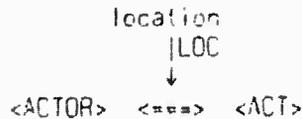


The special symbol *NOW* represents the 'current' time -- i.e., the time of an utterance or, more exactly, the time of creation of a conceptualization. TIME relations will be shown on a time line, left representing PAST; right, FUTURE.

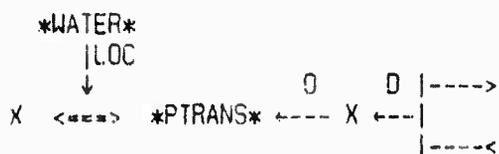


In the implementation, every EVENT, STATE, and STATE-CHANGE has a TIME associated with it. In our diagrams, however, TIME will be left out unless it is relevant to the point being discussed.

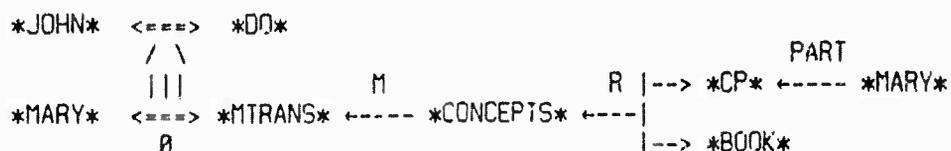
Another modification of EVENTS is LOCATION. It will be represented simply as a modification on the main link of the EVENT:



'Swimming' would be represented by:



Other modifications of conceptualizations permit negation and 'questioning', but these will not be needed for understanding this paper. One that is needed is the MODE 'CANNOT' which can modify an EVENT, and is symbolized by a θ on the <====>.



is the representation provided for "John prevented Mary from reading the book." (*CONCEPTS* is a 'dummy' OBJECT: it represents 'some unspecified conceptual information.')

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".

(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*, as well as such conceptual information as

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

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

(ENGLISH-NAME *JOHN* JOHN)

In Conceptual Dependency, these object-naming units are termed 'PP's (picture producers). Considerable work must still be done on defining the precise nature of these units, both theoretically and computationally. It is expected that future versions of MARGIE will include extremely different handling of nominal references.

III. ANALYSIS OF NATURAL LANGUAGE SENTENCES

The analyzer described below is one that is conceptually oriented not only in the output it produces, but also in the kinds of processes it uses to achieve its answers. Its primary task is not to discover the syntactic relationships in a sentence, but to discover what that sentence is communicating. Syntactic relationships are used to help this process. This distinguishes the analyzer from previous attempts, such as Woods' parser [11], where the semantics was needed to help build the syntactic structures.

For the analyzer, the job of discovering what a sentence is communicating means discovering what Conceptual Dependency network should be generated from that sentence. One source of information used to do this is a simple description of certain relationships between words. But more important than such patterns between word types are the passive features and active expectations that are associated with each word in a language. These expectations look for certain events, certain features or structures, and if these things are found then certain actions are performed.

This emphasis on words rather than on syntactic structures, on content rather than form, is in keeping with the general philosophy of Conceptual Dependency and in contrast to previous linguistic and computational linguistic work. We are interested not in syntactic structures but in those processes that allow people to communicate their thoughts using language.

The features of a word are facts associated either with that

word itself or with the concept referred to by that word. That "John" is a proper name is a fact about the word "John." That "John" is a male human is a fact about the concept referred to by the word "John". Features are represented in the system in the C.D. notation described in section II. They are not special flags or marks built specifically for the analyzer, and though they are used primarily by the analyzer, they are still pieces of world knowledge and are represented like other pieces of world knowledge.

While the features are described with primitives and relationships that are generally used in representing information, the expectations are described with functions and flows of control that are oriented more towards language processing. The basic control structure, involving a set of conditions plus a set of actions to be performed if the conditions occur, is a reasonable mechanism for many other memory processes. Charniak [1] uses a similar device to describe the way sentences tie themselves together in children's stories.

The functions that specify these conditions and actions are ones that have been found useful for analysis. As our knowledge of memory processes increases, some will remain as they are and others will be generalized to do more than language processing. The functions that have been developed fall into several groups.

CONCEPTUAL DEPENDENCY GRAPH MANIPULATORS

These functions create, and change internal counterparts of

conceptual dependency representations. Graph locations, which can be fully specified by strings of conceptual role markers such as "the actor of the caused event", are holders of information. That is, the graph is both the final analysis result and also the source of many of the expectations that are made while analysis is going on.

One function then just takes a string of role markers, e.g. "(\Leftarrow actor)", and returns the conceptual piece found at the end of that path. Another function follows such a path and puts in a conceptual piece. The first function is called CHOICE and the latter CHOOSE. Both of these functions work with a conceptualization. There exist two related conceptualization builders, REPLACE and IMBED. REPLACE replaces the current conceptual graph (which may be empty) with a new one, perhaps built from all or part of the old. This is called mainly when the verb found in an utterance provides a conceptual network tying together the other elements in the sentence, or when some word, like "again", tells the analyzer that the conceptual network from the verb is part of some other network.

IMBED doesn't change the conceptual graph as such but affects how the above functions behave. Basically when IMBED is called with a string of role markers, it causes the conceptualization referenced by CHOICE, CHOOSE, and REPLACE to be moved to the conceptual piece referenced by that string of markers. Suppose the analyzer had so far built a network involving the communication of a causal conceptualization, e.g., "advise" which is the communication of the belief that if the person being told does something he will be

happier for it. Now IMBED would be called with the argument "(OBJECT CON)" to reset the conceptualization to be the action, in the communicated idea, which would cause pleasure. Any further work done by CHOOSE, CHOICE, and REPLACE would be in building up this action. There is of course a function complementary to IMBED called RESET_ALL which resets the conceptualization to be the one which IMBED was called upon. At the moment there is no stacking of these embeddings and a disinclination to do so. Stacking is a mechanism that can be programmed in fairly well-defined ways and it has been the basis of many programs for operating on data bases. However its psychological validity is questionable. At best functions that operate recursively on trees are convenient ways of simulating some human mental processes. In the analyzer however recursion is not a basic mechanism. Hence if the analyzer IMBEDs more than once it will be able to reset only to the most recent embedding or else to the outermost level of the conceptualization. Such an approach is related to the representation we have chosen. Had our system been based on graphs of a more mathematical nature, with a few primitives and lots of trees to represent everything, then embedding would be occurring constantly and the natural way to work with these trees would be with recursive routines. However Conceptual Dependency is oriented about structures where closely related elements of a conceptualization appear together at the same level, where a processor doesn't have to keep looking up and down a tree for information. When the focus of manipulations moves up or down a

level in this kind of format it means something significant, and can be expected to take more effort, and thus be less likely to be as simple a mechanism as recursive stacking.

In talking about functions that add conceptual pieces to the graph one point should be made about these pieces. While on printed output these graphs look like the linear version of C.D. graphs, there is one extra feature about them which doesn't show. Many times a conceptualization will have some piece appearing in several places in a graph. The simplest example is with "give" where we have an ATRANSing with an identity between the actor and the donor. When specifying conceptual pieces to REPLACE we can enforce this identity to the extent that the same graph is pointed to in both places. Not only does it become obvious to other programs, in inference and in generation, when two elements are meant to be the same, but with respect to the building of these structures, it gives the result that any changes made to an element show up in all of its occurrences automatically. Although this is only a small part of it, this ability to do explicit references indicates a representation that can handle the results of more complex reference determination from the memory processor.

As we shall see, often the verb will explicitly provide REPLACE and CHOOSE with the conceptual pieces that it needs. However there are also times when there are significant conceptual structures coming from other words in the sentence. For example, in "John gave Mary a headache," "a headache" is the name of a conceptual structure

involving the feeling of pain, and the analyzer needs to incorporate this structure into one that says "John caused Mary to feel pain in her head." Hence there also exists a routine, called UTILIZE, that takes words that refer to structures and turns them into forms for incorporation with REPLACE.

SYNTACTIC STRUCTURE MANIPULATORS

Another set of functions used is needed to operate on the syntactic structure of a sentence. The description of these functions will be somewhat brief. They have not been the main focus of our effort. This is because much work has already been done on syntactic analysis. Most other approaches, computational and linguistic and even psychological, have been concerned with what could be obtained using just syntax, until it became necessary to add on a little semantics to help out. The approach here is the exact opposite, to see what can be done from the conceptual side and include syntactics when they become important. The first form of the analyzer didn't even have word order. Not even taking into account all the arguments that have been made in favor of semantics over syntactics, it would seem that this attack on the problem has interest in that it does relegate syntax to a truly subordinate position.

The syntactics used by the analyzer are quite simple. This is partly because less time has been spent on them and partly because the existence of a conceptual network means the syntax doesn't have

to carry the semantic load that it does in a syntactically based system.

There are three surface cases used, SUBJ, OBJ, and RECIP, which save places for items until they can be given conceptual roles to play. These roles are primarily determined by word order, with a secondary distinction between humans and objects, so that RECIP is generally a human, if it occurs at all. When embedding occurs these cases are saved as well, and reset by RESET_ALL with the same comments about stacking applying. Further, CHOICE and CHOOSE both know how to handle these cases, and the analyzer can add and extract information from them just as with the conceptualization.

These word order cases are supplemented by the use of prepositional markers. The analyzer usually knows what relationship a preposition is expressing either from what has already been understood or from the nature of the object of the preposition. The verb, which plays a central role in this system, usually does most of the work in giving an expected meaning to the use of a preposition. Still, the analyzer needs to save the fact that such and such item was governed by such and such preposition, particularly when prepositions introduce a sentence ("By the car was a...") and when backup routines are called.

There is another place where simple syntactic action occurs: while waiting for the accumulation of enough information to make a conceptual representation. This happens in the building of noun phrases. Starting with the recognition of an article or adjective,

words as they are brought in are not converted into a unified conceptualization until something is seen that indicates the noun phrase is ended. The end of the sentence, a verb, or the start of a new noun phrase are some of these signals. Knowing what the main item is that is being modified by the previous string of adjectives and nouns the analyzer can make a conceptual whole. But many adjectives used commonly like "short" or "sweet" cannot be said to have meaning until they have something to modify. Granted there may be things that seem common between "a short stick" and "a short pause", between "a sweet candy" and "a sweet voice", but these common elements are too vague to be sufficient to be definitions for the adjectives. That is, given some such unifying theme, we still couldn't predict reliably what modification the adjective meant with many nouns. There are times when we generalize word usages, when metaphors are involved, but for the moment we are concerned with the common, ingrained uses of words. Hence we find ourselves here with fairly ambiguous words, i.e. the adjectives, and the major source of information on what to do with these words coming last. There is also the complicating factor of noun pairs, such as "kitchen table" and "police state". There exists a program by Sylvia Weber Russell (5) that handles a number of these, and eventually it will be tied in with the analyzer.

There are, then, two functions for handling noun phrases. One takes new words and collects them into a simple list, waiting for the end of the phrase. The other is called when the phrase end is noted

and converts this list into a normal conceptual structure. This new structure is then returned as the meaning of the noun phrase and behaves as a unit for such functions as CHOOSE and FEATURE.

MEMORY INTERFACE FUNCTIONS

FEATURE brings us to another open-ended set of functions, which interrogate the memory's world knowledge for information about things. These things may be either words or concepts. FEATURE is the only memory interrogation function currently used by the analyzer (other possibilities are the class III predicates used by generator, as discussed in section IV). It takes as one argument either a word or a simple conceptual piece consisting of a PP plus modifying conceptualizations and as the other argument some property value, such as "human" or "proper" (as in proper nouns). These property values belong to what are called contrast sets, such as "(human, animal, physical object)". These contrast sets are needed because there are often times when the analysis depends on which element of the set a particular word or concept is associated with. It is important to note that these contrast sets are not hierarchical, at least to a great degree. Although Mary being a human implies that she is an animal which implies she is a physical object, the way in which "Mary" is handled in language differs depending on whether she is no more than an object or no more than an animal. FEATURE is a very simple information retrieval function. A particular complex of features has been chosen for some reason and FEATURE is used to find

out what else is true in this complex. Thus, by a criterion of commonness, "John" is chosen as referring to "JOHN1" which is "the man called John" sense of "John". FEATURE then tells us that "JOHN1" is a man, and that an English name is involved.

There are, as mentioned, other functions in the analyzer, but they are subservient to the ones discussed above. Only one more piece of the analyzer needs to be described before some examples are given. This piece is the monitor, or supervisor, the piece that takes definitions of words in terms of these functions and executes their instructions. This monitor is, and is meant to be, very simple. Its job is to do bookkeeping on the following variables.

SENTENCE - this is the utterance being analyzed. It is constant through the analysis.

WORD - this is the current word in the sentence that is being looked at. Normally WORD is set to each successive word in SENTENCE, going from left to right.

SENSE - this is the current sense, or meaning, that is being worked with. It is usually either the meaning of WORD or of the noun phrase containing WORD. A sense of a word is a name for a set of requests and features. Features are simply conceptualizations. Requests are analysis instructions.

REQUESTS - this is a list of requests which is unordered with one exception. The monitor continually rechecks this list to see if changes to WORD, SENSE, CONCEPT, or REQUESTS itself have caused any of the requests to become applicable. Requests are representations

of the expectations a word sets up of situations that might occur and actions to take in those cases. The unordered rechecking is meant to be a simulation of a parallel control structure where each request looks to see if it should do anything, independent of the other requests. The only exception to this concerns those requests that are activated when some phrase or clause ends. For example, in "John wanted Mary..." the analyzer assumes that "Mary" is beginning a clause involving something involving Mary that John wants. If instead that is the whole sentence and nothing more has been found out when the end of the sentence is reached then a default assumption is made that John wants Mary to come to him. These requests that are called by the end of something are always placed at the end of the request list. This is equivalent to considering them as independent processes that, in being called by the absence rather than the presence of something, wait to make sure that "more real" requests have had their say.

ANSWER - this is the conceptual representation of SENTENCE that the analyzer is building. It is the variable whose value is returned by the analyzer.

CONCEPT - this is a pointer to either ANSWER or to some subconceptualization in ANSWER. This points to the place where the building activity is going on at any point in the analysis. Thus it starts off the same as ANSWER but when an embedded conceptualization is being built it points to that instead.

Attached to each word that appears in SENTENCE are one or more

senses, that is, labels of sets of features and requests. Requests are of the form "(TEST ACTION FLAG)". TEST and ACTION are the crucial elements of a request. TEST is a (Lisp) predicate and ACTION is a (Lisp) function, both built from Lisp functions and those functions that have been described above. When WORD changes, the monitor first checks REQUESTS for instructions, adds any requests attached to WORD, then finds the current sense for WORD (setting SENSE equal to it), then checks REQUESTS again, then adds the requests that are part of SENSE to REQUESTS and steps WORD along in SENTENCE. In general, TEST predicates make reference only to CONCEPT and the feature aspects of WORD and SENSE. Checking a request means evaluating the TEST. If TEST is not true nothing happens and the monitor goes on to the next request. If TEST is true, then ACTION is executed and FLAG is altered. FLAG is a bookkeeping mark. When it is NIL it means the request has not been used yet, while T means that the request has already been used.

REQUESTS is changed by either the monitor or an ACTION. In the former case words and senses have their requests added to the list. In the latter case, either the function IMBED which introduces clauses, or a simpler one for starting prepositional phrases, saves the current REQUESTS and replaces it with another set. RESET_ALL restores REQUESTS to the original set when it is called. IMBED thus works with three information sets: the conceptualization being built, the syntactic structure being built, and the expectations being made.

One other operation that the monitor performs is to initialize

REQUESTS to a request which looks for any noun phrase that will be the subject. This is done whenever a new sentence is begun.

The best way to describe how these functions are put together to form requests is by examples. The first example will be straightforward, the second will show how words, like "give", can work to tie together the contents of other words, and the last example will show how words, like "by", can work to tie together large conceptual structures.

The first example is "John advised Mary to drink the wine." The requests attached to the words in the example are:

advised - (T (CHOOSE TIME (BEFORE (NEW_TIME) (CHOICE TIME) X) NIL)

drink - ((NEED_TIME) (CHOOSE TIME (CHOICE TIME)) NIL);

Most of these functions haven't been discussed and the requests are here for completeness. Basically if words have requests at all they are ones like time choices. The above requests say that "advised" always refers to a past event and "drink", if a time is needed, refers to a present one. However the "to" that is set by "advise" sets "(NEED_TIME)" to false, so the time will be untouched by "drink". Past, present and future mean before, during and after the time of the surrounding conceptualization, respectively.

The senses for the words in this sentence are, for programming convenience, usually the same as the word with a numeric suffix attached. The requests attached to the senses that appear in this sentence are:

JOHN1, MARY1: none;

```

ADVISE1: (T (REPLACE CONCEPT (QUOTE ((ACTOR (# SUBJ) <=> (*MTRANS*)
TO (*CP* PART (# RECIP) REF (*THE*))
FROM (*CP* PART (# SUBJ) REF (*THE*)) MOBJECT
((CON (NIL TIME (>)) MODE (NIL)) <=>C
((ACTOR (# RECIP) <=>T (*JOY*) <=>F (*JOY*)) INC (2)
TIME (= MOBJECT CON TIME) MODE (NIL)))))) FOCUS ((ACTOR)
MODE (NIL) TIME (NIL)))) NIL)

```

This request produces a conceptual form equivalent to

```

          |----> ( RECIP )
          |
          M   R
( SUBJ ) <=> *MTRANS* --- (#) --- |
          |
          |----< ( SUBJ )

```

where # is the following conceptual form:

```

      ( )
      /c\
      ||| |--> *JOY*
( RECIP ) <==||
          ↑ |--< *JOY*
      INC|
      (+2)

```

that is, someone is being told that doing something will please him.

```
(T (DEFPROP TO TO0 CURRENT) NIL)
```

This request makes a prediction about future use of "to".

```
((FEATURE SENSE (QUOTE HUMAN)) (CHOOSE RECIP SENSE) NIL)
```

This request says that the next human is the person receiving the MTRANSing. It could be written so as to put the human into the graph directly but it is stored in RECIP just in case a parallel syntactic structure is needed.

```

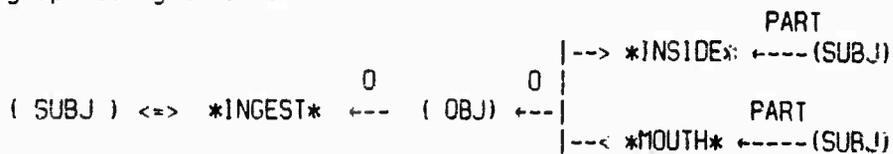
TO0: (T (PROG NIL (IMBED (MOBJECT CON) ((SUBJ CHOICE RECIP)
(TIME AFTER (NEW_TIME) (CHOICE TIME) X)) ((BREAK_POINT)
(RESET_ALL) NIL)) (SETQ USE_TIME NIL )) NIL)

```

This says that when "to" is found start the conceptual building at CON in the MOBJECT, set the time to be after the communication, set the SUBJ of the infinitive verb to be the person being advised, and set REQUESTS to one looking for the end of the clause.

```
DRINK1: (T (REPLACE CONCEPT (QUOTE ((ACTOR (# SUBJ) <=> (*INGEST*)
OBJECT (# OBJ) TO (*INSIDE* PART (# SUBJ)) FROM
(*MOUTH* PART (# SUBJ)) ) MODE (NIL) TIME (NIL)))) NIL)
```

graphically this is:



another request with DRINK1 is:

```
((FEATURE SENSE (QUOTE PP)) (CHOOSE OBJ SENSE) NIL)
```

This request says that the next object it finds is the thing being drunk.

The features of the words are:

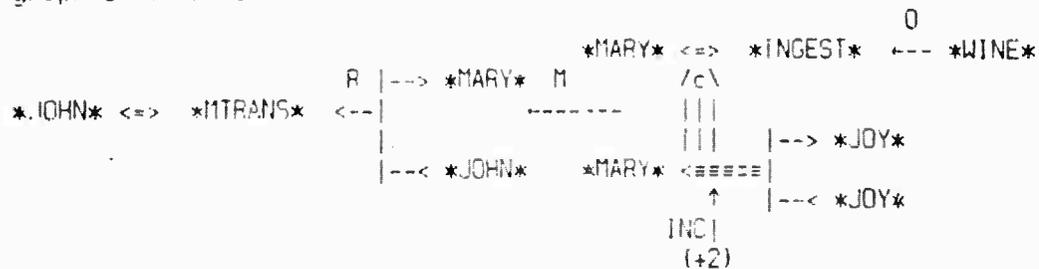
John, Mary - in a contrast set called *WORDTYPE* they have the value *NAME*, which means they don't require an article;

JOHN1, MARY1 - both have the feature "HUMAN", and JOHN1 also has the feature "MALE" versus MARY1's "FEMALE" but that is not needed here. Both also have the feature "PP";

WINE1 - has the feature "PP", which is the only one needed here.

The analysis of the sentence "John advised Mary to drink the wine" proceeds simply enough. The initial request looking for a

subject (SUBJ) is satisfied by "John". "Advised" and "ADVISE1" satisfy no requests but add their own to the set, and further change CONCEPT (and hence ANSWER) to a conceptual skeleton of the MTRANS action. "Mary" and "MARY1" satisfy the request looking for a recipient of the MTRANS. "To" and "TO1" move CONCEPT to point to the conceptualization being MTRANSed and reset REQUESTS. "John" and "JOHN1" satisfy the request now being made for a SUBJ. "Drink" and "DRINK1" put the conceptual skeleton for a drinking action into head of the causal in the MOBJECT slot. "Wine" and "WINE1" satisfy the request looking for an OBJ of the drinking. The end of the sentence causes REQUESTS and the syntactic cases and CONCEPT to be returned to the values they had before "to" was encountered. REQUESTS is checked again, and then the analysis is over. The value of ANSWER (in graphic form) is:



Because of space, the fact that the recipient case of MTRANS involves the Conscious Processors (CPs) of the people, not the people themselves, is not shown in this diagram. Also, the times have been left out.

The second example is "John gave Mary a beating." The focus here is on the way in which "give" is used mainly to pull together

This says that when "to" is found start the conceptual building at CON in the MOBJECT, set the time to be after the communication, set the SUBJ of the infinitive verb to be the person being advised, and set REQUESTS to one looking for the end of the clause.

```
DRINK1: (T (REPLACE CONCEPT (QUOTE ((ACTOP (# SUBJ) <=> (*INGEST*)
OBJECT (# OBJ) TO (*INSIDE* PART (# SUBJ)) FROM
(*MOUTH* PART (# SUBJ)) ) MODE (NIL) TIME (NIL)))) NIL)
```

graphically this is:

```

                                PART
                                |---> *INSIDE* <---- (SUBJ)
                                0   0 |
( SUBJ ) <=> *INGEST* <--- ( OBJ ) <---|
                                |---< *MOUTH* <----- (SUBJ)
                                PART
```

another request with DRINK1 is:

```
((FEATURE SENSE (QUOTE PP)) (CHOOSE OBJ SENSE) NIL)
```

This request says that the next object it finds is the thing being drunk.

The features of the words are:

John, Mary - in a contrast set called *WORDTYPE* they have the value *NAME*, which means they don't require an article;

JOHN1, MARY1 - both have the feature "HUMAN", and JOHN1 also has the feature "MALE" versus MARY1's "FEMALE" but that is not needed here. Both also have the feature "PP":

WINE1 - has the feature "PP", which is the only one needed here.

The analysis of the sentence "John advised Mary to drink the wine" proceeds simply enough. The initial request looking for a

the other elements of the sentence to yield a meaning paraphrasable in English as "John beat Mary." This is a very common use of "give" and there are many other words that can function the same way. For example "John took a walk" means the same as "John walked for a while," and "John got Mary a job" is related to "John gave Mary a job." In all these examples the object is the name of some action or situation, and "give", "get" and "take" take these situations and apply them in specific ways to the other elements they govern.

Some of the requests associated with "GIVE1", which is the sense of "give" that handles the above example, plus ones like "John gave Mary a headache," and "John gave Mary a book," are like the ones described in the previous example. Thus:

```
GIVE1: (T (DEFPROP TO TO, CURRENT) NIL)
        ((FEATURE SENSE HUMAN) (CHOOSE RECIP SENSE) NIL)
        ((FEATURE SENSE POBJ) (CHOOSE OBJ SENSE) NIL)
        (T (REPLACE CONCEPT (QUOTE ((ACTOR (# SUBJ) <=> (*ATRANS*)
        TO (# RECIP) FROM (# SUBJ) OBJECT (# OBJ)) FOCUS ((ACTOR))
        TIME (NIL) MODE (NIL)))) NIL)
```

graphically this last is:

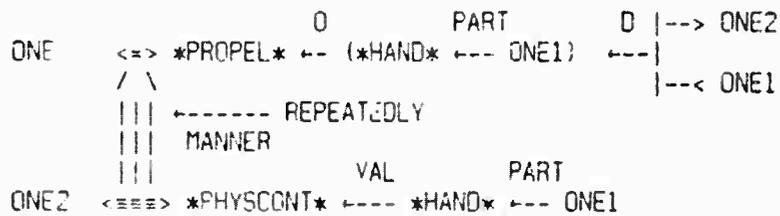
```

                O           R |--> ( RECIP )
( SUBJ ) <=> *ATRANS* --- ( OBJ ) ---|
                                   |---< ( SUBJ )
```

but the important request for the example (which is paraphrased here in English for readability) is this one:

TEST if SENSE is a conceptualization of the form
 one1 \rightarrow do
 / \
 |||
 one2 \leftarrow state
 (that is, someone puts someone into a state) then
 ACTION REPLACE CONCEPT (with UTILIZE) with SENSE where
 one1 is replaced by SUBJ and one2 by RECIP

The sense of "beating" that is assumed here has the following meaning (graphically):



This is the representation for repeated hitting. The analysis returned for "John gave Mary a beating" will look like this except that "JOHN1" will appear everywhere that "ONE1" does and "MARY1" everywhere that "ONE2" does.

The next example is "John killed Mary by choking Mary." It contrasts with the first example in the kind of manipulation that occurs. In "John gave Mary a beating," the meaning of "give" was a set of actions, not some conceptual piece. The actions built a conceptualization from the other words in the sentence. In this example, "John killed Mary by choking Mary," the word "by" ties together two large conceptual pieces, "John killed Mary" and "John choked Mary." "By" asks questions about conceptualizations rather than about words and differs from "give" in that way. To kill

someone means to do something to make that someone die. To choke someone means to grasp his (or her) neck causing him to be unable to breathe.

"BY1", the name assigned to this use of "by", has the following job to do. It has to tie together two conceptualizations, making one "instrumental" in the occurrence of the other. If the two actions are simple EVENTS, then the main act has the other in its INSTRUMENTAL case. (Any ACT can take an INSTRUMENTAL case, which must always be filled by an entire EVENT. This INSTRUMENT further specifies the nature of the ACT on which it is dependent.) If the main action is a causal and the causing action is unspecified (graphically there is a dummy "do" written for the act) then the secondary action is helping to specify this causing act. If the secondary action is a simple act then it is a straightforward replacement of this act in the unspecified slot. This happens in "John angered Mary by giving Bill the book." If the secondary action is a causal itself then the result event of this secondary action is in turn the antecedent event of the main action. This happens in our example.

"BY1" also has a few other duties, like preparing the analyzer for an "ing" form of a verb, and making the current subject the new subject of the "by" clause. An English paraphrase of the request set for "BY1" (which involves too many of those format functions mentioned before to be usefully written out here) reads like this:

```

TEST if CONCEPT is a causal then
ACTION REPLACE CONCEPT with ((CON (NIL) ^ CONCEPT)) , that is,
      form a space for the secondary concept,
and also add the following request:
  TEST until the secondary concept has been found then
  ACTION if the old concept had an unspecified causing action
          then if the secondary is also a causal then
                replace the unspecified action with the
                result event of the secondary, else
                CONCEPT is REPLACEd by the whole secondary action
                causing the result event of the main action,
          else CONCEPT is left the conjunction of two events
and also (IMBED CON ((SUBJ CHOICE SUBJ))
          ((BREAK_POINT) (RESET_ALL) NIL))

```

The last action, the IMBED, says that from this point on conceptual building will be done in the CON space just attached with the REPLACE, and it also says that REQUESTS will be set to look for the end of the clause.

```

TEST if CONCEPT is a simple act then
ACTION (IMBED INST ((SUBJ CHOICE SUBJ)) ((BREAK_POINT) (RESET_ALL) NIL))

```

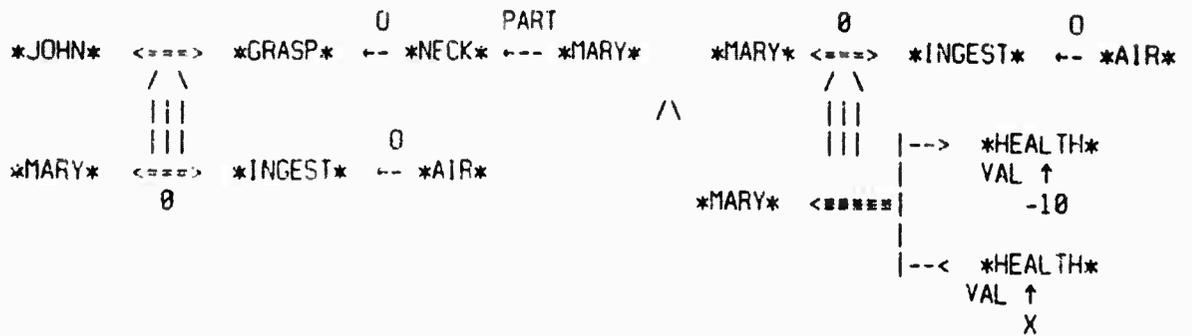
This request is applicable in the simple instrumental use of "by", such as "John gave Mary the book by handing it to her." The last request looks for an "ing" form of a verb to follow the "by".

```

TEST if WORD is a word with suffix ING
ACTION then give it the meaning of the root

```

When this sense of "by", which is set by "KILL1", is used the final analysis of "John killed Mary by choking Mary" is:



We have seen some examples now of the core of the conceptual analyzer. We have seen the kinds of functions that are used and the kinds of results that are constructed. The discussion has been brief and the analyzer described is far from complete. However it can be seen that the basic philosophy of Conceptual Dependency has been continued here. Not only has the stress been on a conceptual rather than just a language oriented semantic output, but the same criterion of naturalness that leads to one representation rather than another has been used in deciding what decisions cause what steps in the analysis process. The assumption implicit in the requests for 'give', that humans are consistently treated differently from physical objects, is such a decision. The control structure itself was worked out from an assumption that natural language processing does not involve global routines that are based on syntactic structures, but rather such processing is carried out by short programs and expectations associated with the words of the language. The idea of sets of requests was a straightforward implementation of this assumption.

In the last few examples, the how of analysis will be omitted and only the input and output will be shown.

JOHN AGGRAVATED MARY BY GIVING BILL THE BOOK

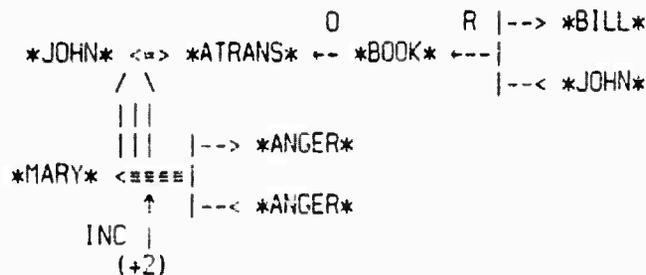
TIM00 : ((VAL *T*))

TIM01 : ((BEFORE TIM00 X))

TIM02 : ((BEFORE TIM01 X))

((CON ((ACTOR (JOHN1) <=> (*ATRANS*) TO (BILL1) FROM (JOHN1) OBJECT (BOOK1 REF (*THE*))) FOCUS ((ACTOR)) MODE (NIL) TIME (TIM02)) <=> ((ACTOR (MARY1) <=>T (*ANGER*) <=>F (*ANGER*)) TIME (TIM01) INC (2))))

This is the internal representation of the following graph structure:



The next two examples show how much concepts with different features can affect the analysis of that sentence. The analyzer assumes that when someone wants someone else, he wants that person to come to him, but when he wants some physical object, he wants to have that object.

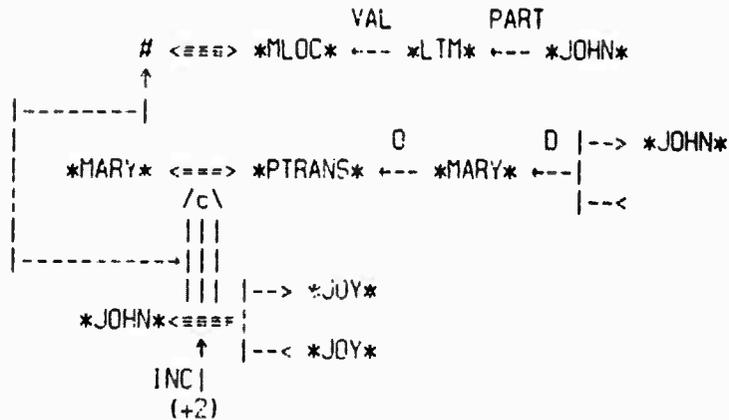
JOHN WANTS MARY

TIM00 : ((VAL *T*))

TIM01 : ((AFTER TIM00 X))

TIM02 : ((AFTER TIM00 X))

((CON ((CON ((ACTOR (MARY1) <=> (*PTRANS*) OBJECT (MARY1) TO (JOHN1) FROM (NIL)) TIME (TIM02)) <=>C ((ACTOR (JOHN1) <=>T (*JOY*) <=>F (*JOY*)) INC (2) TIME (TIM01)))) <=> (*MLOC* VAL (*LTM* PART (JOHN1) REF (*THE*))))) MODE (NIL) FOCUS ((<=> VAL PART)) TIME (TIM00))



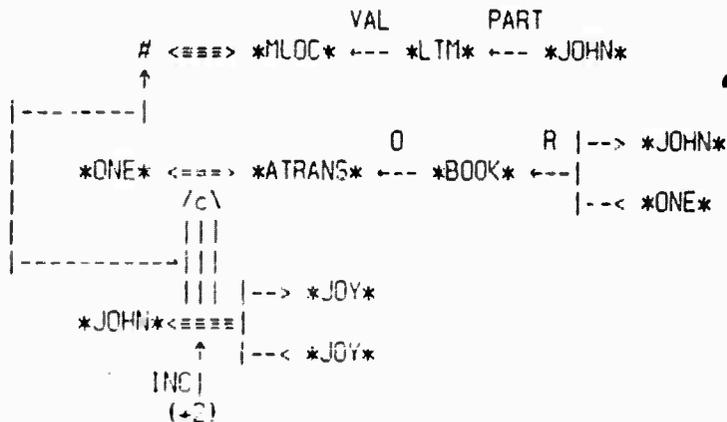
JOHN WANTS A BOOK

TIM00 : ((VAL *T*))

TIM01 : ((AFTER TIM00 X))

TIM02 : ((AFTER TIM00 X))

((CON ((CON ((ACTOR (*ONE*) <=> (*ATRANS*) OBJECT (BOOK1 REF (*A*))
TO (JOHN1) FROM (*ONE*)) TIME (TIM02)) <=> ((ACTOR (JOHN1) <=>T (*JOY*)
<=>F (*JOY*)) INC (2) TIME (TIM01)))) <=> (*MLOC* VAL (*LTM* PART
(JOHN1) REF (*THE*)))) MODE (NIL) FOCUS ((<=> VAL PART)) TIME (TIM00))



IV. PRODUCTION OF NATURAL LANGUAGE SENTENCES FROM CONCEPTUAL REPRESENTATIONS

When the analysis of a source sentence has been completed, control passes to the memory model, which integrates the conceptual structure produced by the analyzer into the existing memory. Neither the source sentence nor the words comprising it are used in the remainder of the paraphrase production. Memory processes are not described in this report, but are discussed in [9]. We can think of the integration process as one of tying references to already known items to internal nodes which represent these items, and of creating new nodes to represent new items and concepts. Thus an associative memory is maintained, in which a node representing the individual 'John Smith' has pointers emanating from it to every conceptualization in which 'John Smith' plays a part.

The problem remaining is to take the conceptual representation which the analyzer produced for the input and find an English sentence which expresses the meaning represented. The words and syntax of the original sentence have been discarded. Thus the fact that the conceptualization was produced from an English sentence which was the input to a paraphrase program is not relevant to this problem. The conceptualization could just as well have come from a German input in a machine translation task, through a chain of deductions in a question answerer, or through some information gathering motivation in an interviewing program. What is required then is a routine which can take an arbitrary conceptual

representation and realize it in English -- i.e., a general Conceptual-English generation program.

Both linguists and computer scientists have designed systems for language generation. These can generally be classified as either random or directed. Random systems [12] attempt to produce grammatical English sentences, starting only with the goal "produce a grammatical English sentence." Such systems can be used to test syntactic theories, and could be used to test semantic theories as well if the goal were "grammatical AND meaningful".

Directed systems posit some underlying structure and have as a goal "produce a sentence having the specified underlying structure." Our goal, to "produce grammatical English sentences with a specified meaning", certainly falls under this paradigm. Unfortunately, the language free aspect of conceptual representation renders approaches which have been previously tried inapplicable to our task. Some of the directed approaches [3] assume a syntactic underlying structure. MARGIE does not know the syntax of the desired output. Others assume a semantic structure [10]; these specify the desired meaning, but do so in terms of linguistic units (word senses) not present in conceptualizations. Thus MARGIE requires a new approach to generation.

The task of producing an English sentence from a language free meaning structure is indeed very complex, but several subtasks may be identified:

- i) Words must be chosen to use in the sentence.
- 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.

Although it may not be necessary to organize these subtasks sequentially, it seems that if (i) and (ii) could be accomplished, then (iii) could make use of the generative mechanisms devised for directed non-conceptual generators.

What MARGIE does in fact is to break up generation into two distinct phases. First a 'syntax network' is created (steps (i) and (ii) occurring in parallel), then a grammar produces an English sentence from the network. The remainder of this section is devoted to describing these syntax networks and how conceptual and linguistic knowledge are used in their formation.

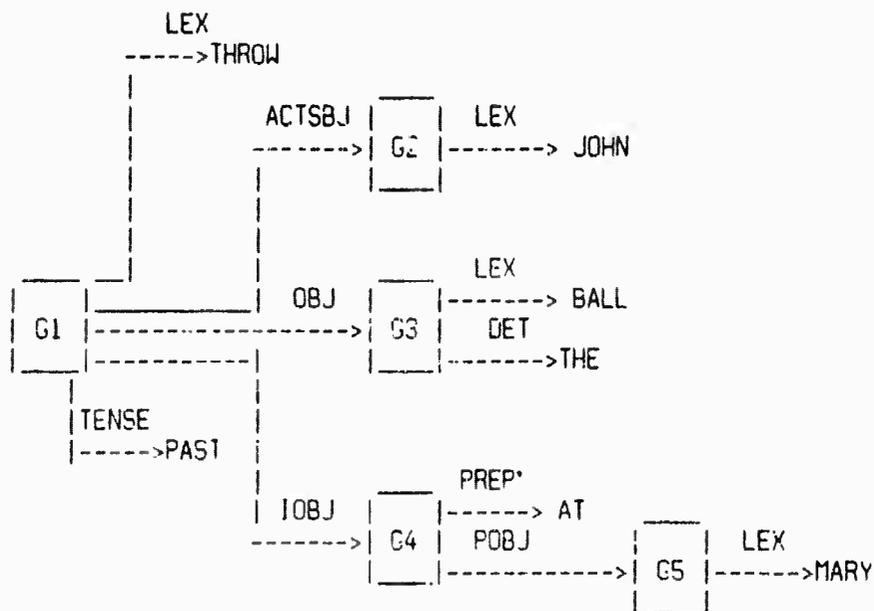
The reader may wonder how MARGIE, having thrown away the source sentence, can be sure that the English realization it arrives at will be a true paraphrase rather than the 'identity' paraphrase. The answer is that it cannot; the generator actually produces several different realizations from the conceptualization. The original sentence may well be among these.

First the process by which a single English realization is obtained will be described; this description will be augmented later to explain the production of multiple realizations (i.e., paraphrases) from a given conceptualization.

Unlike the conceptual representation, the syntax network is very much language dependent: both the tokens and relations in the network are English specific. Structurally the network is identical to the semantic networks of Simmons [10]. Unlike Simmons' nets, however, these syntax networks will connect lexical entries with syntactic relations. The syntactic network from which

"John threw the ball at Mary"

would be generated is



This same network can be more concisely written as:

G1:	LEX	THROW	G3:	LEX	BALL
	ACTSBJ	G2		DET	THE
	OBJ	G3	G4:	PREP	AT
	IOBJ	G4		POBJ	G5
	TENSE	PAST			
G2:	LEX	JOHN	G5:	LEX	MARY

The elements which are objects of LEX relations are lexical entries. The lexical entry THROW will contain only morphological

information such as PAST=THREW. THROW is NOT a word sense -- the same lexical entry serves for "throw a boxing match", "throw a tantrum", and "throw a ball".

The relations may seem to be merely renamings of the relations used by Simmons or Fillmore [2], but that they are not can be seen from further examples. IOBJ (indirect object) serves for the "at Mary" in the above example, as well as for the "to Mary" in "John gave the book to Mary" and the "from Mary" in "John bought the book from Mary". Semantic systems would tend to break down these IOBJ relationships into SOURCE, GOAL, and other relations.

On the other hand, the syntax network may make distinctions which a semantic network would not. Networks for the two sentences

- (1) "John wants Mary to sell him her Chevy"
- (2) "John hopes Mary will sell him her Chevy"

both contain an embedded structure representing

"Mary sell John Mary's Chevy; tense=future "

This embedded structure would be placed in the same relation to 'want' and 'hope' by most semantic models. They are placed in different relations in our syntactic networks because of the necessity of performing an 'infinitive-izing' transformation in (1) but not in (2). Such syntactic information about 'want' and 'hope' will not be processed by the grammar which generates from the syntax nets, but is handled by the routines which create the syntax nets.

(Note that these nets could be subjected to a transformational process as are the syntax trees of a transformational grammar. This

would result in the production of paraphrases of the sort described as 'syntactic' -- e.g., active to passive voice -- in section I. No such transformational process is incorporated in the present program.)

The Production of Syntax Nets from Conceptualizations

To produce a syntax net from a conceptualization, a 'synthesis by analysis' process is undertaken. The conceptualization is analyzed to detect noteworthy patterns in the conceptual syntax and noteworthy relations in the conceptual semantics. 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. For instance, in generating English from:

		O		D	----->	*INSIDE*	PART	
(C1)	*JOHN*	<====>	*INGEST*	-----	*MILK*	-----		*JOHN*
					-----<	*MOUTH*	PART	
								JOHN

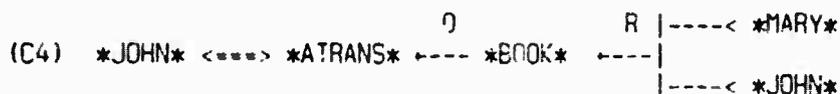
the fact that *MILK* is a FLUID is of interest, since English makes an 'EAT-DRINK' distinction. However, in

		O		D	----->	*INSIDE*	PART	
(C2)	*BEAR*	<====>	*INGEST*	-----	*FISH*	-----		*BEAR*
					-----<	*MOUTH*	PART	
								BEAR

it is not important that BEARS are ANIMALS and not HUMANS. However, to generate a German realization of (C2) the distinction is important, since German makes an 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.)

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.

A portion of a discrimination net which would find the response 'GIVE1' for the stimulus:



is shown in figure 1. (In drawing discrimination nets, root nodes will be assigned index '1'; sons of a node with index N will be assigned indices 2N, 2N+1)

Figure 2 traces the application of the net of figure 1 to stimulus (C4), following the discrimination net algorithm given above.

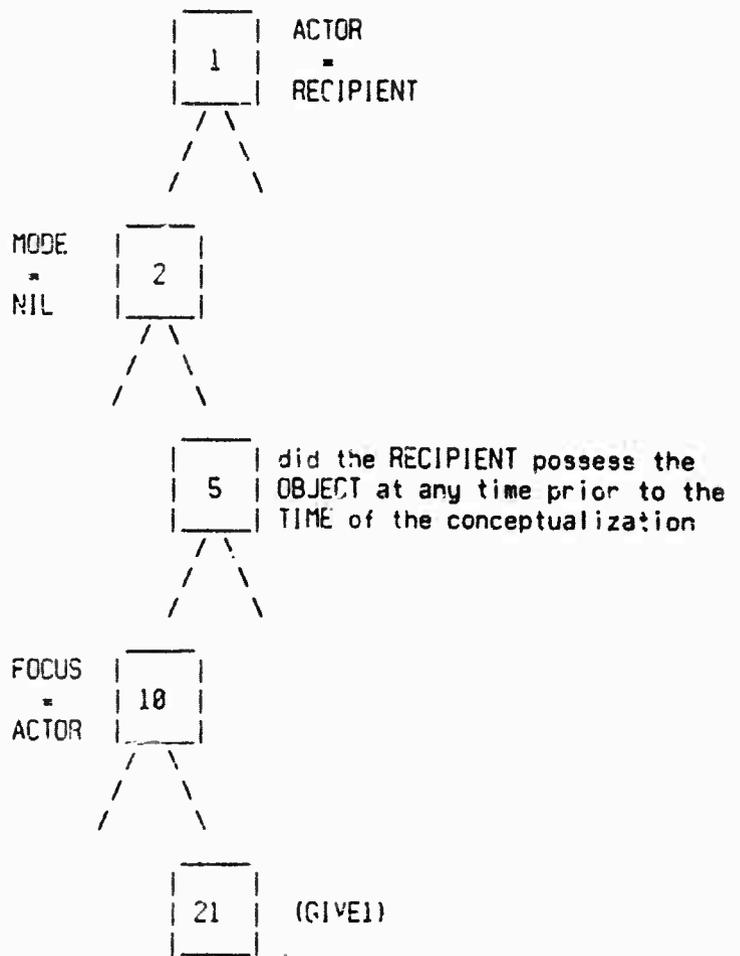


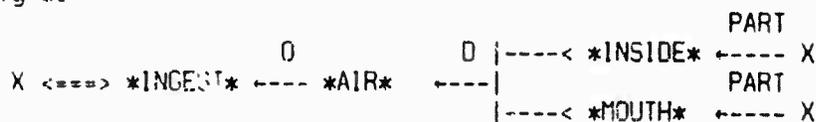
FIGURE 1

CURRENT NODE (CN)	PREDICATE	VALUE	ACTION
1	are the ACTOR and RECIPIENT identical	FALSE	CN←2*CN
2	is there no MODE associated with the stimulus	TRUE	CN←2*CN+1
5	did the recipient possess the object at any time prior to the TIME of the stimulus	FALSE	CN←2*CN
10	is the ACTOR to be 'focussed'	TRUE	CN←2*CN+1
21	none (terminal node)	-----	return GIV

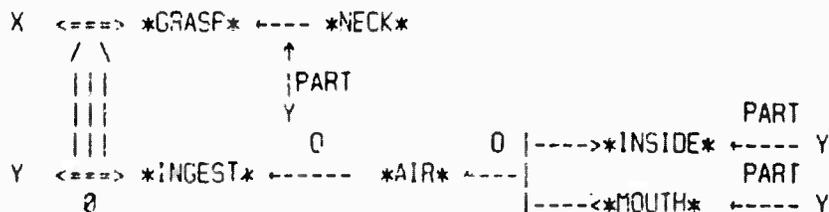
FIGURE 2

The predicates within the discrimination nets may be separated into three distinct classes. Class I predicates are those which perform pattern matching within the stimulus conceptualization. These include tests for the identity of two conceptual fields, e.g., the predicate ACTOR = RECIPIENT at node 1 of figure 1. Other predicates in this class test for the presence of particular conceptual elements in the stimulus -- e.g., is there a \emptyset modifying the RESULT of a conceptualization? -- or test the structure of a stimulus -- e.g., is it of the form EVENT-CAUSE-EVENT?

Class II predicates are logically unnecessary but are included for purposes of keeping the nets compact. They allow a single node in one net to perform an entire set of tests from the same or a different net. An example will clarify the idea behind this. The English verb "to breathe", in its most common sense, is represented conceptually as



while "to choke (someone)" is represented as



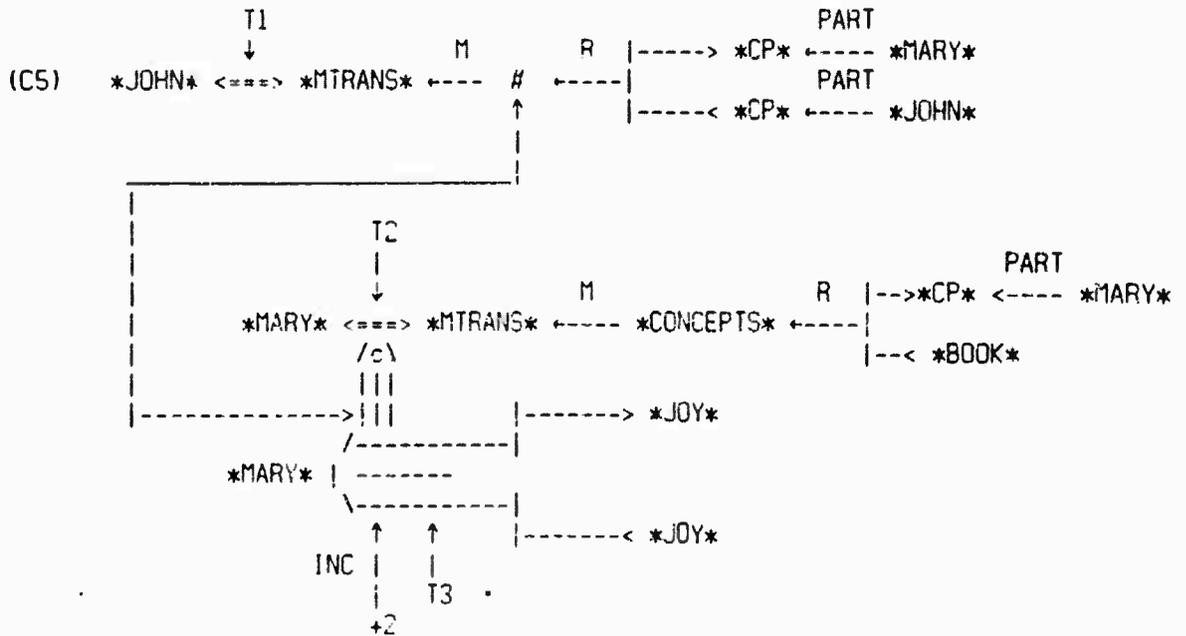
The RESULT in this representation of "choke" is just the representation of "breathe" modified by \emptyset . Rather than repeat the tests necessary for recognizing 'BREATHE1' on the path leading to 'CHOKE1', the predicate

POT_HEAD (<#) = BREATHE1

is evaluated at some node of the discrimination net. Evaluation of this predicate consists of testing whether the structure found in the (<#) slot (i.e., the RESULT) of the stimulus could, if used as a stimulus, evoke the 'response' BREATHE1 (i.e., whether BREATHE1 is a POTential 'HEAD' of a syntax net for this structure). The exact relationship between the 'responses' in the discrimination nets and the production of the syntax nets will be explained shortly. In this case the savings obtained are not considerable, since 'BREATHE1' does not require a large set of tests for its characterization. In other cases, however, considerable storage savings result from this form of recursion in the discrimination process. The price paid for the savings is, of course, extra processing time, since the discrimination net which recognizes 'BREATHE1' may make unnecessary tests in doing so.

Class III predicates test properties which are 'semantic' in nature. They all involve interaction with the memory model. 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*.

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 the generator, such relational information is not included unless it is specifically desired that it be expressed. However, linguistic choices may be dependent on this information. For example,



can be realised as

" John (tell + tense) Mary she (like + tense) reading the book"

However, if the generator finds out (by asking the memory model) that

- 1) T1 is prior to *NOW*, and
- 2) T1 is prior to T2.

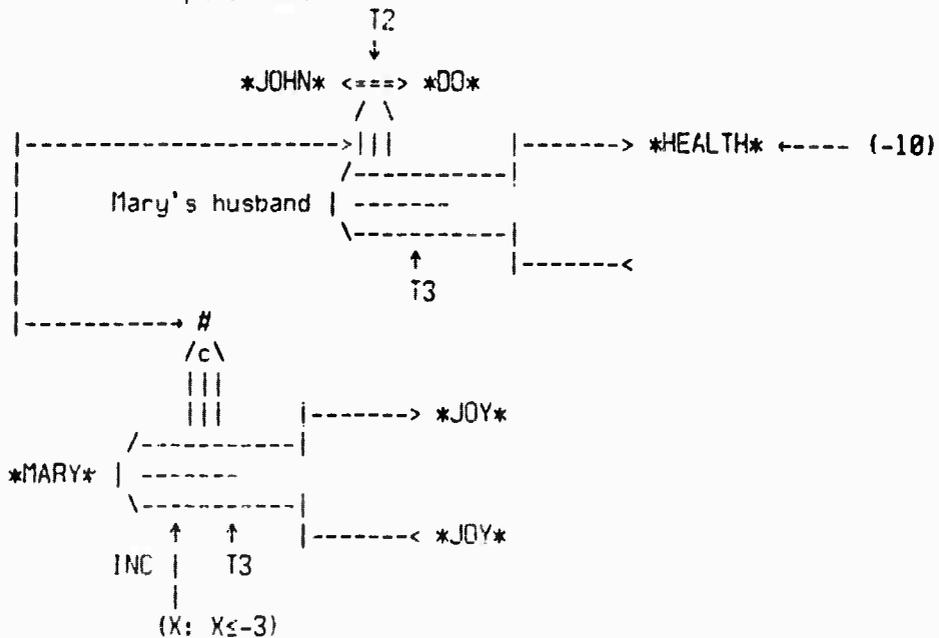
then the realization

"John advised Mary to read the book"

is possible. If, however, T2 is prior to T1, then the sentence

"John told Mary she would have enjoyed reading the book"

In order to choose between 'tell', 'threaten', and 'promise' BABEL uses a predicate MEM-QUERY. The distinction is made on the basis of whether the OBJECT of the *MTRANS* could cause the RECIPIENT of the *MTRANS* to become much less happy (or much happier). A conceptualization:



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.

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 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 not specific to a paraphrase program, nor even to the subtask of language generation in general. A psychiatric interviewing program, for example, 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 analysis has nothing to do with the program's expressing in English the fact that what was said was a threat. 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.

It is interesting to note how small changes in some conceptual roles may have large effects in the linguistic realization of conceptualizations. The time relations in (C5) were one example. But not only the time relations are required for reading (C5) as 'advise'; the identity of the several instances of *MARY* is also necessary. Suppose that the *MARY* in the STATE-CHANGE of (C5) were

changed to *JOHN* -- i.e., we had the 'meaning' "John tell Mary that Mary read the book can cause John become happier." No longer can this be realised as advise, regardless of time relations. But if the time relations necessary to get 'advise' from (CS) still hold, the new meaning could yield the reading 'request' or 'ask':

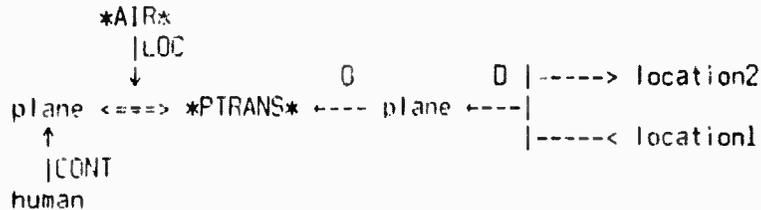
"John requested that Mary read the book"

It is the job of the discrimination nets, employing the three types of predicates provided, to make the subtle distinctions required for the selection of words.

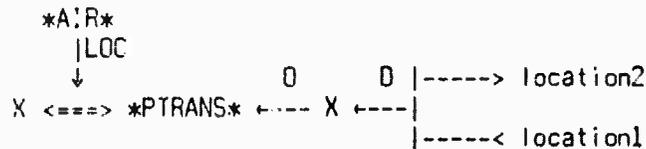
The core of BABEL is a collection of discrimination nets utilizing these kinds of predicates. Given a conceptualization, it is first necessary to decide which nets are applicable. Of course, all of them could be tried, most of them failing to find any response. For efficiency, a quick structural analysis is performed to determine the set of applicable nets. For example, a stimulus with the structure EVENT-CAUSE-STATE CHANGE will be found to have two relevant nets, EKC, which is specifically for EVENT-CAUSE-STATECHANGE structures, and KAUS, which applies to all CAUSAL structures.

Each of the discrimination nets found is applied to the stimulus until a 'response' is found. If all trees are applied without a response being found, BABEL gives up trying to express the conceptualization. If a response is found, it will be a unit called a CONCEXICON pointer. As shown in Figures 1 and 2 above, the CONCEXICON pointer GIVE1 may be found as a response to the stimulus:

FLY2



FLY3



It is the job of the discrimination nets to find the particular pattern present in a stimulus and return the appropriate CONEXICON pointer.)

The FRAMEWORK of the concexicon entry 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, OBJ2, IND-OBJ, INF, and INF2 mentioned earlier. A FIELD SPECIFICATION is a designation of a substructure of a conceptualization. It consists of a list of elements from the set

```

{ ACTOR OBJECT TO FROM <=> <=> <=> <=> CON
  VAL PART MOBJECT TIME MODE /\ <=> <=> }
  
```

These are the internal names used by the system to refer to roles in conceptual relations. Most correspond closely to the names used in section II; the less obvious ones are:

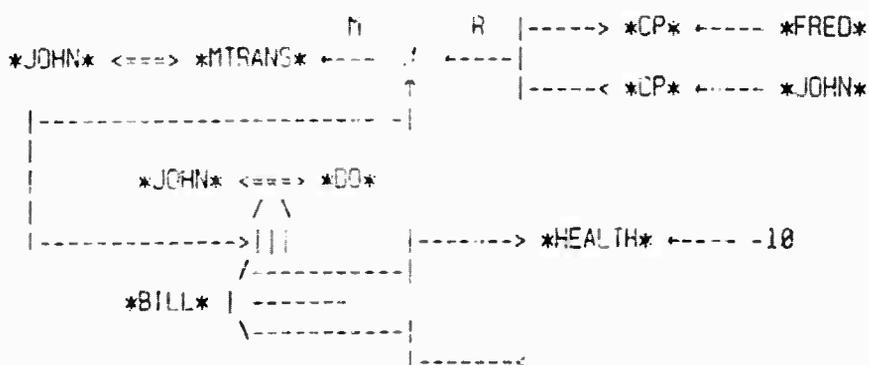
ACTOR	refers to the <ACTOR> in EVENTS, the <CONCEPT> in STATES and STATE-CHANGES
CON	refers to the <ANTECEDENT> in causal relations, the first conceptualization of conjunctive relations
/\	refer to the second conceptualization of a conjunctive relation
<=, <=C	refers to the <RESULT> of the corresponding type or causal relationship
<=F, <=T	refer to the initial and terminal states of a statechange relation

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 empty (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 CON ACTOR) applied to



is the PP *JOHN*.

Special requirements are used mainly for the introduction of prepositions into the syntax nets and will be described in an example.

In processing a conceptual structure, a 'base node' for the syntax net is created. This is termed the ACTIVE NODE. When a concexicon entry for the structure has been found (by applying discrimination nets to this CURRENT CONCEPTUAL STRUCTURE (CCS)), the relation LEX is attached to the ACTIVE NODE with its value being the reference in the LEXICAL POINTER field of the concexicon entry. Next, all SPECIAL ACTIONS specified in the concexicon entry are taken, and finally the FRAMEWORK is processed as follows:

- (1) Get the next FRAME. If no more exist, go to step 3. Otherwise, add a node N to the network connected to the active node by the SYNTAX RELATION specified in the frame. Make N the ACTIVE NODE, saving the old ACTIVE NODE, the CCS, and the unprocessed FRAMES of the FRAMEWORK on the NODE PUSHDOWN LIST (NPL).
- (2) Get a new CCS by applying the FIELD SPECIFICATION of the FRAME to the old CCS. Apply the network generation algorithm to the new CCS, thereby expanding the syntax net from the ACTIVE NODE.
- (3) If the NPL is empty, generation of the syntax net is completed. Otherwise, 'pop' the NPL, restoring an existing node to ACTIVE NODE status, and restoring a CCS and unprocessed set of FRAMES. Return to step 1.

The structure specified by a FIELD SPECIFICATION (in step 2) may turn out to be a conceptual nominal. Since these are represented

only as pointers into memory (*MARY*, *E T*) instead of by complex conceptual structures, they are treated specially by the generation algorithm. Specifically, no discrimination net is applied, but a lexical pointer is found associated with the concept in memory (i.e., the predicate ENGLISH-NAME mentioned earlier is used). Treatment is entirely equivalent to considering the PP as itself being a concexicon entry with a regular LEXICAL POINTER, but with no FRAMEWORK or SPECIAL ACTIONS.

Consider how this process builds a net from our simple example (C4) above. The conceptualization would be taken and recognized as an event structure (main link <=>) with ACT = *ATPANS*. The discrimination nets for such structures would be applied and a concexicon pointer, say 'RECEIVE1', found. The concexicon entry RECEIVE1 consists of:

LEXICAL POINTER	FRAMEWORK			SPECIAL ACTIONS
RECEIVE	ACTSBJ	(TO)	NIL	NIL
	OBJ	(OBJECT)	NIL	
	IOBJ	(ACTOR)	(PREP FROM)	

An active node G1 would be established with relation LEX and value RECEIVE (the LEXICAL POINTER field of the entry)

G1: LEX RECEIVE

Processing the first two frames attaches an ACTSBJ MARY (the

lexical pointer associated with *MARY*, found in the (TO) field), and an OBJ BOOK (the lexical pointer associated with *BOOK*, found in the (OBJECT) field).

G1: LEX RECEIVE
 ACTSBJ G2
 OBJ G3

G2: LEX MARY

G3: LEX BOOK

The third frame specifies the relation IOBJ to be found in the ACTOR field (*JOHN*). The SPECIAL REQUIREMENT (PREP FROM) in this FRAME has a minor transformational effect on the standard processing which can be seen in the network created.

G1: LEX RECEIVE	G4: PREP FROM
ACTSBJ G2	POBJ G5
OBJ G3	
IOBJ G4	G5: LEX JOHN

G2: LEX MARY

G3: LEX BOOK

There exist both syntactic and semantic elements not yet present in this network. Every sentence has a TENSE (PAST, PRESENT, FUTURE, PAST PERFECT, etc.), a FORM (SIMPLE, PROGRESSIVE), a MOOD (INDICATIVE, INTERROGATIVE, SUBJUNCTIVE, IMPERATIVE), and a VOICE (ACTIVE, PASSIVE). Only a simple analysis of these problems has been incorporated into the current program. Whenever the LEX relation attached to an ACTIVE NODE in the syntax net has a 'verb' as its value, (VERB being a lexical category), TENSE, FORM, MOOD, and VOICE relations must be attached to the node. This information is derived

from the CCS at the time. TENSE is chosen from the set IPRES, PAST, FUT1 depending on whether the TIME of the conceptualization is *NOW*, before *NOW*, or after *NOW*. The TIME of a conceptualization is taken as that attached to the main link if it is one of the simple types, or the TIME of the ANTECEDENT in a causal relation. FORM is always chosen as SIMPLE; extension to PROGRESSIVE will occur when the representation of time is expanded to include intervals as well as points. MOOD is chosen as INTERROG if a (?) MODE modifies the conceptualization. SUBJUNCTIVE is chosen for certain <=C structures. Otherwise, INDICATIVE is used. VOICE is currently always chosen as ACTIVE; presumably FOCUS could be used to choose PASSIVE in some instances.

In addition to these relations which are required by English syntax, information may be present in a conceptual structure which will not be processed by the connexion entry retrieved for that structure. The modifying relations of conceptual dependency (PART as in *NECK* ----- *MARY*, REF in *BOOK* ----- INDEF) each have a language specific function associated with them. When these relations are noticed on a conceptual structure, the corresponding functions are executed to modify the syntax node created for the head of the structure.

These considerations result in a completed syntax net for our example:

G1:	LEX	RECEIVE	G4:	PREP	FROM
	ACTSBJ	G2		POBJ	G5
	OBJ	G3			
	I OBJ	G4	G5:	LEX	JOHN
	TENSE	PAST			
	FORM	SIM			
	VOICE	ACT			
	MOOD	INDIC			
G2:	LEX	MARY			
G3:	LEX	BOOK			
	DET	A			

From this net the sentence "Mary received a book from John" is generated.

As described above, the syntax net created will always have a tree structure. In actuality, new nodes are created for conceptual nominals only if no node already exists for that element. Otherwise, another connection to the existing node is made. In the syntax net for "John told Mary he saw Mr. Smith" only one node for "John" will be present, standing in an ACTSBJ relation to two different nodes. The only use of this fact by the surface generator is in the inclusion of pronouns, and is thus not of great significance.

The surface generation grammar will not be described here. It is based on the grammar and program used by Simmons and described in (10); the principal techniques used do not differ from those described there. Our grammar generates understandable, but not totally correct, English sentences from most of the syntax nets created by the program. Certainly there is still a great deal of work to be done in this area; the conceptual approach taken by MARGIE does not alleviate most of the problems caused by natural language syntax in generation.

MULTIPLE REALIZATIONS

The process described in this last 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 was 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 examples given earlier. Rather, they are obtained by allowing the discrimination net algorithm to find more than one response.

It was pointed out earlier that there may be more than one discrimination net applicable to a given stimulus. Sometimes more than one of these will produce a response. Since the nets are organized to group 'related' meanings into a single net, however, it often is the case that more than one appropriate response exists within a single net. This case is handled by the addition of two simple devices to the discrimination nets.

First, terminal nodes are allowed to have associated with them not just a single concexicon entry, but a set of such entries. Thus a 'stimulus' which finds the response LIKE₄ may find ENJOY₁ and PLEASE₁ at the same time. Each of these entries may be used as the source for generating a distinct net, leading to distinct paraphrases.

This handles cases of what might be called 'conceptual synonymy'. Such cases do not explain a great deal of paraphrase, however, and become rarer as conceptual representations are refined. The second device used is to permit any terminal of a discrimination net to hold, in addition to a list of concexicon pointers, a pointer back to some node in the network. (This will be represented by the presence of a " ↑ <integer> " at the terminal, <integer> being the index of some node in the network. Some terminals may contain only such pointers, and no responses at all.) In addition to using the responses found at a terminal, it is possible to follow the pointer and resume the net application process from the specified node. More formally, it is necessary to modify the discrimination net algorithm as follows:

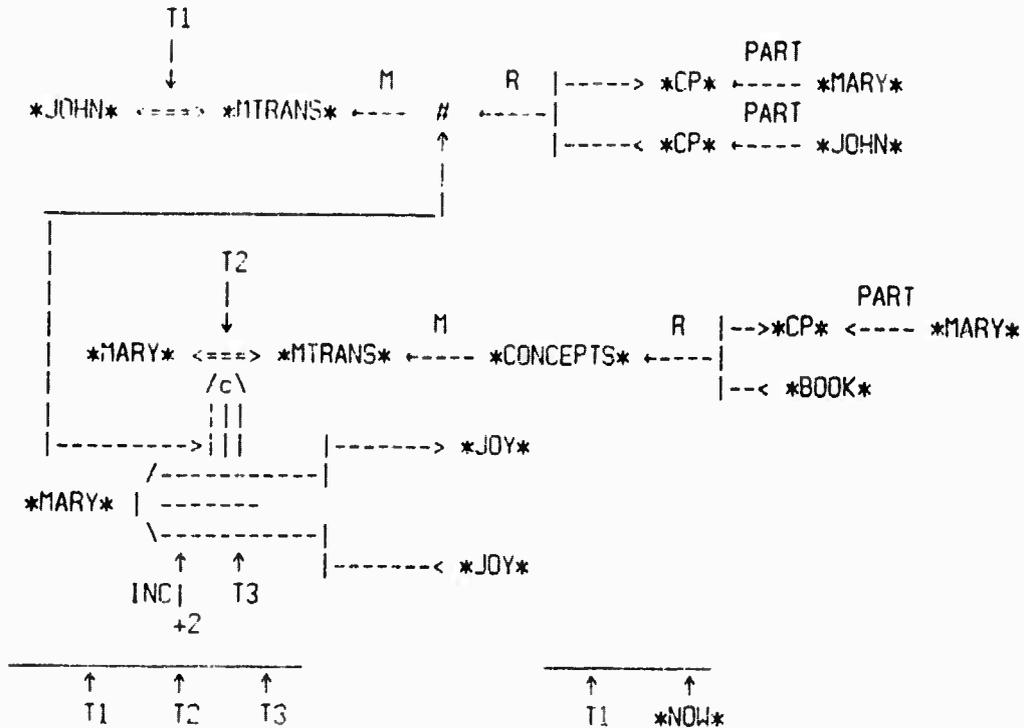
add step

0. set RESPONSES to NIL.

replace step 6 with

6. add the responses associated with CURRENT-NODE to RESPONSES. If CURRENT-NODE has no associated pointer, then return RESPONSES. Otherwise set CURRENT-NODE to the node indicated by the pointer and go to step 2.

This may lead to the discovery of 'conceptually' distinct responses. Intuitively, this process corresponds to 'ignoring' some feature of the stimulus which English provides a special way of expressing and finding a more general way of expressing the information. The reader is invited to apply the partial discrimination net of Figure 3 to the stimulus



following the modified discrimination net algorithm, and verify that

the concexicon entries

ADVISE1 SUGGEST1 TELL1

are all found. The paraphrases

"John advised Mary to read the book"
 "John suggested to Mary she would like to read the book"
 "John told Mary she would like to read the book"

are generated from this stimulus, as well as several others which result from the conceptual synonymy of LIKE4, ENJOY1 and PLEASE1, and the paraphrase of

"Mary would like to read the book" as
 "Mary would become happier if she read the book".

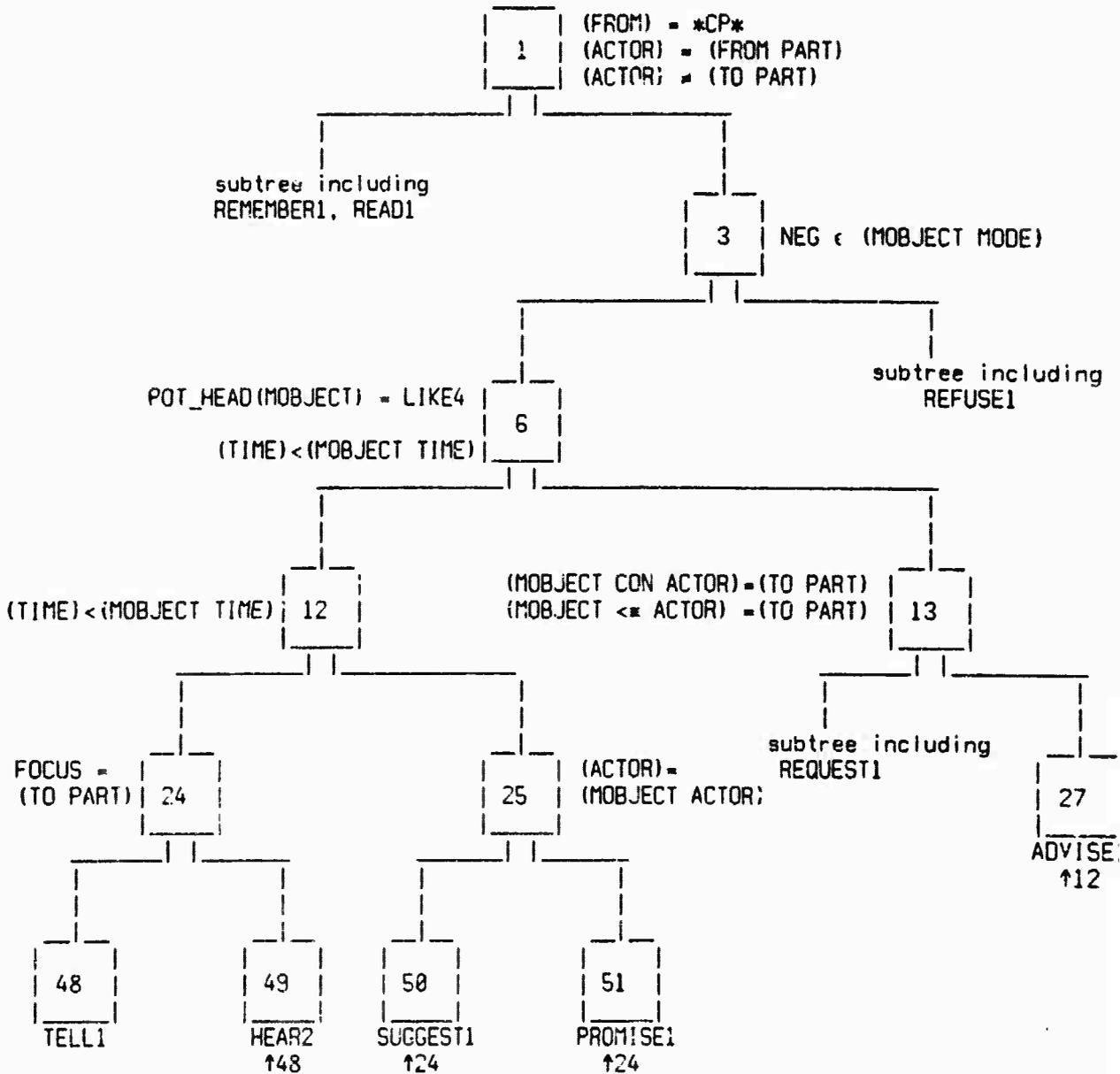


FIGURE 3
partial discrimination net for *MTRANS* EVENTS

FIELD SPECIFICATIONS in predicates refer to fields of an *MTRANS* conceptual stimulus. Multiple predicates at a node form a conjunction.

In part I of this report we described a natural language processing task which we termed 'sentence paraphrasing'. It was claimed that this task is of interest for three major reasons:

- 1) its relation to machine translation
- 2) the need to 'understand' natural language in order to produce paraphrases
- 3) the effect of context on sentence paraphrases

MARGIE is a computer program which, given an English sentence as input, can produce English sentences which are paraphrases of that input. In order to describe its operation it was necessary to define the notion of conceptual representation. Section II enumerated the properties intrinsic in such representations and gave more detailed examples of the particular representation employed by MARGIE.

The production of paraphrases requires two basic processes. The first takes English sentences and produces conceptual representations of their meaning; the second performs the inverse operation.

Part III discussed a conceptual analyzer for English sentences. The analyzer's goal was to find, for a given English sentence, the conceptualization that represented the meaning a human would assign to that sentence in the same context. The basic mechanism was the request. Words had both features and requests associated with them. Encountering a word made both sets available to the analyzer. A feature was a conceptualization. A request was a predicate plus a

set of actions. If the predicate became true while the request was still active, the actions were performed. The predicates could ask questions about the words and concepts found or about the conceptualization being built. The actions could modify that conceptualization or alter the set of requests active. In general, verbs and prepositions contributed most of the requests while nouns were important mainly for the features associated with them.

Part IV detailed the operation of a program to express a conceptually represented meaning in English. This was accomplished via an intermediate 'syntax net'. To produce the net it was necessary to discriminate conceptual patterns for which English provides particular verbs. To choose verbs not only pattern matching, but conceptual knowledge and even theorem proving capabilities were seen to be necessary. The discriminations provided a link between pure conceptual structures and units in a 'concexicon'. These units in turn provided a link from the conceptual to the syntactic, and enabled the construction of the syntax nets.

The paraphrase task did not motivate either the analysis or generation algorithm. The analyzer performs no operations specific to this task. And it was shown that only a slight modification to a general conceptual to English generator was needed to accomplish paraphrasing. The property of explicit representation of similarities in meaning at the conceptual level makes this possible. It also means that no manipulations of the meaning representation are

needed for paraphrasing. This stands in contrast to representations based on words or word senses, which would require the application of transformational or inference rules to produce paraphrases.

V1.

EXAMPLES OF PROGRAM OUTPUT

Lines preceded by a '*' were typed by the user. Lower case is used for explanatory remarks. All other lines were typed by the program.

TYPE INPUT

*(JOHN GAVE MARY A BICYCLE) the input sentence.

OUTPUT FROM PARSER: in two parts; first time relationships:

TIM00 : ((VAL *T*)) TIM00 has value *T* (analog of *NOW*)

TIM01 : ((BEFORE TIM00 X)) TIM01 is before TIM00 by an unspecified amount X

then the actual conceptualization, utilizing these times:

((ACTOR (JOHN) <=> (*ATRANS*) OBJECT (BICYCLE REF (*A*)) FROM (JOHN) ~
TO (MARY)) TIME (TIM01) FOCUS (ACTOR)):

PARTIALLY INTEGRATED RESULT: this is the memory model's internal format
((*ATRANS* (#JOHN1) (G0004) (#JOHN1) (#MARY1)) (TIME _ (G0006)))

THINGS TO SAY: this is the conceptualization after being internalized
and reconstructed by the memory model.

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

the syntax net produced by the generator:

G0020:	JBJ2	(G0023)	G0021:	LEX	(JOHN)
	OBJ	(G0022)			
	ACTSBJ	(G0021)	G0022:	DET	(A)
	VOICE	(ACT)		LEX	(BICYCLE)
	FORM	(SIM)			
	TENSE	(PAST)	G0023:	LEX	(MARY)
	MOOD	(INOIC)			
	LEX	(GIVE)			

the string produced by the surface grammar:

(JOHN GAVE MARY A BICYCLE) the 'identity' paraphrase in this case

more nets and surface strings:

G0022: IOBJ (G0025)
 OBJ (G0024)
 ACTSBJ (G0023)
 VOICE (ACT)
 FORM (SIM)
 TENSE (PAST)
 MOOD (INDIC)
 LEX (GIVE)

G0024: DET (A)
 LEX (BICYCLE)

G0025: POBJ (G0026)
 PREP (TO)

G0026: LEX (MARY)

G0023: LEX (JOHN)

(JOHN GAVE A BICYCLE TO MARY)

by having concexicon entries GIVE1 and GIVE2, both with lexical pointer GIVE, and both occurring at the same terminal of a discrimination net, but with slightly different syntax FRAMEs, the forms "X gave Y Z" and "X gave Z to Y" are generated.

G0023: IOBJ (G0026)
 ACTSBJ (G0025)
 OBJ (G0024)
 VOICE (ACT)
 FORM (SIM)
 TENSE (PAST)
 MOOD (INDIC)
 LEX (GET)

G0025: LEX (MARY)

G0026: POBJ (G0027)
 PREP (FROM)

G0027: LEX (JOHN)

G0024: DET (A)
 LEX (BICYCLE)

(MARY GOT A BICYCLE FROM JOHN)

a pointer from the terminal holding GIVE was followed, having the effect of ignoring the focus on the ACTOR (=JOHN)

G0024: IOBJ (G0027)
 ACTSBJ (G0026)
 OBJ (G0025)
 VOICE (ACT)
 FORM (SIM)
 TENSE (PAST)
 MOOD (INDIC)
 LEX (RECEIVE)

G0026: LEX (MARY)

G0027: POBJ (G0028)
 PREP (FROM)

G0028: LEX (JOHN)

G0025: DET (A)
 LEX (BICYCLE)

(MARY RECEIVED A BICYCLE FROM JOHN;
 also ignores the focus on JOHN)

In the remaining examples only the input sentence, parser output, and paraphrase surface strings will be shown. The reader should keep in mind that the thrust of our work in generation has been toward the production of syntax nets from conceptual structures. No more effort has been put into the surface phase of generation than was necessary to obtain minimally readable realizations. While syntactic problems in the resulting surface strings occasionally reflect actual inadequacies in the production of the syntax nets, the great majority of such faults result from the primitive transition network grammar being employed to map the syntax nets into sentences.

TYPE INPUT

*(JOHN ADVISED MARY TO DRINK THE WINE)

OUTPUT FROM PARSER:

TIM00 : ((VAL *T*

TIM01 : ((BEFORE TIM00 X))

TIM02 : ((AFTER TIM01 X))

TIM03 : ((AFTER TIM01 X))

It might be noted that the times given may not appear in the output. Some are generated at one point in the program and then overwritten by later actions. No information has been lost and usually the overwritten time was a reference point. TIM00 and TIM02 are replaced here.

((ACTOR (JOHN1) <=> (*ITRANS*) TO (*CP* PART (MARY1) REF (*THE*)) FROM
 (*CP* PART (JOHN1) REF (*THE*)) MOBJECT ((CON ((ACTOR (MARY1) <=> (~
 INGEST) OBJECT (WINE1 REF (*THE*)) TO (*INSIDE* PART (MARY1)) FROM ~
 (*MOUTH* PART (MARY1)) INST (*ONE*)) MODE (NIL) TIME (TIM03) FOCUS (~
 ACTOR1)) <=>C ((ACTOR (MARY1) >T (*JOY*) <=>F (*JOY*)) INC (2) TIME ~
 (TIM03) MODE (NIL)))))) FOCUS ((ACTOR)) MODE (NIL) TIME (TIM01))

paraphrases

(JOHN ADVISED MARY TO DRINK SOME WINE)

even though the input contained 'THE WINE', the memory was unable to determine what 'THE WINE' referred to. (No wine existed in the current context.) In passing the conceptualization on to the generator, the 'definite' reference on WINE was changed to 'indefinite'. The generator expresses the indefinite reference as 'A' or 'SOME', depending on conceptual properties of the governor. Since the concept referred to by WINE in the conceptualization is a physical substance, but not an 'entity', the modifier SOME is chosen.

(JOHN ADVISED MARY TO INGEST SOME WINE)

(JOHN SUGGESTED TO MARY SHE WOULD LIKE TO DRINK SOME WINE)

(JOHN SUGGESTED TO MARY SHE WOULD LIKE TO INGEST SOME WINE)

(JOHN SUGGESTED TO MARY SHE DRINKS SOME WINE WOULD PLEASE HER)

(JOHN SUGGESTED TO MARY SHE INGESTS SOME WINE WOULD PLEASE HER)

(JOHN SUGGESTED TO MARY SHE WOULD ENJOY SHE DRINKS SOME WINE)

(JOHN SUGGESTED TO MARY SHE WOULD ENJOY SHE INGESTS SOME WINE)

(JOHN SUGGESTED TO MARY SHE BECOMES HAPPY IF SHE DRINKS SOME WINE)

(JOHN SUGGESTED TO MARY SHE BECOMES HAPPY IF SHE INGESTS SOME WINE)

TYPE INPUT
*(MARY WANTS TO CHOKe JOHN)

OUTPUT FROM PARSER:

TIM00 : ((VAL *T*))

TIM01 : ((AFTER TIM00 X))

TIM02 : ((AFTER TIM00 X))

TIM03 (AFTER TIM00 X))

((CON ((CON ((CON ((ACTOR (MARY1) <=> (*GRASP*) OBJECT (*NECK* PART (~
JOHN1))) TIME (TIM03)) <=> ((ACTOR (JOHN1) <=> (*INGEST*) OBJECT (*AIR~
* REF (*A*) FROM (*MOUTH* PART (JOHN1)) TO (*INSIDE* PART (JOHN1))) ~
TIME (TIM03) MODE ((*CANNOT*))))) FOCUS (CON ACTOR)) <=>C ((ACTOR (MARY~
)) <=>T (*JOY*) <=>F (*JOY*)) INC (2) TIME (TIM02)))) <=> (*MLOC* VAL~
(*LTM* PART (MARY1) REF (*THE*))) MODE (NIL) FOCUS ((-> VAL PART))~
TIME (TIM00))

paraphrases

(MARY WANTS TO CHOKe JOHN)

The original input is again the first realization found. It is natural to organize the discrimination nets so that the first 'response' found is the one which expresses a 'maximal' conceptual substructure. This normally results in the most concise linguistic expression.

(MARY WANTS TO PREVENT JOHN BREATHEs BY SHE GRABS HIS NECK)

(MARY WANTS TO PREVENT JOHN INHALES SOME AIR BY SHE GRABS HIS NECK)

(MARY WANTS TO CAUSE JOHN IS UNABLE TO BREATHE BY SHE GRABS HIS NECK)

(MARY WANTS TO CAUSE JOHN IS UNABLE TO INHALE SOME AIR BY SHE GRABS HIS NECK)

(MARY WANTS TO CAUSE JOHN NOT CAN BREATHE BY SHE GRABS HIS NECK)

(MARY WANTS TO CAUSE JOHN NOT CAN INHALE SOME AIR BY SHE GRABS HIS NECK)

(MARY WANTS JOHN IS UNABLE TO BREATHE BECAUSE SHE GRABS HIS NECK)

(MARY WANTS JOHN IS UNABLE TO INHALE SOME AIR BECAUSE SHE GRABS HIS NECK)

(MARY WANTS JOHN NOT CAN BREATHE BECAUSE SHE GRABS HIS NECK)

(MARY HOPES SHE CHOKES JOHN)

all the paraphrases of "choke" are produced again, this time combined with "hope" instead of "want".

(MARY THINKS SHE WOULD LIKE TO CHOKE JOHN)

all the paraphrases of "like" (seen in the preceding example) are also produced in the embedded sentence.

TYPE INPUT

*(JOHN KILLED MARY BY CHOKING MARY)

The analysis of this example plus a graphic equivalent of the output can be found in section iii.

OUTPUT FROM PARSER:

TIM00 : ((VAL *T*))

TIM01 : ((BEFORE TIM00 X))

TIM02 : ((BEFORE TIM01 X))

((CON ((CON ((ACTOR (JOHN1) <=> (*GRASP*) OBJECT (*NECK* PART (MARY1)) ~
)) TIME (TIM02)) <=> ((ACTOR (MARY1) <=> (*INGEST*) OBJECT (*AIR* REF ~
(*A*)) FROM (*MOUTH* PART (MARY1)) TO (*INSIDE* PART (MARY1))) TIME (~
TIM02) MODE ((*CANNOT*))))) FOCUS (CON ACTOR)) ^ ((CON ((ACTOR (MARY1) ~
<=> (*INGEST*) OBJECT (*AIR* REF (*A*)) FROM (*MOUTH* PART (MARY1)) ~
TO (*INSIDE* PART (MARY1))) TIME (TIM02) MODE ((*CANNOT*)) <=> ((ACTO~
R (MARY1) <=>T (*HEALTH* VAL (-10)) <=>F (*HEALTH* VAL (NIL))) MODE (~
NIL) TIME (TIM01))))))

paraphrases

(JOHN STRANGLED MARY)

Here the first paraphrase does not match the input. It is, in fact, more concise; the generator sees the representation produced by the analyzer for "killed by choking" as sufficient for use of the word "strangle".

(JOHN CHOKED MARY AND SHE DIED BECAUSE SHE WAS UNABLE TO BREATHE)

(JOHN CHOKED MARY AND SHE DIED BECAUSE SHE WAS UNABLE TO INHALE SOME AIR)

(JOHN CHOKED MARY AND SHE DIED BECAUSE SHE NOT COULO BREATHE)

(JOHN CHOKED MARY AND SHE DIED BECAUSE SHE NOT COULD INHALE SOME AIR)
 (JOHN CHOKED MARY AND SHE BECAME DEAD BECAUSE SHE WAS UNABLE TO BREATHE)
 (JOHN CHOKED MARY AND SHE BECAME DEAD BECAUSE SHE WAS UNABLE TO INHALE
 SOME AIR)
 (JOHN CHOKED MARY AND SHE BECAME DEAD BECAUSE SHE NOT COULD BREATHE)

TYPE INPUT

*(JOHN TOLD MARY THAT JOHN WOULD HIT MARY)

OUTPUT FROM PARSER:

TIM00 : ((VAL *T*))

TIM01 : ((BEFORE TIM00 X))

TIM02 : ((AFTER TIM00 X))

((ACTOR (JOHN1) <=> (*MTRANS*) TO (*CP* PART (MARY1) REF (*THE*)) FROM
 (*CP* PART (JOHN1) REF (*THE*)) MOBJECT ((CON ((ACTOR (JOHN1) <=> (~
 PROPEL) OBJECT (*HAND* PART (JOHN1)) TO (MARY1) FROM (JOHN1) INST (~
 (ACTOR (JOHN1) <=> (*MOVE*) OBJECT (*HAND* PART (JOHN1)))))) TIME (TIM~
 02) MODE (NIL)) <=> ((ACTOR (*HAND* PART (JOHN1)) <=> (*PHYSCONT* VAL ~
 (MARY1))) TIME (TIM02) MODE (NIL) FOCUS (CON ACTOR)))))) TIME (TIM01))

paraphrases

The preceding examples were run in a mode in which queries from the generator to the memory which the memory was not yet capable of handling resulted in a uniform response of FALSE. This example produces more interesting results when run in a mode which allows these queries to be answered by human intervention at the teletype:

TIME TO PLAY GOD -- IS THIS TRUE?

```
(CON ((COM) ((ACTOR (JOHN) <=> (*PROPEL*) OBJECT (*HAND* REF (*A*) PART (JOHN)) FROM (JOHN) TO (MARY) INST ((ACTOR (JOHN) <=> (*MOVE*) OBJECT (*HAND* REF (*A*) PART (JOHN)) FROM (*ONE*) TO (*ONE*)) FOCUS ((ACTOR))) TIME (G0008) FOCUS ((ACTOR))) <=> ((ACTOR (*HAND* REF (*A*) PART (JOHN)) <=> (*PHYSCONT* VAL (MARY))) TIME (G0003))) <=>C ((ACTOR (~MARY) <=>T (*PSTATE*) <=>F (*PSTATE*)) INC (*?* LEQUAL (-3)) TIME (*?* AFTER (G0008))))
*i
```

The program asks (conceptually) whether John's hitting Mary (at time G0008) could cause a change in Mary's position on the 'physical state' scale (*PSTATE*) at some time after G0008) by an increment ≤ -3 . The human respondent answered True.

(JOHN THREATENED TO HIT MARY)

The knowledge of the potentially injurious nature of the event communicated by John allowed the program to choose "threaten".

(JOHN THREATENED TO HIT MARY WITH HIS HAND)

The 'instrument' of the hitting is normally expressed. When it is the hand of the 'hitter', however, it can be left off, as in the preceding realization. 'Hand' was not present in the input; the analyzer made the assumption that John's hand was what he would hit Mary with. The generator assumes other people use this default too, and thus permits both realizations.

(JOHN TOLD MARY HE WILL HIT HER)

(JOHN TOLD MARY HE WILL HIT HER WITH HIS HAND)

TYPE INPUT
*(JOHN LOANED A BICYCLE TO MARY)

OUTPUT FROM PARSER:

TIM00 : ((VAL *T*))

TIM01 : ((BEFORE TIM00 X))

TIM02 : ((AFTER TIM01 X))

((CON ((ACTOR (JOHN1) <=> (*ATRANS*) TO (MARY1) FROM (JOHN1) OBJECT (~BIKE1 REF (*A*)) TIME (TIM01)) ^ ((CON ((ACTOR (MARY1) <=> (*ATRANS*~) TO (JOHN1) FROM (MARY1) OBJECT (BIKE1 REF (*A*)) TIME (TIM02)) <=>~ (*MLOC* VAL (*LTM* PART (JOHN1)))) FOCUS ((<=> VAL PART) TIME (TIM0~1))))))

The conceptual representation found is a conjunction:

- 1) John gave Mary the bicycle, and
- 2) John believed at that time that Mary would give the bicycle to him at some future time.

paraphrases

TIME TO PLAY GOD -- IS THIS TRUE?

((ACTOR (BIKE REF (*A*)) <=> (*POSS* VAL (MARY))) TIME (*?* BEFORE (G~0009)))

*NIL

the generator asked whether Mary had the bicycle at any time before John loaned it to her. The answer given was "no".

TIME TO PLAY GOD -- IS THIS TRUE?

((ACTOR (BIKE REF (*A*)) <=> (*POSS* VAL (JOHN))) TIME (*?* BEFORE (G~0012)))

*T

the generator asks whether John had the bicycle at any time before the time at which he believes she will be giving it to him. The answer given was "yes".

(JOHN GAVE A BICYCLE TO MARY AND HE EXPECTED SHE TO RETURN IT TO HIM)

(JOHN GAVE A BICYCLE TO MARY AND HE EXPECTED SHE TO GIVE HIM IT)

(JOHN GAVE A BICYCLE TO MARY AND HE EXPECTED SHE TO GIVE IT TO HIM)

(JOHN GAVE A BICYCLE TO MARY AND HE EXPECTED TO GET IT FROM HER)

(JOHN GAVE A BICYCLE TO MARY AND HE EXPECTED TO RECEIVE IT FROM HER)

(JOHN GAVE A BICYCLE TO MARY AND HE THOUGHT SHE WILL RETURN IT TO HIM)

TYPE INPUT

*(JOHN SOLD A BICYCLE TO MARY)

OUTPUT FROM PARSER:

TIM00 : ((VAL *T*))

TIM01 : ((BEFORE TIM00 X))

((CON ((ACTOR (JOHN1) <=> (*ATRANS*)) OBJECT (BIKE1 REF (*A*)) TO (MARY1) FROM (JOHN1)) TIME (TIM01)) <==> ((ACTOR (MARY1) <=> (*ATRANS*)) OBJECT (MONEY1) TO (JOHN1) FROM (MARY1)) FOCUS ((CON ACTOR)) TIME (TIM01))))

paraphrases

(JOHN SOLD MARY A BICYCLE FOR MONEY)

(MARY BOUGHT A BICYCLE FROM JOHN FOR MONEY)

(MARY PAYED JOHN MONEY FOR A BICYCLE)

(JOHN TRADED MARY A BICYCLE FOR MONEY)

(MARY TRADED JOHN MONEY FOR A BICYCLE)

TYPE INPUT

*(JOHN AGGRAVATED FRED)

OUTPUT FROM PARSER:

TIM00 : ((VAL *T*))

TIM01 : ((BEFORE TIM00 X))

((CON ((ACTOR (JOHN1) <=> (*DO*))) <= ((ACTOR (FRED1) <=>T (*ANGER*) ~ <=>F (*ANGER*)) TIME (TIM01) INC (2)) FOCUS ((CON ACTOR))))

paraphrases

(JOHN ANNOYED FRED)

(JOHN AGGRAVATED FRED)

(JOHN MADE FRED BECAME ANGRY)

(JOHN CAUSED FRED BECAME ANGRY BY HE DID SOMETHING)

(FRED BECAME ANGRY BECAUSE JOHN DID SOMETHING)

REFERENCES

- 1) Charniak, E., "Towards a Model of Children's Story Comprehension", AI TR-266, Massachusetts Institute of Technology, Cambridge Massachusetts, December 1972.
- 2) Fillmore, C. "The Case for Case" in Back and Harms (eds.) Universals in Linguistic Theory, Holt, Rinehart, and Winston, New York, 1968.
- 3) Friedman, J. "Directed Random Generation of Sentences", CACM, Vol. 12, No. 1, January 1959.
- 4) Roberts, K., "An Investigation of Paraphrasing: The Effects of Memory and Complexity", Tech. Report No. 30, Human Performance Center, University of Michigan, June 1971.
- 5) Russell, S., "Semantic Categories of Nominals for Conceptual Dependency Analysis of Natural Language", AIM-172, Computer Science Department, Stanford University, Stanford California, July 1972.
- 6) Schank, R., "Conceptual Dependency: A Theory of Natural Language Understanding" Cognitive Psychology, Vol. 3, No. 4, October 1972.
- 7) Schank, R., "The Fourteen Primitive Actions and their Inferences", AIM-183, Computer Science Department, Stanford University, Stanford California, March 1973.
- 8) Schank, R., Goldman, N., Riesbeck, C., and Rieger, C., "Primitive Concepts Underlying Verbs of Thought", AIM-162, Computer Science Department, Stanford University, Stanford California, February 1972.
- 9) Schank, R. and Rieger, C., "Inference and the Computer Understanding of Natural Language", AIM-197, Computer Science Department, Stanford University, Stanford California, April 1973.

- 10] Simmons, R. and Slocum, J., "Generating English Discourse from Semantic Networks", CACM, Vol. 15, No. 10, October 1972.
- 11] Woods, W., "Transition Network Grammars for Natural Language Analysis", CACM, Vol. 13, No. 10, October 1970.
- 12] Yngve, V.H., "Random Generation of English Sentences", in 1961 International Conference on Machine Translation of Languages and Applied Language Analysis, Her Majesty's Stationery Office, London, 1962.