MARGIE MEMORY, ANALYSIS, RESPONSE GENERATION, and INFERENCE on ENGLISH

Roger C. Schank
Ntit Goldman
Charles J. Riager III
Chris Rissbeck

Stanford University
Stanford , California USA

A program is described that accepts natural language input and makes inferences from it and paraphrases of it. The Conceptual Dependency framework is the basis of this system.

1. Introduction

A program is now running at the Stanford Artificial Intelligence Project that analyzes natural language sentences into an underlying language-free conceptual representation; stores this representation in memory,establiahes references and makes inferences] encodes the conceptual representation into an English syntactic representation; and then encodes the syntactic representation into an English string. The program has two modes: PARAPHRASE and INFERENCE. In PARAPHRASE mode up to 150 semantic paraphrases can be generated from an input sentence by reading the conceptual representation underlying that sentence using different words and concept combinatione. 1n INFERENCE mode, most of the possible inferences that can be made from an input sentence are produced as output.

2. Examples

INFERENCE MODE

INPUT: John told Mary that Bill wants a book.
OUTPUT1: A book about what?
OUTPUT2: Mary knows that Bill wants a book.
OUTPUT3: Bill wants to come to have a book.
OUTPUT4: Bill wants someone to cease to have a book.
OUTPUT5: Bill wants to read a book.

INPUT: John gave Mary a beating with a stick.
OUTPUT1: A stick touched Mary.
OUTPUT2: Mary became hurt.
OUTPUT3: John wanted Mary to become hurt.
OUTPUT4: What did Mary do to make John angry?

INPUT: John gave Mary an aspirin.
OUTPUT1: John believes that Mary wants an aspirin.
OUTPUT2: Mary is sick.
OUTPUT3: Mary wants to feel better.
OUTPUT4: Mary will ingest the aspirin.

INPUT:John is going to go to the store.
OUTPUT1: John wants to be at the store.
OUTPUT2: What does John want to buy?
OUTPUT3: John will trade money to the store for something.

PARAPHRASE MODE

INPUT: John advised Mary to drink the wine.
OUTPUT: John told Mary that drinking the wine would benefit her.

INPUT: John killed Mary by choking Mary.
OUTPUT1: John strangled Mary.
OUTPUT2: John choked Mary and she died because she not could breathe.
OUTPUT3: Mary died because she was unable to inhale some air and she was unable to inhale some air because John grabbed her neck.

INPUT: Reading the book reminded Rita to loan the book to Bill.
OUTPUT: Rita remembered to give the book to Bill and she expects him to return it to her because someone read the book.

INPUT: John prevented Bill from giving a banana to Mary by selling a banana to Rita.
OUTPUT1: Bill was unable to give a banana to Mary because Rita traded John some money for a banana.
OUTPUT2: Rita bought a banana from John caused Mary not could get a banana from Bill.

3. Organization

The theoretical basis for this system is Conceptual Dependency as expounded by the authors [6],[7], and [9]. The natural language analyzer was written by Chris Riesbeck and is described in [5]. It is a different program than we have used before,with an emphasis on analyzing meaning before considering syntax. The conceptual memory is the work of Chuck Rieger [4]. Neil Goldman [2] wrote the program that encodes conceptual representations into English syntactic ones. We have devoted almost no effort towards encoding our syntactic representations into well constructed English strings. Instead we have modified Simmons' [10],[11] program to accomplish this task.

4. Theoretical Approach

It is really not possible to adequately judge a system on the basis of a small sample of its output. Ad hoc procedures could be written that would perform well on small data bases. It has been our intent here to have this system reflect our thinking along theoretical lines as well as possible. We have, on occasion sacrificed efficiency for theory. The conceptual representations used are intended to account for the concepts that are remembered when a human hears a sentence. Likewise, the processes used in our programs are intended to mimic human processing insofar as that can be determined.

The core of the system is Conceptual Dependency theory. This theory postulates that there are six basic concept types: things,actions, attributes of things, attributes of actions, times and locations. These concept types are connected in only sixteen possible ways. The principle ones are: Thing <=> Action,which signifies an actor acting; Action <D- Thing which signifies the object of an Action; Action <R[Thing / Thing] which signifies the recipient and the donor of an Action; Action <D[Location / Location] which signifies the direction of an Action; Action <O-O> signifies the mental object of an action; Thing <= [Attribute / Attribute] signifies that a Thing has changed Attributes on some scale and, Thing <=> Action signifies that an action has caused a state change. Thing <= [Attribute / Attribute]

Whenever a two-headed arrow is present a conceptualization is said to exist.

Conceptual Dependency uses only fourteen possible Actions, these Actions are primitive building blocks from which verbs describing complex chains of these Actions can be built. These Actions ares

PROPEL	GRASP
HOVE	INGEST
EXPEL	SMELL
SPEAK	LISTEN-TO
LOOK-AT	MTRANS
ATRANS	PTRANS
CONC	MBUILD

These Actions are described in detail elsewhere [7]. The most important are MTRANS (transfer of mental information)! PTRANS (change of location)t and ATRANS ttransfer of possession).

Uhat is important about the system described here it that for the examples it does, it functions the way we would claim an ideal language understanding system must. Namely, the analysis of the input is done to a deep conceptual level using conceptual information rather then by doing a syntactic analysis first. Then it is operated upon using the language-free nature of this depth, and then repacked into English.

5. The Analuzer

The natural language analyzer attempts to extract the conceptual dependency representation underlying a sentence. There are several things that distinguish the approach to parsing described here from others. The first matter is the use of a predetermined representation of the meaning of sentences, i.e. conceptual dependency. The determination of the syntactic structure of a sentence is an instrument in the primary task of figuring out what that sentence means. In fact syntax has been denigrated to the status of something to use when all else fails.

Our analyzer also differs from traditional parsers in having been written with an emphasis on uhat individual words can and do communicate, not on uhat possible syntactic structures there are and what they mean. Thus the analyzer does not look for the presence or absence of a particular structure by template matching nor by feature recognition. Rather those words that might appear as features for the structure have attached to them programs that perform the task which would be considered as the meaning Df the structure. For example, the analyzer does not look for an active or a passive construction. Attached to the various forms of the verb "be" is an instruction that says that the following words will be telling some fact about the subject and further if the verb form following is a past participle (i.e. a verb form with indications of tense) then something is being told about what happened to the subject. Implicit then in the instructions attached to "be" is the active /passive distinction but it not looked for in those terms but rather is part of a general semantic function of "be" just like "John is dead."

The following description of the analysis of a sentence may help to clarify the processes involved. The sentence is "John told nary that Bill wants a book." The first word of the sentence, "John" tells the analyzer nothing about what is going to happen. but only gives a subject for some verb or predication to follow. Subject is a syntactic relationship that has semantic Importance only in that it is a place to save something while the conceptualization involving that something is yet unknown. There are other place-holders avallabia, like object and recipient which are useful to distinguish words from each other without making a commitment about any particular conceptual role they will play. Thus in "John gave Mary a beating," the word "give" assigns a syntactic role of recipient to "Mary" but eventually the conceptual role for "Mary" will be as the conceptual object of "beat". Syntactic roles thus serve a purely utilitarian function, highly specific to the needs of the verb involved. No semantic reality is attached to them and thus one verb's USE of what the analyzing program labels an object will have no necessary relationship to uhat some other verb does with uhat it calls an object. A verb needs only enough places to save those things for which it hasn't determined conceptual roles.

Returning to the sample sentence, following "John" we have the verb "told". Now this word tells us many things. It tells us that there is a communicative act involved (which is called MTRANS in conceptual dependency). By its morphology it tells us the act took place before the time of speaking. It tell us that the subject of the verb is the actor of the action and also that the information was originally in the active part of his mind, called the Conscious Processor (CP). It tells us that if s human follows then that person's *CP* is the conceptual recipient of the communication. It tells us that there will be a another conceptualization,like "Bill wants a book," or at least a word signifying a set of conceptualizations, liks a "lie" or a "story", which will follow soon.

The next word ,"Mary", satisfies the expectation of a recipient for the action. Like "John" the major action on the part of "Mary" is to say it is a human. Out of context it doesn't add any new expectations to the handling of the sentence.

The next word, "that", is one that does have functional effect on the analysis of the sentence. It says that if it is followed by a simple noun it is being used to say that the object following, e.g. "that joke", is one the hearer should know about, either from past knowledge or from information that is about to come. If "that" is followed by anything else it says that the conceptualization expected by the previous verb is being started and the analyzer should be prepared for a new set of syntactic and conceptual roles and role fillers.

The next word,"Bill", is not a simple noun and so the analyzer, obeying uhat "that" told it, assumes that "Bill" is a new subject, just like "John" was. Other than that and information about the starting point for tense determination the analyzer knows nothing, out of context.

The next word, "wants", says quite a bit. As should be obvious the verbs and acts and function words .like "that" and prepositions .are the driving fore behind the analyzer. They setup conditions that tell where what has come and uhat is predicted to cone should fit and the nouns and adjectives and previous context serve as the data base for these instructions to operate upon. "Wants" gives us the conceptual information that the topic of the sentence is something that "Bill" believes will cause him pleasure. In Conceptual Dependency a person's belief of some conceptualization is rspresented as the presence of that conceptualization in the passive part of his mind, called the Long Term Memory (LTfl). "IJante" tells us that if a verb form "to X" follows then that is the action Bi11 would like and further either he is the actor or, if the "to" was preceded by another person, then that person is the actor of the action specified by the verb. If the only thing that follows "wants" is a noun phrase then the action that would give Bill pleasure must be assumed from the nature of the object and the context. There ars othsr options following "wants" of course but the above Includes the relevant one.

The next word is "a" which causes a syntactic holding to occur in the analysis. That is "a" tells the analyzer to collect from the sentence first a complete phrase, ending with a noun, attach the preceding adj ectivs and nouns to this final noun w1th the appropriate conceptual relationships and then pass this result back to the list of expectations set up by the previous function words Df the sentsnce. The finding of a phrase boundary is not independent of the context of the sentence, nor is it a clear cut process, as can be seen

from the two sentences, "The city people like is New York" and "The city people like New York." But then very l.ittle in the comprehension of sentences)e clear cut.

The next uiord, "book", is absorbed into the phrase being built, but we don't know yet if the phrase is ended or if "end" or "marker" or something will follow.

The next thing to come is the end of the sentence. This has a major effect on many analyses. The various function words have provisions for what kind of assumptions to make if the sentence ends before they get all the information they need. The first use of the sentence end is by the instructions attached to "a". The article recognizes that the end of a sentence ends a phrase and so the concept of a "book", noted as being indefinitely referenced, is passed back to the instructions attached to "wants". These instructions, seeing that the sentence is finished and seeing that no action has yet been specified involving the book , make an assumption that Bill wants to have the book. This is a general assumption made with all inanimate physical objects. Another assumption that might have been made if the object were a person is that Bill wanted that pernon to come to him. Uhen "wants" has taken care of all the business that it needs to, the structure of Bill wanting a book is passed back to the instructions attached to "told". Since everything that "told" has said was crucial has been done the analysis of the sentence is finished. The final result follows, first as the analyzer outputs it and then as one normally finds it represented in articles on conceptual dependency.

(JOHN TOLD MARY THAT BILL WANTS A BOOK)

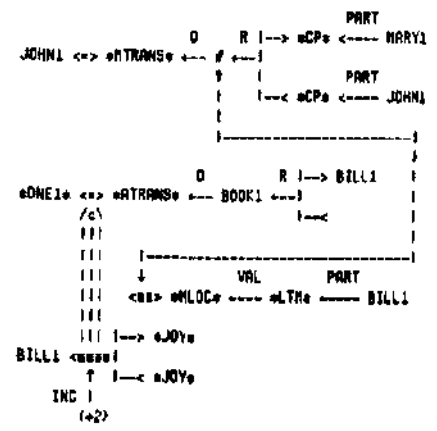
((ACTOR (JOHN1) <=> (*MTRANS*) TO (*CP* PART (MARY1)) FROM (*CP* PART (JOHN1)) MOBJECT ((CON ((CON ((ACTOR (*ONE1*) <=> (*ATRANS*) OBJECT (BOOK1 REF (*A*)) TO (BILL1)) TIME (TIM03)) <=C ((ACTOR (BILL1) <=>T (*JOY*) <=>F (*JOY*)) INC (2) TIME (TIM02)))) <=> (*MLOC* VAL (*LTM* PART (BILL1)))) TIME (TIM00))) TIME (TIM01))

TIM00 : ((VAL *T*))

TIM01 : ((BEFORE TIM00 X))

TIM02 : ((AFTER TIM00 X))

TIM03 : ((AFTER TIM00 X))



We have seen now what sort of concerns have guided the construction of this analyzer. The stress has been on the finding of the meaning, not the structure, of a sentence. The processes available have been ones based on a left to right handling of sentences, where each word contributes something to the choices to be made and the analysis program itself is basically a monitor executing these contributions. Further not only is the meaning the ulitmate goal of the analysis but meaning in conceptual representation, is the major source of information to guide the getting of more meaning. For example, the fact that a communication was involved makes the analyzer look for another conceptualization, not the particular syntax of "tell".

This approach has not solved immediately any of the classic problems of ambiguity and contextual relevance, Uhat it does do is provide a format where the solutions to such problems can be formulated in terms that match commonsense descriptions of how people figure out meanings.

The sentence whose analysis was followed above will be discussed further with respect to reasonable inferences that can be made upon it. The section on generation usee different examples which give a better indication of the paraphrase capability of the generator. These other examples can also be successfully analyzed by the program and the analyses differ in no fundamental way from what has just been described. An example is "John advised Mary to drink the wine." "John", "Nary", and "wine" are like the nouns in "John told Mary that Bill wants a book." "Advised" is the communicative act HTRANS as is "told", wi th the added expectation that the word "to" will probably introduce an action for the recipient to perform, and further if the recipient performs this action the recipient will be better for it. Hence the analyzer's output for "John advised Mary to drink the wine" is ;

((ACTOR (JOHN1) <=> (*MTRANS*) 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 (NIL)) MODE (NIL) TIME (TIM03) FOCUS ((ACTOR))) <=C ((ACTOR (MARY1) <=>T (*JOY*) <=>F (*JOY*)) INC (2) TIME (TIM02) MODE (NIL))))) FOCUS ((ACTOR)) MODE (NIL) TIME (TIM01))

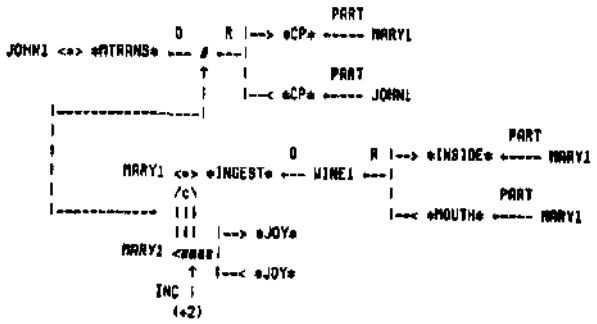
TIM00 : ((VAL *T*))

TIM01 : ((BEFORE TIM00 X))

TIM02 : ((AFTER TIM01 X))

TIM03 : ((AFTER TIM01 X))

which graphically is the following:



6. Memory

MEMORY's principal tasks are the following:

(1) to convert the format of the input, as just described, to positional internal memory form. This process also involves the linking-in of immediately-identifiable concepts with the occurence sets of those concepts (occurence sets are explained shortly).

(2) to establish references to tokens of real world concepts for as many of the other concepts and tokens as possible from the descriptive set returned for each from the analyzer. An example of a descriptive set for "John" is:

X: ((ISA X #PERSON) (MALE X) (NAME X "JOHN"))

When the descriptive set unambiguously identifies some

#JOHN in MEMORY, this #JOHN ie used In the internal form. Uhen no #JOHN satisfies the descriptive set, or when severel do, a (possibly temporary) token i• created, storing the facts in the descriptive set as its occurence set, and noting this event on the Hat IUNKNOUNREF.

(3) to extract subpropositions from the conceptualization at points before and during infereneing* Those extracted before usually cone from REL link («-», as in "the red dog", (DOG «-» (ACTOR (DOG) <■> (COLOR VAL (RED))))) embeddings, and, together uith the main propoeition, comprise the initial list of propositions submitted to the breadth-flrtt inferences

(4) to generate inferences of five basic types from each subproposition. These five types are ("X -* Y" means "Y ie inferred from X"h

(a) NORMATIVE. Uhat is the normal stats of affairs in the world? EXAMPLE! "Mhere is John at 9AM Tuesday?" "Probably at work."

(b) PERIPHERAL. Uhat "surrounds" a sltuationt uhat do people automatically assume uhen hearing something? EXAMPLE! "John told Mary that Bill saw Rita." - "John knows that Bill saw Rita."

(c) CAUSATIVE. Uhat uere probable causes of some state or action. EXAMPIEt "John hit Mary." * "John ua« *ad at Mary."

(dt RESOLUTIVE. Uhat are the probable results of some state or action? EXAMPLE: "John bit Bill"-. "Bill is hurt."

(a) PREDICTIVE. Uhat might an actor do, given his current state? EXAMPLE: "John wants an icecream cone." -» "John might go to the store."

A very important goal for making inferences ie to analyze an input on both the ACTUAL level (uhat the conceptual dependency diagram conveys literally) and the level of INTENTION of the actors (why did they do what they did.) For all inferences, particular attention is accorded to maintaining a record of uhat generated what. This means that for any proposition in MEMORY, there is a list of other propositions which participated in its generation, and a list of propositions in whose generation It has participated. Also, for RESULTATIVE and CAUSATIVE Inferences, in addition to the new inference, a (CAUSE X Y) relationship is generated between the old and new information.

(S) to make use of RESULTATIVE and CAUSATIVE inferences to fill in missing causal chains. For example, the analysis of "Mary kissed Bill because he hit John." wouId reeu11 in a causal whieh stood for severel intervening unspecified causa 1st (a) Bill'* hitting John caused John to be hurt, (b) John's being hurt pleased Mary (a peripheral inference is that Mary feels a negative emotion toward Bill), (c) Mary's pleasure was caused by John's action, (d) Mary therefore feels a positive emotion toward John, le) this causes her to kiss John. This is a very common and important MEMORY function, and is called CAUSAL CHAIN EXPANSION.

(5) to detect when the same inferred information astsee from two distinct sources (for example, uhen a PREDICTIVE inference mads from one line of a etory is confirmed by some later story l ins). This is cal led KNITTING (see *M),* end generally indicates that MEMORY has, by inference*, been able to show how information in a story le connected. A vary simple example of this ie the folloulngt

"Mary wae hit by a car."
"Sha want to the hospital."

or

"John wanted a wrench."
"Ha went to the hardware store."

uhere, inferring that Mary was badly hurt, MEMORY predicts that she will go to the hospital.'and this is immediately confirmed by the next line (notice that this prediction would help understand "The nurses usre very kind." if this sentence were the second line of this story).

(7) to detect and try to fill in missing or unspecified concepts (tokens) or events during Inferancing. This includes supplying information to the analyzer in cases such as "John hit Mary", where, in the abaence of other information, the object of the PROPEL ACT involved in the hit ie inferred (by a NORMATIVE inference) to be (HAND PART (JOHN)). In the sequence "John picked up a rock." "He hit Mary.", MEMORY would apply the knowledge that people hit other people with whatever they are currently holding, and would in this case return (ROCK *-• (ACTOR (ROCK) <«> (*L0C* VAL (HAND PART (JOHN))))), In the example which will be described shortly. MEMORY will detect a missing concept during inferencing, and will dsk a qusstion about it.

MEMORY relies on two main data types for the storage of conceptual information and concepts and their tokens: CONCEPTS and BOWS- A bond is a "8t of concepts which is stored with property BONDVALUE under a system-generated atom, called a SUPERATQM. A concept, or a token of a concept is simply a LISP-generatsd atom. If it has a name, the name ie stored in the same way all other conceptual information about the concept is stored. Both bonds and concepts have an OCCURENCE SET, whose value is a list of euperatoms under which are stored bonds in which X occurs. MEMORY is therefore a fully inverted structure. Conceptually, the occurence set of a bond or concept is a catalog of all conceptual knowledge about that bond or concept. This form of data base facilitates rapid lookup and simplifies simulation of parallel associative searches.

In addition to its occurence set, each bond, X, has associated with it two very important propertiee: its REASON SET and OFFSPRING SET. The reason set is the list of other proposi tions in MEMORY which contributed to the generation of X, and the offspring set is a list of propositions in whose generation X has participated. Thus MEMORY is preserves lines of reasoning as well as the facts lying along those lines. Other properties of bonds and concepts are: TRUTH (is this fact currently believed or not), RECENCY (when was this bond or concept last accessed), and STRENGTH (if believed, with uhat strength). Strengths are propagated and can be dynamically updated when the strength of propositions on some bond's reason set changes.

The flow of information in MEMORY in response to conceptual input ie as follows. The graph is transformed syntactically into internal positional notation. During this process, subproposi tions communicated by REL *(*■■*)* links and main-conceptualization modifiers (like tine and location) are noted as potential subpropositions. Also during this phase, direct references to predicates and concepts are immediately linked into the correct occurence sets. This includes the creation of new time tokens. Next, as many references to tokens are established from descriptive eets as are possible. Potential REL subproposi tions used in this step are processed no further. Possibly temporary tokens are created for ail unidentified referents (using the available descriptive sets), and bonde are created for the structure's propositions and stored. At this point, there is the main conceptuallzation, a list of subproposi tions, and a list of unidentified references.

The main conceptualization and subpropositions are eubmitted in parallel to the inference mechanism. Thie Is a simple breadth-first monitor which looks at the predicate of the proposition for which inferences are desired, and locates the INFE,R£NCJi MOLECULE for that predicate. No pattern matching is done in the Monitor, only in inference molecules. The monitor also co11ects as much TIME information a* can be found about the bond into a condensed form and makes it available to the inference molecule. All inference molecules are time and context sensitive. The molecules are highly structured LISP PROGs which the moni tor executes.

The reeult of executing an inference molecule is a list of inferences (which are pointers to newly created bonds in MEMORY), organized by inference TYPE and a simple INTEREST measure. Inferences are reordered, cut off below certain interest and strength levels, and stored on the master list. After inferencing has ceased, the monitor rescans the master .list to detect actions, asserts the volition of actors by assuming the actor WANTED the results of hie action, and submits these new WANT propositions to the inferencer again.

As each 'inference is generated, an "evaluation function" is applied to it to detect one of three situations; confirmation, contradiction or augmentation. These are the heart of the causal chain expansion and knitting mechanisms described above, but are beyond the scope of this paper (see 14), Also during the inferencing, the list of "unident)fled concepts mag be augmented by some new inference requiring knowledge which MEMORY cannot guess (the example will enow such a case). These go on 'UNK'NOUNREF. There is a similar list, called 'UNKNOUNCON on which missing actions or states (which MEMORY cannot predict) are noted.

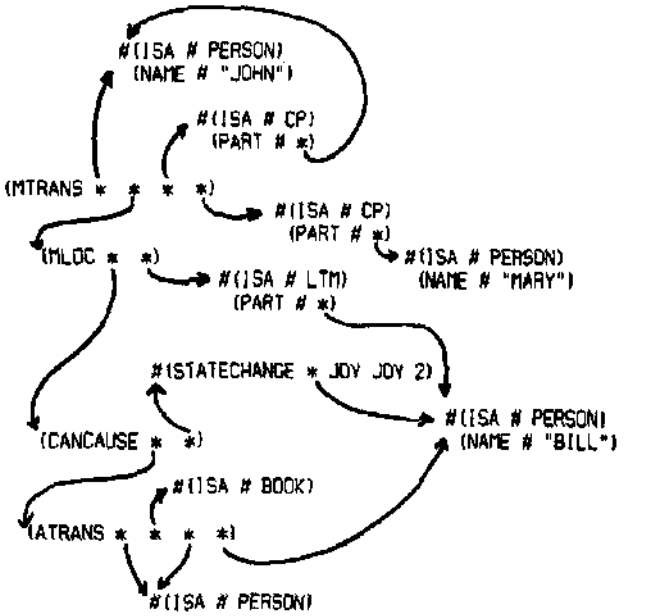
After all inferencing ceases, the referencer is re-entered in the hope that the inferencer has generated new information about unidentified references at the beginning. If any more can be solved at this time, the inference mechanism is re-app1ied, not duplicating previoue work since inference molecules remember which inferences were successful on the first pass. This second pass is done in case the now-accessible occurence sets of the newl y identified concepts will open new inference paths.

After these processes, potential responses come from the following sources: interesting inferences, IUNKNOWNREF and IUNKNOUTCON. At present there are no heuristics for deciding what to say, so MEMORY merely dumps everything on the generator for expression. For each conceptualization to be expressed, this involves conversion from internal to external format. Part of this is just syntactic, but part is deciding what information to include about each bond and concept since there are generally many members of each concept's and each bond's occurence set. This is a theoretical issue which has not been addressed by this research. Hence, MEMORY uses some fairly simple heuristics for deciding what to include in the expression of each bond and concept. Examples of what to include are: for all concepts, a NAME if possible; for bodyparts, who is it part ofj for a state, its begin and end times (or just a time); for an action, time, mode, location, and so on.

The proper handling of time relatione has been a central issue in this project. In MEMORY, this means that, as well as time-sensitive inference molecules, there must exist proof procedures for determining BEFORE (and otherl relationships. Also this means that outdated propositions can be kept around by proper maintenance of their time dependencies. The implications of this approach from the standpoint of the frame problem are discussed in *W.*

A summary of what happens to the parser output from the sentence "John told Mary that Bill wants a book." will now be given. We will not pursue the problems of reference establishment, but assume a perfect internal form can be created immediately. Only the inferences which go to the generator will be shown.

The internal form which results from this input is: (time relations have been omitted for clarity)



The resulting set of inferences generated (their content) i s;

(P: peripheral, R: resultative, C: causative, PR: predictive, N: normative, MIOs missing informal ion question)

(P) John believes that Bill wants a Dook.
(People generally believe what they communicate to others.)

(R) Mary, now Knows that Bill wants a book..
(People normally believe factual information they are told.)

(P) Bill wants a book.
(MEMORY believes what it hears too.)

(P,R) Bill wants to come to possess a Book.
(People want actions because of the predictable results of those actions. One certain result of an ATRANS is that the recipient acquires possession of the object.)

(P.R) Bill possibly wants someone else to cease to have a book,
(Another result of the ATRANS is that the donor ceases to have the object, This has a much lower strength than the previous one.)

(N) Bill wants to read a book.
(People usually want to have objects to use them in their normal function.)

(P;R) Bill wants to know the concepts contained in a book.
(This would be a result of reading a book.)

(MIQ) A book about what?
(The concepts predicted by the last inference are not known. Other useful inferences could be generated if MEMORY knew what the book is about.)

(PR) Bill might get himself a book.
(Knowing someone's state, he may be predicted to do certain things.)

(PR) John might give Bill a book.
(Knowing someone's wants, another person might attempt to satisfy them.)

(PR) Mary might give Bill a book.
(etc.)

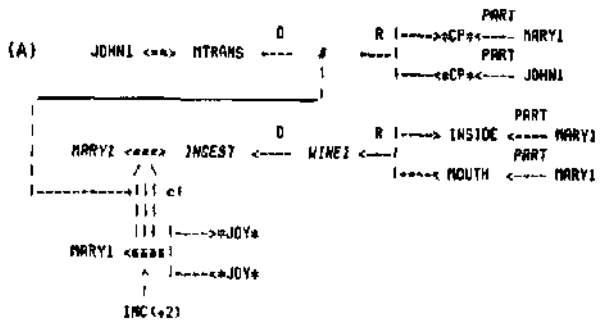
(C) John may want Mary to give Bill a book.
(A person's motivation for communicating a want might be to have that want satisfied.)

Memory passes each of these in turn to the conceptual generator.

7. Generator

The third major component of this system is the generator, which has the task of expressing arbitrary conceptual structures in surface lexical form. This is accomplished by two sequential transformations of representation. The first pnase produces a syntactic "case" network from the conceptual representation. The second phase is carried out by our version of Simmons' program, uhich uses an augmented finite-state transition network to map these case networks onto surface forms,

When the memory model decides it wishes to express some information as natural language strings, it passes a list of conceptualizations containing that information to the generator. The generator is capable of expressing the inferences made by the memory model from our example sentence; however, to better explain how generation is accomplished and to demonstrate the paraphrase capabilities of the program, it will be useful to consider the second example mentioned in sections. When paraphrasing, the generator might receive the encoding produced by the parser for "John advised Mary to drink the wine". The structure which the generator works with is:



The first step of the generative process is to obtain the (English) syntax network for this structure:

(B)	N1:	LEX	ADVISE	N4:	LEX	DRINK
		ACTSBJ	N2		ACTSBJ	N3
		OBJ2	N3		OBJ:	N5
		INF	N4			
	N2:	LEX	JOHN	N5:	LEX	WINE
	N3:	LEX	MARY			

This network consists of a set of nodes (N1.N2,...) each associated with a set of syntax relation-value pairs. Both the tokens ~ e.g., JOHN1 — and the relation© — e.g., 0BJ2 -- of (B) are assumed to be language-dependent. The tokens in (B) are not Engl ish words. They are, however, entries in an English lexicon, which can be easily mapped (by dictionary look-up) onto English words. The relations are chosen because of the syntactic rules of English. In order to create the syntax net, the generator must choose the proper lexical entries and syntax relations based on the contents of the language free conceptual structure.

The simple conceptual ■» lexical transformations JQHN1 —> JOHN and UINE1 —> UINE *are* accomplished via dictionary look-up in the current program. It is the method of discovery of the tokens ADVISE and DRINK, which do not correspond to any isolated conceptual units in (A), uhich makes the generation process both interesting and pouerful.

The generator has available to it a set of discrimination nets through which conceptual structures are filtered to discover word sense units. Each discrimination net is a binary tree, with predicates at branching nodes and word senses at terminal nodes. The predicates are of three distinct types:

1. pattern matching within a conceptual structure
2. high-level definition -- does a substructure of the conceptualization satisfy the predicates of a particular word sense
3. memory queries -- requests for information from world model, possibly requiring deduction

Given a conceptual structure, the generator must first find an appropriate discrimination net. The skeletal structure of the conceptualization is checked (<==>, <==>, etc.). In some cases the structure alone specifies the proper network. In example (A) above, we have a simple event, and the primitive ACT involved (here, MTRANS) is checked to determine which net to use. Figure 1 shows a portion of the network which would be chosen for this example.

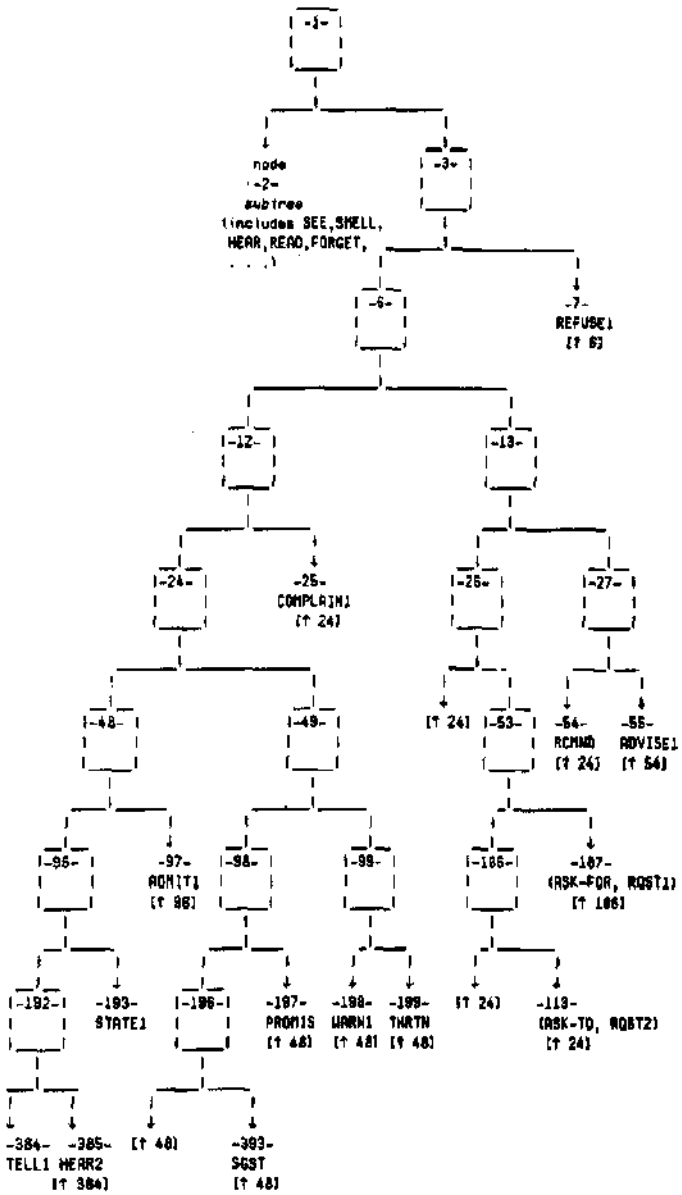


Figure 1

When this net is applied to (A) initial tests at node 1, which discriminate representations of 'perception' verbs from 'communication' verbs, pass control to node 3. Here it is determined that REFUSE1 is not an appropriate sense for (A) and control passes down to node 6. A test here discriminates a class of verbs meaning "communicate that an event would benefit someone". (A) falls into this

meaning class, so control passes to node 13. At this node a choice based on the identity of the "potentially benefitted" individual is made. Since in our example this is the RECIPIENT of the MTRANS (MARY1), the branch leading to A0VISE1 and RCHN0 (a sense of "recommend") is taken. Eventually terminal node 55 uith the word sense AOVISEl is reached.

'ADVISE1' has associated uith it a lexicon entry (ADVISE) and a framework — a list of syntax relations with a specification of a substructure of the conceptualization where the value for each can be found. For 'AOVISEl* the framework contains the pairs:

RELATION	CONCEPTUAL LOCATION
ACTSBJ	ACTOR
0BJ2	RECIPIENT
INFinitive	OBJECT

Each of these substructures is then used as an argument to the generator program, resulting in the construction of syntax net (B). Further processes add such information as TENSE, VOICE, and determiners to the syntax net.

Thie exatnple has shown how the generator may reconstruct a syntactic case representation of an English input directly from its conceptual representation. It is not necessarily the case, however, that the syntax net derived will correspond directly to the original input. There are two independent sources of the paraphrase capabilities of the generator. The first is implicit in the theory on which this form of generation depends. Since the conceptual networks from which the generator works are

(1) language-free, and
(2) unique representations of meaning,

the generator is not tied to the same vocabulary as the parser, nor even to a synonymous one. In generating (B) from (A), there is no clue that the word 'advise' was used in the original statement.

There is also an explicit paraphrase capability in the generator. Terminal nodes may contain, in addition to word-sense tokens, pointers back into the discrimination net. (These pointers are written as It node-index] in Figure 1. In fact, some terminals contain only such pointers.) In paraphrase mode, the control algorithm may ignore a word-sense token at a terminal and resume the filtering operation at the node indicated by the pointer, thus finding a different 'word-sense' for the conceptual (sub-) structure. This sense may have a very different lexical unit and syntax framework associated with it, resulting in a syntax net differing in many aspects from that previously obtained. If 'AOVISEl'were ignored in the preceding example, filtering would resume at node 54 of the discrimination net, and the head 'RCtlND' (among others) would be found. The following paraphrases are among those obtained for conceptualization (A);

John recommended to Mary she drink the wine.
John suggested Mary would like to drink the wine.
John told Mary she would enjoy drinking the wine.

No further description of the surface generation phase will be given here. It Bhould be emphasized, however, that the relations we are dealing with in the syntax nets have only syntactic significance to the process, and no semantic properties are ascribed to them. All 'meaning' is treated at the conceptual level; thus the notion of a semantic net as proposed by Simmons has no place in MARGIE'S generation system. In addition to obvious theoretical implications of this difference, there is the practical consequence that we can, in general, expect a simpler netuork grammar to handle a given subset of a language. Consider, for example,

(a) John broke the window.
(b) John remembered to go home.

The 'eemantic' deep case interpretation may involve marking John as an AGENT (instigator of an action) in (a), but ae a DATive or EXPeriencer (one affected by an action) in (b). We can mark 'John' as ACTSBJ in both caees, since the only property of the relation in which we are interested it that it becomes the subject NP in active sentences.

8. Conclusion

MARGIE it Intended primarily to be the beglnninge of a larger more complex system for language understanding. Many effective natural language understanding programs. work In severely restricted domains. Thus, many of the techniques that *work, for example,* in Woods* soon rock* system [13J or Winograd'e [12] blocks world are not necessarily expandable to a larger domain. Ue have felt that a consistent conceptual representation is very important before attempting computer understanding. Our analyzer utilizes the properties of this system to aid its analysis of input sentences. The memory, which certainly owes much of its design to notions brought forward by Quil Man [31 and Becker [1], operates on these conceptual representations, thus simplifying the problem of handling identical information by allowing only one possible deep format for a given meaning. Inference is thus more straightforward because of thB very few primitive actione that exist in the system (see [7] and [8]. By using a deep conceptual system of the type we propose, the EngliBh-like quality of the input is lost. Untie this facilitates understanding it also creates the problem of recoding information back into English. *Um* have found Simmons' program to be helpful in this regard, but most of the problem was left unsolved by hie program eince his representations are considerably less deep than ours.

Ue have found MARGIE encouraging uith reapsct to the prospects of a larger system and thus we stress that while our computer results are interesting, it is the theoretical design of the system that is important for future efforts.

9. References

1. Becker, J. 'A Model for the Encoding of Experiential Information' in Schank and Colby (eds), Computer Models of Thought and Language, W.H. Freeman & Co., San Francisco 1973.
2. Goldman, N. 'Computer Generation of Natural Language from a Deep Conceptual Base' Ph.D. thesis, Computer Science Dept., Stanford University, Stanford Cal., 1973.
3. Quillian, R. 'Semantic Memory', Bolt Beranek & Newman, Cambridge, Mass., 1966.
4. Rieger, C. 'Conceptual Memory', Ph.D. thesis, Computer Science Dept., Stanford University, Stanford Cal. 1973.
5. Riesbeck, C. 'Computer Analysis of Natural Language in Context', Ph.D. thesis, Computer Science Dept., Stanford University , Stanford Cal, 1973.
6. Schank, R. 'Conceptual Dependency: A Theory of Natural Language Understanding' in Cognitive Psychology vol. 3, no. 4, 1972.
7. Schank, R. 'The Fourteen Primitive Actions and Their Inferences', Stanford AIM-183, Computer Science Dept., Stanford University, Stanford Cal. 1973.
8. Schank, R. and Rieger, C. 'Inference and the Computer Understanding of Natural Language', Stanford AIM-197, Computer Science Dept., Stanford Cal, 1973.
9. Schank, R., et al 'Primitive Concepts Underlying Verbs of Thought', Stanford AIM-162, Computer Science Dept., Stanford University, Stanford Cal., 1972.
10. Simmons, R. 'Semantic Networks: Their Computation and Use for Understanding English Sentences' in Schank & Colby (eds) Computer Models of Thought and Language, W.H. Freeman 1973.
11. Simmons, R. and Slocum, J. 'Generating English Discourse from Semantic Networks', Communications of the ACM, vol. 15, no. 10, October 1972.
12. Winograd, T. Understanding Natural Language , Academic Press, New York, 1973.
13. Woods, W. 'Transition Network Grammars for Natural Language Analysis', Communications of the ACM, vol. 13, no. 10, October 1970.