A CONCEPTUAL PARSER FOR NATURAL LANGUAGE*

Roger C. Schank and Lawrence G. Tesler

Computer Science Department
Stanford University
Stanford, California

This paper describes an operable automatic parser for natural language. It is a conceptual parser, concerned with determining the underlying meaning of the input utilizing a network of concepts explicating the beliefs inherent in a piece of discourse.

1. Introduction

The parser described in this paper is a conceptual parser. Its primary concern is to explicate the underlying meaning and conceptual relationships present in a piece of discourse in any natural language. Its output is a language-free network consisting of unambiguous concepts and their relations to other concepts. Pieces of discourse with identical meanings, whether in the same or different languages, parse into the same conceptual network.

The parser is not a syntactic parser in that its output is not concerned with the syntax of the input language. It bears some similarity to certain deep structure parsers4,7,12,13 only insofar as all these parsers are concerned to an extent with the meaning of the piece of discourse being operated upon. However, the conceptual parser is not limited by the problems inherent in transformational grammar (such as the difficulty in reversing transformational rules and the notion that semantics is something that 'operates' on syntactic output). Also, the parser does not have as a goal the testing of a previously formulated grammar7,13 so that the underlying theory has been able to be changed as was warranted by obstacles that we encountered.

The intention of this work is to handle natural language utilizing a semantics-based system, and thus our paper bears some similarity to the work of Quillian". However, the conceptual dependency framework, though semantics-based, is intended to function as a more complete linguistic system. Thus, a grammar of a language is employed.

The grammar of the system is bipartite. The first part is a universal grammar exemplified by the conceptual rules employed by the system. The second part is language-specific and is made up of realization rules intended to map pieces of the conceptual network into linguistic items. The realization rules may be used for both parsing and generating. However, it is not necessary to use all the realization rules in order to parse. That is, the system is capable of making sense of a piece of language containing only a few words that it knows since its conceptual framework is capable of making predictions. Thus, it can understand while using only a few realization rules, whereas it would need a great many more to map the same structure back into language. This phenomenon is similar to that observed in a man attempting to learn a foreign language.

The network produced by the parser contains conceptualizations. A conceptualization is a statement about a single conceptual subject. The subject may be abstract or concrete; it may be one thing or a combination of things. The statement may tell what the subject is, what it does, what is done to it, etc. Furthermore, the entire conceptualization may be qualified as to time and place of occurrence, reasons, causes, consequences, and explanations. The numerous conceptualizations in a discourse are interrelated not only by casual, logical, spatial and temporal ordering, but also by anaphoric references and by multiple mention of the same concepts.

Several conceptualizations may occur in a single sentence, and words from several sentences may be required to complete one conceptualization. Furthermore, information from some conceptualizations in a discourse may serve to disambiguate the interpretation of other conceptualizations. Consequently, the parser is not inherently sentence-bound.

2. Domain and Capabilities

The parser is being used to understand natural language statements in Colby's on-line dialogue program for psychiatric interviewing , but is not restricted to this context. In interviewing programs like Colby's, as well as in question-answering programs, a discourse-generating algorithm must be incorporated to reverse the function of the parser. The conceptual parser is based on a linguistic theory that uses the same rules for both parsing and generating, thus facilitating man-machine dialogues.

In an interviewing program, the input may contain words that the program has never encountered, or which it has encountered only in different environments. The input may deal with a conceptual structure that is outside the range of experience of the program, or even use a syntactic combination that is unknown. The program is designed to learn new words and word-senses, new semantic possibilities, and new rules of syntax both by encountering new examples during the dialogue and by receiving explicit instruction.

*This research is supported by Grant PHS MH 06645-*0?* from the National Institute of Mental Health, and (in part) by the Advanced Research Projects Agency of the Office of the Secretary of Defense (SD-I83).

3• Underlying Theory

The parser is based on the Conceptual Dependency model of language9, in this model, a conceptualization is a network of linguistic concepts[10] that fall into the following conceptual categories;

Governing categories

PP An actor or object; corresponds roughly to a syntactic noun or pronoun.

ACT In English, corresponds syntactically to verbs, verbal nouns (e.g., gerunds) and certain abstract nouns.

LOC A noun denoting the location of a conceptualization.

T Denotes the time of a conceptualization.

Assisting categories

PA PP-assister; corresponds roughly to an adjective.

AA ACT-assister; an adverb.

Most words or idiomatic phrases in a piece of discourse represent one or possibly several concepts in the network, and therefore fall into the categories above.

However, connectives such as conjunctions, prepositions, punctuators, auxiliaries, and determiners are intended linguistically to represent the structure rather than the content of the network. They serve in a parser to disambiguate the parse. In a generator, they are generated so that the hearer will have clues to disambiguate the discourse. He thereby can understand what meaning was intended by the speaker.

The Conceptual Dependency model is stratidealional insofar as it involves a mapping from one level to another (e.g., see Lamb6). Its highest level is an interlingua consisting of a network of language-free dependent concepts. (The dependency considered here is partially derived from the notions of Hays3 and Klein[5], however the dependencies are not at all restricted to any syntactic criterion.) The linguistic process can be thought of, in Conceptual Dependency terms, as a, mapping into and out of some mental representation. This mental representation consists of concepts related to each other by various meaning-contingent dependency links. Each concept *in* the network may be associated with some word that is its realizate on a sentential level.

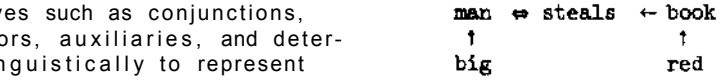
The conceptual categories and relationships of the system are arrived at by studying a one-by-one analysis, identical to the way in which a person hears a sentence. According to the theory, as each word is input, its representation is determined and then is stored until it can be attached to the concept which directly governs it. The rule-of-thumb in representing concepts as dependent on other concepts is to see if the dependent concept will further explain its governor and if the dependent concept cannot make sense without its governor.

For example, in the sentence, "The big man steals the red book," the analysis is as follows: 'The' is input and stored for possible use in connecting sentences in paragraph, i.e., in this case, 'the' specifies that 'man' has been referred to previously. Next, 'big' is input. But 'big' cannot stand alone conceptually, and it is stored until its governor appears. 'Man' can stand alone and is modified conceptually by 'big', so it is stored as a governor with its dependent.

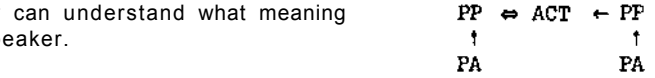
'Steals' denotes an action that is dependent on the concept that is doing the acting. A conceptualization cannot be complete without a concept acting (or an attribute statement), so a two-way dependency link may be said to exist between 'man' and 'steal'. That is, they are dependent on each other and govern each other. Every conceptualization must have a two-way dependency link.

Next, 'the' is stored as before as is 'red'. When 'book' is input, 'red' is attached to it as before, and the whole entity is stored as dependent on 'steals' as the object of the action (represented by a horizontal single arrow).

The entire structure is as follows:



The categories assigned to these concepts are as follows:



This *system,* with more symbols and categories added, has been made to effect a set of rules which can account for all possible conceptualizations. The relations inherent in these conceptualizations are intended to provide a *system* for representing the meaning of a sentence in any language, in language-free terms. (Language enters into the conceptual representation only in a naming capacity.) A list of the allowable conceptual dependencies is presented in the Appendix. In addition to dependencies, other relations are allowed by the theory, e.g., conjunctions, disjunctions and comparatives.

This conceptual framework can be used to write realization rules for any language and rules for English have been written[9]. The realization rules are responsible for placing the entities realized from the conceptual level in the correct grammatical order to form a sentence. These rules are then the parsing rules for a particular language.

The system for analyzing a sentence into its conceptual representation works backwards through the realization rules of a language. In places

where either of two rules could apply, the system can build upon both of them, aborting if one analysis leads to an impossible conceptual structure, or producing multiple analyses for ambiguous sentences. All conceptualizations are checked against a list of experiences to see if that particular part of the construction has occurred before. If the construction has not occurred, or has occurred only in some peculiar context, this is noted. Thus, in the construction 'ideas ⇐ sleep[1]', it is discovered that this connection has never been made before, and is therefore meaningless to the system. If the user says that this construction is all right, it is added to the memory; otherwise the construction is looked up in a metaphor list or aborted. The system thus employs a record of what it has heard before in order to analyze what it is presently hearing.

In order for the system to choose between two analyses of a sentence both of which are feasible with respect to the conceptual rules (see Appendix) a conceptual semantics is incorporated. The conceptual semantics is a data base which limits the possible conceptual dependencies to statements consonant with the system's knowledge of the real world. The definition of each concept is composed of records organized by type of dependency and by conceptual category of the dependent. For each type of dependency, semantic categories (such as animate object, human institution, animal motion) are delimited with respect to the conceptual category of a given concept, and defining characteristics are inserted when they are known. For example, concepts in the semantic category 'physical object' all have the characteristic 'shape'. Sometimes this information is intrinsic to the particular concept involved, for example, 'balls are round'.

The semantic categories are organized into hierarchical structures in which limitations on any category are assumed to apply as well to all categories subordinate to it. The system of semantic categories and a method of constructing semantic files is discussed more fully in a previous paper9.

In the present system, the files are constructed by incorporating information derived from rules presented as English sentences. The program parses each of these sentences, observes which dependencies are new, and then adds them to the files.

4. Conceptual Analysis

In the Conceptual Dependency theory on which our program is based, the parsing procedure begins by looking up the conceptual category of a word. If the conceptual category is either PP or ACT, the concept evoked by the word being considered is placed directly into the conceptual network. Otherwise the concept is queued until a permissible conceptual governor enters the system. Prepositions, conjunctions, and determiners are similarly queued. A permissible conceptual governor is one whose category is on the left hand side of a conceptual rule (see Appendix) and where the dependency effected by the use of that rule is allowed by the conceptual semantics. 11 (An example of the conceptual semantics is given in the Appendix.)

If a permissible conceptual governor has been found for a queued concept, the queued concept is placed in the network with the appropriate dependency. A governor is not placed directly in the network unless it satisfies the conceptual semantics with respect to its dependencies.

Ambiguous interpretations of a piece of discourse are built up if more than one conceptual rule may apply. If the conceptual semantics disallows an interpretation, the dependency is aborted. If more than one network remains after the conceptual semantics have been checked multiple analyses are built up. Semantic ambiguity (that is, multiple meanings for a word) is also handled by the conceptual semantics. If a word connects to more than one concept, each concept is checked for the appropriate possibilities of dependence. The concept that fits according to the semantics is the word-sense chosen. If more than one concept fits, more than one network is built up. The semantics are checked as each dependent is added, so it is possible to abort a multiple network at a later poin in the parse.

As an example of the strategy employed in the theory it is illuminating to follow the parse of an example sentence. Consider:

'The tall boy went to the park with a girl[1].

The machine tries to simulate the behavior of a human in perceiving this sentence. Thus, it is continually operating and making hypotheses as each word enters the system.

When 'The' enters, it is held in waiting as it may be a link to a previous mention of the next PP which will enter the system. That is, 'the' would be replaced by 'a' in an ordinary dialogue if its specific referent were previously unmentioned or unknown.

'Tall' is marked as a PA. It is therefore queued until a rule that uses PA as a dependent can apply. Since 'boy' is a PP, it is placed directly in the network. Previous networks generated by the user are searched for an instance of 'boy' to which the 'the' refers and a link is made if one is found. The conceptual rule 'PP <- PA' is keyed by the realization rule 'PA PP: 2;
1 2 ↑
1

the numbers are place markers representing relative position in the piece of discourse. The PA is below the line to indicate that this dependent is an attribute of its governor. The 'PP <- PA' semantics for 'boy' are checked to see if the conjunction 'boy ← tall' can exist. Since the system knows that any animal can have height the connection is allowed and our network looks as follows:

```
boy
 ↑
tall
```

(The check with the semantics is made at every connection but from now on we will only mention it when it is necessary.)

When 'went' is operated on it is transformed into *go-p' (p means past) and since 'go' is an ACT the realization rule that applies will connect it to a previous PP by a two-way link. The 'p' modifies the two-way link and is moved over it. We now have the following:

```
     p
boy ⇔ go
 ↑
tall
```

'To', the next word into the system, is queued since it may be a preposition or part of an infinitive. If the next governor encountered is not an ACT, 'to' is a preposition and is translated into a 'to' link.

4=

[illegible] cy. When the link is written horizontally, it represents a dependency between an ACT and a PP where the dependent is not the object of a direct action. Vertical prepositional dependency specifies additional information about a concept that is only indirectly an attribute of that concept (e.g., a location rather than a physical attribute). Prepositional links may have many different forms, each represented by a tag (e.g., "to", "of") written over the link. Prepositional dependency is different from simple dependency in certain specified ways, one being that strings of prepositionally dependent PP's may exist whereas this cannot exist with simple dependency.

When 'the' is encountered it is treated as before. 'Park' is marked as both a LOC and a PP_{LOC} However, the system will demand the PP interpretation since 'to LOC' is not allowed; by definition, LOC's can only modify two-way links. 'Park' is then placed in the network giving:

```
     p    to
boy ⇔ go ⇐ park
 ↑
tall
```

'With' is held in waiting as a link until the PP to which it connects is encountered. 'A' is ignored and the construction 'with girl'is made. We are are now faced with the problem of where to attach this construct. A problem exists since at least two realization rules may apply:

```
                    2                    1
'ACT PREP PP:  1 ⇐  3;  'PP PREP PP ⇑ 2
  1   2   3                          3
```

The problem is resolved by the conceptual semantics. The semantics for 'go' contains a list of conceptual prepositions. Under 'with' is listed'any movable physical object' and since a girl is a physical object the dependency is allowed. The semantics for 'park' are also checked. Under 'with' for 'park' are listed the various items that parks are known to contain, e.g., statues, jungle gyms, etc. 'Girl' is not found so the network (l) is allowed while (2) is aborted.

```
          p      to    with
(1)  boy ⇔ go ⇐ park ⇐ girl
      ↑
     tall

          p      to
(2)  boy ⇔ go ⇐ park
      ↑          ⇑ with
     tall        girl
           with
```

Although ' *= girl' is dependent on 'go' it is dependent through 'park'. That is, these are not isolated dependencies since we would want to be able to answer the question 'Did the girl go to the park?' affirmatively. In (2) the below-the-line notation indicates that it is the 'park with a girl' as opposed to another 'park'. [illegible] well be the case that this is what was intended.

The conceptual semantics functions as an experience file in that it limits conceptualization to ones consonant with the system's past experience. Since it has never encountered 'parks with girls' it will assume that this is not the meaning intended. It is possible, as it is in an ordinary conversation, for the user to correct the system if an error was made. That is, if (2) were the intended network it might become apparent to the user that the system had misunderstood and a correction could easily be made. The system would then learn the new permissible construct and would add it to its semantics. The system can always learn from the user[10] and in fact the semantics were originally input in this way, by noticing occurrences in sample sentences.*

Thus, the system purports to be analyzing a sentence in a way analogous to the human method. It handles input one word at a time as it is encountered, checks potential linkings with its own knowledge of the world and past experience, and places its output into a language-free formulation that can be operated on, realized in a paraphrase, or translated.

5. Implementation

The parser is presently operating in a limited form. It is coded in MLISP for the FDP-10 and can be adapted to other LISP processors with minor revisions. The algorithm used differs from the theoretical analysis given in the previous section because a computer program must deal with machine limitations and must cope with special cases that may be encountered in the input.

---

*Sylvia Weber and Kenneth Mark Colby of Stanford University prepared most of the initial data for the semantic files.

Rather than building up the network structure a little bit at a time during the parse, the program determines all the dependencies present in the network and then assembles the entire network at the end. Thus, the sentence 'The big boy gives apples to the pig.' is parsed into:

1) boy
   ↑
   big

2) boy ⇔ give

3) gives ← apples
   to
4) give ⇐ pig

and then these are assembled into:

boy ⇔ give ← apples to ⇐ pig
↑
big

The input sentence is processed word-by-word. After "hearing" each word, the program attempts to determine as much as it can about the sentence before "listening" for more. To this end, the network is built up a little at a time as each word is processed. Furthermore, the program anticipates what kinds of concepts and structures may be expected later in the sentence. If what it hears does not conform with its anticipation, it may be "confused", "surprised", or even "amused".

In case of semantic or syntactic ambiguity, the program should determine which of several possible interpretations was intended by the "speaker". It first selects one interpretation by means of miscellaneous heuristics and stacks the rest. In case later tests and further input refute or cast doubt upon the initial guess, that guess is discarded or shelved, and a different interpretation is removed from the stack to be processed. To process an interpretation, it may be necessary to back up the scan to an earlier point in the sentence and rescan several words. To avoid repetitious work during rescans, any information learned about the words of the sentence is kept in core memory.

The parse involves five steps: the dictionary lookup, the application of realization rules, the elimination of idioms, the rewriting of abstracts,and the check against the conceptual semantics.

The dictionary of words is kept mostly on the disk, but the most frequently encountered words remain in core memory to minimize processing time. Under each word are listed all its senses. "Senses" are defined pragmatically as interpretations of the word that can lead to different network structures or that denote different concepts. For example, some of the senses of "fly" are:

fly. - (intransitive ACT): what a passenger does in an airplane

fly_2 - (intransitive ACT): what an airplane or bird does in the air.

fly_3 - (PP): an insect

fly_4 - (transitive ACT): what a pilot does by operating an airplane.

fly_5 - (intransitive ACT -- metaphoric): to go fast.

fly_6 - (PP): a flap as on trousers.

If there are several senses from which to choose, the program sees whether it was anticipating a concept or connective from some specific category. Recent contextual usage of some sense also can serve to prefer one interpretation over another. To choose among several senses with otherwise equal likelihoods, the sense with lowest subscript is chosen first. Thus, by ordering senses in the dictionary according to their empirical frequency of occurrence, the system can try to improve its guessing ability.

The realization rules that apply to each word sense are referenced in the dictionary under each sense. Most of these rules fall into categories that cover large conceptual classes and are referenced by many concepts. Such categories are PP, PA, AA, PPLOC, PPT> LOC, T, simply transitive ACT, intransitive ACT, ACT that can take an entire conceptualization as direct object ("state ACT"), and ACT that can take an indirect object without a preposition ("transport ACT"). In contrast to most concepts, each connective (e.g., an auxiliary, preposition, or determiner) tends to have its own rules or to share its rules with a few other words.

A realization rule consists of two parts: a recognizer and a dependency chart. The recognizer determines whether the rule applies and the dependency chart shows the dependencies that exist when it does. In the recognizer are specified the ordering, categories, and inflection of the concepts and connectives that normally would appear in a sentence if the rule applies. If certain concepts or connectives are omissible in the input, the rule can specify what to assume when they are missing. Agreement of inflected words can be specified in an absolute (e.g., "plural") or a relative manner (e.g., "same tense"). Rules for a language like English have a preponderance of word order specifications while rules for a more highly inflected language would have a preponderance of inflection specifications.

Realization rules are used both to fit concepts into the network as they are encountered and to anticipate further concepts and their potential realizates in the networks. When a rule is selected for the current word sense, it is compared with the rules of preceding word senses to find one that "fits". For example, if "very hot" is heard, one realization rule for "very" is:

```
                    1
very    PA :        ↑
 0      1           0
```

where the tags "0" and "1" indicate the relative order of the word senses in the recognizer and identify them for reference by the dependency chart; "0" means the current word. One rule for "hot" is:

```
                  0
AA    PA :        ↑
-1    0          -1
```

The program notices that "very" fits in the "-1" slot of the "hot" rule and verifies that "hot" fits in the "1" slot of the "very" rule. Therefore, the dependency suggested by the chart can be postulated for the network:

```
hot(PA)
 ↑
very(AA)
```

If the rules for two adjacent word senses do not fit together, other rules are tried or more distant word senses are checked.

Whenever a dependency is postulated, it is looked up in an idiom file to see if it is an idiom or a proper name and should be restructured. Thus, the construct:

```
make
 ↑
up
```

is reduced to the single concept

make-up.

This idiom will be detected by the parser even if several words intervene between "make" and "up" in the sentence.

After eliminating idioms from the network, there still may be constructs that do not reflect language-free beliefs. The most conspicuous cases are caused in English by abstract nouns. Most such nouns do not correspond to PP's but rather are abbreviations for conceptualizations in which the concept represented is actually an ACT or a PA.

The program treats an abstract noun as a PP temporarily in order to obtain its dependents, because abstract nouns have the syntax not of ACT's but of PP's. After obtaining its dependents, the PP is rewritten as an entire conceptualization according to rules obtained from an abstract file. These rules also specify what to do with the dependents of the PP; they may be dependent on the entire conceptualization, dependent on the ACT only, or appear elsewhere in the conceptualization.

By way of example, the sentence:

Tom's love for Sue is beautiful.

leads to the following dependencies:

```
love(PP)          love(PP)
⇑  of             ⇑   for
Tom(PP)           Sue
```

After hearing "is", the program expects no more dependents for "love" (by a heuristic in the program), so it checks the abstract file and finds rules for "love" including:

```
      ACT
of ⇑   ⇑ for  :  (a) ⇔ 0 ← (b)
  PP   PP
 (a)  (b)
```

where "(a)" and "(b)" identify concepts without reference to sentential order. The network is now rewritten:

```
Tom
⇳ ⇔
love
 ↑
Sue
```

where the horizontal main link represents "is", waiting for a right-hand concept. When "beautiful" is heard, the network is complete, giving:

```
Tom
 ⇳ ⇔ beautiful
love
 ↑
Sue
```

The network above may be realized alternatively as either of the paraphrases:

That Tom loves Sue is beautiful.
For Tom to love Sue is beautiful.

In conceptual dependency theory, connectives like "that", "for","to", and "of" are cues to the structure of the network and need not appear in the network at all. The network above demonstrates such a situation. Conversely, portions of the network may be absent from the sentence. For example, the sentence:

It is good to hit the ball near the fence.

is parsed as:

```
          one
           ⇳  = good
          hit
           ↑
          ball
           ⇑   to
fence  ⇒  place
          near
```

Here, "one" and "place" are not realized. Notice that the relevant realization rule for "it" is:

$$\begin{bmatrix}\text{for} & \text{PP}\\ (\text{a0}) & (\text{a1})\end{bmatrix}\ \text{it}\ \ \text{be}\ \ \text{PA}\ \ \text{ACT}\ :\ \begin{matrix}(\text{a1})\,|\,\text{one}\\ \Updownarrow \Longleftrightarrow 2\end{matrix}$$
$$\begin{matrix}0 & 1 & 2 & 3 & & 3\end{matrix}$$

The square brackets indicate optional words. The tags "(a0)" and "(a1)" indicate that "for" precedes the "PP" but the whole phrase may occur in any position of the construct. "(a1)|one" in the dependency chart means that if "(a1)", i.e., "for PP", is omitted, and the subject of the action is not obvious from context, then the concept "one" is to be assumed.

The conceptual network should reflect the beliefs inherent in the original discourse in a language-free representation. The interlinguistic file of conceptual semantics is checked to verify that the dependencies are indeed acceptable. This check is made after abstracts have been rewritten.

After the five parsing steps are completed, the program proceeds to the next word. At the end of the sentence, it outputs the final network in two dimensions on a printed page or on a display console.

6. Examples of Algorithm

Only a few of the relevant realization rules will be shown in the examples.

Example 1

'John saw birds flying to California.'

Words	Realization Rule Patterns (for possible senses)	Dependencies (rectangle around new dependencies)
John	1: (all PP patterns) 1: (all PP patterns)	-
	2(to see, past tense) PP ACT PP:-1 ⇔ 0←1 -1 0 1	John ⇔(p) see ← PP (note: "to" means "tense" of ACT Number 0)
	PP ACT by PP:2 ⇔ 0←1 -1 0 1 2	
	3 (to sew): ... etc.	
birds	1:(all PP patterns)	see←birds
flying	1:a) PP ACT-ing:-1⇔0 1 0	birds ⇔ fly

But now there are two main links on one line. So, go back and try as object of "see":

b) -3PP
-2 ⇕ ACT-ing:
-1ACT 0

-3 ⇔ -1
-2
↑ t-1
-3 ⇔ 0

see ← birds ⇕ fly

to: 1:ACT to PP_{LOC} -1 ⇐ 1 (0)
-1 0 1

to fly ⇐ PP_{LOC}

2:to ACT ⇔ 1
0 1

etc.

California: 1:(all PP_{LOC} patterns)

to fly ⇐ California

Final output:

p
John ⇔ see←birds
⇕
fly
to ⇑
California

Example 2

John saw Texas flying to California

Words	Patterns	Dependencies
John	1:(all PP patterns)	
saw	(as above)	John ⇔(p) see ← PP
Texas	1:(all PP_{LOC} patterns)	see ← Texas
flying	(as above)	Texas ⇔ fly: rejected by semantics. Laugh and go back and try (1b):

John ⇔(p) see
↑
John ⇔(p) fly

(rest as above)

Final output:

John ⇔(p) see ← Texas
↑(p)
John ⇔ fly ⇐(to) California

Example 3

Jane ate the hamburgers in the park.

Words	Patterns	Dependencies
Jane	1:(all PP patterns)	
ate	1(eat-past tense):	John ⇔(p) eat

Words	Patterns	Dependencies
	PP ACT PP:-1 ⇔ 0←1 (to) -1 0 1	eat ← PP
	etc.	
etc.		
hamburger	1:(all PP patterns)	eat←hamburgers
in the park	1:ACT in PP:-1⇐1 (0) (in) -1 0 1	eat ⇐ park: rejected by semantics
	2:PP in PP:-1 -1 0 1 0⇑1	hamburgers ⇑in : park set aside as unlikely (would accept if no other alternatives)
	3: (PP ⇔ ACT) in PP_{LOC}: (-3 -2 -1) 0 1	John [p ⇔ ⇑in park] eat

-2
-3 ⇔ -1
⇑ 0
1

7. Examples of Parses

'Flying planes can be dangerous'

one
⇕ &⇔dangerous (c denotes conditional)
fly
↑
plane

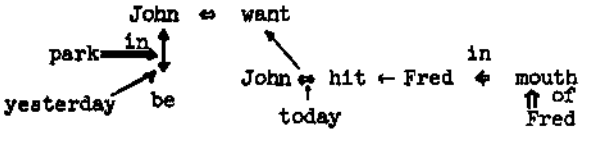
If prompted to find an alternate parse, the program would produce:

c
planes ⇔ dangerous
⇕
fly

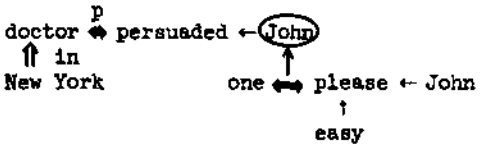
'The shooting of the hunters was terrible'

	Alternate parse:
hunters ⇕ shoot ⇒ terrible	one ⇕ shoot ⇒ terrible ↑ hunters

'John, who was in the park yesterday, wanted to hit Fred in the mouth today'.



'John was persuaded by the doctor in New York to be easy to please'.



(A circle around a concept denotes that it is the conceptual subject, or topic, of the network. If there is no circle, the conceptual subject is the item to the left of the first two-way link.)

'The girl I liked left'.

p
girl ⇔ leave
↓
I ⇔ like

8. Conclusion

Before computers can understand natural language, they must be able to make a decision as to precisely what has been said. The conceptual parser described here is intended to take a natural language input and place the concepts derivable from that input into a network that explicates the relations between those concepts. The conceptual network that is then formed is not intended to point out the syntactic relations present and there is some question as to why any system would want this information. Although Chomsky's deep structures [1] convey a good deal more information than just syntactic relations, it is clear that a parser that uses deep structures for output would be oriented syntactically. We see no point in limiting our system by trying to test out a previously formulated grammar. The output of a transformational parser, while making explicit some important aspects of the meaning of the sentence, does not make explicit all the conceptual relationships that are to be found, does not limit its parses with a check with reality, and most importantly is syntax based. The parser presented here is semantics based. We aver that the system that humans employ is also semantics based. It seems clear to us that our parser satisfies the requirements that a parser must satisfy and in so doing points out the advantages of regarding language from a Conceptual Dependency point of view.

APPENDIX

1. Conceptual Rules (permissible dependencies):

PP ⇔ ACT; PP ⇔ PP; PP ⇔ PA; ACT ← PP;

	PP	PP	ACT	PA	AA	PA
ACT ⇔ PP;	⇑;	↑ ;	↑ ;	↑ ;	↑ ;	⇑
	PP	PA	AA	PA	PA	PP

```
T  LOC  ACT  ⇒
↓;  ⇓ ;  ↑ ; ↑; ⇕ ⇒ .
⇒   ⇒    ⇒   ⇒
```

II. Realization Rules

The realization rules of the theory fall into four categories.

1) Those that determine the subject and verb (or attributive) of a conceptualization.

2) Those that attach T's LOC's, reasons, etc. to a two-way link.

3) Those that attach objects to the verb.

4) Those that attach dependents below the line.

These categories play somewhat different roles in parsing. In a way, the parser uses them in order: first, identifying the subject and placing the main link; next, modifying the main link; next, finding objects; last, qualifying the concepts involved in the first three steps. However, the processing of discourse actually proceeds from left to right, so some amount of guesswork is necessary, for example, to attach dependents to the subject before identifying it as such.

The categorization of rules also helps when a previously unknown word is encountered. Depending on whether the parser's subgoal is to complete a category four rule, a category three rule, etc., a different rule or rules will be invoked to guess the category of the new work.

(There are about 100 of these rules, presently. A few are shown here).

```
PP ACT : 1 ⇔ 2 ;
1   2

PP ACT to ACT : 1 ⇔ 2  ;
1   2      3        ↑
                  1 ⇔ 3

PA PP : 2 ;
1  2    ↑
        1

ACT PP PP : 1 ⇐ 2 ← 3   ;
 1  2  3

PP Prep PP : 3  ;
1   2   3   ⇑ 2
                        3
PP ACT Prep PP : 1 ⇔ 2 ⇐ 4 | or 1 ⇔ 2 ;
1   2   3   4                    ⇑ 3
                                 4

PP PP ACT ACT : 1 ⇔ 4
1  2   3   4    ↓
            2 ⇔ 3

                    1
ACT PP ACT-ing :    ↑      ;
 1  2   3         2 ⇔ 3

PP who ACT ACT : 1 ⇔ 3
                 ↕
                 2
```

III. A Sample of the Conceptual Semantics for 'ball'.

ball,

inanimate motion object

←PA

size	any
shape	round
color	any
texture	usually smooth
elasticity	bounces

⇐PP

in	phys obj
on	phys obj
for	phys obj
by	place
of	animal
at	no
to	no

⇒ACT

specific	bounce
motion object	roll, come, spin,
concrete	fall,hit . . .
any	begin, cause . . .

Bibliography

1. Chomsky, N., Aspects of the Theory of Syntax, MIT Press, Cambridge, 1965.

2. Colby, K., and Enea, H., "Heuristic Methods for Computer Understanding of Natural Language in Context-Restricted On-Line Dialogues," Mathematical Biosciences, 1967.

3. Hays, D., "Dependency Theory: A Formalism and Some Observations", Language v. 40, 1964.

4. Kay, M., "Experiments with a Powerful Parser" RAND, Santa Monica, California, 1967.

5. Klein, S., "Automatic Paraphrasing in Essay Format" Mechanical Translation, 1965.

6. Lamb, S., "The Sememic Approach to Structural Semantics", American Anthropologist, 1964.

7. Petrick, S., "A Recognition Procedure for Transformational Grammer" MIT Ph.D. Thesis, Cambridge, 1965.

8. Quillian, R., "The Teachable Language Comprehender: A Simulation Program and Theory of Language", Bolt Beranek and Newman, Cambridge, Massachusetts, 1969*

9. Schank, R., "A Conceptual Dependency Representation for a Computer-Oriented Semantics", Ph.D. Thesis University of Texas, Austin 1969 (Also available as Stanford AI Memo 83, Computer Science Department, Stanford University, Stanford, California, March 1969)-

10. Schank, R., "A Notion of Linguistic Concept: A Prelude to Mechanical Translation", Stanford AI Memo 75. Computer Science Department, Stanford University, Stanford, California, December 1969*

11. Schank, R., "Outline of a Conceptual Semantics for Generation of Coherent Discourse", TRACQR-68-U62-U, Tracor Inc., Austin, Texas March 1968.

12. Thorne, J., Bratley, P., and Dewar, H., "The Syntactic Analysis of English by Machine" in Machine Intelligence III, University of Edinburgh, 196b.

13* Walker, D., Chapin, P., Geis, M., and Gross, L., "Recent Developments in the METRE Syntactic Analysis Procedure", MITRE Corp., Bedford, Mass., June 1966.