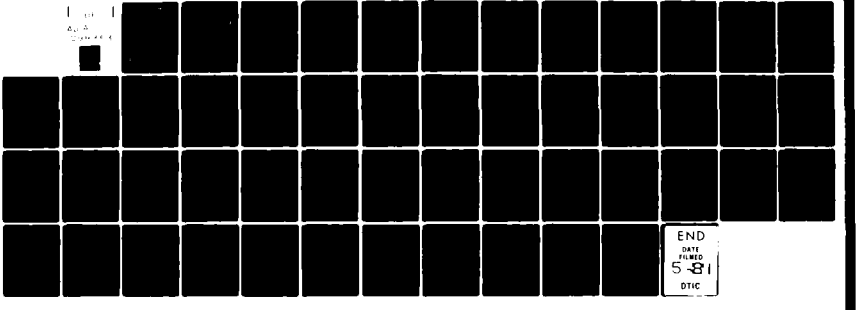
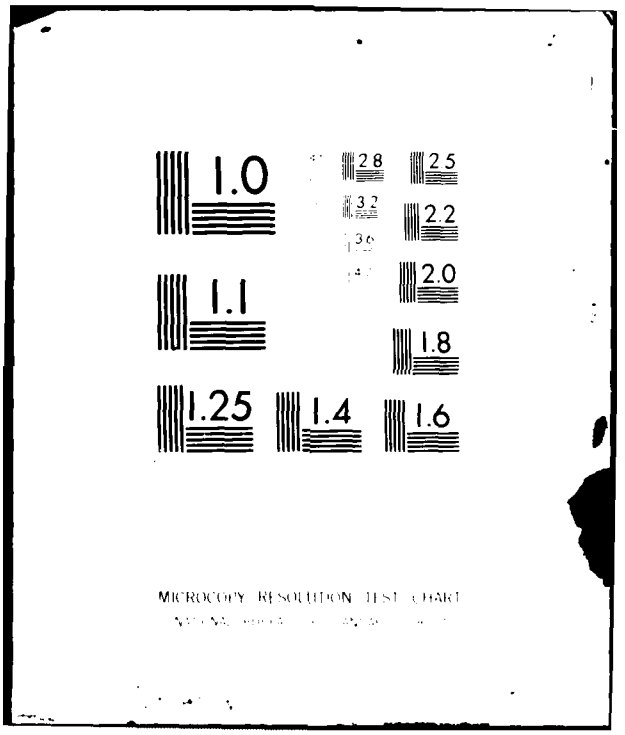
AD-A098 353 CONNECTICUT UNIV STORRS LAB FOR COMPUTER SCIENCE RE--ETC F/6 9/2
WORD MEANING SELECTION IN MULTIPROCESS LANGUAGE UNDERSTANDING P--ETC(U)
FEB 81 R E CULLINGFORD, M J PAZZANI N00014-79-C-0976
UNCLASSIFIED TR-CS-80-12A NL




1.0
2.8
2.5
3.2
2.2
3.6
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART

(12)

COMPUTER SCIENCE TECHNICAL REPORT

Laboratory for Computer Science Research
The University of Connecticut

COMPUTER SCIENCE DIVISION

Electrical Engineering and Computer Science Department
U-157
The University of Connecticut
Storrs, Connecticut 06268

12

WORD MEANING SELECTION IN MULTIPROCESS LANGUAGE UNDERSTANDING PROGRAMS

R. E. Cullingford
and
M. J. Pazzani

TR-CS-80-12A

Date: February 1981

WORD MEANING SELECTION IN MULTIPROCESS LANGUAGE UNDERSTANDING PROGRAMS

by

R. E. Cullingford
and
M. J. Pazzani

Department of Electrical Engineering and Computer Science
The University of Connecticut
Storrs, CT 06268

ABSTRACT

An understander reading or listening to someone speak has to repeatedly solve the problem of word-meaning ambiguity, the selection of the intended meaning of a word from the set of its possible meanings. For example, the problem of pronominal reference can be considered as a choosing of the intended referent from the collection of entities which have already been mentioned or which can be inferred.

Human understanders apply rules of syntax, surface semantics, general world knowledge, and various types of contextual knowledge to resolve word-sense or pronominal ambiguity as they process language. We describe a mechanism, called a cooperative word-meaning selector, which allows the computer to use various knowledge sources as it "understands" text. The word-meaning selector is part of a conceptual analyzer which forms the natural-language interface for a pair of multiprocess language processing systems. The first, called DSAM (Distributable Script Applier Mechanism), reads and summarizes newspaper articles making heavy reference to situational scripts. The second, ACE (Academic Counseling Experiment), is a conversational program which automates certain parts of the academic counseling task. In each of these systems, a variety of knowledge sources, each managed by a distinct 'expert' process, is brought to bear to enable the word-meaning selector to form the most plausible reading of a sentence containing ambiguous words.

The research summarized here was sponsored in part by the Advanced Research Projects Agency of the Department of Defense, and monitored by the Office of Naval Research under Contract N00014-79-C-0976. The support of these agencies is gratefully acknowledged.

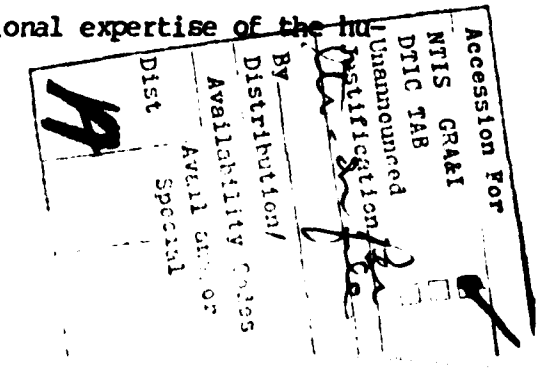
M. J. Pazzani is currently with the MITRE Corporation in Bedford, MA.

I. Introduction

In his noted paper describing what has come to be known as the "Turing test" [Turi50], the British mathematician A. M. Turing proposed to answer the question "Can a machine think?" in terms of its ability to play a certain "imitation game." The game has three players: a man, a digital computer, and an interrogator in a room apart from the other two. The interrogator is supposed to decide, on the basis of questions put to the two players, which is the man and which the computer. Questions and answers are communicated back and forth over a teleprinter. If the computer, by imitating human-like responses to the interrogator's questions, could lead the interrogator to decide wrongly in the same percentage of cases as when the imitation game is played between, say, a man and a woman, we would conclude that it was indeed "thinking" in a truly human fashion.

The point of separating the interrogator and making him talk to the others over a teleprinter is of course that thinking is an intellectual activity having little to do with physical appearance or communication apparatus. Moreover, the question-answering format of the imitation game is a suitable means of focussing on the various capabilities that are normally considered to require thinking. For example, the interrogator could ask the computer to write a poem, play chess, or solve an algebra word problem.

There is, however, a prior problem to be solved before the computer can demonstrate its ability to perform intelligently in tasks such as these. To engage in a question-answering session it must be able conduct a conversation with the interrogator. If it cannot imitate the conversational expertise of the human, it will never succeed at the imitation game.



The thinking computer, therefore, must be expert at natural-language communication. This seems to require three distinct but strongly interdependent processes: the ability to analyze a textual input to get at its meaning content [cf. Wilk75, Ries75, Wile80, Smal80]; to make inferences about the components of meaning that are only implicit in the input [cf. Char72, Rieg75, Scha77, Cull79, Carb79]; and to generate appropriate responses based upon the current input and conversational history [cf. Bobr75, Gold75, Meeh76, Cull80].

This paper discusses a key problem in the computer analysis of natural language, the selection of intended word meanings in context. Section II presents the problem in terms of the class of programs known as conceptual analyzers, and describes knowledge sources needed for word-meaning selection. In Section III, we propose a partial solution to the problem, embodied in a computer algorithm called a cooperative word-meaning selector. The algorithm provides a mechanism whereby alternative word meanings can be explicitly manipulated by various memory processes as these attempt to cooperatively home in on the most plausible readings of sentences.

The algorithm is part of a conceptual analyzer which maps natural-language strings into a Conceptual Dependency representation [Scha72, Scha75] of their meaning. The analyzer, which is based upon the one described in [Birn81], forms the natural-language interface for a pair of multiprocess text "understanding" systems. The first, called a Distributable Script Applier Mechanism (DSAM [Cull81]), reads and summarizes newspaper stories making heavy reference to situational scripts [Scha77]. The second, ACE (Academic Counseling Experiment), is a conversational program currently being developed to automate certain parts of the academic counseling task. An Appendix gives annotated output from the word-meaning selector as it runs during a story-understanding task.

II. Conceptual Analysis and Word-Meaning Selection

A. Conceptual Analysis

The machine's ability to understand its input is a key component in any automatic natural-language processing task. Hence, language analysis has been the subject of considerable research attention in artificial intelligence (AI) and computational linguistics. The model of understanding we wish to describe in this paper is connected with the class of language-input programs called conceptual analyzers.

Analyzers of this kind are designed to map an input string directly into a representation of the meaning of the string, using whatever morpological, syntactic, semantic, contextual, etc., cues are available. Conceptual analyzers normally operate from left to right, in one pass, a lexical or phrasal unit at a time. Their output is stated in terms of the conceptual representation system used by inference and memory-search processes [e.g., Wilk72, Scha72, Rume75]. Ideally, "well-formed" components of this output (called "conceptualizations" [Scha72]) are made available to the memory functions as they are formed from the input stream.

Conceptual analyzers are distinguished from other types in that they do not attempt to first analyze the input syntactically, then assign a semantic reading to the syntactic structure [see, e.g., Katz63, Chom65, Wood70, Gins78, Marc80]. Nor do they conduct a simultaneous syntactic and semantic analysis [cf. Wino72, Brow75]. (These latter types of analyzer are often called "parsers".) Syntactic features such as word order and noun-group constituency are used by a conceptual analyzer only to guide the conceptual mapping process.

B. Problems in Selecting Word Meanings

The most formidable problem faced by a conceptual analyzer is that of word meaning selection. The phrasal/lexical units that it sees in the input are usually analyzable into more than one meaning structure. If the analyzer defers choosing a representation, the number of possibilities grows as the product of the number of individual readings, and the analysis process soon gets out of hand. Therefore, the analyzer has to decide on a representation as quickly as possible.

The best-known case of the meaning-selection problem is word-sense disambiguation. A word sense is a distinct meaning of a word (found, e.g., under its dictionary entry), with a distinct underlying representation. Consider, for example, the following usages of "sense:"

(1a) Most words have more than one sense.
(1b) Sight is our most important sense.
(1c) Some people have no sense at all.

"Sense" is a well-behaved word in that it has only a relatively few alternative meanings. Other words have dozens of possible readings. For example, "give" and "take" have been metaphorically extended to so many situations that they are essentially meaningless in isolation. Their disambiguation requires access to substantial amounts of context. A computer word disambiguation scheme, therefore, will require a model of context consisting of both the meanings of surrounding words and higher level expectations.

Choosing the referent of a definite noun phrase is another example of the word-meaning selection problem. A definite noun phrase consists of a definite name, a pronoun, or a construction introduced by the definite article or certain types of modifiers:

(2a) John kicked the ball.
(2b) The Celtics extended their streak.
(2c) He threw John out.

The problem here is choosing the real-world referent of phrases such as "John," "the Celtics," "the ball," "their streak," and "he." Notice that the choice of referent often interacts with the word-sense disambiguation process. For example, memory's ability to identify a real-world ball in John's vicinity reinforces the selection of "round toy" as the intended meaning of "ball" in (2a). Example (2b) illustrates the process of finding a referent in the current clause unit (i.e., identifying "Celtics" with "their"), which in turn reinforces the reading of "streak." Finally, the identification of "he" as a bartender in (2c) gives a different meaning to the sentence than if "he" were a third baseman.

A final example of the word-meaning selection problem occurs in ellipsed inputs. These are sentence fragments (often noun phrases) presented without their accompanying propositions, most often during a conversational interaction. For example, in:

(3a) Q. Where did you go on New Year's Eve?
A. 3 parties.
(3b) Q. Who's eligible for Federal matching funds in the '80 election?
A. 3 parties.

reference to the conceptual form of the immediately preceding question is needed to select the intended sense of "parties."

C. Knowledge Sources for Word-Meaning Selection

Clearly, to solve the word-meaning selection problem the computer will have to be given various sources of knowledge about natural-language phenomena and means for applying the knowledge as appropriate. This section contains a brief discussion of some kinds of knowledge that seem to be needed, as an introduction to a Conceptual Dependency analyzer capable of certain kinds of word-meaning selection.

The simplest, and probably best understood, knowledge source for this task is rules of syntax. The intended reading of "visiting" in the following example cannot be determined without examining context:

(4a) Visiting relatives can be a nuisance.

If we change the syntactic form of (4a) slightly, however, the meaning is clear:

(4b) Visiting relatives is a nuisance.
(4c) Visiting relatives are a nuisance.

In (4b), the singular form of the copula selects the "I-visit-relatives" meaning of "visiting," while the plural form in (4c) selects the "relatives-visit-me" meaning. Syntactic phenomena of this type have been extensively studied [e.g., Hobb76, Hobb79, Nash78].

More powerful knowledge sources use "semantic," "contextual" and "pragmatic" features. Surface semantics exploits the constraints which certain words place on other words in a sentence. The term "surface" is used because the conceptual class of a word, rather than deeper contextual information, sets up expectations about, or selectional restrictions [Katz53] on, the senses of surrounding words.

For example, the meaning structures associated with certain verbs (e.g.,

eat, think, sleep) require an animate actor. In the Conceptual Dependency representational formalism, one sense of the verb "eat" is based upon the "primitive" action INGEST, whose conceptual actor must be an animal or a person. This actor is to be found in the syntactic subject spot in a declarative sentence. Therefore, the expectation for an animate actor constrains the meaning of the head noun in a noun group preceding the verb. Thus, a sentence such as:

(5) Colorless green ideas sleep furiously.

has no intelligible meaning structure even though it is syntactically acceptable. One reason for this is that there is no sense of "ideas" in (5) which meets the selectional restrictions of "sleep."

If a word has multiple senses and one sense belongs to an expected class, then the sense which belongs to the expected class should be the intended sense. Example 6 illustrates how surface semantics can be used in word sense selection:

(6) John kicked the green ball.

The word "ball" has at least two meanings, a spherical toy or a formal dance. The spherical toy sense is appropriate here because of the selectional restrictions imposed by "kick." Because the conceptual object of the meaning structure underlying "kick" must be a physical object (Note 1), the intended sense is clear. Once the "toy" sense of "ball" has been selected in (5), the noun group "green ball" has only one distinct meaning, even though there are three senses of "green" (a "color," "unripe," and "inexperienced"). As a similar example, there are two possible meanings of the phrase "the colorful ball," but only one reading in each of the following:

John kicked the colorful ball.
John attended the colorful ball.

In many cases, however, surface semantics alone does not give sufficient information to perform word-meaning selection. General world knowledge, our shared commonsense understanding of the features and functions of objects and events, is also available to assist the understander. Use of world knowledge requires access to deeper memory functions than does surface semantics, which sets up constraints based solely on the meaning structures associated with word senses. World knowledge can be utilized in disambiguation to eliminate readings of sentences which, while not logically ruled out, are nevertheless highly unlikely in commonsense terms:

(7a) John has hair on his chest.
(7b) John has a padlock on his chest.

In (7a), it is possible that John has a hairy piece of luggage. Similarly, one can imagine that John is locked up in a straitjacket in (7b). Nevertheless, application of ordinary world knowledge yields the most plausible reading in both cases. Selectional restrictions do not help in these examples because both meanings of "chest" are possible in each sentence. To perform the necessary disambiguation, a computer program must consult a memory containing descriptions of entities such as body parts and luggage. This would point out that luggage is a kind of container and that locks are often used to prevent undesired access to a container. Alternatively, it would assert that body parts of certain animate objects can have hair.

In the sense in which we have introduced it here, world knowledge encompasses general sources of information not tied to any specific experience, which an understander can use to constrain the possible meanings of a word. Detailed knowledge about special situations constitutes important further sources of data concerning context. In the Conceptual Dependency paradigm, knowledge about stereotypical situations such as eating at restaurants, riding subways, seeing

plays, etc., is organized for use by the computer into knowledge structures called situational scripts [Scha77]. Higher level knowledge structures, called goals and plans, encode the objectives of human actors and methods for achieving these objectives. [Scha77, Wile78]. The complex social, psychological and physical states from which goals in turn arise are called themes [Scha77].

Once a scriptal, planning or thematic context is established, there are many expectations concerning events that are likely to occur which can aid the word-meaning selection process. Consider, for example,

(8) As we left the restaurant, we left a tip.

The first clause in Example 8 establishes a context (the restaurant script). Word senses are selected with respect to that context. The first occurrence of "left" can be disambiguated by surface semantics. The only possible reading is a physical change of location of its actor (in Conceptual Dependency terms, a PTRANS). The second occurrence of "left" in (8) would be ambiguous if it were not for context. Two possible readings of "left a tip" are "give money to the server," or "tell someone a useful piece of information." The most likely meaning in this example, "to give money to the server," is selected because this event is expected to occur in the restaurant script at the point introduced by the clause "as we left the restaurant."

As a second example of meaning selection in context, consider:

(9) John insulted the bartender at "21."
He threw him out.

Here we have an unexpected event occurring in a restaurant. "Insult" is a social act implying the possiblity of the insulted person's forming the goal of getting revenge. This establishes a planning context within which to interpret

this person's actions [Wile78]. The actions which are most likely to occur are conditioned by a thematic relationship mentioned in Story 9, viz., that the insulted person has the role of bartender in the restaurant script.

Suppose that "throw out" can mean "eject someone," "force someone out in baseball," or "discard." Then, if "he" and "him" can be either "John" or "the bartender," there are six possible meaning structures which could be associated with the second sentence of (9). The "discard" possibilities can be ruled out on surface semantic grounds. Only one of the remaining four viz., "bartender eject John from restaurant," conforms with an expectation set up as a result of the intricate interplay among scriptal, planning and thematic inferencing.

D. Computer "Understanding" Systems

No existing computer language-processing system is capable of handling complex examples such as (9). Two systems have been developed at The University of Connecticut, however, which can apply various kinds of knowledge to perform several important types of word-meaning selection. These systems use a Conceptual Dependency analyzer (to be described more fully in Section III) as their input interface, and assist the analyzer in selecting word meanings as it runs.

One program is a story understander, DSAM (Distributable Script Applier Mechanism), which reads and summarizes newspaper stories about plane crashes. DSAM, a flexible version of an earlier program (described in [Cull78]) developed at Yale University, was built to investigate in detail the inferencing processes needed for "careful" reading, as opposed to skimming [DeJo79]. Here is a typical example of input and output from DSAM:

Story 1:
An airliner flying from Denver to Salt Lake City struck
a mountain peak today as it was leaving Stapleton Airport,
authorities reported. All 33 persons aboard were killed.

Summary:
33 persons died in a plane crash near Denver, Colorado.

Several kinds of commonsense knowledge are needed for a reasonably deep comprehension of Story 1. We need to know about causal relationships among stereotyped events in the world. For example, it is perfectly possible for a plane to crash while taking off, mountains are suitable objects for planes to run into, and the death of passengers and crew is likely when this happens. Information of this type is contained in a situational script [Scha77] describing the details of the context surrounding plane crashes. We also need to know about objects, persons and places as these are typically described in newspaper stories. For example, the "it" that was leaving the airport is the plane that crashed. Note that solving this pronominal reference problem is crucial in deciding on the appropriate sense of the word "leave."

DSAM is "distributable" because it is configured as a system of cooperating processes, each an "expert" at tasks such as applying scripts, answering questions about persons, etc. Thus, the story understander could be implemented on a distributed-processing system, although it is running on a single computer at present. New sources of knowledge can be added to the system (using an integration technique called hierarchical task management [Cull81]) to make it incrementally more powerful. For example, DSAM was given a "flare" mechanism to allow it to focus on "interesting" things in plane crash stories:

Story 2:
An airliner carrying Marshal Lin Biao and his aides struck a mountain peak today as it was leaving Shanghai Airport, authorities reported. All 33 persons aboard were killed.

Summary:
Marshall Lin Biao died in a plane crash near Shanghai, China.

The effect of the flare mechanism can be seen in the summary of Story 2, which mentions the death of the "very important person," but not the others on

the plane.

The second program is an interviewing system, called ACE (ACademic Counseling Experiment), which is currently being developed to simulate an academic counselor advising students on course selection, scheduling, etc. The main knowledge source ACE contains is a model of the curriculum followed by engineering students who intend to be computer science majors at The University of Connecticut. The model gives a script-like description of course sequences, co-requisites, and distribution requirements (but not course content), as well as the typical or expected student schedule.

One of ACE's tasks is to conduct a pre-registration, in which a student signs up for courses in the semester preceding when they are actually taken. The following is actual input/output from a question-answering session between ACE and a (simulated) first-semester engineering student:

```
ACE     : What courses are you taking at UConn this semester ?
STUDENT : computer science, chemistry 130 and history 101.

ACE     : Have you completed Chem 129 ?
STUDENT : Yes, I got credit for Chem 129 by examination.

ACE     : What is the number of the computer science course ?
STUDENT : 110.

ACE     : What other courses besides CS 110, Hist 101 and Chem
          130 are you taking ?
STUDENT : I'm also taking english and a group 3 course.

ACE     : Have you completed Math 133 ?
STUDENT : I passed it in high school.

ACE     : What is the number of the English course ?
STUDENT : I am taking English 105.

ACE     : What is the name of the group 3 course ?
STUDENT : Art 101.

ACE     : You will have to take Math 134 next semester .
ACE     : You may take CS 111, CS 207, a group 2 course and a
          group 3 course .
```

The basic idea here is to set up a course schedule for the student which makes sense in terms of the student's current standing. To do this, ACE needs to know what the student is taking now. The first question in the dialog above is designed to elicit this information. Since engineering students typically take four or five courses a semester, the response to this question is incomplete. ACE's understanding of the ellipsed answer depends upon the analysis of ambiguous terms such as "computer science" in the context of the question which was asked. On this basis, "computer science" is taken as referring to a course rather than the field. Even so, the response as given indicates several problems.

First, there are several ways to complete the undergraduate chemistry requirement. One way is to take the sequence Chem 129/Chem 130. It is highly unusual for a first-semester freshman to be taking Chemistry 130 since it has a prerequisite. In some cases, a student may be able to get advanced credit by passing an examination based on the prerequisite's subject matter. ACE is aware of this possibility as it asks the second question.

Having solved the problems caused by the answer to the first question, ACE attempts to find out what other courses the student is taking. It notices that the highly expected mathematics course sequence has not been mentioned, and immediately tries to find out why. The curriculum specifies that Math 133 is the expected course at this point, and this determines the form of the question. Because the question explicitly mentions Math 133, the referent of "it" in the response "I passed it in high school" can be supplied to the analyzer. This in turn disambiguates "passed."

Having determined the student's current course load to its satisfaction, ACE then consults the curriculum again to find out which courses are mandatory

at this point, and which are optional. Note, however, that the responses it generates at the end of the dialog are critically dependent on its understanding of what was said before.

III. An Algorithm for Word-Meaning Selection

Having sketched some important analysis problems posed for systems such as DSAM and ACE, we now turn to a more detailed discussion of how these problems are solved by the conceptual analyzer they use.

A. Conceptual Dependency Analysis

Language analysis in the Conceptual Dependency (CD) paradigm, motivated by the way that people seem to approach the task, has attempted to use predictions or expectations about what will be heard as the driving force behind the understanding process (Note 2). Syntactic, surface semantic, scriptal, and planning contexts are all rich sources of predictions. Therefore, the main line of development in CD analyzers has been the attempt to incorporate more and more context into the analysis process.

Word definitions describing the meaning structure(s) built by a word and suggestions for using this structure are typically kept off-line in a dictionary, and are not called into active memory until the word is actually seen in the input stream. Expectations associated with a word definition are encoded in a special type of production (or test-action pair [Newe72, McDe78]) called a request [Ries75].

Requests are activated when the associated word definition is loaded. The activation process places the requests in a short term memory of requests to be considered. Request consideration repeatedly selects a request and evaluates its test part. If the test is true, the request is said to have "fired," and its action part is executed.

Requests can check semantic, lexical, or contextual features of the run-time environment, and create or connect Conceptual Dependency structures.

Moreover, they can cause other requests to be loaded or deleted. Associated with the meaning structure built by a word (sense) are a set of roles and a set of expectations embodied in requests indicating how the roles are to be filled.

Consider, for example, the sense of the verb "to take" which means "to execute the academic-course script" from the point of view of the student. (To "give a course" or to "teach a course" is to execute the same script from the point of view of the teacher.) In a simple English format, the requests associated with this sense of "take" would be:

```
REQUEST1:
    TEST:
      Is the "object" of take a course?
    ACTIONS:
      Create the concept for an execution of the course
         script.
      Fill the conceptual object slot of this concept
         with the course that was found.
      Activate REQUEST2

REQUEST2:
    TEST:
      Is the "subject" of take a person?
    ACTIONS:
      Fill the conceptual actor of the course script
         with the person that was found.
```

These requests contain two different types of information. "Positional" specifications predict where in the sentence the conceptual actor and object of the take-course script will be found. For example, the "object" spot in REQUEST1 is the syntactic object of the clause containing "take" if the sentence has the active voice, the syntactic subject if it is passive. "Semantic" specifications constrain the entities that will be used to fill the conceptual actor and object role in the take-course concept.

Existing CD analyzers such as ELI (English Language Interpreter: [Ries78]) and CA (Conceptual Analyzer: [Birn79]) use the test part of a request to imple-

ment a form of word-sense disambiguation based on surface semantics. For certain words, a group of tests checking lexical, syntactic, or semantic features can be used to determine which sense is intended. In such a case, the tests are said to be orthogonal, i.e., the tests check mutually exclusive cases and cover all possibilities in such a way that exactly one request is fired. The action of the request which has fired will create the meaning representation for the intended word sense. In this manner, one word sense is selected and the others are suppressed.

So, for example, we could add a second word sense to the definition of "take" corresponding to a sentence such as "John took an aspirin" in the following way:

```
REQUEST3:
    TEST:
      Is the "object" of take a drug?
    ACTIONS:
      Suppress the other requests of "take"
      Create the concept for an INGEST action
      Fill the conceptual object slot of this concept
         with the drug that was found.
      Activate REQUEST4 (to find an actor, as above)
```

This method of selecting word senses by using orthogonal requests works well for many verbs. The intended word sense is determined by the class of actor or object associated with the verb. The method is also useful in disambiguating some adjectives. For example, two meanings of "rich" can be discerned in "a rich man" and "a rich cake." Here, the conceptual type of the modified noun (person vs. ingestible object) is enough information to select the proper meaning of "rich." Words which can be disambiguated by orthogonal requests have a common feature. Their meaning is embodied in a case frame of conceptual cases and fillers, with a request looking for a conceptual entity to fill each case. The type of conceptual entity found can determine which sense is intended. Nouns which build Picture Producers [Scha72], on the other hand, do not have this feature

since typically they build case frames with all the cases already filled in. (Picture Producers are concepts corresponding to entities such as persons, places and objects which tend to produce a mental image in the mind of the understander.) For example, in the sentence:

(10) John shot two bucks

"John" builds a Picture Producer for a male person named John. This concept is "complete" in the sense that it can be understood in isolation, as, for example, "shot" cannot. Similarly, although the phrase is ambiguous in (10), "two bucks" builds well-formed structures for either "amount of money" or "male deer." It would be difficult, if not impossible, to define a set of orthogonal tests to select the intended meaning of ambiguous nominals such as "buck." The problem with this approach is that each word is responsible for <u>disambiguating itself</u>. To select the intended sense of a word which can create several different Picture Producers, <u>expectations</u> about the intended conceptual class must be used.

B. Related Work in Word-Meaning Selection

Riesbeck's ELI and Small's Word Expert Parser (WEP: [Smal80]) are two conceptual analyzers which implement proposed solutions to the problems caused by words with multiple senses. In ELI, expectations are used to choose among word senses. Requests associated with a word are activated only if their actions build a Conceptual Dependency structure which is expected by an already existing request. "Expected" here means that the test part of the existing request would become true if the meaning structure which the new request builds were added. (The process of extracting the meaning representation from the action part of a request is called rehearsal.) Note that this is a top-down approach. ELI matches meaning structures created (through rehearsal) by new input to its expectations. There are several problems with this approach, all caused by the requirement for a pre-existing expectation. Since conceptual entities must have been predicted before they are accepted into the system, and since initially there are no expectations, there must be a standard set of initiating requests at the beginning of each new sentence. However in sentences such as:

The pilot and co-pilot died, authorities announced,

the second clause is not expected and the initiating requests are no longer active. ELI can properly handle the disambiguation of "ball" in:

(10a) John kicked the ball.
(10b) John attended the ball.

because "kick" activates a request containing the proper prediction. However, it cannot handle the passive forms of these sentences:

(11a) The ball was kicked by John.
(11b) The ball was attended by John.

because the expectation needed to disambiguate "ball" comes after the word. ELI

does not have the ability to delay deciding among requests to activate.

The Word Expert Parser is a complex and interesting conceptual analyzer capable of performing in a bottom-up fashion several of the types of word-meaning selection considered here. In WEP, each word comprising a bundle of word-senses is assigned an individual "word expert." A word expert is represented as a coroutine which cooperates with neighboring words to select its intended sense and eventually to build a meaning structure for the entire sentence. (The meaning representation system is a variant of Rieger's Commonsense Algorithm notation [Rieg76].) A word expert's basic mechanism for selecting one of its senses is a discrimination net in which n-way discriminations (called multiple-choice tests) can be made on the basis of lexical and semantic properties of neighboring words.

The sources of knowledge which the Word Expert Parser uses are surface semantics and, to a lesser degree, general world knowledge coded into an individual word expert. However, it apparently has no way of using the higher level forms of predictions provided, for example, by scripts, plans, and discourse context. So, for example, it could not disambiguate "took" in:

(12a) David was arrested because he took
a bottle of aspirin.
(12b) David died because he took a bottle
of aspirin.

In other cases, the information requested by the multiple-choice tests of the discrimination net will not be present. For example, the semantic category of the object of "take" is needed to discern between "take a course" and "take medicine." In Example (13), however, this is not immediately available:

(13) David took it.
Question from the word expert of "take:"
What did David take?

A method for performing pronominal reference has not been implemented in WEP. In fact, such an algorithm could not use the standard word expert discrimination net mechanism because this requires the possible senses (or antecedents in the case of pronominal reference) to be known when the word expert is coded.

C. A Representation for Multiple Word Meanings

Existing conceptual analyzers, then, fail to handle many important cases of word meaning selection. There are two main reasons for this: (1) the handling of expectations is constrained too much, as in ELI, by the top-down nature of the control structure; or (2) as in WEP there is no way to make use of high-level expectations during disambiguation. Both approaches suffer from the fact that the alternative senses of a word cannot be explicitly seen by the disambiguating processes, being hidden in ELI, for example, inside the action parts of requests to be rehearsed.

The approach taken in this work builds upon the design of Birnbaum and Selfridge's CA, which implements an essentially bottom-up approach to conceptual analysis. CA allows word definitions to add concepts and expectations to the analyzer's short-term memory (called the "concept list," or C-LIST) essentially at will. Thus, it avoids ELI's excessively top-down nature. On the other hand, expectations embodied in requests are handled uniformly no matter what their source. Thus, the opportunity exists to have memory processes examine and modify the current state of the analysis process.

Two things are needed to make this work. One, we must have representational system for words with multiple meanings which makes the alternatives visible. Secondly, a uniform repertoire of procedures to manipulate the alternatives must be made available to the processes capable of making a selection.

Our approach is implemented in a conceptual analyzer called APE (A Parsing Experiment), which extends CA to handle these more general kinds of meaning selection processes. We use the following simple declarative representation for a word with several word senses:

```
(VEL V1 "sense 1"
     V2 "sense 2"
     . . .
     Vn "sense n")
```

where VEL (Latin "or") indicates a mutually exclusive set of possible meanings.

The dictionary definition of a word with multiple meanings always contains a request to add all its senses (i.e., add a VEL) to the analyzer's short-term memory, the C-LIST. At the same time, a pool of requests may be activated to aid in the disambiguation of the VEL.

Selection of the intended component of the VEL follows these procedures:

1. The request creating the VEL may activate another request to examine the C-LIST for a concept with a semantic feature, to check the input sentence for a lexical feature, or to query a script or plan applier or other memory module for a contextual feature. A request of this type could assert which meaning is intended or eliminate senses which are not intended.

2. A pre-existing request may be looking for one of the possible meanings of the word creating the present VEL. Typically, expectations of this kind come from surface semantics (e.g "attend" expecting an event as its object).

3. Most importantly, a VEL may be disambiguated by expectations explicitly set up because some sort of context has been established:
   a. Knowledge structures such as scripts or plans;
   b. Discourse contexts such as permeate narratives (which allow pronoun and other definite nominal references to be established) or question-answering dialogs (which fill out ellipsed answers using expectations about the answer to a question).

   These expectations are also encoded as requests but the source of the request comes from the understanding system itself.

The first and second of these techniques implement word meaning selection based on surface semantics, since expectations are set up by the requests asso-

ciated with input words. The last method relies on the integration of the analyzer with various kinds of memory modules so that context can assist the parser.

Note that the advantages of integration are two-way. Recourse to context will be often be decisive in eliminating ambiguity, thus reducing drastically the number of ambiguous readings to be considered by the analyzer. The analyzer in turn can inform the contextual knowledge sources that their predictions have been substantiated. As a result, the absorption of the concepts created by the analyzer into the larger knowledge structures encoding the computer's understanding of a domain are dramatically speeded up. (The analysis procedure discussed in [DeJo79] illustrates precisely this point.)

D. Use of Surface Semantics in Disambiguation: An Example

VEL Representation and Operations

The following simple example illustrates the use of the VEL, and the operations which create VELs or select a conceptual entity from VELs. The word definitions to follow are essentially encoded in the request format definition defined by the CA conceptual analyzer. See [Birn79] for a detailed description of its operation. Here we stress parts of CA of relevance to the VEL mechanism.

The sentence to be be analyzed is "The ball was kicked." The words "kick" and "ball" are of interest here. The LISP-format dictionary definition of "kick" as it is used by APE, the natural-language front end for DSAM and ACE, is given below ("~" indicates a comment):

```
{def kick
(requests
 (req
   [test t]  ~add propel concept
   [actions
    <:= str0 (add-con '<*act* type (*propel*)
                          actor(nil)
                          object(nil)
                          inst (*act* type (*move*)
                                   actor(nil)
                                   object (*pp* type(#bod-prt)
                                              stype (*foot*)) > )>

    ~activate requests to find the actor and object
    (activate
     <req  ~find person who is the actor
      [test (:= str1 (if-avail(cc)(and(in-act-spot cc str0)
                                       (feature cc #person))]

      [actions (fill-gap '(actor) str0 str1)
               (fill-gap '(inst actor) str0 str1)]>

     <req  ~find the object of the propel
      [test (:= str2 (if-avail(crc)(and(in-obj-spot crc str0)
                                        (feature crc *pp*))]
      [actions (fill-gap '(object) str0 str2) ]>
    )
   ]
}
```

The definition of "kick" contains a request whose test is always true and whose action contains the function add-con which builds a CD structure for a PROPEL (i.e., the application of a physical force) and adds it to the C-LIST. The act which is INSTrumental to the PROPEL is a MOVE (i.e. movement of a body part.) The object which is MOVEd is a foot. The name of the concept which add-con creates is substituted throughout the body of the request by the run-time binding operator, :=.

At activation time, "kick" also creates expectations that an actor and object with specific conceptual properties will be found. The function activate creates a pool of requests which encode these expectations. Here, two requests are spawned. The test of the first request contains the function if-avail which applies a predicate to the concepts on the C-LIST. In this case, the predicate is looking for an "available" concept (i.e., one which has not been absorbed by a larger concept) which is a person, and is in the syntactic "actor spot" of the sentence with respect to the PROPEL concept created by "kick." In this example, the sentence is passive, so if-avail attempts to apply the semantic predicates to the concept built by a noun phrase following the preposition "by." Thus, APE can distinguish between constructions such as "the ball was kicked by John," "the ball was kicked by Tuesday," and "the ball was kicked by the bridge."

For purposes of meaning selection, if-avail can apply a semantic/positional predicate to all the components of a VEL. The predicate returns true if at least one of the components of a VEL satisfies its requirements. Those components which fail to make the predicate true are removed from the VEL by a process called VEL compression. If only one component remains, the VEL is replaced by that one, the intended sense of the word. Thus, a request which expects a concept belonging to a certain semantic category can find it and assert that the

VEL has been disambiguated.

The second request activated under "kick" searches for a Picture Producer in the syntactic "object" (here, the subject) spot with respect to "kick." If this request fires, it fills the conceptual object spot of the PROPEL concept.

Now we need the definition of "ball:"

```
{def ball
(requests
 (req   ~create a formal dance or a toy.
  [test t]
  [actions <:= str0 (add-con '(*vel* v1 (nil) v2 (nil))>
           (velrole str0 'v1 '(*pp* type (#toy)))
           (velrole str0 'v2 '(*event* type ($formal-dance)) ]
 )
}
```

This definition adds a VEL with two senses of "ball." The function velrole is used to create the alternate senses: one a Picture Producer of type "toy;" the other a "formal-dance" script. Note that the definition of "ball" contains no disambiguating requests. It relies on other concepts to select its intended sense.

In the analysis of the sentence "the ball was kicked," the intended sense of "ball" is found by APE because "kick" is expecting its conceptual object to be a Picture Producer.

E. Selecting the Meaning of Definite Noun Phrases

In Section II, we identified the need to find the referent of definite noun phrases as an important part of the word meaning selection process. One example of this problem is pronominal reference. A pronoun normally refers to a conceptual entity mentioned or inferrable elsewhere in a story or dialog. Pronominal ambiguity arises in examples in which there are several possible referents meeting (in English) the restrictions of gender and number for a pronoun. Selecting the referent of a pronoun is not a simple task. It may require surface semantics, world knowledge, or contextual knowledge. Consider, for example, the usages of "he" and "him" in the following sentences:

(13a) Bill hates John, so he hit him.
(13b) Bill hates John, because he hit him.

In Example (13a), Bill is most likely the actor of the PROPEL. Syntactic cues don't help much here. A human understander knows that hitting is an act that can be explained by a desire to cause injury, which can be explained by a state of hatred. Because Bill hates John, it follows that Bill may wish to hit John. In (13b), on the other hand, the probable actor of the PROPEL is John. Hitting someone often leads to their hating you. Once again, an extremely complicated inference process involving the goals and plans of the actors in Example (13) is required before the intended referent can be located.

Suppose we represent (through a memory call) the set of possible referents for a pronoun in a VEL format. Then pronominal ambiguity can be resolved in a manner analagous to using VELs in word sense disambiguation. (Any definite noun phrase whose referents can be proposed at the time the phrase is encountered can be handled in the same fashion. See [Cull81] for details.)

The VEL is used in pronominal reference to help select the intended re-

ferent by providing an explicit representation for the possible referents. However, in pronominal reference, unlike word sense selection, the components of the VEL are not fixed in a dictionary definition. From the VEL representation, the operations used by surface semantics, world knowledge and contextual knowledge can select the intended sense.

For example, consider the following stories.

Story 3:
John knew aspirin upset his stomach.
He took it anyway.

Story 4:
John knew Computer Science 265 was difficult.
He took it anyway.

In Story 3, the possible referents of "it" are "aspirin" and "stomach." One sense of "take" expects its conceptual object to be a drug. Since "aspirin" meets this selectional restriction, one could safely assume it is the referent. This in turn disambiguates "take."

Similarly, in Story 4, Computer Science 265 can be selected as the proper referent of "it" and "take" is intended to mean execute the "course" script.

Computer output from the story understander, DSAM, which illustrates the use of VELs in pronominal reference is given below. The mechanism of word meaning selection involves an interaction between APE and PP-Memory (the module of DSAM which knows something about the properties of Picture Producers).

The APE dictionary definition of the word "it" shows how pronominal reference is done in DSAM:

```
{def it
(requests
 <req
   [test t]
```

```
   [actions
        <:= str0 (add-con '(*pp* type (nil)
                                  ref (pron)
                                  gender (*neut*)) >
        (find-ref str0)]>
)]
```

The function find-ref is used by APE to request the set of possible referents for "it." Find-ref, therefore, is effectively a distributed function call, in which the module responsible for keeping track of Picture Producers, PP-Memory, supplies the analyzer with all the entities meeting the requirements of gender and number in the current context.

Below we present output from the story understander, DSAM, as it processes Story 3. The output has been edited slightly for readability, and various comments (indicated by ";") have been inserted to explain what's going on. The pronominal reference interchange described above, and all the interactions among the expert processes comprising DSAM and ACE, is controlled by the integration "expert," the hierarchical task manager [Cull81]. This module is referred to as the Gateway (GW) in the computer run to follow. DSAM's dictionary and morphology specialist, which supplies request clusters to APE, is called TTIN.

DSAM...v1.0

```
TTIN: file  take7  text:
((david knew aspirin upset his stomach pr) (he took it pr ))
        ;The story to be processed


        ;We pick up the computer run after the first
        ;sentence has been analyzed. Here is its CD
        ;representation:
APE: sentence concept:  apc5
* (xpn apc5): (*act* mode (*tf*)
                    type (*mtrans*)
                    actor (*pp* type (#person)
                              persname (david)
                              gender (*masc*))
                    from (*ltm* part (*pp* type (#person)
                                           persname (david)
```

```
                                        gender (*masc*)))
                        mobject (*conrel* type (*cause*)
                                precon
                                (*act* type (nil)
                                       actor (*pp* type (#ingobj)
                                                  ingtype (*med*))
                                postcon
                                (*state* var (*health* part
                                             (*pp* part (*pp* pptok pphum0
                                                              type (#person)
                                                              persname (david)
                                                              gender (*masc*))
                                                        type (#bod-prt)
                                                        stype (*stomach*))
                                         toward (*-5*)))))
        ;The meaning structure for the sentence is based upon
        ;an MTRANS, i.e., a mental transfer of information,
        ;from David's long-term memory (ltm)
        ;The information transferred is that some unknown act
        ;involving aspirin has caused the physical state of David's
        ;stomach to decline. Note that PP-memory and APE have
        ;disambiguated "his," which at this point can only be
        ;"David." pphum0 is the memory pointer to "David,"
        ;"David's stomach" is pptok1, and "aspirin" is pptok0.

        ;DSAM starts on the second sentence
sent = (|he |perf$ |take |it |pr )

the current word is  |he
inr9 being considered

       ;At this point, the referents for "he" must be requested.
       ;Find-ref is called to get the data from PP-Memory (PPMEM).
APE  seeking referent apc31
ape : requesting
     (find-ref apc31)

       ;a complicated gateway interaction has been omitted here
PPMEM: apc31 could be (pphum0)
       ;once again, david (pphum0) is only referent

inr9  has fired
:c-list=  (apc35)
available=  (apc35)
       ;apc35 is "david"

* (xpn apc35):  (*pp* pptok pphum0
                      type (#person)
                      persname (david)
                      gender (*masc*))

the current word is  |take
inr11 being considered
inr11  has fired
:c-list=  (apc35 apc36 apc37 apc40 apc42)
```

```
available=  (apc37 apc35)
        ;apc37 is a VEL with two senses of take

aprl7 being considered
aprl7  has fired
:c-list=  (apc35 apc36 apc37 apc40 apc42)
available=  (apc37)
        ;take locates "David" (apc35) as its actor.
        ;Either sense of take requires a person for
        ;the conceptual actor
        ;Here is the still-ambiguous concept:
* (xpn apc37): (*vel* mode (*tf*)
                      vl (*act* actor (*pp* pptok pphum0
                                             type (#person)
                                             persname (david)
                                             gender (*masc*))
                                course (nil)
                                type ($course))
                      v2 (*act* actor (*pp* pptok pphum0
                                             type (#person)
                                             persname (david)
                                             gender (*masc*))
                                object (nil)
                                type (*ingest*)))

the current word is  |it
inrl2 being considered
        ;ape needs a referent for "it"
APE  seeking referent apc45
ape : requesting
     (find-ref apc45)

PPMEM: apc45 could be (pptokl pptok0)
        ;There are two possible "it's"
        ; pptok0 - aspirin
        ; pptokl - david's stomach
inrl2  has fired
:c-list=  (apc35 apc36 apc37 apc40 apc42 apc44 apc5l apc49 apc50)
available=  (apc5l apc37)
        ;apc5l is the VEL containing "david's stomach" and "aspirin"

* (xpn apc5l): (*vel* v0 (*pp* pptok pptok0
                                type (#ingobj)
                                ingtype (*med*))
                      v0 (*pp* pptok pptokl
                                type (#bod-prt)
                                stype (*stomach*)
                                part (*pp* pptok pphum0
                                           type (#person)
                                           persname (david)
                                           gender (*masc*))>
        ;the VEL of possible referents of "it"

aprl9 being considered
        ;If we find a drug, assert INGEST
```

```
        ;and fill object slot

VEL assert old:  apc51  new:  apc50
        ;According to PPMEM, "it" can indeed be a drug,
        ;so compress the VEL for "it" to "drug"

VEL assert old:  apc37  new:  apc42
        ;Now we can disambiguate "take"
        ;Compress its VEL to "ingest medicine"

apr19  has fired
:c-list=  (apc35 apc36 apc37 apc40 apc42 apc44 apc51 apc49 apc50 apc52)
available=  (apc42)

APE: sentence concept:  apc42


        ;The final representation of the sentence:
* (xpn apc42): (*act* object (*pp* pptok pptok0
                                type (#ingobj)
                                ingtype (*med*))
                      mode (*tf*)
                      actor (*pp* pptok pphum0
                                 type (#person)
                                 persname (david)
                                 gender (*masc*))
                      type (*ingest*))
```

In this example, "it" has two possible antecedents. Selectional restrictions imposed by one sense of "take" choose the intended referent. This also disambiguates "take."

F. Word-Meaning Selection in Scriptal Context

Throughout this paper, we have argued that understanding must be done in context. Contextual information is gleaned from "knowledge structures" which encode people's repeated experiences in familiar (i.e., scriptal) and not-so-familiar (i.e., planning) situations. Simulating such knowledge structures for the machine gives it a source of "experience" by which it can evaluate new input.

Situational scripts are the model of context used by DSAM. A script consists of a set of roles, the standard participants in the script; a set of possible entry conditions which describe the state of the roles at the beginning of the episode; a set of scenes, containing the events which typically occur in a script; the causal and temporal relationships among the events; and a set of possible resulting states of the script. All these items are expressed in language-independent Conceptual Dependency patterns. The expert process called the script applier attempts to locate the conceptualizations produced by the analyzer in its collection of scripts. If it succeeds in this, it can build an inference chain of events, both those which were explicitly mentioned and those which can reasonably assumed to have occurred, as well. This "trace" through a script is the story representation which the machine consults to demonstrate its "understanding," e.g., by summarization or question-answering.

The understanding process depends critically on the system's ability to select word meanings. In DSAM, the event patterns of the script have been augmented so that each event can have a named request associated with it. Such requests enable the script applier to inform APE of its expectations in a format APE can use. This gives APE the common sense knowledge we share of the domain being currently considered.

An expectation-based system such as DSAM can have the problem of combinatorial explosion unless there is some method of controlling the number of expectations. DSAM contains a windowing mechanism (described in [Cullingford 78]), which keeps track of what has been seen and what is immediately expected, as the story follows a path through a script. This mechanism is aided by the communication channel between the analyzer and the script applier. In addition to sending concepts as soon as they are completed, APE informs the script applier of the relations between concepts specified by words such as "as," "after," and "because." With knowledge of the relation between a concept just seen and the one currently being formed, the script applier can give the analyzer the named requests for predicted events.

As an example of this communication consider the simple story:

(14) As David left the restaurant, he left a tip.

The second use of "left" must be disambiguated by context, i.e., by the restaurant script. The Appendix contains an annotated protocol of DSAM performing this operation. Here we sketch the main features of the understanding process.

When APE has completed analyzing the first clause of (14), it places its result where the memory modules (PP-Memory and the script applier) can see it. The script applier finds a match for this in the "leaving scene" of the restaurant script. At this point, since "as" indicates that the next concept to be produced will be in the causal/temporal vicinity of "leaving," the script applier can form some fairly specific predictions. It expects, among other things, that the customer will give a tip to the waiter. This expectation embodies a rule which people have about restaurants. The script applier thus sends a named request for the tipping event (and others as well) to APE, which looks it up and

activates it. The request has two purposes. First, if the predicted concept is found in the input, APE can tell the script applier that its expectation has been substantiated, thus eliminating the need for memory search. Secondly, if APE has a conceptualization which is ambiguous, but one reading is expected by context, i.e., by the named request, that one will be asserted as the proper reading.

IV. Some Conclusions

We have approached the word meaning selection problem from a semantic point of view. This process is proposed as one which unifies the problems of word-sense disambiguation, definite noun-phrase (including pronominal) reference, and discourse ellipsis, phenomena which are normally considered separately. We identified surface semantics, general world knowledge and episodic or discourse context as three sources of expectations which aid in this process. We then described a computer algorithm, the cooperative word-meaning selector, capable of using all three. The algorithm is part of a working conceptual analyzer, APE, which is in turn part of two different computer "understanding" systems. Although these systems are typical toy artificial intelligence programs, they work well enough for us to believe that "real" text-processing systems capable of flexible and reasonably deep (though not real-time) comprehension could be designed using them as models.

Our approach does have several limitations. We have purposely ignored any notion of favored meanings or probabilities. With some difficulty, for example, we could compute that the word "ring" refers to a piece of jewelery 85% of the time. When all else failed, we could use this sense. This would work well (85% of the time, in fact), but we don't believe it has a place in a cognitive theory.

We have also ignored syntactic phenomena, feeling that "conceptual" factors are more fundamental. Many words have multiple parts of speech (i.e., the senses belong to different parts of speech). Syntactic knowledge would help to eliminate senses which belong to a certain part of speech. For example, in:

Hearing aids the blind.

a syntactic analysis would favor the verb sense of "aid," which would help in the conceptual analysis of this sentence.

Just as the control structure of ELI limited its power, the use of requests in APE creates some problems. First of all, requests tend to make word definitions long and awkward. Secondly, requests either fire or they don't. This limits the ability of the rules to use surface semantics. The tests of requests which fill slots tend to look for only very general semantic categories to account for all possible cases. This limits the power of selectional restrictions when it is necessary to choose among components of a VEL which are similar. A notion of minimal requirements and more specific optional restrictions which increase the certainty of the selection would be useful.

A type of ambiguity we have not considered is caused by modifiers. For example, in "small car salesman," "small" could refer to the car or the salesman. The problem here for the analyzer is not in disambiguating a VEL, which the current process should be able to do, but in creating one in the first place. This is because the request associated with "small" that looks for a concept to modify is satisfied when it finds one. A more general approach would be to look for all such concepts, and create a VEL if there is more than one.

Finally, the use of named requests to make high-level predictions available to the analyzer is unwieldy and difficult to generalize beyond simple scriptal or planning contexts. What is really needed is a separate "expectation expert," which would match expectation patterns from whatever source against the stream of concepts flowing in from the outside world, or circulating internally. How such an expert would be designed is only very dimly understood at present.

Nevertheless, in spite of these shortcomings, we believe that the coopera-

tive word-meaning selector is a viable approach to a key problem in applying knowledge to understand natural language. Therefore, it provides a first-pass model of one type of processing the machine intelligences of the future will have to perform.

Implementation Note

The implementations of DSAM and ACE discussed in this paper run on a PDP-11/60 minicomputer under the UNIX operating system [Ritc74]. Both understanders are configured as a set of up to seven 65 kilobyte processes programmed in Maryland VLISP [Kirb77]. Process creation and message passing are implemented using the UNIX "fork" and "pipe" system facilities. To give a feeling for runtime, processing of Story 14 under DSAM requires about 6 minutes of elapsed time, with DSAM using the computer in single-user mode.

Acknowledgement

The authors wish to acknowledge several fruitful conversations with Larry Birnbaum and Mallory Selfridge, the designers of CA, concerning conceptual analysis. Kevin Mullens collaborated extensively in the implementation of ACE, the academic counselor.

Notes

(1) We are ignoring metaphorical uses of "kick" such as "John kicked himself for his stupidity," and "John kicked his habit."

(2) The arguments for predictive understanding, and for conceptual analysis in general, are covered in detail in [Ries75, Ries78, and Scha78].

APPENDIX
Annotated Computer Run: Context and Disambiguation

```
                        DSAM...v1.0
TTIN: file  rest2  text:
((as david left the restaurant comma he left a tip pr))

:c-list=  (apc0 apc4 apc8 apc12 apc14 apc18)
available=  (apc0)
        ;the state of the world after the words "As david left"
        ;have been processed is dispalyed
        ;those concepts on the available list accumulate

* (xpn apc0):
(*conrel* type (*when*)
          cona (*vel* v1 (*act* actor (*pp* type (#person)
                                                 persname (david)
                                                 gender (*masc*))
                                      type (*atrans*))
                         v2 (*act* object (*pp* type (#person)
                                                  persname (david)
                                                  gender (*masc*))
                                      actor (*pp* type (#person)
                                                 persname (david)
                                                 gender (*masc*))
                                      type (*ptrans*)
                                      from (*inside* part (nil)))
                         v3 (*act* actor (*pp* type (#person)
                                                 persname (david)
                                                 gender (*masc*))
                                      type (*mtrans*)))
          conb (nil))

        ;the CD representation for "as david left"
        ;"as" indicates a temporal relation between two concepts
        ;three senses of left are considered,
        ;       1. atrans- left a dollar
        ;       2. ptrans- left the resturant
        ;       3. mtrans- left a note
        ;note that meanings 1 and 3 are inferred (i.e. "david
        ;left a dollar" = david ptrans david from dollar. The usual
        ;intention of a ptrans from a dollar is an atrans.)

:c-list=  (apc0 apc4 apc8 apc12 apc14 apc18 apc20 apc21)
available=  (apc21 apc0)
        ;the state of the parser after the "the restaurant" is seen.
        ;apc21 is the conceptualization for "a restaurant"

apr8 being considered
        ;apr8 is a request activated by "leave" looking for a location
        ;as the conceptual object of the ptrans created by "leave"

VEL assert old:  apc8  new:  apc14
        ;the vel is disambiguated by surface semantics
        ;the ptrans sense of "leave" requires a location to leave from
```

```
        ;apc8 is the vel created by "leave" apcl4 is the ptrans sense

* (xpn apcl4):
(*act* object (*pp* type (#person)
                     persname (david)
                     gender (*masc*))
       actor (*pp* type (#person)
                    persname (david)
                    gender (*masc*))
       type (*ptrans*)
       from (*inside* part (*pp* ref (*def*)
                                 type (#struc)
                                 stype (*restaurant*))))
        ;the CD representation for "david left the restaurant"

aprll being considered
APE: shipping:  apcl4
        ;aprll notices that apcl4 (david left the restaurant)
        ;is a completed concept and ships it

the current word is  |he
inr6 being considered
APE  seeking referent apc25
        ;ape needs a vel of all possible "he's"
ape : requesting
     (find-ref apc25)

PPMEM: apc25 could be (pphum0)
        ;PPMEM finds referent, a vel is not formed
        ;since there is only one possible referent
* (xpn apc29):
(*pp* pptok pphum0
      type (#person)
      persname (david)
      gender (*masc*))
        ;the CD representation of "he" i.e. "david"

APPLY: trying  f13
        ;apply is trying to activate a script
        ;f13 is a scene from the $fly script
APPLY: trying  f12

APPLY: backbone match on  f12 with bindings
((&passgrp . apc4) (&plane . apc4))
        ;f12 is close- it is a "plane ptrans passagers"
        ;if the actor of the ptrans is a plane
        ;and the object could be passengers
        ;the fly script would be valid
APPLY: need bindings:
((&passgrp . apc4) (&plane . apc4))

PPMEM: requesting
     (sctoutf ppmem ((&passgrp . pphum0) (&plane)))
        ;PPMEM says apc4 (whose permanant token is pphum0)
        ;could be the passenger group
```

```
        ;but apc4 cannot be the plane

APPLY: invalid binding:  (&plane . apc4)
        ;apply realizes its error


        ;meanwhile, back in APE...
the current word is  |tip
inr9  has fired
        ;the vel for "tip" is created
:c-list=  (apc0 apc4 apc8 apc12 apc14 apc18 apc20 apc21 apc29
 apc30 apc34 apc36 apc40 apc42 apc43 apc46 apc48)
available=  (apc43 apc42 apc0)
        ;apc43 is the vel for "tip"
        ;apc42 is the vel for "leave"
        ;apc0 is the concept "Something occured when david
        ;     exit from the restaurant"

* (xpn apc43)
(*vel* ref (*indef*)
       v1 (*pp* type (#money))
       v2 (*info* type (nil)))
        ;This is the CD representation for "tip"
        ;However, the money sense of tip also has a request associated
        ;with it. A tip is not just money, it is money given to someone
        ;whose occupation is a type of service (*service*).
        ;The request (apr17) looks for an atrans whose object is the
        ;money sense of "tip". If its TO slot is not filled
        ;(e.g., "John gave the barber a tip.") then it is filled with a
        ;person whose occupation type is *service*.

APPLY: searching for  apc14  in  $restaurant
APPLY: trying  rs0
APPLY: trying  rs4
        ;the leaving the restaurant event
APPLY: backbone match on  rs4 with bindings
((&rest . apc14) (&cust . apc4))

APPLY: need bindings:
((&cust . apc4))

ppmem : requesting
     (setoutf ppmem ((&cust . pphum0)))
        ;ppmem says that david could be the customer
        ;it has been established that pptok0 (apc14) "the restaurant"
        ;is the restaurant

APPLY: instantiated event  rs4
APPLY: context active:  $restaurant
         ;rs4 has been established
APPLY: hi-level predictions: (exp-tip)
        ;at the time of leaving, a tip is expected

APPLY: new role bindings:
((&cust . pphum0) (&rest . pptok0))
```

```
        ;meanwhile, back in APE
aprl3 being considered
VEL assert old:  apc43  new:  apc48
        ;the info sense of tip is found by the mtrans sense of leave

* apc43: (*vel* ref apc42 v1 apc46 v2 apc48)
* (xpn apc48): (*info* ref (*indef*) type (nil))
        ;the VEL and its info sense

VEL assert old:  apc30  new:  apc40
        ;the mtrans is asserted

aprl3  has fired
aprl4 being considered
        ;the ptrans sense doesn't find its object

aprl5 being considered
        ;the atrans find its object

VEL assert old:  apc43  new:  apc46
        ;the money sense of tip is asserted.

VEL assert old:  apc30  new:  apc34
        ;the atrans sense of leave is asserted

aprl5  has fired
* !exps!: (exp-tip)
     ;exp-tip is a named request, sent to the parser from APPLY
     ;ape retrieves this request, activates it (aprl8)
     ;and considers it just like other requests
     ;the test looks for an event which could be the tip
     ;and the action puts a confirmation marker on concept
     ;which is used by the script applier so it does not have
     ;to search all the scenes for this event.


VEL assert old:  apc30  new:  apc51
        ;a conflict results because of the two asserts associated
        ;with leave. The conflict resolution scheme results in
        ;a new vel (apc51) with an atrans and a mtrans component.
        ;note that "tip's" disambiguation is not complete
        ;however, its disambiguation is now  dependent on
        ;the disambiguation of "leave"

* (xpn apc51):
(*vel* v0 (*act* mobject (*info* ref (*indef*)
                                 type (nil))
                 actor (*pp* pptok pphum0
                             type (#person)
                             persname (david)
                             gender (*masc*))
                 type (*mtrans*))
       v0 (*act* object (*pp* ref (*indef*)
```

```
                        type (#money))
                actor (*pp* pptok pphum0
                        type (#person)
                        persname (david)
                        gender (*masc*))
                type (*atrans*)))

        ;the vel for "leave a tip"

aprl7 being considered
        ;aprl7 fills the to slot with a person whose occ is (*service*)
aprl7  has fired
:c-list=  (apc0 apc4 apc8 apcl2 apcl4 apcl8 apc20 apc21 apc29
 apc30 apc34 apc36 apc40 apc42 apc43 apc46 apc48 apc50 apc51)
available=  (apc50 apc0)

* (xpn apc34):
(*act* to (*pp* type (#person)
                occ (*service*))
        object (*pp* ref (*indef*)
                        type (#money))
        actor (*pp* pptok pphum0
                        type (#person)
                        persname (david)
                        gender (*masc*))
        type (*atrans*))
        ;the atrans sense of tip now has a to slot
        ;filled by the dictionary definition of "tip"

aprl8 being considered
        ;named request: exp-tip

VEL assert old:  apc51  new:  apc34
        ;one effect of this request disambigutes apc51
        ;in an ambiguous situation, the more expected reading is preferred

aprl8  has fired
:c-list=  (apc0 apc4 apc8 apcl2 apcl4 apcl8 apc20 apc21 apc29
 apc30 apc34 apc36 apc40 apc42 apc43 apc46 apc48 apc50 apc51)
available=  (apc0)

APE: sentence concept:  apc0

* (xpn apc0):
(*conrel* type (*when*)
          cona (*act* object (*pp* type (#person)
                                persname (david)
                                gender (*masc*))
                        actor (*pp* type (#person)
                                persname (david)
                                gender (*masc*))
                        type (*ptrans*)
                        from (*inside* part (*pp* ref (*def*)
                                                type (#struc)
                                                stype (*restaurant*))))
```

```
conb (*act* to (*pp* type (#person)
                       occ (*service*))
             object (*pp* ref (*indef*)
                          type (#money))
             actor (*pp* pptok pphum0
                          type (#person)
                          persname (david)
                          gender (*masc*))
             type (*atrans*)))
;the final conceptualization
```

References

[Birn79] Birnbaum, L., and Selfridge, M., "Problems in Conceptual Analysis of Natural Language," unpublished Computer Science Department Research Report No. 168, Yale University, New Haven, CT.

[Bobr77] Bobrow, D. G., et. al., "GUS, A Frame-Driven Dialog System," Artificial Intelligence, Vol. 8, No. 1.

[Brow75] Brown, J. S., and Burton, R. R., "Multiple Representations of Knowledge for Tutorial Reasoning," in Bobrow, D. and A. Collins (eds.), Representation and Understanding, Academic Press, New York: 1975

[Carb79] Carbonell, J. G., "Subjective Understanding: Computer Models of Belief Systems," unpublished doctoral diss., Computer Science Department Research Report No. 150, Yale University, New Haven, CT.

[Char72] Charniak, E., "Towards a Model of Children's Story Comprehension," unpublished doctoral diss., AI-TR No. 266, Massachusetts Institute of Technology, Cambridge, MA.

[Chom65] Chomsky, N. Aspects of the Theory of Syntax, Cambridge, MIT Press.

[Cull78] Cullingford, R. E., "Script Application: Computer Understanding of Newspaper Stories," unpublished doctoral diss., Computer Science Department Research Report No. 116, Yale University, New Haven, CT.

[Cull79] Cullingford, R. E., "Pattern-Matching and Inference in Story Understanding," Discourse Processes, 2, 319-334, Oct. 1979.

[Cull80] Cullingford, R. E., and M. W. Krueger, "Automated Explanations as a Component of a CAD System," Proc. Int. Conf. on Cybernetics and Society, Cambridge, MA.

[Cull81] Cullingford, R. E., "Integrating Knowledge Sources for Computer 'Understanding' Tasks," IEEE Trans. Systems, Man & Cybernetics, (in press).

[DeJo79] DeJong, G. F., "Skimming Stories in Real Time: An Experiment in Integrated Understanding," unpublished doctoral diss., Computer Science Department Research Report No. 158, Yale University, New Haven, CT.

[Gins78] Ginsparg, J. M., "Natural Language Processing in an Automatic Programming Domain," unpublished doctoral diss., Computer Science Department Research Report No. 78-671, Stanford University, Stanford, CA.

[Gold75] Goldman, N., "Conceptual Generation," in [Scha75]

[Hobb76] Hobbs, J. R., "Pronoun Resolution," unpublished Computer Science Department Research Report No. 76-1, City University of New York, New York, NY.

[Hobb79] Hobbs, J. R., "Coherence and Coreference," Cognitive Psychology, Vol. 3, No. 1.

[Katz63] Katz, J. J., and Fodor, J. A., "The Structure of Semantic Theory," Language, Vol. 39, 170-210.

[Kirb77] Kirby, R. L., "VLISP for PDP-11s with Memory Management," unpublished Research Report TR-546, Department of Computer Science, University of Maryland, College Park, MD.

[McDe78] McDermott, J., and Forgy, C., "Production System Conflict Resolution Strategies," in Waterman & Hayes-Roth, Pattern Directed Inference Systems, Academic Press, New York, NY.

[Meeh76] Meehan, J. R., "The Metanovel: Writing Stories by Computer," unpublished doctoral diss., Computer Science Department Research Report No. 74, Yale University, New Haven, CT.

[Nash78] Nash-Webber, B. L., "Syntax beyond the Sentence: Anaphora," in Spiro, Bruce & Brewer (eds.), Theoretical Issues in Reading Comprehension, Lawrence Erlbaum Associates, Hillsdale, NJ.

[Newe72] Newell, A., and Simon, H. A., Human problem Solving, Prentice-Hall, New York, NY.

[Marc80] Marcus, M., A Theory of Syntactic Recognition for Natural Language, MIT Press, Cambridge, MA.

[Rieg76] Rieger, C., "An Organization of Knowledge for Problem Solving and Comprehension," Artificial Intelligence, Vol. 7, No. 2.

[Ries75] Riesbeck, C. K., "Conceptual Analysis," in [Scha75]

[Ries78] Riesbeck, C., "An Expectation-Driven Production System for Natural Language Understanding," in Waterman & Hayes-Roth, Pattern Directed Inference Systems, Academic Press, New York, NY.

[Ritc74] Ritchie, D. M., and K. Thompson, "The UNIX Time-Sharing System," Comm. ACM, July, 1974.

[Rume75] Rumelhart, D., and D. Norman, "The Active Structural Network," in Norman, D., D. Rumelhart and the LNR Research Group, Explorations in Cognition, Freeman, San Francisco: 1975

[Scha72] Schank, R., "Conceptual Dependency: A Theory of Natural Language Understanding," Cognitive Psychology, Vol. 3, 552-631.

[Scha75] Schank, R. (ed.), Conceptual Information Processing, North Holland, Amsterdam: 1975

[Scha77] Schank, R. and R. P. Abelson, Scripts, Plans, Goals and Understanding, Erlbaum Press, Hillsdale, N. J.: 1977

[Scha78] Schank, R., "Predictive Understanding," in Campbell & Smith (eds.), Recent Advances in the Psychology of Language, Plenum Publishing, New York, NY.

[Schm76] Schmidt, C., et. al., "Recognizing Plans and Summarizing Actions," Proc. Conf. on AI and the Simulation of Behaviour, Univ. of Edinburgh, Edinburgh, UK.

[Smal80] Small, S., "Word Expert Parsing: A Theory of Distributed, Word-Based Natural Language Understanding," unpublished doctoral diss., Computer Science Department Technical Report No. 954, University of Maryland, College Park, MD.

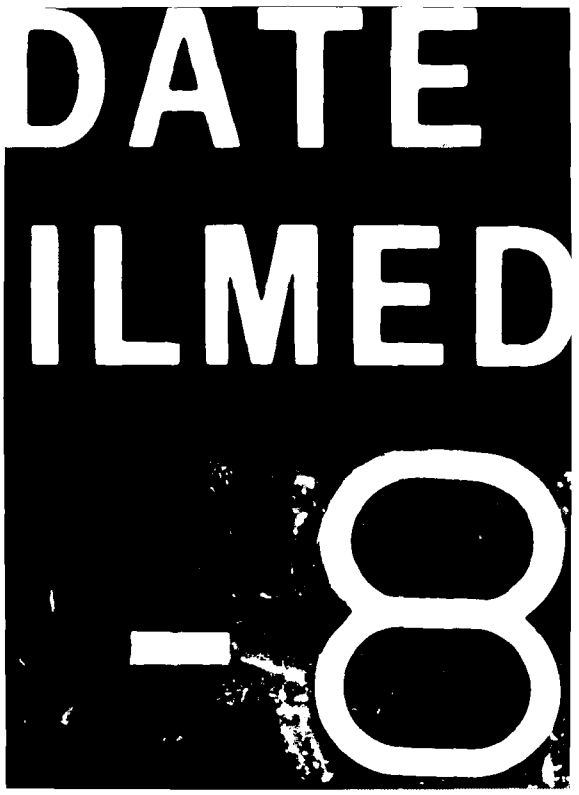
[Turi50] A. M. Turing, "Computing machinery and intelligence," Mind, vol 59, pp. 433-460, Oct. 1950.

[Wile78] Wilensky, R., "Understanding Goal-Based Stories," unpublished doctoral diss., Computer Science Department Research Report No. 140, Yale University, New Haven, CT.

[Wilk75] Wilks, Y., "A Preferential, Pattern-Seeking Semantics for Natural Language Understanding," Artificial Intelligence, 6, pp. 53-74.

[Wino72] Winograd, T., Understanding Natural Language, Academic Press.

[Wood70] Woods, W. A., "Transition Network Grammars for Natural Language Analysis," Comm. ACM, Vol. 13, No. 10.


DATE
ILMED
-8