PRODUCTION NOTE

TECHNICAL REPORTS

Technical Report No. 236

HWIM: A COMPUTER MODEL OF LANGUAGE COMPREHENSION AND PRODUCTION

Bertram Bruce

Bolt Beranek and Newman Inc.

March 1982

Center for the Study of Reading

UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN
51 Gerty Drive
Champaign, Illinois 61820

The National Institute of Education
U.S. Department of Health, Education and Welfare
Washington, D.C. 20208

BOLT BERANEK AND NEWMAN INC.
50 Moulton Street
Cambridge, Massachusetts 02138

CENTER FOR THE STUDY OF READING

Technical Report No. 236

HWIM: A COMPUTER MODEL OF LANGUAGE COMPREHENSION AND PRODUCTION

Bertram Bruce

Bolt Beranek and Newman Inc.

March 1982

University of Illinois
at Urbana-Champaign
51 Gerty Drive
Champaign, Illinois 61820

BBN Report No. 4800
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02238

Bill Woods, Bonnie Webber, Scott Fertig, Andee Rubin, Ron Brachman, Marilyn Adams, Phil Cohen, Allan Collins, and Marty Ringle contributed useful suggestions and criticisms to this paper. Cindy Hunt helped in the preparation. The system development and much of the writing was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533. Additional work on the paper itself was supported by the National Institute of Education under Contract No. HEW-NIE-C-76-0116. Views and conclusions contained here are those of the author and should not be interpreted as representing the official opinion or policy of DARPA, NIE, the U.S. Government, or any other person or agency connected with them.

Abstract

This paper discusses a computer natural language system called "HWIM," which accepts either typed or spoken inputs and produces both typed and spoken responses. HWIM is an example of a relatively complete language system in which one can see how the many components of language processing interact. The paper focuses on HWIM's discourse processing capabilities and on the integration of diverse types of knowledge for the purpose of comprehending and producing natural language.

HWIM: A Computer Model of Language Comprehension and Production

People design computer programs to understand natural language for two reasons. One is to make computers more useful by facilitating communication between the computer and computer users. The other is that the design of such programs can inform theories of human language use. The program then becomes a model for human language comprehension or production. Decisions made in the design process in order to enhance efficiency, or simply to make the program succeed, suggest characteristics of the processes humans carry out when performing similar tasks. Also, the extent to which a theory of language use actually "works" when it is implemented as a computer model is one measure of its correctness. Moreover, the act of expressing a theory in a computer program forces us to be explicit and unambiguous about what the theory is. By examining the development, the structure, and the operation of computer programs that successfully use natural language, one may gain insights into human use of natural language, including reading and writing, speaking and listening.

This paper discusses some general issues of language comprehension and production through examination of a complete natural language system called "HWIM" (for "Hear what I mean"). HWIM was developed over a five year period as the Bolt Beranek

and Newman speech understanding system. It was designed to understand natural language (typed or spoken); to answer questions, perform calculations, and maintain a data base; and to respond in natural language (typed and spoken). Both its inputs and outputs used a relatively rich grammar with a 1000 word vocabulary. Utterances were assumed to be part of an on-going dialogue so that HWIM had to have a model of both the discourse and the user. This paper focuses on the component of the system embodying semantic and pragmatic knowledge. A fuller treatment of the system can be found in Bruce (in press) and in Woods, Bates, Brown, Bruce, Cook, Klovstad, Makhoul, Nash-Webber, Schwartz, Wolf, and Zue (Note 9). For discussions of other speech understanding systems, see Erman, Hayes-Roth, Lesser, and Reddy (1980); Lea (1980); and Walker (1978).

HWIM's Job Responsibilities

HWIM was designed to serve as an assistant for the manager of a travel budget. The task of the system was to assist the travel budget manager, helping to record the trips taken or proposed and to produce such summary information as the total money allocated to various budget items. In order to carry out its duties, HWIM needed to converse with the travel budget manager. Though their conversations were simplified relative to natural inter-personal conversation, the design of the system can

help in formulating more general models of language understanding. Some salient features of HWIM are the following:

- the maintenance of dynamic models of the discourse, the task, the user, and specific facts about the world
- the use of such knowledge to constrain possible interpretations of utterances, thus increasing the likelihood of successful understanding
- the use of the same knowledge in generating both written and spoken responses; a formalism for expressing response generation rules which provides a framework for the application of semantic and discourse level knowledge
- facilities for making the system's own knowledge state known to the travel budget manager
- inference mechanisms and data base structures that facilitate freer expression of commands and questions by the travel budget manager

Steps in Processing

In this section we see a trace of HWIM's processing. The sentences shown here would normally occur as part of an ongoing

dialogue. While their interpretations and resulting responses might be significantly different in context, what can be seen here is the flow of control within the system, the types of interpretations produced, the inference capabilities, and response generation. The structure of HWIM is shown in Figure 1 (and discussed further in Appendix A).

When HWIM engages in a text dialogue, the system reduces to just two processes, Trip and Syntax. In that mode Syntax is a subprocess to Trip, though it may itself invoke Trip as its own subprocess to evaluate tests on arcs in the grammar. To see one possible flow of control, (Figure 2), consider the sentence -

Enter a trip for Jerry Wolf to New York.

Trip reads the input and does a morphological analysis of each word. It then calls Syntax to parse and interpret the sentence. During processing, Syntax may encounter a test, e.g., "Does the name 'Jerry Wolf' denote a known person?", which is to be performed by Trip. Control passes to Trip, which answers "yes," and then back to Syntax again. When the parse is complete, control returns to Trip. Ambiguity in the input can cause a question to be formulated for the travel budget manager. The manager's response would cause a return to Syntax, and so on to the end of the session.

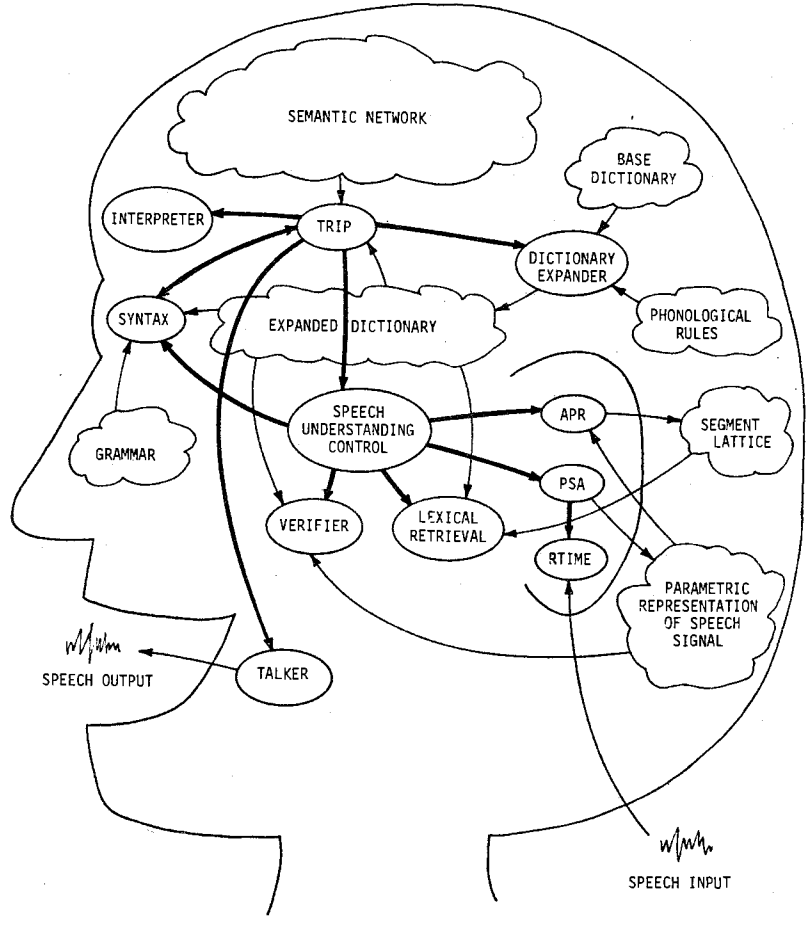

Figure 1. The structure of HWIM.

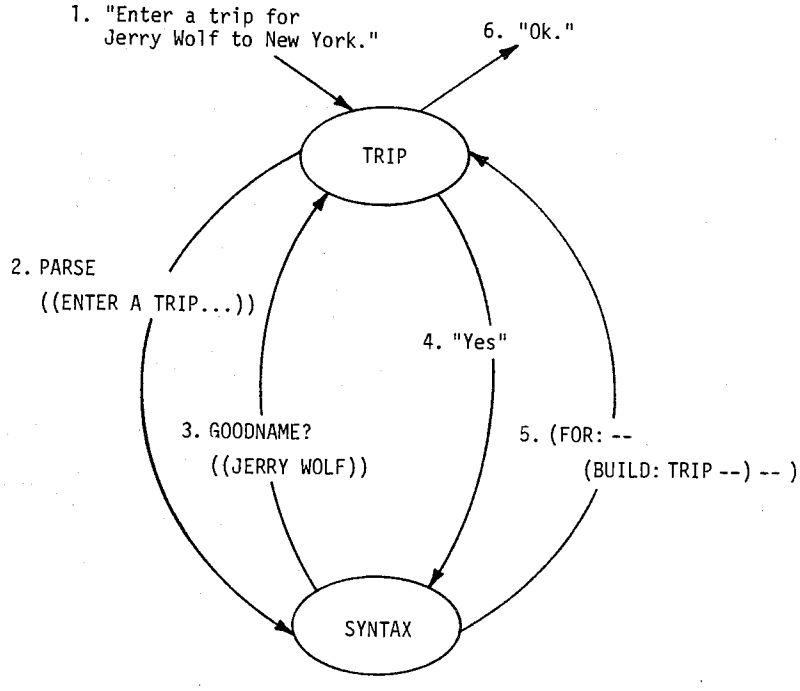

Figure 2. One possible flow of control for a dialogue interchange.

To see the steps HWIM takes in reading, comprehending, and then responding to some text, imagine that the travel budget manager types:

When did Bill go to Mexico?

The program makes a first pass on the input, converting the words as typed into words as they appear in its dictionary and removing punctuation. For example, "$23.16" would become "twenty three dollar-s and sixteen cent-s". In this case the modified word list is simply

(WHEN DID BILL GO TO MEXICO)

For spoken inputs HWIM has to consider many possible word lists because the input is ambiguous. This could be viewed as analogous to problems faced by a person in reading (see Rumelhart, 1977; Woods, 1980b). For typed inputs, HWIM is a perfect decoder; thus, only one word list needs to be considered. In order to simplify the discussion, the example here will focus on a single word list.

The parser in HWIM uses a pragmatic grammar which contains significant static knowledge of the travel budget management domain. Collapsing the task specific knowledge into the grammar was viewed as an expedient to gain computational (processing) efficiency. It helped to constrain parses early and enabled

simultaneous parsing and semantic interpretation. This is in contrast with the two phase method used in the LUNAR parser (Woods, Kaplan & Nash-Webber, Note 10) which produced purely syntactic parse trees that were subsequently given to a semantic interpreter to transform into executable interpretations. More recently, techniques such as the use of cascaded ATN's (Woods, 1980a) as in the RUS parser (Bobrow, Note 2) have been developed to gain much of the efficiency of pragmatic or semantic grammars while maintaining a clean separation between general syntactic knowledge and task specific knowledge.

For the example sentence, the parser derives the syntactic structure:

```
S Q
  QADV WHEN
  SUBJ NPR BILL
  AUX TNS PAST
      VOICE ACTIVE
  VP V GO
     PP PREP TO
        NP NPR MEXICO
  ADV THEN
```

Taking advantage of the semantic knowledge in the grammar, we get

in addition:

```
[FOR: THE A0009 / (FIND: LOCATION (COUNTRY 'MEXICO))
     : T ; (FOR: THE A0008 / (FIND: PERSON (FIRSTNAME 'BILL))
                T ; (FOR: ALL A0010 /
                          (FIND: TRIP (TRAVELER A0008)
                                  (DESTINATION A0009)
                                  (TIME (BEFORE NOW)))
                          : T ; (OUTPUT: (GET: A0010 'TIME]
```

The interpretation is a quantified expression which can (almost) be directly executed. Translated it says, roughly:

> Find the location which has the country name "Mexico" and call it A0009. Find the person whose first name is "Bill" and call him A0008. Find in turn, all trips whose traveler is A0008, whose destination is A0009, and which occurred prior to now. As each is enumerated, label it A0010 and output its specific time.

The T's which appear in the interpretation are there in place of potential restrictions which might have been applied to the location, person, or trip classes being searched.

Before the interpretation can be executed it undergoes an optimization (see Reiter, 1976) and modification to match the

data base representations. In this example there is no entity called "LOCATION" in the semantic network data base which serves the purpose implied in the interpretation. There are cities, states, coasts, countries, etc. and there is a link called "LOCATION", but no concept with instances as we might expect. The interpretation thus references a fictitious node. The optimizer recognizes this and modifies the interpretation so that a search is done among all the possible "locations". In HWIM, this modification is specific to the fictitious node that has been referenced. More recent knowledge representation systems, such as KL-ONE would formalize the ability to form abstractions such as "location" from the more specialized concepts (Brachman, Bobrow, Cohen, Klovstad, Webber, & Woods, Note 3).

The optimized interpretation is placed on a queue of to-be-executed commands. The queue represents a first step towards a "demand model of discourse" (see next section). In principle it can contain previous queries or commands from the speaker as well as system initiated commands. However, most utterances translate into single commands which are directly executed.

The (FOR: --) expression above is a LISP form which can be evaluated directly. The FOR: function manages quantification in the expected way. Here it looks for a unique location and a

unique person which match the respective descriptions. There is a single place whose country name is "Mexico," i.e., the country, Mexico. However, there are 8 items in the data base representing people whose first name is "Bill." Given the apparent inconsistency between the data base and the command, FOR: is forced to take the initiative of the dialogue and ask the manager for help. The response generation program produces the question:

> Do you mean Billy Diskin, Bill Huggins, Bill Levison, Bill Patrick, Bill Plummer, Bill Russell, Bill Merriam, or Bill Woods?

When the system asks a question (as in this example) it expects an answer. This means that otherwise "incomplete" utterances may be accepted. Here the parser will allow a name as a complete utterance. In this context, the person types a second sentence:

Bill Woods.

This sentence undergoes a pre-pass to produce the word list

(BILL WOODS)

which the parser interprets as

```
(!  THE A0011 PERSON ((FIRSTNAME BILL)
                       (LASTNAME WOODS)))
```

The optimized interpretation is

```
(IOTA 1 A0011 / (FIND:  PERSON
                        (FIRSTNAME 'BILL)
                        (LASTNAME 'WOODS)) : T )
```

This interpretation uses the IOTA operator, a function that returns the one item in the class defined by the (FIND: - -) expression which meets the restriction, T (i.e., no restriction). In this case there is exactly one item matching the description, namely BILL WOODS.

At this point, we are in the middle of what Schegloff (1972) has called an "insertion sequence." In the process of answering the user's question, the system began executing a FOR: expression. Evaluating the FOR: expression required information that was obtainable only by asking the user a question. HWIM's question to the user, plus the user's answer constitute an insertion sequence with respect to the primary question-answer sequence. The execution of the FOR: expression is suspended for the duration of the insertion sequence, but can be resumed when the needed information is processed. Having now a unique DESTINATION and a unique TRAVELER, FOR: can find all trips whose TIME is (BEFORE NOW) and for each, output its TIME.

Finding the trips, checking TRAVELER, DESTINATION, and TIME can require considerable search in the data base. For example,

the DESTINATION of a TRIP is computed from the DESTINATIONs of its component LEG/OF/TRIPs. The DESTINATION of a LEG/OF/TRIP may have to be computed from the PURPOSE of the LEG/OF/TRIP, e.g., attending a specific conference. Information about trips, budgets, conferences, etc. is never assumed to be complete since it is not so in the real world. Furthermore, the range of askable questions precludes computing in advance all the implicit information. (The procedures that do these computations are discussed in Woods et al., Note 9.) In this example there is only one trip that matches the description and its time is printed out by the response generation programs:

> Bill Woods's trip from Boston, Massachusetts to Juarez, Mexico was from October 15th to 17th, 1975.

Discourse Model

In any conversation there are many forces operating to determine what will be said next, including the memories, the intentions, and the perceptions of the participants in the conversation (Bruce, 1980, Note 4). A discourse model is an idealized representation of such forces as they are used by a speaker in constructing an utterance and by a listener in understanding one. (See also Reichman, 1978; Stansfield, 1974; Deutsch, 1974; Sidner, Note 8).

The discourse context is an important factor in even the restricted domain of person-computer communication, but in many interactive natural language understanding systems there is no explicit "discourse model". Most artificial problem domains are deliberately constructed so that only a few interaction modes are sensible. For example, the person may only ask questions; the system may only answer the questions. This greatly simplifies the determination of the speaker's intent, and hence, the full understanding of the utterance. There is also no need to maintain a data base of participants in a conversation (together with their presumed purpose, attitudes, models of the world, etc.) when the only other participant in the conversation is "the USER." Even in those cases where the problem definition permits a general discourse and the system is able to cope with that generality, the means of coping are rarely delineated (Bruce, 1975b).

Incremental simulations (see Woods & Makhoul, 1973) of dialogues between a travel budget manager and HWIM (a system builder played the part of an ideal HWIM) suggested a discourse model in which, at any point, the manager could be seen as being in one of several states, e.g., trying to determine the consequences of a proposed trip, examining the state of the budget, or entering new trip information. We considered several formulations of a discourse model in which these states would be

made explicit and used to advantage.

One such formulation involved the concepts of modes of interaction and intents (Bruce, 1975c). An intent is the assumed purpose behind an utterance. Patterns of intents such as,

user-enter-new-information
    system-point-out-contradiction
        user-ask-question
        system-answer-question
    user-make-editing-change
system-confirm-change,

constitute the modes of interaction.

An augmented transition network (ATN) grammar was used to represent some of the common modes of interaction found in travel budget management dialogues. Then a modified ATN parser was written that steps through the grammar on the basis of the input sentence structure and the then-current state of the data base. At any given state the parser can predict the most likely next intent and hence such things as the class of likely verbs for the next utterance. This model was not used in the final version of the system, primarily because it was too rigid to function alone as a discourse model.

Another formulation of the user/discourse model involves the

notion of demands made by participants in the dialogue. (Both of these formulations are discussed more fully in Bruce, 1975b). Demands include such things as unanswered questions and contradictions which have been pointed out. This latter formulation allows us to model how one computation of a response can be pushed down, while a whole dialogue takes place to obtain missing information, and how a computation can spawn subsequent expectations or digressions. Some elements of this demand model, together with other aspects of the discourse model, are explained below:

(1) Demands: These are demands for service of some sort made upon the system by the user or by the system itself. An active unanswered question is a typical demand with high priority. The fact that some questions cannot be answered without more information leads to an embedding of modes of interaction. Demands of lower priority include such things as a notice by the system that the manager is over his budget. Such a notice might not be communicated until after direct questions had been answered.

(2) Counter-demands: These are questions the system has explicitly or implicitly asked the user (e.g., "This trip will put you over budget" implicitly asks the user to review the budget, canceling budget items or adjusting cost estimates).

While it should not hold on to these as long as it does to demands, nor expect too strongly that they will be met, the system can reasonably expect that most counter-demands will be resolved in some way. This is an additional influence on the discourse structure.

(3) Current discourse state: The discourse area of the data base also contains an assortment of items that define the current discourse context, including:

- LOCATION, a pointer to the current location of the speaker, e.g., the city
- TIME, a pointer to the current time and date
- SPEAKER, a pointer to the current speaker
- the last mentioned person, place, time, trip, budget, conference, etc.

There is also a representation of the current TOPIC (see Figure 3). This is the active focus of attention in the dialogue. It could be the actual budget, a hypothetical budget, a particular trip, or a conference. The current topic is used as an anchor point for resolving definite references and deciding how much detail to give in responses. It can also generate certain modes of interaction. For example, if the manager says "Enter a trip,"

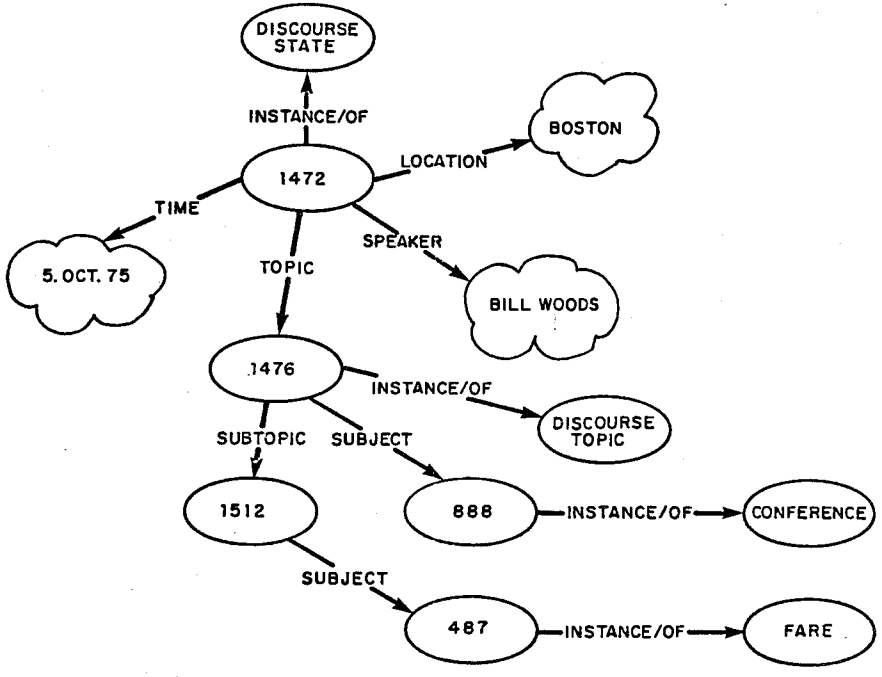

Figure 3. A discourse state.

the system notes that the current topic has changed to an incompletely described trip. This results in demands that cause standard fill-in questions to be asked. If the manager wants to complete the trip description later, then the completion of the trip description becomes a low priority demand.

The system has a primitive one-queue implementation of the "demand model." This queue contains executable procedures which represent the speaker's previous queries and commands as well as commands initiated by the system to examine the consequences of its actions, give information to the user, or check for data base consistency. These procedures are related by functional dependencies and relative priorities. The major types of demands are the following:

- DO: means execute the specified command.
- TEST: means evaluate the form and answer "no" if NIL or "yes" otherwise.
- RESPOND: means give the user some information (which may or may not be part of an answer to a direct query).
- PREVENT: means monitor for a subsequent possible action and block its normal execution (as in "Do not allow more than three trips to Europe.").

Understanding

The process of understanding written natural language requires the integration of diverse knowledge sources. An optimal process must apply various types of knowledge in a balance that avoids over- or under-constraining the set of potential accounts of the input.

Ultimately the understanding process should produce a meaning representation for the natural language input, whether the input be written or spoken. This representation may be produced directly or via some number of contingent knowledge structures (Bobrow & Brown, 1975). The exact form of these structures varies from one system to another, but typically includes such things as parse trees.

In HWIM, integration of knowledge sources for linguistic processing is accomplished by means of (1) an ATN grammar (see Figure 4) that incorporates much of the syntactic, semantic, and pragmatic knowledge in the system, (2) a semantic network (see Nash-Webber, 1975) that represents remaining linguistic and world knowledge, and (3) tests on arcs in the grammar used to link the two representations. These structures are used to produce the account of the input that serves as a representation of its meaning.

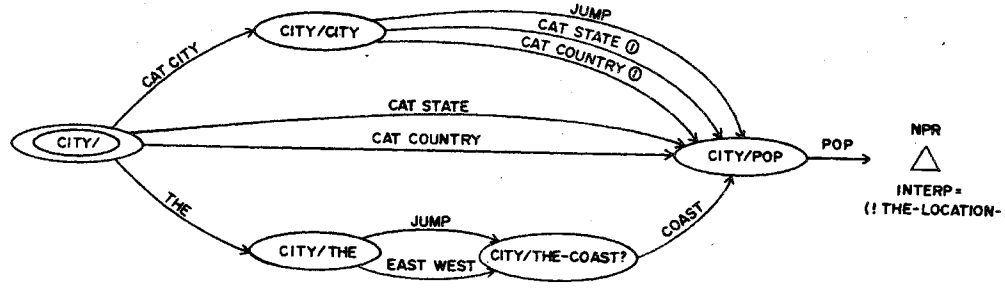

Figure 4. A small portion of the pragmatic grammar.

The decision to include semantic knowledge in a grammar has both its merits and its defects. On the one hand there is an obvious advantage in terms of efficiency when semantic knowledge can be applied early on in the most direct way to constrain the possible interpretations of an utterance. Moreover, one avoids the seemingly unnecessary step of building a syntactic model, such as a parse tree, for an utterance. All of this points to the desirability of a unified linguistic processor as exists in HWIM (see also Erman, Hayes-Roth, Lesser, & Reddy, 1980; Burton & Brown, Note 5). On the other hand, the fact that we could incorporate semantic and pragmatic knowledge in the grammar and use that knowledge as if it were fundamentally no different from syntactic knowledge was a consequence of having a limited domain of discourse. No computer natural language system has yet approached the variety and complexity that natural human communication exhibits. As semantic and pragmatic knowledge becomes more complex, more fluid, and more intricately interconnected, it is not clear to what extent the pragmatic grammar approach will work.

Despite the inclusion of non-syntactic knowledge, the pragmatic grammar is not a complete knowledge source by itself for linguistic processing. It is, however, closely linked to the semantic network via tests on the arcs. These tests allow the grammar to capture more volatile constraints on the input, such

as those provided by the discourse model or the factual data base. For example:

- Is "Jerry Woods" a valid name?
- Is "San Francisco, California" a valid place?
- What words can go with "speech" as a project descriptor (e.g., "understanding")?
- Does "What is the registration fee?" make sense in this context?
- Does "How much is in the budget?" make sense in this context?
- Does "When is that meeting?" make sense in this context?
- Does "November 15th" make sense in this context?

With the grammar encoding semantic and pragmatic, as well as syntactic information, it is possible for the parser to build a procedural "meaning" representation as well as a purely syntactic one. The meaning representations are in a command language whose expressions are atomic, consisting of a functional operator applied to arguments, or compound, making use of a quantification operator (FOR:) applied to another expression. The operators in

atomic expressions specify operations to be performed on the data base or interactions with the user.

The parser builds interpretations by accumulating in registers the semantic head, quantifier, and links of the nodes being described in the sentence (see Woods et al., Note 9, for a more complete description). For example, the sentence

I will go to the ASA meeting.

yields an interpretation in the command language (Section COMMAND) of the form

```
(FOR: THE A0018 / (FIND: MEETING (SPONSOR 'ASA)) : T ;
     (FOR: 1 A0019 / (BUILD: TRIP
                             (TO/ATTEND A0018)
                             (TRAVELER SPEAKER)
                             (TIME (AFTER NOW)))
                : T ; T))
```

This interpretation is built up in the following way. A constituent describing a person is found at the start of the sentence. The parser transforms the pronoun "I" into the link-node pair (TRAVELER SPEAKER) and returns this as the interpretation of that constituent. The word "will" adds (TIME (AFTER NOW)) to the list of link-node pairs being accumulated. (The grammar does not accept constructions like "will have gone,"

so "will" can always be interpreted as marking a future event.) The word "go," in the context of our travel domain, indicates that a trip is being discussed. Next, the constituent "the ASA meeting" produces

```
(TO/ATTEND (! THE Y MEETING ((SPONSOR ASA)))).
```

(The ! indicates that a FOR: expression will have to be built as part of the interpretation.) The top level of the grammar has thus accumulated the link-node pairs

```
(TIME (AFTER NOW))
(TRAVELER SPEAKER)
(TO/ATTEND (! THE Y MEETING ((SPONSOR 'ASA))))
```

with the semantic head TRIP. The appropriate action (in this case a BUILD:) is created, to produce

```
(BUILD:  TRIP
         (TO/ATTEND (! THE Y MEETING ((SPONSOR 'ASA))))
         (TRAVELER SPEAKER)
         (TIME (AFTER NOW)))
```

Finally, the necessary quantificational expressions are expanded around the BUILD: expression.

Response Generation

Once a satisfactory theory for an utterance has been constructed, it must be acted upon by the system. Regardless of

the type of action taken, some appropriate response should also be made. From the speaker's point of view the response should be explicit, concise, and easy to understand. The response may affect the speaker's way of describing entities in the domain as well as illuminate the capabilities and operation of the system.

The effects of generated responses are many. In a domain such as travel budget management there are many objects without standard names, e.g., "the budget item for two trips to a West Coast conference". Names chosen (constructed) by HWIM provide a handle for the manager to use and thus strongly influence the ways the objects are referred to subsequently. Responses can also indicate the capabilities of the program. Careful construction of responses to exhibit exactly the knowledge structures handled by HWIM gives the manager the legitimate confidence to pursue just the paths which rely on those structures. A response can also show the system's focus of attention. The manager can thus get a better idea, not only of the facts she or he seeks, but also of how well the system is understanding and where more clarification may be useful.

Types of Knowledge Used in HWIM

In order to carry out its role effectively, HWIM must have a large amount of knowledge in a readily accessible form. This knowledge is needed for understanding both spoken utterances and

written sentences, carrying out commands, answering questions, and generating responses. HWIM's success in performing these tasks is evidence that the types of knowledge embodied in the system are sufficient for natural communication, at least at the level shown by sample dialogues. The history of the development of HWIM (Nash-Webber & Bruce, 1976) suggests that each category of knowledge is also necessary for natural communication to take place. The kinds of knowledge used by HWIM can be categorized as follows:

- Acoustic-Phonetic forms--Knowledge of phonemes and their relation to acoustic parameters.
- Phonological rules--Knowledge of phoneme clusters and pronunciation variations describable by rules.
- Lexical forms--Knowledge of words, their inflections, parts of speech and phonemic spellings. For example, "entered" is the past form of "to enter".
- Sentential forms--Knowledge of grammatical structures at the phrase, clause, and sentence level. For example, a sentence may start with a command verb, such as "schedule."
- Discourse form--Knowledge of idealized discourse,

e.g., what type of utterance is likely to be produced in a given state of the discourse.

- Semantic--Knowledge of how words are related and used and structurally conveyed meaning. This is used in parsing and in constructing responses. For example, the benefactive case for schedule should be filled by a person who will be taking the trip being scheduled.

- World--Knowledge of specific projects, trips, budgets and conferences. Whereas HWIM's semantic knowledge is essentially fixed, its world knowledge changes every time a trip is taken or money is shifted from one budget item to another.

- On-Going discourse--Knowledge of the current discourse state, e.g., the current topic and the objects available for anaphoric reference. This knowledge is used to relax constraints in the pragmatic grammar. For example, if a conference is under discussion, then "What is the registration fee?" is a meaningful utterance. If not, then the speaker would have to say something like, "What is the registration fee for the ACL conference?"

- Parameters of the communicative situation--HWIM's operation is a function of the modality in which it operates; speech or text input and speech or text output. In general, communicative situations can vary along a number of dimensions (see Rubin, 1980) and a successful communicator must use knowledge of the situation to interpret and respond appropriately.

- User--Knowledge about each possible travel budget manager, what groups they belong to, and what they may know about the data base. For example, only certain users may be allowed to modify budget totals.

- Self--Knowledge about the system's own knowledge (often called "meta-knowledge"). One example is that data on trips and budgets is marked to indicate whether it came from the travel budget manager or was computer by HWIM. Another is that HWIM knows what elements constitute a complete description of any object, such as a trip. Thus it knows when its own knowledge is incomplete.

- Task--Knowledge of the task domain, e.g., that a manager is concerned with maintaining an accurate

record of all trips, taken or planned, and with staying within the budget.

- o Strategic--Knowledge concerning the representation and use of other knowledge. This knowledge that would be needed in even a "blank" system; i.e., one that had no word- or domain-specific knowledge.

Conclusion

One reason for building a computer model of language understanding is that in the course of designing and debugging a computer program, one must resolve theoretical questions about details of the process that can be glossed over in less procedure-oriented models. For example, designers of computer models of speech act generation (Cohen & Perrault, 1979) have increased the precision of speech act definitions. A second reason for computer models is that the task of the system can be defined to require complete use of language (within a restricted domain, of course). Thus, one can see what components, each representing aspects of language theory, are needed and how they might interact. Winograd's (1972) blocks world program, a system which accepted typed questions, commands, and assertions, performed actions, and generated natural language responses, could be put in the latter category. There is value in both of these approaches; in fact, they tend to complement each other,

with the second showing what can and cannot be done and the first addressing specific design questions.

HWIM is in both categories, but its principal value probably lies more in the second than the first. To some extent, it represents a synthesis of a number of lines of research in areas such as parsing, inference, data base design, and generation. It shows what can be done in a fairly complete system that understands natural language (typed or spoken), answers questions and performs various actions, and responds in natural language (typed and spoken).

Problems that arose in the design of HWIM were precursors of those that are central issues in AI research today. For example, the speech act issues for HWIM are similar to those studied by Cohen (Note 6), Cohen and Perrault (1979), and Allen (Note 1). Questions of knowledge representation closely related to those faced in HWIM have been pursued by Bobrow and Winograd (1977), Brachman (1979), and Fahlman (1979). (Also see Brachman & Smith, 1980). Language generation in a discourse context similar to HWIM's has been studied by McDonald (Note 7), for instance. Finally, the issue of interactions among syntax, semantics, and pragmatics is crucial in the work of many, including Schank and Abelson (1977), Woods (1980a), and Bobrow (Note 2). The characteristics of HWIM reflect the goal of natural communication

between a person and a computer assistant. Even in its limited domain, it illustrates the extent to which natural communication depends upon diverse kinds of knowledge in both communicants. The structure of HWIM can provide a useful framework for obtaining a better understanding of natural communication.

Reference Notes

1. Allen, J. A plan-based approach to speech act recognition (Tech. Rep. No. 131/79). Toronto: University of Toronto, Department of Computer Science, January 1979.

2. Bobrow, R. J. The RUS system (BBN Report No. 3878). Cambridge, Mass.: Bolt Beranek and Newman Inc., 1978.

3. Brachman, R. J., Bobrow, R., Cohen, P., Klovstad, J., Webber, B. L., & Woods, W. A. Research in natural language understanding Annual report (1 Sept. 78--31 August 1979).

4. Bruce, B. C. Belief systems and language understanding BBN Report No. 2973). Cambridge, Mass.: Bolt Beranek and Newman, Inc., 1975. (a)

5. Burton, R., & Brown, J. S. Semantic grammar: A Technique for constructing natural language interfaces to instructional systems (BBN Report No. 3587). Cambridge, Mass.: Bolt Beranek and Newman Inc., May 1977.

6. Cohen, P. R. On knowing what to say: Planning speech acts (Tech. Rep. No 118). Toronto: Department of Computer Science, University of Toronto, 1978.

7. McDonald, D. Natural language production as a process of decision making under constraint (MIT AI Lab Tech. Rep.). Cambridge, Mass.: Massachusetts Institute of Technology, 1980.

8. Sidner, C. L. Towards a computational theory of definite anaphora comprehension in English discourse (MIT AI Lab Tech. Rep. No. 537). Cambridge, Mass.: Massachusetts Institute of Technology, 1979.

9. Woods, W. A., Bates, M. Brown, G., Bruce, B., Cook, C., Klovstad, J., Makhoul, J., Nash-Webber, B., Schwartz, R., Wolf, J., Zue, V. Speech understanding systems: Final technical progress report (30 October 1974--29 October 1976). Cambridge, Mass.: Bolt Beranek and Newman Inc., 1976.

10. Woods, W. A., Kaplan, R. M., & Nash-Webber, B. L. The lunar science natural language information system: Final report (BBN Report No. 2378). Cambridge, Mass.: Bolt Beranek and Newman, Inc., 1972.

References

Bobrow, D. G., & Winograd, T. An overview of KRL, a knowledge representation language. Cognitive Science, 1977, 3, 3-46.

Bobrow, R. J., & Brown, J. S. Systematic understanding: Synthesis, analysis, and contingent knowledge in specialized understanding systems. In D. G. Bobrow & A. Collins (Eds.) Representation and understanding: Studies in cognitive science. New York: Academic Press, 1975.

Brachman, R. J. On the epistemological status of semantic networks. In N. V. Findler (Ed.), Associative networks: The representation and use of knowledge in computers. New York: Academic Press, 1979.

Brachman, R. J., & Smith, B. C. SIGART Newsletter, Special Issue on Knowledge Representation, No. 70, February 1980.

Bruce, B.C. Pragmatics in speech understanding. Proceedings Fourth International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 1975.(b)

Bruce, B. C. Discourse models and language comprehension. American Journal for Computational Linguistics 1975, 35, 19-35.(c)

Bruce, B.C. Analysis of interacting plans as a guide to the understanding of story structure. Poetics, 1980, 9, 295-311.

Bruce, B. C. Natural communication between person and computer. In W. Lehnert and M. Ringle (Eds.), Strategies for natural language processing. Hillsdale, N.J.: Erlbaum, in press.

Cohen, P. R., & Perrault, C. R. Elements of a plan-based theory of speech acts. Cognitive Science, 1979, 3, 177-212.

Deutsch, B. G. The structure of task oriented dialogues. In L.D. Erman (Ed.), Proceedings of the IEEE Symposium on Speech Recognition. Pittsburgh, PA: Carnegie Mellon University, 1974.

Erman, L. D., Hayes-Roth, F., Lesser, V. R., & Reddy, D. R. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. Computing Surveys, 1980, 12, 213-253.

Fahlman, S. E. NETL: A system for representing and using real-world knowledge. Cambridge, Mass.: MIT Press, 1979.

Lea, W. A. (Ed.) Trends in speech recognition. Englewood Cliffs, N.J.: Prentice-Hall, 1980.

Nash-Webber, B. L. The role of semantics in automatic speech understanding. In D. G. Bobrow & A. Collins (Eds.), Representation and understanding: Studies in cognitive science. New York: Academic Press, 1975.

Nash-Webber, B., & Bruce, B. C. Evolving uses of knowledge in a speech understanding system. Proceedings of the COLING-76 Conference, Ottawa, Canada, June 1976.

Reichman, R. Conversational coherency. Cognitive Science, 1978, 2, 283-327.

Reiter, R. Query optimization for question-answering systems. Proceedings of the COLING-76 Conference, Ottawa, Canada, June 1976.

Rubin, A. D. A theoretical taxonomy of the differences between oral and written language. In R. J. Spiro, B. C. Bruce, and W. F. Brewer (Eds.), Theoretical issues in reading comprehension. Hillsdale, N.J.: Erlbaum, 1980.

Rumelhart, D. E. Towards an interactive model of reading. In S. Dormic (Ed.), Attention and performance VI. Hillsdale, N.J.: Erlbaum, 1977.

Schank, R. C., & Abelson, R. P. Scripts, plans, goals and understanding: An inquiry into human knowledge structures. Hillsdale, N.J. Erlbaum, 1977.

Schegloff, E. A. Notes on a conversational practice: Formulating place. In D. Sudnow (Ed.), Studies in social interaction. New York: Free Press, 1972.

Stansfield, J. L. Programming a dialogue teaching situation. Unpublished doctoral dissertation, University of Edinburgh, 1974.

Walker, D. E. (Ed.) Understanding spoken language. New York: Elsevier North-Holland, 1978.

Winograd, T. Understanding natural language. New York: Academic Press, 1972.

Woods, W. A. Cascaded ATN Grammars. American Journal of Computational Linguistics, 1980, 6, 1-12. (a)

Woods, W. A. Multiple theory formation in high-level perception. In R. J. Spiro, B. C. Bruce, & W. F. Brewer (Eds.), Theoretical issues in reading comprehension. Hillsdale, N.J.: Erlbaum, 1980. (b)

Woods, W. A., & Makhoul, J. Mechanical inference problems in continuous speech understanding. Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, Calif., 1973.

APPENDIX A

The Structure of HWIM

The structure of HWIM is shown in Figure 1. Components of the system are shown in ovals with thick arrows representing flow of control between components. Major data structures are shown in clouds with thin arrows indicating data flow. The system is implemented on TENEX, a virtual memory time-sharing system for the DEC PDP-10. Each component exists in a separate TENEX process (called a "fork"). Among the reasons for this multiple-process job structure (see Woods, et al., 1976) are that most of the components are so large, in terms of program and data structure requirements, that they cannot exist in the same TENEX address space with another component.

The Trip component is the one primarily discussed in this paper. It controls the system and is the major component for acting on and responding to an utterance. If it is given a typed input, it calls Syntax to parse the sentence. Otherwise it calls Speech Understanding Control.

Most of the other components perform single functions: Dictionary Expander expands a dictionary of baseform pronunciations using a set of phonological rules (applied once at system loadup time). Real Time Signal Acquisition (RTIME)

digitizes and stores the speech signal. Signal Processing (PSA) converts the speech signal into a parametric representation. Acoustic-Phonetic Recognizer (APR) operates on the parametric representation of the speech signal to produce a set of phonetic hypotheses represented in the segment lattice. Lexical Retrieval searches the expanded dictionary and matches pronunciations against portions of the segment lattice. Verifier generates an idealized spectral representation of a word (or words), using a speech synthesis-by-rule program and matches it against a region of the parametric representation of the speech signal. Syntax judges the grammaticality of a given word sequence; predicts possible extensions at each end of the sequence; builds a formal representation of an utterance. Interpreter (present in an early version only) takes a syntactic representation of an utterance and builds a procedural representation of the meaning. Talker generates speech from phoneme-prosodic cue strings.

This page is intentionally blank.