TOWARDS A THEORY OF COMPREHENSION OF DECLARATIVE CONTEXTS

Fernando Gomez
Department of Computer Science
University of Central Florida
Orlando, Florida 32816

ABSTRACT

An outline of a theory of comprehension of declarative contexts is presented. The main aspect of the theory being developed is based on Kant's distinction between concepts as rules (we have called them conceptual specialists) and concepts as an abstract representation (schemata, frames). Comprehension is viewed as a process dependent on the conceptual specialists (they contain the inferential knowledge), the schemata or frames (they contain the declarative knowledge), and a parser. The function of the parser is to produce a segmentation of the sentences in a case frame structure, thus determininig the meaning of prepositions, polysemous verbs, noun group etc. The function of this parser is not to produce an output to be interpreted by semantic routines or an interpreter, but to start the parsing process and proceed until a concept relevant to the theme of the text is recognized. Then the concept takes control of the comprehension process overriding the lower level linguistic process. Hence comprehension is viewed as a process in which high level sources of knowledge (concepts) override lower level linguistic processes.

1. Introduction

This paper deals with a theory of computer comprehension of descriptive contexts. By "descriptive contexts" I refer to the language of scientific books, text books, this text, etc.. In the distinction performative vs. declarative, descriptive texts clearly fall in the declarative side. Recent work in natural language has dealt with contexts in which the computer understanding depends on the meaning of the action verbs and the human actions (plans, intentions, goals) indicated by them (Schank and Abelson 1977; Grosz 1977; Wilensky 1978; Bruce and Newman 1978). Also a considerable amount of work has been done in a plan-based theory of task oriented dialogues (Cohen and Perrault 1979; Perrault and Allen 1980; Hobbs and Evans 1980). This work has had very little bearing on a theory of computer understanding of descriptive contexts. One of the main tenets of the proposed research is that descriptive (or declarative as we prefer to call them) contexts call for different theoretical ideas compared to those proposed for the understanding of human actions, although, naturally there are aspects that are common.

An important characteristic of these contexts is the predominance of descriptive predicates and verbs (verbs such as "contain," "refer," "consist of," etc.) over action verbs. A direct result of this is that the meaning of the sentence does not depend as much on the main verb of the sentence as on the concepts that make it up. Hence meaning representations centered in the main verb of the sentence are futile for these contexts. We have approached the problem of comprehension in these contexts by considering concepts both as active agents that recognize themselves and as an abstract representation of the properties of an object. This aspect of the theory being developed is based on Kant's distinction between concepts as rules (we have called them conceptual specialists) and concepts as an abstract representation (frames, schemata). Comprehension is viewed as a process dependent on the conceptual specialists (they contain the inferential knowledge), the schemata (they contain structural knowledge), and a parser. The function of the parser is to produce a segmentation of the sentences in a case frame structure, thus determining the meaning of prepositions, polysemous verbs, noun group, etc.. But the function of this parser is not to produce an output to be interpreted by semantic routines, but to start the parsing process and to proceed until a concept relevant to the theme of the text is recognized. Then the concept (a cluster of production rules) takes control of the comprehension process overriding the lower level linguistic processes. The concept continues supervising and guiding the parsing until the sentence has been understood, that is, the meaning of the sentence has been mapped into the final internal representation. Thus a text is parsed directly into the final knowledge structures. Hence comprehension is viewed as a process in which high level sources of knowledge (concepts) override lower level linguistic processes. We have used these ideas to build a system, called LLULL, to understand programming problems taken verbatim from introductory books on programming.

2. Concepts, Schemata and Inferences

In Kant's Critique of Pure Reason one may find two views of a concept. According to one view, a concept is a system of rules governing the application of a predicate to an object. The rule that

tells us whether the predicate "large" applies to the concept Canada is a such rule. The system of rules that allows us to recognize any given instance of the concept Canada constitutes our concept of Canada. According to a second view, Kant considers a concept as an abstract representation (vorstellung) of the properties of an object. This second view of a concept is akin to the notion of concept used in such knowledge representation languages as FRL, KLONE and KRL.

Frames have played dual functions. They have been used as a way to organize the inferences, and also as a structural representation of what is remembered of a given situation. This has caused confusion between two different cognitive aspects: memory and comprehension (see Ortony, 1978). We think that one of the reasons for this confusion is due to the failure in distinguishing between the two types of concepts (concepts as rules and concepts as a structural representation). We have based our analysis on Kant's distinction in order to separate clearly between the organization of the inferences and the memory aspect. For any given text, a thematic frame contains structural knowledge about what is remembered of a theme. One of the slots in this frame contains a list of the relevant concepts for that theme. Each of these concepts in this list is separately organized as a cluster of production rules. They contain the inferential knowledge that allows the system to interpret the information being presently processed, to anticipate incoming information, and to guide and supervise the parser (see below). In some instances, the conceptual specialists access the knowledge stored in the thematic frame to perform some of these actions.

3. Linguistic Knowledge, Text Understanding and Parsing

In text understanding, there are two distinct issues. One has to do with the mapping of individual sentences into some internal representation (syntactic markers, some type of case grammar, Wilks' preference semantics, Schank's conceptual dependency etc.). In designing this mapping, several approaches have been taken. In Winograd (1972) and Marcus (1979), there is an interplay between syntax, and semantic markers (in that order), while in Wilks (1973) and Riesbeck (1975) the parser rely almost exclusively on semantic categories.

A separate issue has to do with the meaning of the internal representation in relation to the understanding of the text. For instance, consider the following text (it belongs to the second example):

> "A bank would like to produce records of the transactions during an accounting period in connection with their checking accounts. For each account the bank wants a list showing the balance at the beginning of the period, the number of deposits and withdrawals, and the final balance."

Assume that we parse these sentences into our favorite internal representation. Now what we do with the internal representation? It is still far distant from its textual meaning. In fact, the first sentence is only introducing the topic of the programming problem. The writer could have achieved the same effect by saying: "The following is a checking account problem". The textual meaning of the second sentence is the description of the output for that problem. The writer could have achieved the same effect by saying that the output for the problem consists of the old-balance, deposits, withdrawals, etc.. One way to produce the textual meaning of the sentence is to interpret the internal representation that has already been built. Of course, that is equivalent to reparsing the sentence. Another way is to map the sentence directly into the final representation or the textual meaning of the sentence. That is the approach we have taken. DeJong (1979) and Schank et al. (1979) are two recent works that move in that direction. DeJong's system, called FRUMP, is a strong form of top down parser. It skims the text looking for those concepts in which it is interested. When it finds all of them, it ignores the remainder of the text. In analogy to key-word parsers, we may describe FRUMP as a key-concept parser. In Schank et al. (1979), words are marked in the dictionary as skippable or as having high relevance for a given script. When a relevant word is found, some questions are formulated as requests to the parser. These requests guide the parser in the understanding of the story. In our opinion, the criteria by which words are marked as skippable or relevant are not clear.

There are significant differences between our ideas and those in the aforementioned works. The least significant of them is that the internal representation selected by us has been a type of case grammar, while in those works the sentences are mapped into Schank's conceptual dependency notation. Due to the declarative nature of the texts we have studied, we have not seen a need for a deeper representation of the action verbs. The most important difference lies in the incorporation in our model of Kant's distinction between concepts as a system of rules and concepts as an abstract representation (an epistemic notion that is absent in Schank and his collobarators' work). The inclusion of this distinction in our model makes the role and the organization of the different components that form part of comprehension differ markedly from those in the aforementioned works.

4. Organization and Communication between the System Components

The organization that we have proposed appears in Fig. 1. Central to the organization are the conceptual specialists. The other components are subordinated to them.

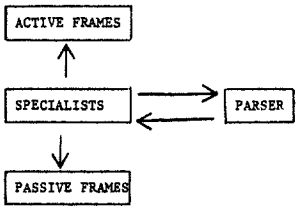

Figure 1 System Organization

The parser is essentially based on semantic markers and parses a sentence in to a case frame structure. The specialists contain contextual knowledge relevant to each specific topic. This knowledge is of inferential type. What we have termed "passive frames" contain what the system remembers of a given topic. At the beginning of the parsing process, the active frames contain nothing. At the end of the process, the meaning of the text will be recorded in them. Everything in these frames, including the name of the slots, are built from scratch by the conceptual specialists.

The communication between these elements is as follows. When a text is input to the system, the parser begins to parse the first sentence. In the parser there are mechanisms to recognize the passive frame associated with the text. Once this is done, mechanisms are set on to check if the most recent parsed conceptual constituent of the sentence is a relevant concept. This is done simply by checking if the concept belongs to the list of relevant concepts in the passive frame. If that is the case the specialist (concept) override the parser. What does this exactly mean? It does not mean that the specialist will help the parser to produce the segmentation of the sentence, in a way similar to Winograd's and Marcus' approaches in which semantic selections help the syntax component of the parser to produce the right segmentation of the sentence. In fact when the specialists take over the segmentation of the sentence stops. That is what "overriding lower linguistic processes" exactly means. The specialist has knowledge to interpret whatever structure the parser has built as well as to make sense directly of the remaining constituents in the rest of the sentence. "To interpret" and "make sense directly" means that the constituents of the sentence will be mapped directly into the active frame that the conceptual specialists are building. However this does not mean that the parser will be turned off. The parser continues functioning, not in order to continue with the segmentation of the sentence but to return the remaining of the conceptual constituents of the sentence to the specialist in control when asked by it. Thus what we have called "linguistic knowledge" has been separated from the high level "inferential knowledge" that is dependent on the subject matter of a given topic as well as from the knowledge that is recalled from a given situation. These three different cognitive aspects correspond to what we have called "parser," "conceptual specialists," and "passive frames" respectively.

5. The Parser

In this section we explain some of the components of the parser so that the reader can follow the discussion of the examples in the next section. We refer the reader to Gomez (1981) for a detailed description of these concepts. *Noun Group:* The function that parses the noun group is called DESCRIPTION. DESCR is a semantic marker used to mark all words that may form part of a noun group. An essential component of DESCRIPTION is a mechanism to identify the concept underlying the complex nominals (cf. Levi, 1978). See Finin (1980) for a recent work on complex nominals that concentrates on concept modification. This is of most importance because it is characteristic of declarative contexts that the same concept may be referred to by different complex nominals. For instance, it is not rare to find the following complex nominals in the same programming problem all of them referring to the same concept: "the previous balance," "the starting balance," "the old balance" "the balance at the beginning of the period." DESCRIPTION will return with the same token (old-bal) in all of these cases. The reader may have realized that "the balance at the beginning of the period" is not a compound noun. They are related to compound nouns. In fact many compound nouns have been formed by deletion of prepositions. We have called them prepositional phrases completing a description, and we have treated them as complex nominals. *Prepositions:* For each preposition (also for each conjunction) there is a procedure. The function of these prepositional experts (cf. Small, 1980) is to determine the meaning of the preposition. We refer to them as FOR-SP, ON-SP, AS-SP, etc.. *Descriptive Verbs:* (D-VERBS) are those used to describe. We have categorized them in four classes. There are those that describe the constituents of an object. Among them are: consist of, show, include, be given by, contain, etc.. We refer to them as CONSIST-OF D-VERBS. A second class are those used to indicate that something is representing something. Represent, indicate, mean, describe, etc.. belong to this class. We refer to them as REPRESENT D-VERBS. A third class are those that fall under the notion of appear. To this class belong appear, belong, be given on etc.. We refer to them as APPEAR D-VERBS. The fourth class are formed by those that express a spatial relation. Some of these are: follow, precede, be followed by any spatial verb. We refer to them as SPATIAL D-VERBS. *Action Verbs:* We have used different semantic features, which indicate different levels of abstraction, to tag action verbs. Thus we have used the marker SUPL to mark in the dictionary "supply", "provide", "furnish", but not "offer". From the highest level of abstraction all of them are tagged with the marker ATRANS. The procedures that parse the action verbs and the descriptive verbs are called ACTION-VERB and DESCRIPTIVE-VERB respectively.

6. Recognition of Concepts

The concepts relevant to a programming topic are grouped in a passive frame. We distinguish between those concepts which are relevant to a

specific programming task, like balance to checking-account programs, and those relevant to any kind of program, like output, input, end-of-data, etc.. The former can be only recognized when the programming topic has been identified. A concept like output will not only be activated by the word "output" or by a noun group containing that word. The verb "print" will obviously activate that concept. Any verb that has the feature REQUEST, a semantic feature associated with such verbs as "like," "want," "need," etc., will activate also the concept output. Similarly nominal concepts like card and verbal concepts like record, a semantic feature for verbs like "record," "punch," etc. are just two examples of concepts that will activate the input specialist.

The recognition of concepts is as follows: Each time that a new sentence is going to be read, a global variable RECOG is initialized to NIL. Once a nominal or verbal concept in the sentence has been parsed, the function RECOGNIZE-CONCEPT is invoked (if the value of RECOG is NIL). This function checks if the concept that has been parsed is relevant to the programming task in general or (if the topic has been identified) is relevant to the topic of the programming example. If so, RECOGNIZE-CONCEPT sets RECOG to T and passes control to the concept that takes control overriding the parser. Once a concept has been recognized, the specialist for that concept continues in control until the entire sentence has been processed. The relevant concept may be the subject or any other case of the sentence. However if the relevant concept is in a prepositional phrase that starts a sentence, the relevant concept will not take control.

The following data structures are used during parsing. A global variable, STRUCT, holds the result of the parsing. STRUCT can be considered as a STM (short term memory) for the low level linguistic processes. A BLACKBOARD (Erman and Lesser, 1975) is used for communication between the high level conceptual specialists and the low level linguistic experts. Because the information in the blackboard does not go beyond the sentential level, it may be considered as STM for the high level sources of knowledge. A global variable WORD holds the word being examined, and WORDSENSE holds the semantic features of that word.

7. Example 1

An instructor records the name and five test scores on a data card for each student. The registrar also supplies data cards containing a student name, identification number and number of courses passed.

The parser is invoked by activating SENTENCE. Because "an" has the marker DESCR, SENTENCE passes control to DECLARATIVE which handles sentences starting with a nominal phrase. (There are other functions that respectively handle sentences starting with a prepositional phrase, an adverbial clause, a command, an -ing form, and sentences introduced by "to be" (there be, will be, etc.) with the meaning of existence.) DECLARATIVE invokes DESCRIPTION. This parses "an instructor" obtaining the concept instructor. Before returning control, DESCRIPTION activates the functions RECOGNIZE-TOPIC and RECOGNIZE-CONCEPT. The former function checks in the dictionary if there is a frame associated with the concept parsed by DESCRIPTION. The frame EXAM-SCORES is associated with instructor, then the variable TOPIC is instantiated to that frame. The recognition of the frame, which may be a very hard problem, is very simple in the programming problems we have studied and normally the first guess happens to be correct. Next, RECOGNIZE-CONCEPT is invoked. Because instructor does not belong to the relevant concepts of the EXAM-SCORES frame, it returns control. Finally DESCRIPTION returns control to DECLARATIVE, along with a list containing the semantic features of instructor. DECLARATIVE, after checking that the feature TIME does not belong to those features, inserts SUBJECT before "instructor" in STRUCT. Before storing the content of WORD, "records," into STRUCT, DECLARATIVE invokes RECOGNIZE-CONCEPT to recognize the verbal concept. All verbs with the feature record, as we said above, activate the input specialist, called INPUT-SP. When INPUT-SP is activated, STRUCT looks like (SUBJ (INSTUCTOR)). As we said in the introduction, the INPUT specialist is a collection of production rules. One of those rules says:

> IF the marker RECORD belongs to WORDSENSE then activate the function ACTION-VERB and pass the following recommendations to it: 1)activate the INPUT-SUPERVISOR each time you find an object 2) if a RECIPIENT case is found then if it has the feature HUMAN, parse and ignore it. Otherwise awaken the INPUT-SUPERVISOR 3) if a WHERE case (the object where something is recorded) is found, awaken the INPUT-SUPERVISOR.

The INPUT-SUPERVISOR is a function that is controlling the input for each particular problem. ACTION-VERB parses the first object and passes it to the INPUT-SUPERVISOR. This checks if the semantic feature IGENERIC (this is a semantic feature associated with words that refer to generic information like "data," "information," etc.) does not belong to the object that has been parsed by ACTION-VERB. If that is not the case, the INPUT-SUPERVISOR, after checking in the PASSIVE-FRAME that name is normally associated with the input for EXAM-SCORES, inserts it in the CONSIST-OF slot of input. The INPUT-SUPERVISOR returns control to ACTION-VERB that parses the next object and the process explained above is repeated.

When ACTION-VERB finds the preposition "on," the routine ON-SP is activated. This, after checking that the main verb of the sentence has been parsed and that it takes a WHERE case, checks the BLACKBOARD to find out if there is a recommendation for it. Because that is the case, ON-SP tells DESCRIPTION to parse the nominal phrase "on data cards". This returns with the concept card. ON-SP activates the INPUT-SUPERVISOR with card. This routine, after checking that cards is a type of input that the solver handles, inserts "card" in

the INPUT-TYPE slot of input and returns control. What if the sentence had said "... on a notebook"? Because notebook is not a form of input, the INPUT-SUPERVISOR would have not inserted "book" into the INPUT-TYPE slot. Another alternative is to let the INPUT-SUPERVISOR insert it in the INPUT-TYPE slot and let the problem solver make sense out of it. There is an interesting tradeoff between understanding and problem solving in these contexts. The robuster the understander is, the weaker the solver may be, and vice versa. The prepositional phrase "for each student" is parsed similarly. ACTION-VERB returns control to INPUT-SP that inserts "instructor" in the SOURCE slot of input. Finally, it sets the variable QUIT to T to indicate to DECLARATIVE that the sentence has been parsed and returns control to it. DECLARATIVE after checking that the variable QUIT has the value T, returns control to SENTENCE. This resets the variables RECOG, QUIT and STRUCT to NIL and begins to examine the next sentence.

The calling sequence for the second sentence is identical to that for the first sentence except that the recognition of concepts is different. The passive frame for EXAM-SCORES does not contain anything about "registrar" nor about "supplies". DECLARATIVE has called ACTION-VERB to parse the verbal phrase. This has invoked DESCRIPTION to parse the object "data cards". STRUCT looks like: (SUBJ (REGISTRAR) ADV (ALSO) AV (SUPPLIES) OBJ). ACTION-VERB is waiting for DESCRIPTION to parse "data cards" to fill the slot of OBJ. DESCRIPTION comes with card from "data cards," and invokes RECOGNIZE-CONCEPT. The specialist INPUT-SP is connected with card and it is again activated. This time the production rule that fires says:

> If what follows in the sentence is <universal quatifier> + <D-VERB> or simply D-VERB then activate the function DESCRIPTIVE-VERB and pass it the recommendation of activating the INPUT-SUPERVISOR each time a complement is found.

The pattern <universal quantifier> + <D-VERB> appears in the antecedent of the production rule because we want the system also to understand: "data cards each containing...". The rest of the sentence is parsed in a similar way to the first sentence. The INPUT-SUPERVISOR returns control to INPUT-SP that stacks "registrar" in the source slot of input. Finally the concept input for this problem looks:

```
INPUT CONSIST-OF (NAME (SCORES CARD (5)))
                    SOURCE (INSTRUCTOR)
                 (NAME ID-NUMBER P-COURSES)
                    SOURCE (REGISTRAR)
        INPUT-TYPE (CARDS)
```

If none of the concepts of a sentence are recognized - that is the sentence has been parsed and the variable RECOG is NIL - the system prints the sentence followed by a question mark to indicate that it could not make sense of it. That will happen if we take a sentence from a problem about checking-accounts and insert it in the middle of a problem about exam scores. The INPUT-SP and the INPUT-SUPERVISOR are the same specialists. The former overrides and guides the parser when a concept is initially recognized, the latter plays the same role after the concept has been recognized. The following example illustrates how the INPUT-SUPERVISOR may furthermore override and guide the parser.

> The registrar also provides cards. Each card contains data including an identification number ...

When processing the subject of the second sentence, INPUT-SP is activated. This tells the function DESCRIPTIVE-VERB to parse starting at "contains ..." and to awaken the INPUT-SUPERVISOR when an object is parsed. The first object is "data" that has the marker IGENERIC that tells the INPUT-SUPERVISOR that "data" can not be the value for the input. The INPUT-SUPERVISOR will examine the next concept looking for a D-VERB. Because that is the case, it will ask the routine DESCRIPTIVE-VERB to parse starting at "including an identification number..."

8. Example 2

We will comment briefly on the first six sentences of the example in Fig. 2. We will name each sentence by quoting its beginning and its end. There is a specialist that has grouped the knowledge about checking-accounts. This specialist, whose name is ACCOUNT-SP, will be invoked when the parser finds a concept that belongs to the slot of relevant concepts in the passive frame. The first sentence is: "A bank would like to produce... checking accounts". The OUTPUT-SP is activated by "like". When OUTPUT-SP is activated by a verb with the feature of REQUEST, there are only two production rules that follow. One that considers that the next concept is an action verb, and another that looks for the pattern <REPORT + CONSIST D-VERB> (where "REPORT" is a semantic feature for "report," "list," etc.). In this case, the first rule is fired. Then ACTION-VERB is activated with the recommendation of invoking the OUTPUT-SUPERVISOR each time that an object is parsed. ACTION-VERB awakens the OUTPUT-SUPERVISOR with (RECORDS ABOUT (TRANSACTION)). Because "record" has the feature IGENERIC the OUTPUT-SUPERVISOR tries to redirect the parser by looking for a CONSIST D-VERB. Because the next concept is not a D-VERB, OUTPUT-SUPERVISOR sets RECOG to NIL and returns control to ACTION-VERB. This parses the adverbial phrase introduced by "during" and the prepositional phrase introduced by "with". ACTION-VERB parses the entire sentence without recognizing any relevant concept, except the identification of the frame that was done while processing "a bank".

The second sentence "For each account the bank wants ... balance." is parsed in the following way. Although "account" belongs to slot of relevant concepts for this problem, it is skipped because it is in a prepositional phrase that starts a sentence. The OUTPUT-SP is activated by a

REQUEST type verb, "want". STRUCT looks like: (RECIPIENT (ACCOUNT UQ (EACH)) SUBJECT (BANK)). The production rule whose antecedent is <RECORD + CONSIST D-VERB> is fired. The DESCRIPTIVE-VERB function is asked to parse starting in "showing," and activate the OUTPUT-SUPERVISOR each time an object is parsed. The OUTPUT-SUPERVISOR inserts all objects in the CONSIST-OF slot of output, and returns control to the OUTPUT-SP that inserts the RECIPIENT, "account," in the CONSIST-OF slot of output and returns control.

The next sentence is "The accounts and transactions ... as follows:" DECLARATIVE asks DESCRIPTION to parse the subject. Because account belongs to the relevant concepts of the passive frame, the ACCOUNT-SP specialist is invoked. There is nothing in STRUCT. When a topic specialist is invoked and the next word is a boolean conjunction, the specialist asks DESCRIPTION to get the next concept for it. If the concept does not belong to the list of relevant concepts, the specialist sets RECOG to NIL and returns control. Otherwise it continues examining the sentence. Because transaction belongs to the slot of relevant concepts of the passive frame, ACCOUNT-SP continues in control. ACCOUNT-SP finds "for" and asks DESCRIPTION to parse the nominal phrase. ACCOUNT-SP ignores anything that has the marker HUMAN or TIME. Finally ACCOUNT-SP finds the verb, an APPEAR D-VERB and invokes the DESCRIPTIVE-VERB routine with the recommendation of invoking the ACCOUNT-SUPERVISOR each time a complement is found. The ACCOUNT-SUPERVISOR is awakened with card. This inserts "card" in the INPUT-TYPE slot of account and transaction and returns control to the DESCRIPTIVE-VERB routine. AS-SP (the routine for "as") is invoked next. This, after finding "follows" followed by ":," indicate to DESCRIPTIVE-VERB that the sentence has been parsed. ACCOUNT-SP returns control to DECLARATIVE and this, after checking that QUIT has the value T, returns control to SENTENCE.

The next sentence is: "First will be a sequence of cards ... accounts." The INPUT-SP specialist is invoked. STRUCT looks like: (ADV (FIRST) EXIST). "Sequence of cards" gives the concept card activating the INPUT-SP specialist. The next concept is a REPRESENT D-VERB. INPUT-SP activates the DESCRIPTIVE-VERB routine and asks it to activate the INPUT-SUPERVISOR each time an object is found. The INPUT-SUPERVISOR checks if the object belongs to the relevant concepts for checking accounts. If not, the ACCOUNT-SUPERVISOR will complain. That will be the case if the sentence is: "First will be a sequence of cards describing the students". Assume that the above sentence says: "First will be a sequence of cards consisting of an account number and the old balance." In that case, the INPUT-SP will activate also the INPUT-SUPERVISOR but because the verbal concept is a CONSIST D-VERB, the INPUT-SUPERVISOR will stack the complements in the slot for INPUT. Thus, what the supervisor specialists do depend on the verbal concept and what is coming after.

The next sentence is: "Each account is described by ..., in dollars and cents." Again, the ACCOUNT-SP is activated. The next concept is a CONSIST D-VERB. ACCOUNT-SP assumes that it is the input for accounts and activates the DESCRIPTIVE-VERB function, and passes to it the recommendation of activating the INPUT-SUPERVISOR each time an object is parsed. The INPUT-SUPERVISOR is awakened with (NUMBERS CARDINAL (2)). Because number is not an individual concept (like, say, 0 is) the INPUT-SUPERVISOR reexamines the sentence and finds ":," it then again asks to DESCRIPTIVE-VERB to parse starting at "the account number...". The INPUT-SUPERVISOR stacks the complements in the input slot of the concept that is being described: account.

The next sentence is: "The last account is followed by ... to indicate the end of the list." The ACCOUNT-SP is invoked again. The following production rule is fired: If the ordinal "last" is modifying "account" and the next concept is a SPATIAL D-VERB then activate the END-OF-DATA specialist. This assumes control and asks DESCRIPTIVE-VERB to parse starting at "followed by" with the usual recommendation of awakening the END-OF-DATA supervisor when a complement is found, and the recommendation of ignoring a PURPOSE clause if the concept is end-of-list or end-of-account. The END-OF-DATA is awakened with "dummy-account". Because "dummy-account" is not an individual concept, the END-OF-DATA supervisor reexamines the sentence expecting that the next concept is a CONSIST D-VERB. It finds it, and redirects the parser by asking the DESCRIPTIVE-VERB to parse starting in "consisting of two zero values." The END-OF-DATA is awakened with "(ZERO CARD (2))". Because this time the object is an individual concept, the END-OF-DATA supervisor inserts it into the END-OF-DATA slot of the concept being described: account.

9. Conclusion

LLULL was running in the Dec 20/20 under UCI Lisp in the Department of Computer Science of the Ohio State University. It has been able to understand ten programming problems taken verbatim from text books. A representative example can be found in Fig. 2. After the necessary modifications, the system is presently running in a VAX11/780 under Franz Lisp. We are now in the planning stage of extensively experimenting with the system. We predict that the organization that we have proposed will make relatively simple to add new problem areas. Assume that we want LLULL to understand programming problems about roman numerals, say. We are going to find uses of verbs, prepositions, etc. that our parser will not be able to handle. We will integrate those uses in the parser. On top of that we will build some conceptual specialists that will have inferential knowledge about roman numerals, and a thematic frame that will hold structural knowledge about roman numerals. We are presently following this scheme in the extension of LLULL. In the next few months we expect to fully evaluate our ideas.

10. A Computer Run

The example below has been taken verbatim from Conway and Gries (1975). Some notes about the output for this problem are in order. 1) "SPEC" is a semantic feature that stands for specification. If it follows a concept, it means that the concept is being further specified or described. The semantic feature "SPEC" is followed by a descriptive verb or adjective, and finally it comes the complement of the specification in parentheses. In the only instance in which the descriptive predicate does not follow the word SPEC is in expressions like "the old balance in dollars and cents". Those expressions have been treated as a special construction. 2) All direct objects connected by the conjunction "or" appear enclosed in parentheses. 3) "REPRESENT" is a semantic marker and stands for a REPRESENT D-VERB. 4) Finally "(ZERO CARD (3))" means three zeros.

(A BANK WOULD LIKE TO PRODUCE RECORDS OF THE TRANSACTIONS DURING AN ACCOUNTING PERIOD IN CONNECTION WITH THEIR CHECKING ACCOUNTS. FOR EACH ACCOUNT THE BANK WANTS A LIST SHOWING THE BALANCE AT THE BEGINNING OF THE PERIOD, THE NUMBER OF DEPOSITS AND WITHDRAWALS, AND THE FINAL BALANCE. THE ACCOUNTS AND TRANSACTIONS FOR AN ACCOUNTING PERIOD WILL BE GIVEN ON PUNCHED CARDS AS FOLLOWS: FIRST WILL BE A SEQUENCE OF CARDS DESCRIBING THE ACCOUNTS. EACH ACCOUNT IS DESCRIBED BY TWO NUMBERS: THE ACCOUNT NUMBER (GREATER THAN 0), AND THE ACCOUNT BALANCE AT THE BEGINNING OF THE PERIOD, IN DOLLARS AND CENTS. THE LAST ACCOUNT IS FOLLOWED BY A DUMMY ACCOUNT CONSISTING OF TWO ZERO VALUES TO INDICATE THE END OF THE LIST. THERE WILL BE AT MOST 200 ACCOUNTS. FOLLOWING THE ACCOUNTS ARE THE TRANSACTIONS. EACH TRANSACTION IS GIVEN BY THREE NUMBERS: THE ACCOUNT NUMBER, A 1 OR -1 (INDICATING A DEPOSIT OR WITHDRAWAL, RESPECTIVELY), AND THE TRANSACTION AMOUNT, IN DOLLARS AND CENTS. THE LAST REAL TRANSACTION IS FOLLOWED BY A DUMMY TRANSACTION CONSISTING OF THREE ZERO VALUES.)

Figure 2 A Programming Problem

```
OUTPUT CONSIST-OF (ACCOUNT OLD-BAL DEPOSITS
   WITHDRAWALS FINAL-BAL)

ACCOUNT INPUT (ACCOUNT-NUMBER SPEC GREATER (0)
   OLD-BAL SPEC (DOLLAR-CENT))
   INPUT-TYPE (CARDS)
   END-OF-DATA ((ZERO CARD (2)))
   NUMBER-OF-ACCOUNTS (200)

TRANSACTION INPUT (ACCOUNT-NUMBER (1 OR -1)
   REPRESENT
   (DEPOSIT OR WITHDRAWAL)
    TRANS-AMOUNT SPEC (DOLLAR-CENT))
   INPUT-TYPE (CARDS)
   END-OF-DATA ((ZERO CARD (3)))
```

Figure 3 System Output for Problem in Figure 2

ACKNOWLEDGEMENTS

This research was supported by the Air Force Office of Scientific Research under contract F49620-79-0152, and was done in part while the author was a member of the AI group at the Ohio State University.

I would like to thank Amar Mukhopadhyay for reading and providing constructive comments on drafts of this paper, and Mrs. Robin Cone for her wonderful work in typing it.

REFERENCES

Bruce, B. and Newman D. Interacting Plans. Cognitive Science. v. 2, 1978.

Cohen, P. and Perrault R. Elements of a Plan-Based Theory of Speech Acts. Cognitive Science, v. 3, n. 3, 1979.

Conway, R. and Gries, D. An Introduction to Programming. Winthrop Publishers, Inc., Massachusetts, 1975.

DeJong, G. Prediction and Substantiation: A New Approach to Natural Language Processing. Cognitive Science, v. 3, n. 3, 1979.

Erman, D. and Lesser V. A Multi-Level Organization for Problem-Solving Using Many Diverse Cooperating Sources of Knowledge. IJCAI-75, University Microfilms International, PO BOX 1467, Ann Arbor, Michigan 48106, 1975.

Finin, T. The Semantic Interpretation of Compound Nominals. Report T-96, Dept. of Computer Science, University of Illinois, 1980.

Gomez, F. Understanding Programming Problems Stated in Natural Language. OSU-CISR-TR-81, Dept. of Computer Science, The Ohio State University, 1981.

Grosz, B. The Representation and Use of Focus in Dialogue Understanding. SRI Technical Note 151, Menlo Park, Ca., 1977.

Hobbs, J. and Evans D. Conversation as Planned Behavior. Cognitive Science. v.4, no. 4, 1980.

Levi, J. N. The Syntax and Semantics of Complex Nominals. Academic Press, 1978.

Marcus, M. A Theory of Syntactic Recognition for Natural Language. MIT Press, 1979.

Ortony, A. Remembering, Understanding, and Representation. Cognitive Science, v. 2, n. 1, 1978.

Perrault, R. and Allen F. A Plan-Based Analysis of Indirect Speech Acts. American Journal of Computational Linguistics, v. 6, n. 3, 1980.

Riesbeck, C. K. Conceptual Analysis. In R. Schank (Ed.), Conceptual Information Processing. N. York, Elvesier-North Holland, 1975.

Schank, R. and Abelson, R. Scripts, Plans, Goals, and Understanding. Laurence Erlbaum Associates, Hillsdale N. J., 1977.

Schank, R. C., Lebowitz, M., and Lawrence, B. Parsing Directly in Knowledge Structures. in IJCAI-79, Computer Science Department, Stanford University, Stanford, CA 94305.

Small, S. Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding. Tech. Report 954, Dept. of Computer Science, University of Maryland, 1980.

Wilks, Y. An Artificial Intelligence Approach to Machine Translation. In Schank and Colby (eds.) Computer Models of Thought and Language. San Francisco, W. H. Freeman and Co., San Francisco, 1973.

Wilensky, R. Understanding Goal-Based Stories. Dept. of Computer Science, Yale University. Tech. Report 140, 1978.

Winograd, T. Understanding Natural Language. N. York, Academic Press, 1972.