The Discourse System Project

James Allen, Stephane Guez, Louis Hoebel,
Elizabeth Hinkelman, Keri Jackson, Alice Kyburg and David Traum

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 317

November 1989

Abstract

There has been a significant amount of work in the last decade on the processing of discourse, be it for modeling two-person extended dialogues, story and text understanding, or extended question answering systems. While there has been important progress in areas such as the use of world knowledge in language interpretation, the use of plan-based models of language (eg. speech act planning), models of reference and focus, and in models of discourse structure itself, work in each area has not been related to work in the other areas. No one has determined how each individual processing technique can be combined with the others to form a fully functional discourse system. Some suggestions on the organization of discourse have arisen recently [Grosz and Sidner, 1986; Litman and Allen, 1987; Allen, 1987], and while showing promise, have not yet been explored in enough detail for an actual application. This report describes an architecture for discourse systems that allows the integration of many different processing modules. In addition, it describes an initial set of modules that have been implemented and tested using the architecture.

This work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13341-5700 and the Air Force Office of Scientific Research, Bolling AFB, DC 20332 under contract no. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC). This work was also supported by ONR/DARPA research contract no. N00014-82-K-0193 and ONR research contract no. N00014-80-C-0197.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 317

2. GOVT ACCESSION NO.:

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): The Discourse System Project

5. TYPE OF REPORT & PERIOD COVERED: technical report

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): J. Allen, S. Guez, L. Hoebel, E. Hinkelman, K. Jackson, A. Kyburg, and D. Traum

8. CONTRACT OR GRANT NUMBER(s): N00014-82-K-0193

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Dept., 734 Comp. Studies Bldg.
University of Rochester
Rochester, NY 14627

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:

11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

12. REPORT DATE: November 1989

13. NUMBER OF PAGES: 82

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research
Information Systems
Arlington, VA 22217

15. SECURITY CLASS. (of this report): unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

discourse structure; blackboard; text understanding; modular processing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

20. ABSTRACT

There has been significant work in the last decade on the processing of discourse for modeling two-person extended dialogues, story and text understanding, and extended question answering systems. While there has been important progress in the use of world knowledge in language interpretation, the use of plan-based models of language (e.g., speech act planning), models of reference and focus, and models of discourse structure itself, work in each area has not been related to work in the other areas. No one has determined how individual processing techniques can be combined to form a fully functional discourse system. Some suggestions on the organization of discourse have arisen recently, and while showing promise, have not been explored in enough detail for an actual application. This report describes an architecture for discourse systems that allows the integration of many different processing modules. It also describes an initial set of modules that have been implemented and tested using the architecture.

Contents

List of Figures

Chapter 1

INTRODUCTION: The Architecture of Discourse Systems

Work in discourse includes the study of all forms of natural language that involves multiple sentences (or utterances). This includes many forms:

- two person, or man-machine, conversations, as needed in an expert- apprentice training situation or in a man-machine control task;
- narrative, as in story understanding and message understanding;
- extended question answering, as with a complex information system where the information desired can only be extracted after a series of questions.

The problems studied involve how sentences relate to one another, and how they relate to the context of the discourse. As such, most work in discourse addresses issues quite different than the parsing and semantic interpretation of individual sentences.

It is important to realize that once a system exists to analyze single sentences, such as existing question-answering interfaces to data bases, it cannot be made into a discourse system by simply giving it more sentences to parse. Systems that handle sentences in isolation do not have to face a wide range of hard technical problems that arise in discourse. They have very limited facilities for determining the referents of noun phrases, they have no facilities for identifying the appropriate semantic connections between sentences, and they have no need to face the problems of generating an interactive dialogue to handle exchange of information in complex environments.

This report describes a system architecture for discourse systems. Before looking at the details, however, consider some of the work that has been done in discourse in the last decade. Rather than concentrating on building complete systems, most of the important results have arisen from projects that have studied a particular aspect of discourse in detail. The following paragraphs will discuss, in turn, the use of world knowledge in language interpretation, the use of plan-based models of language (speech act planning and recognition), models of reference and local focus, and models of discourse structure.

The importance of using world knowledge to discover the connections between sentences has been stressed in the work of Schank and his students [Schank, 1975; Wilensky, 1983; Cullingford, 1981]. The development of script and plan-based models were important in drawing connections between sentences in a story, for identifying the appropriate referents of noun phrases, and for filling in "understood" information needed to answer questions

appropriately. [Grosz, 1977] used a plan-based representation to show that the structure of task-oriented dialogues followed the structure of the task being discussed, and developed a theory of global focus that assigned priority to the more salient parts of the plan at any given time in the conversation. This allowed her to develop a theory of reference that explained some problematic uses of definite noun phrases and pronouns. In particular, she showed that maintaining a simple "history list" of all objects mentioned so far in the discourse could not solve many problems in analyzing pronouns and definite descriptions.

Another important aspect of world knowledge is our knowledge of the way language is used. This was developed into a plan-based theory of speech acts by Cohen, Perrault and Allen [Cohen and Perrault, 1979; Allen and Perrault, 1980]. This work provided the missing link between the domain of discourse (say task planning) and language itself. With speech acts, language became just another activity that an agent could plan, and a crucial part of understanding what another agent said was recognizing their plan that led to the utterance. This model lead to an analysis of indirect speech acts that identified the appropriate intention using the knowledge of the context, and knowledge of the speakers and hearers beliefs and intentions. As an example of this problem, notice that the simple sentence *Do you know the time* can be intended as a literal yes/no question, a request for the time, an offer to tell the time, or a reminder that it is getting late. The sentence remains the same, but its interpretation changes as the context changes.

There has been significant work looking at the problem of reference, both for definite descriptions and for pronouns. [Appelt, 1985] extended the speech act planning formalism to produce a system that could plan the content of its descriptions so that they would successfully refer. In task oriented dialogues, Grosz, [1977] showed how the interpretation of noun phrases depended on knowledge of the task being discussed. Sidner [1983], Grosz, et al[1983] and Reichman [1978] studied the use of pronominal reference and developed theories of focus (or local focus) to explain their use. These theories depend on a hypothesis that each sentence has a particular focus (or center), which is the most likely object to be referred to by a pronoun. In fact, Grosz et al [1983] claim that the center must be pronominalized if other objects are also pronominalized and propose that there are only a few limited ways in which focus can naturally shift.

The last area is work on discourse structure. In this, discourse is divided into segments concerning a common topic. These segments are hierarchically organized and naturally represented by a tree structure. As a result, models are proposed that suggest a grammar for discourse (with sentences as the "terminal" symbols), as well as many models based on segmentation and the relation between segments. Important information on discourse structure is signalled by the use of *cue phrases* (or *clue words*) that serve as explicit signals of segment boundaries. For instance, the phrase "by the way" signals a digression is about to begin, while the phrase "for instance", as used in this sentence, is used to signal a subsegment describing an example.

While progress has been made on each of these areas, there is a fundamental problem that is preventing the construction of a full discourse system, namely, that no one knows how to fit all the pieces together. The best attempt at this integration is in a recent paper by Grosz and Sidner [1986]. Lets examine this briefly. They propose that discourse processing consists of three major components: the segmentation structure, the attentional state and the intentional state.

The segmentation structure represents a structural theory of discourse, as described above - discourse consists of sentences that are grouped into segments, which in turn may be parts of other segments. Grosz and Sidner consider the cue phrases to be a major determiner of segmentation, and mention that other linguistic structure, such as tense and modality, are important as well. Closely linked to the segmentation structure is the attentional state. This is a generalization of the reference and focus of attention work described above. There is a direct 1-1 correspondence to the discourse segmentation and focus states that make up the attentional state. In particular, each segment has a focus state, and the attentional

state at any time is a stack of focus states, each state on the stack representing a not yet completed segment.

The last component is the intentional state. This captures the linguistic purposes of each discourse segment. For example, if one is telling a story, one may have many different goals - telling the story clearly, impressing the listener, keeping the listener occupied, etc. It is only those goals that relate to the actual telling of the story that are discourse purposes. Each focus space has a particular discourse purpose and Grosz and Sidner examine two constraints between the discourse purpose organization (into goals and subgoals), and the attentional stack.

As a starting point, this is a promising organization. What Grosz and Sidner do not say is how the reasoning about the general topic of conversation fits in, or propose in detail how the various techniques developed previously can be integrated into the organization.

The system described here provides a general framework for maintaining information about discourse segments in a blackboard style [Lesser and Erman, 1977] architecture. In particular, each segment is represented as a partition of the blackboard that maintains the relevant information necessary for the different modules of the discourse system (e.g. tense analysis, reference, plan-based analysis, and so on). A uniform hypothesis structure allows modules to co-ordinate their activities by maintaining multiple hypotheses (both within and across segments) that compete based on the rating evaluations returned by the individual modules.

Segments may be organized hierarchically by explicitly identifying relationships between segments. Thus, part of the information in a segment is what segment contains it (if any), and what segments does it contain (again, if any). Segments do not have to be related at all, however, for there are no constraints on the overall organization of segments on the blackboard.

The idea is that each module is responsible for two types of processing. The first is intra-segment reasoning, on which a new sentence must be incorporated into an existing segment on the blackboard. The second is inter-segment reasoning, where a new discourse segment representation is created, possibly using information from an existing segment on the stack. Each module communicates to the other modules only via the blackboard structure.

Given a new sentence in a discourse, there are a limited range of possible analyses that each module must consider. The sentence could be a continuation of one of the existing segments; it could be a beginning of a new subsegment of an existing segment on the stack, or it could be the beginning of a segment unrelated to those on the blackboard (eg. an interruption).

This model can be constrained to capture the Grosz and Sidner stack-based model, and the constraints they suggest between segmentation and the attentional state, and the intentional state and the attentional state can be embedded in this system. Likewise, different organizational constraints can also be explored. The goal is to construct a system in which such system organizations can be easily evaluated.

In this report we demonstrate how different processing techniques can be recast into this framework. In particular, concrete descriptions are provided for modules that perform lexical, syntactic and semantic analysis, tense analysis, reference analysis, and speech act analysis. These are applied to a concrete application involving two conversants interacting to extract information from a library database system. Transcripts of actual terminal-to-terminal conversations in this domain are provided in an appendix.

Chapter 2

GENERAL ARCHITECTURE OF THE DISCOURSE SYSTEM

The Discourse System is designed for the interpretation of dialogues. The system consists of several modules, each one devoted to a certain level of analysis and specialized to perform a particular task. Modules communicate through a central blackboard which contains public information accessible by all the modules. The overall control of the system is event driven and is moderated by a monitor. When new information is posted to the blackboard, the system records all the modules that may be interested in this information. The system operates in a cycle of selecting a module, to invoke with certain information recorded earlier, calling the module and then processing all results in order to record future module calls.

One sentence is analyzed at a time. Information in the blackboard is organized in such a way that different hypotheses about the analysis may be generated and evaluated concurrently. New hypotheses are created each time one or several modules produce different values for a same object.

At each step, the blackboard monitor chooses the hypothesis to be further analyzed. The analysis is pursued until no more information is produced by any module or until a complete and coherent interpretation of the sentence is found, which accounts for the current discourse in the most plausible way. The actual stopping criteria must be defined as part of any particular implementation. The results of the chosen hypothesis are used to update the description of the analysis of the current discourse in order to be able to proceed to the next sentence.

So far, the following modules have been considered:

- the LEXICAL module: takes words as soon as they are read by the system, and returns a lexical analysis, including the root, the category of the word, and when appropriate, the gender, the number, the tense, the aspect and the semantic type.
- the SYNTACTIC module: parses the input sentences and returns syntactic structures.
- the SEMANTIC module: accepts syntactic structures and returns the logical form for these structures.
- the REFERENCE module: interprets semantic structures, solves references, returns frame structures using the knowledge representation system and asserts relations between objects and referents.
- the SPEECH ACTS module: analyzes the speech acts in each sentence.

- the TENSE module: analyzes the tense and aspect of the sentences.
- the DISCOURSE STRUCTURE module: maintains structural constraints between segments of discourse, and analyzes the cue phrases in the current sentence to identify transitions, etc.
- the PLAN TRACKING module: performs a plan recognition to find the plans involved in the current discourse.

Chapter 3

AN EXAMPLE

For motivation, here is an example of a dialogue between a library data base operator and a user looking for bibliographical references in order to write a paper. This example is taken from a collection of real dialogues obtained experimentally. The complete experiment and its results are described in Appendix A.

```
O: This is the data base operator. (1)
O: What can I do for you? -o- (2)
U: I'm interested in getting some information about the stock
   market crash of 1929? (3)
U: specifically I want to know about causes and safeguards.-o- (4)
O: Okay,(5)
O: lets see. (6)
O: I'm checking under **stocks**.-o-  (6)
U: Could you look under **crash of '29** or something to find sources
   that discuss the crash?-o- (7)
O: Okay, lets see. (8)
O: I'm checking under stocks. -o- (9)
U: Sure. (10)
U: lets try that. (11)
O: There's nothing listed under crash of '29, or even crash. (12)
O: Lets see what's under stocks -U.s. (13)
O: That doesn't look good either. -o- (14)
U: How about under the **Stock market?** -o- (15)
O: No! (16)  This is surprising. (17)
O: Do you think **Economy -U.s.** might be good? -o- (18)
U: Hmmm, Its worth a shot. (19)
U: If you can find anything related to the Economy of the twenties
   that would be good. (20)
U: Also you might look at **banking** ?-o- (21)
O: There we go. (22)
O: I have economics 1900 - 1934. (23)
O: I'm now looking at some specifics. (24)
U: Ahhh, Could you look under **FSLIC** (Fedral Savings Loan Insurance ?) (25)
U: that was established to prevent further crashes... (26)
U: that might be useful. -o- (27)
O: First,(28)
O: let me give you a few books that look useful. (29)
```

```
O: First, (30)
O: I have Economic Basis of Tax Reform, by H.G. Brown, (31)
O: I also have something which may or may not be useful:
   Industrial Crisis, its causes and...by Edie. -o- (32)
U: I definitely would like to see the first one. (33)
U: The second is at least worth exploring. (34)
U: Anything on the FSLIC? -o- (35)
O: Okay, (36) that sounds good. (37)
O: One moment. (38) ... Yes, (39)
O: we have a book called Proposals for reform
   of the Deposit Insurance System. (40)
O: The author must be American Enterprise Institute. (41)
O: just in case, here is the call number: (42)
O: KF 1023.p123 -o- (43)
U: Is that Rhees or Carlson? (44)
O: That is in Rhees. -o- (45)
U: Also could I get the call number & locations of the other books?-o- (46)
O: Well, I'll have to look them up again. (47)
O: Do you have time to wait a few minutes.-o- (48)
U: Sure-o- (49)
O: O.k. (50)
O: why don't you give me the titles it will go more quickly. (51)
O: don't have them written down. -o- (52)
U: Yes, (53)
U:     the first is economics 1900-1934, (54)
U:     the second is industrial crisis-o- (55)
O: Ok, The call Number is HJ 2305/b23 (56)
O: It is in Rhees ... (57)
O: The second is HJ 463.h143. (58)
O: That is also in RHEES.-o- (59)
U: Ok, (60) that should get me going. (61)
U: Thanks for the help.-o- (62)
O: You're welcome. (63)
O: Good Luck on your project.-o- (64)
```

This dialogue provides a lot of examples of what happens in a real dialogue and that we should try to deal with in a system for analyzing such dialogues. It emphasizes the role of clue words on the discourse structure (see (28) for instance). It provides an example of a topic first suggested, then rejected (or put on hold), then resumed later in the dialogue (see (25-26-27-28) then (35)).

To give an idea of how the system should operate, consider a hypothetical trace of the system on the first few sentences of this dialog. Chapter 7 gives an actual trace of the current system on sentence (3).

- We begin with sentence (3).

```
U: I'm interested in getting some information about the stock
   market crash of 1929. (3)
```

- The sentence is sent as a string of words to the lexical analysis module.

- The lexical module posts lexical data about each word of sentence to the Chart. These postings trigger calls to syntactic module to be scheduled.

```
(LEX W1 (WORD "I") (ROOT W1 "I") (CAT PRO) (NUM 1S))
(LEX W2 (WORD "am") (ROOT "be") (CAT AUX) (NUM 1S) (TENSE PRES))
...
(ENTRY EW1 (SYN EW1))
...
```

- The syntactic analysis begins. Resulting syntactic structures are posted to the Chart and trigger calls to the semantic module to be scheduled. Calls to the syntactic and semantic modules are interleaved.

```
Executing Call (SYN-PARSE E6 HYP4044) ...

(NP NP2 (DET EW6)     ;; "some")
        (HEAD EW7)    ;; "information"
         ... )

Executing Call (SEM-INTERPRET E1 HYP4404) ...

(SEM E1 SEMS1)

(BE-INTERESTED SEMS1 (EXPERIENCER SEMNP1)
                     (THEME SEMS2) ... )
```

- The syntactic and semantic analyses are completed. Let us assume that only one interpretation is found at this level for sentence (3). The relevant syntactic and semantic information is duplicated and re-posted to the main blackboard in each open segment. Currently there is only one open segment.

```
RePosting Object (INFORMATION SEMNP2 (THEME ((SEMNP3 10 10)))
                                     (SPEC ((SOME 10 10))))
from Context HYP4404 to NEW Context HYP4711

RePosting Object (ENTRY E1 (SEM ((SEMS1 10 10)))
                           (SYN ((S1 10 10))))
from Context HYP4404 to NEW Context HYP4711
...
```

- Re-posting of facts to the segments triggers calls to other modules: reference module, speech act module, tense and aspect module, discourse structure module, planning module.

```
Scheduling Call (REFERENCE-ANALYSIS E1 HYP4711)
...
Scheduling Call (SSA E1 HYP4711) ;; Speech Acts Analysis
...
```

- Module calls are executed, resulting in new facts to be posted to the blackboard and new calls to be scheduled and executed. As modules may post partial results, or multiple values for certain slots, a tree of hypotheses is built. Facts and hypotheses are rated. The most highly rated hypothesis is always chosen as the next one to be further explored.

```
Executing Call (REFERENCE-ANALYSIS E4 HYP4711)
               ;; analysis of "getting some information..."

;; two possible referents for "getting some information... "
;; are found by the reference module

     (REF S2 ((REF-S2-ONE 9 9)      ;; highly rated referent
              (REF-S2-TWO 4 8)))    ;; poorly rated referent

;; the current hypothesis is split into two hypotheses
;; one for each interpretation

Splitting Hypothesis HYP4711
Creating NEW Hypothesis HYP4712 in Segment SEG4402 as daughter of HYP4711

(REF S2 ((REF-S2-ONE 9 9)))  ;; posted in HYP4712

Creating NEW Hypothesis HYP4713 in Segment SEG4402 as daughter of HYP4711
...

(SA E1 ... )  ;; posting more results from different modules
...

;; Hypotheses Structure for Segment SEG4402
;;
;;                /-----  HYP4712 ...                 -----------
;;    HYP4711 --<                              /----- | HYP4750 |
;;                \-----  HYP4713 -------<             -----------
;;                                         \
;;                                          \---- HYP4751 ...
```

- When no more module calls are scheduled or only correspond to very low rated hypotheses, then the most likely hypothesis is chosen and declared as the interpretation of the current sentence. The data associated with this interpretation is copied as permanent data in the current segment in order to be later accessed when processing the next sentences.

```
Selecting the Hypothesis HYP4750  (rating 125)
Transfering slot values to the permanent hypothesis PERM-HYP4403
...
```

- The blackboard is cleaned up and the system is ready to process the next sentence.

```
U: Specifically I want to know about causes and safeguards. (4)
```

- The sentence is sent to the lexical modules, then syntactic and semantic analyses are completed and the results are posted to the chart.
- The data relevant to the chosen syntactic/semantic interpretation is re-posted to the only active segment on the blackboard, and higher level processing are performed. A tree of hypotheses is built and one hypothesis is eventually picked up as the accepted interpretation for sentence (4).

- Sentence (4) is attached to the same segment as sentence (3). The facts associated with the interpretation of sentence (4) are copied as permanent data to this hypothesis.
- The blackboard is cleaned up and the processing of the following sentences may go on.

```
O: Okay,(5)
O: lets see. I'm checking under stocks.  (6)

U: Could you look under crash of '29 or something to find sources
   that discuss the crash? (7)

O: Okay, lets see. (8)
```

- At certain points, new segments are activated and the discourse structure is progressively built.

```
O: There's nothing listed under crash of '29, or even crash.
```

- Some segments may also be deactivated, when a topic ends or when clue words indicate that the end of a part of the conversation has been reached.
- The actual conditions under which a new segment is activated or an active segment is deactivated remain to be explored. The incremental building of the discourse structure still raises difficult theoretical questions which are the subjects of current research efforts. The Discourse System provides the framework for such studies.

Chapter 4

THE MANAGEMENT OF THE BLACKBOARD

4.1 Structure of the Blackboard

The blackboard (BB) is the central data structure where all the information to be exchanged between the modules is posted. It is divided into several parts: one part, the CHART, is reserved to the use of the lexical, syntactic and semantic modules; the other parts correspond to different segments of the current discourse.

Each of these parts of the BB is structured as a tree of hypotheses based on a context hierarchy mechanism as in CONNIVER [Sussman and McDermott, 1972]. Each hypothesis represents a particular state of the analysis, which is the result of all the choices that have been previously made in the course of this analysis. The information available in a given hypothesis is that stored with the particular hypothesis and all the information available through inheritance from its supercontexts. In its broader sense, a hypothesis consists of the information in the entire set of nodes included in the path from the current node to the root node.

4.1.1 The CHART

The part of the BB referred to as the CHART is reserved for the use of the lexical, syntactic and semantic modules and is organized as a chart as used in chart parsers (e.g. [Kaplan, 1973], also see Winograd [1983] or Allen [1987]. The chart is currently accessed by these three modules only, but there may be exceptions however when other modules (e.g. the cue phrases analyzer or the tense analyzer) need to check specific information at the lexical or syntactic level. The data produced within the CHART that may be of public interest for other modules is duplicated and re-posted to each of the segments of the BB. This mainly consists of the semantic structures produced by the semantic analysis.

Currently, the CHART space is used to create plausible syntactic/semantic interleaved parses of the input before other modules have unrestricted and asynchronous access to the data. The information posted from the CHART to the BB segments becomes the root hypothesis for each open segment. This organization provides some island of certainty about the lower levels of processing. The growth of hypotheses is thereby restricted. The early restriction of the solution space is a serious requirement for any potentially useful system. Hypothesis growth is further constrained in that only the most promising hypotheses are selected for further processing and evaluation. This leads to a system that can possibly determine solutions very directly from relatively unambiguous inputs.

4.1.2 Segments

Except for the CHART just mentioned, the BB is divided into several parts or "segments", each part corresponding to a segment of the current analyzed discourse. For a good introduction of segmentation in discourse, see [Grosz and Sidner, 1986].

A sentence may be attached to any current segment, provided that this interpretation is consistent with all the constraints expressed by the different modules. This view departs from the traditional view where the discourse structure is more strictly constrained, as in the stack model for the attachment of sentences to segments. The present organization allows a more flexible discourse structure which can, in principle, account for many cases the stack model can not describe.

Each segment of the BB is associated with a tree of hypotheses. This tree of hypothesis represents the evolution of the interpretation of the current sentence in the course of the analyses performed by each relevant module.

Besides the tree of hypotheses, each segment contains information relevant to that particular segment and whose value goes beyond the analysis of the current sentence. This general information includes for example the focus of the segment, the tenses used in the segment, and the history list (the list of all the conceptual structures mentioned within the segment). This information is updated after the analysis of each sentence.

Each analyzed sentence has to be attached to a segment. In fact, during the analysis, attachment to each of all the active segments is considered. For each sentence, there is at least one hypothesis in each segment which may considered for further analysis.

Though the BB is divided into several parts corresponding to different segments, every hypothesis competes with every other hypothesis independently of the segment that contains them. In the same way, the interpretation finally chosen as the most plausible one for the current sentences is chosen from all the hypotheses of the BB. In other words, all possible attachments of a sentence to a segment are considered concurrently, and only one attachment is selected at the end. At this point, no ambiguity is maintained beyond the current interpretation of one sentence, and no backtrack on this choice is allowed. This aspect of the organization of the system is clearly a simplification and might be revised in the course of the development of the project.

In the previous paragraphs, we used the phrase "active segments" to refer to all the segments which are a priori valid candidates to have a new sentence attached to them. In some cases, for example the presence of a strong cue phrase, a segment may be closed. The number of active segments is not predetermined. At the beginning of the analysis of a new discourse, there is only one segment. New segments are created to account for the structure of the discourse, when new sentences are interpreted.

4.1.3 Trees of Hypotheses

As already mentioned, each segment of the BB is associated with a tree of hypotheses. If no ambiguities have appeared yet as the result of the analysis, this tree consists of only one hypothesis. Hypotheses are split only when one or several modules produce different results for the same object.

There may be several trees attached to each segment, if the syntactic and semantic analyses have already resulted in different hypotheses due to ambiguities. However, all these trees are technically attached to one unique hypothesis for each segment. The data accessible from each hypothesis include all the public data about syntactic and semantic structures duplicated from the CHART.

A hypothesis represents a state of analysis of the current sentence, that is one possible interpretation of the input in the context of an open dialogue segment. A hypothesis contains all the facts so far available and relevant to the current analysis. These facts are

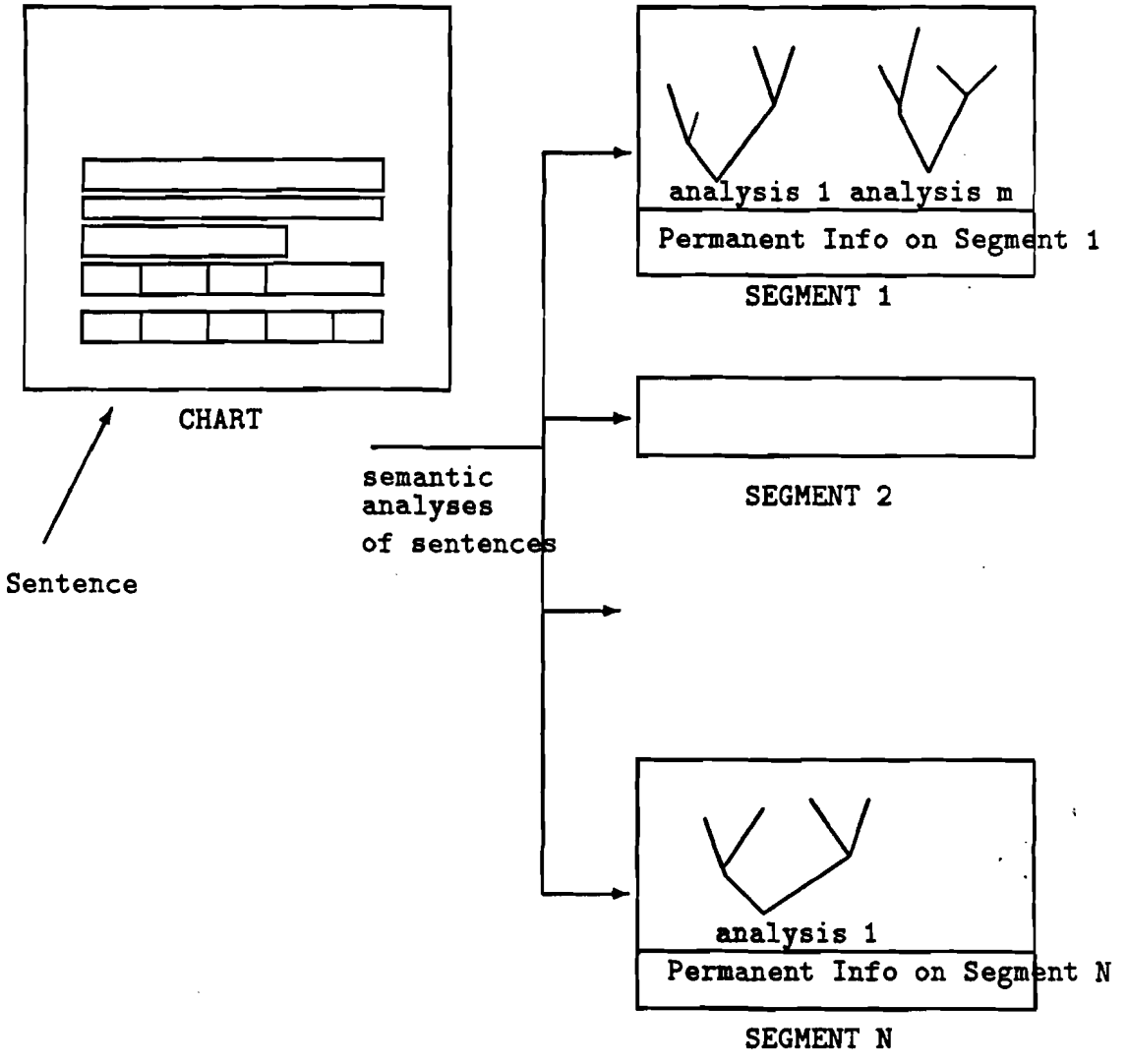

Figure 4.1: Architecture of the BlackBoard

directly associated with the current analysis or available through inheritance relations. A module is always called relatively to a hypothesis node. A context, as viewed by a module, is a collection of the current facts on a path from that hypothesis node to the root. All facts of each hypothesis are public to all modules.

A hypothesis also contains a list of module calls to be executed if the hypothesis is selected as the most plausible one (or plausible enough to be further analyzed). These calls are always propagated to the new hypotheses created from the currently existing ones, so that only hypotheses of leaf nodes are explored and expanded. No task is ever performed from a non leaf hypothesis.

4.2 Representation of Objects

The basic data structure in the Discourse System consists of objects with slot values. Objects are named frame-like structures. The representation of objects is spread out, with one structure for each slot of an object and one structure to record the type of the object. Except for type information which is independent of context, each of these structures represents a fact relative to a particular hypothesis, with a rating and a weight (to be discussed later):

(SLOT-NAME OBJECT-NAME SLOT-VALUES CONTEXT RATING WEIGHT)

In this representation, OBJECT-NAME is a unique identifier used to refer to each object. CONTEXT is the name of a hypothesis structure. SLOT-VALUEs is a disjunctive list of possible values for the slot. Until there is some necessity to differentiate between the possible values and to split the current context into different hypotheses (one for each possible value or any other combination), the complete list is kept within the same representation. Each value in SLOT-VALUES has a rating and weight to specify the plausibility of the value and the confidence in this rating. Values are represented as triples:

(VALUE VALUE-RATING VALUE-WEIGHT)

The overall RATING for a slot is automatically computed as the maximum rating of the possible values recorded in SLOT-VALUES, with WEIGHT being the confidence of that maximum rate. The computation of ratings is discussed in detail in section 4.4.

For example, an object OB1 corresponding to some noun phrase NP37, with logical form LF3, might be ambiguous in reference between two objects. Let the objects be JACK1 and JOHN1, with JACK1 slightly better rated due to a focusing heuristic. The information about OB1 is represented as follows:

```
(OB1 SYN (NP37 10 10)
     SEM (LF3 10 10)
     REF ((JACK1 8 10)(JOHN1 6 10)))
```

If this object is asserted with a context of C12, then the actual representation in our system would be:

```
(SYN OB1 (NP37 10 10) C12 10 10)
(SEM OB1 (LF3 10 10) C12 10 10)
(REF OB1 ((JACK1 8 10)(JOHN1 6 10)) C12 8 10)
```

The representation is distributed into separate assertions in order to interact easily with the trees of contexts that form hypotheses. For instance, if the choice of referent for OB1 has an effect on some other slot value, say S11, then this information is represented by creating two subcontexts of C12, call them C13 and C14. The value of S1 in C13, say AA12, and the value of S1 in C14, say BB12, would be asserted in each context:

```
(REF OB1 (JACK1 8 10) C13 8 10)
(S11 OB1 (AA 10 10) C13 10 10)

(REF OB1 (JOHN1 6 10) C14 6 10)
(S11 OB1 (BB 10 10) C14 10 10)
```

Now, from context c13, OB1 has referent JACK1 and the S1 value AA, and from context C14, OB1 has the referent JOHN1 and the S1 value BB. In context C12 the referent of OB1 is still ambiguous, while the values for the SYN and SEM slot in all three contexts is the same, namely NP37 and LF3.

Each defined object is associated with a type:

(**Type** OBJECT-NAME TYPE)

There is no specification of context. Objects exist and are accessible throughout all the hypotheses and segments. Only slot values may be different in different hypotheses. There is no rating for objects, only slot values have ratings.

To summarize, slot values are implemented as a data structure with the following fields:

- SLOT-NAME: a non unique identifier specifying the kind of information stored in the slot (SEM, NP, PLAN, SPEECH-ACT, etc)
- OBJECT-NAME: the unique identifier of the typed object to which this slot belongs.
- SLOT-VALUE: the list of possible values a slot may take, each value being a triple (VALUE RATING WEIGHT). A value may also be "?" indicating that the slot may have more values than the already specified ones, or GEN(M) where M is some module that can generate more values than it previously has or will check any value that may be proposed by another module. (see section 4.3.2)
- CONTEXT: the name of the hypothesis that the slot is in.
- RATING: a numerical value of the certainty that this structure contains the correct value for the slot, computed as the maximum of all the individual slot values ratings.
- WEIGHT: a numerical value indicating the confidence of a previous rating.

The most basic functions to define objects, assert slot values and retrieve them are:

(**BBDeclareObject** OBJECT-NAME TYPE)

- OBJECT-NAME is the ID under which the object will be posted to the BB. This ID is provided by the module which first creates the object.
- TYPE is the type of the object (e.g. NP, INFO, WANT-EVENT...). Type information is independent of the context (hypothesis).

(**BBDefineSlotValue** OBJECT-ID SLOT L-VALUES CONTEXT)

- OBJECT-ID is the name of the object, that is the identifier under which it is known and has been declared (using BBDeclareObject or BBDefineObject).
- L-VALUES is a list of possible values represented as triples with rating and weight: (VALUE RATING WEIGHT). The list is merged with any previously existing list of values, according to the rules specified in next section.

- When this function is called, a rating and a weight for the overall slot value is automatically computed, according to the rules specified in the section on ratings.
- CONTEXT is the name of the hypothesis in which the list L-VALUES are attributed to the slot. Slot values depend on the hypothesis and an object may have different slot values in different contexts.

(**BBDefineObject** OBJECT-ID TYPE CONTEXT &rest SLOT-VALUES)

- This function is a shorthand combining BBDeclareObject and BBDefineSlotValue. It declares a new object OBJECT-ID of type TYPE and attributes to it the slot values specified in SLOT-VALUES for the hypothesis CONTEXT.
- OBJECT-ID and TYPE are respectively the name and the type of the object.
- SLOT-VALUES is a list of lists of the form (SLOT-NAME VALUE1 VALUE2 ...) where each value (VALUE1, VALUE2,...) is a triple (VALUE RATING WEIGHT).
- CONTEXT is the name of the hypothesis in which the slot values specified in SLOT-VALUES are attributed to the current object.

(**BBGetObject** OBJECT-ID CONTEXT)

- This function returns the list of the slot/value pairs known in CONTEXT for the object OBJECT-ID with the format:
  (TYPE OBJECT-ID (SLOT1 VALUE1) (SLOT2 VALUE2) ...)
  VALUE1, VALUE2... are lists of triples of the form (VALUE RATING WEIGHT).
- CONTEXT is the name of a hypothesis. Its specification is obligatory since an object may have different slot values in different hypotheses.

(**BBGetSlotValue** OBJECT-ID SLOT CONTEXT)

- This function returns the list of possible values in CONTEXT for the slot SLOT of OBJECT-ID, giving each value with its rating and weight:
  ((VALUE1 RATING1 WEIGHT1) (VALUE2 RATING2 WEIGHT2) ...)

(**BBMatchObjects** PATTERN CONTEXT)

- This function retrieves information in the BB through pattern matching.
- It returns the list of the IDs of the objects which match PATTERN in the hypothesis CONTEXT. The syntax of patterns is the same as in pattern declarations and is completely specified in the section on pattern matching.

4.3 Segments and Hypotheses

4.3.1 Segments and their Implementation

The segments of the BB contain general information for the corresponding segment in the discourse. The scope of this information goes beyond the analysis of each sentence, and it is updated each time the interpretation of a sentence has been completed. Each segment is the root of a tree of hypotheses about the incoming sentence.

The structure "segment" contains the following slots:

- NAME: the name of the segment
- STATUS: indicates whether the segment is "active" or "closed".
- HYPOTHESIS: the name of the hypothesis identified with the current segment and root of the tree of hypotheses for that segment.
- PARENT: the name of the segment the current one is part of (if the discourse model maintains a hierarchy of segments)
- BLACKBOARD: the blackboard the current segment belongs to (the system is able to consider several blackboards concurrently).

For each segment, we associate a unique context representing a hypothesis. This hypothesis is the root of the tree of hypotheses developed in the segment. It is identical to all the other hypotheses except that it is never erased. This "permanent" hypothesis contains the permanent information associated with the segment, that is data whose interest and scope go beyond the analysis of one sentence, such as the history list, the current tense of the segment, the focus of the segment or other data resulting from the analysis of previous sentences.

When a new sentence is analyzed, the system creates at least one hypothesis in each segment, as a daughter of the permanent hypothesis. There are more than one such hypothesis when there are ambiguities at the syntactic/semantic level. In each active segment, the system creates exactly as many new hypotheses as there are syntactic/semantic interpretations of the current sentence.

Permanent data is handled with the functions BBDefinePermanentFact and BBGetPermanentFact specified as follows:

(BBDefinePermanentFact FACT-NAME FACT CONTEXT)

- FACT-NAME is the label under which the fact is to be recorded, that is the slot name of the permanent object of the current segment (e.g. history-list, tense, focus...).
- FACT is the value given to FACT-NAME (no rating or weight need to be specify).
- CONTEXT can be either the name of the current hypothesis, the name of the root hypothesis or the name of the current segment. Modifications induced by this function are always performed at the level of the permanent hypothesis associated with the current segment.

(BBGetPermanentFact FACT-NAME CONTEXT)

- FACT-NAME is the label under which the information is recorded (e.g. history-list, focus...).
- CONTEXT can be either the name of the current hypothesis, the name of the root hypothesis or the name of the current segment.

4.3.2 Organization of the Hypotheses

A hypothesis represents a state of analysis of the current sentence. Each hypothesis contains references to its parent, the list of all the facts (objects slot values) which differentiate this hypothesis from its parent, and a list of module calls or actions to be executed to expand this hypothesis.

In a naive implementation, a new hypothesis would be created each time new information is added to the BB, or each time values are modified. However, this view can easily lead to an explosive creation of new hypotheses and a very high rate of work done several times, since the results of a module call on one hypothesis can not be shared with an other hypothesis in a different branch of the tree.

To limit the redundancy and ensure the consistency of the system, we have adopted the following organization:

1. A slot value can never be changed within a hypothesis. A value may be poorly rated, at any time, in order to discard the whole hypothesis, or a different value for the slot may be considered in another branch of the tree of hypotheses. Under some strict conditions (specified below), new values for a slot may be added to existing ones. The system is essentially monotonic within a single hypothesis tree.

2. When a module returns values for a slot, it must return what this module considers as a complete set of possible values for this slot. This set of values may be restrictive if it is fully specified, which means that no other module will be allowed to generate other different values for the same slot. This set may also be incompletely specified in such a way that some of the values will be fully specified later, either by this module or by another one. The special value "?" indicates that other modules may generate other values later. The value GEN(M) indicates that the module M is able to provide other values if required. When a module proposes values for slot which already has values, they are combined according to unification rules described in next paragraph.

3. No module is ever run on a hypothesis that is not a leaf node. So splitting a hypothesis can never occur at a point within a tree.

4.3.3 Combining Slot Values

When a slot value is proposed by a module for a hypothesis that already has values for that slot, the list of values are unified according to certain rules.

Each list of values represents the set of all possible values. These lists consist of actual "explicit" values, the special value '?' and values of the form GEN(X), where X is the name of a module in the system. The presence of the value '?' indicates that the list is open to any other value not yet explicitly mentioned. GEN(X) indicates that any value acceptable for module X can be added to the list. The acceptability is checked by calling module X (with mode "evaluator") on the proposed value (unless we already know that this value has been generated by the same module X).

A module always proposes a complete set of values. We call "restrictive" a set with no '?' and no value of the form GEN(X), otherwise it is an open set of values. A module is not allowed to produce a set containing both a '?' and a value of the form GEN(X).

The unification rules are such that it is never possible to build a list containing at the same time a '?' and a value of the form GEN(X), nor is it possible to have a list with more than one value of the form GEN(X). However, we need to consider sets of values open to new values acceptable at the same time by both module M and module N. We represent such sets by including the value GEN(M,N). This notation is generalized to as many modules as necessary: GEN(M1,M2,M3,M4,...).

The unification rules are as follows:

1. if the first list is restrictive (no '?', no GEN(X)), then:

    a. if the second list is also restrictive: take the intersection, e.g.

        {1, 2} and {2, 3, 4} gives {2}

    b. if the second list contains a value GEN(M): take all the values of the first list which are either explicitly present in the second list or acceptable for module M, e.g.

        assuming 1 is acceptable for module M and 3 is not
        {1, 2, 3, 4} and {2, 4, 5, 6, GEN(M)} gives {1, 2, 4}

    c. if the second list contains a '?': take all the values of the first (restrictive) list, and only them, e.g.

        {1, 2, ?} and {2, 3, 4} gives {2, 3, 4}

2. if both lists are open (they can contain only one element of the form '?' or GEN(X)), then:

    a. if both lists contain a '?': take the union, including the '?' value, e.g.

        {1, 2, ?} and {2, 3, 4, ?} gives {1, 2, 3, 4, ?}

    b. if the first list contains a '?' and the second list contains a value GEN(M): take all the values of the second list, including GEN(M), and the explicit values of the first list which are acceptable for module M, e.g.

        assuming that 1 is acceptable for module M and 3 is not
        {1, 2, 3, 4, ?} and {2, 4, 5, 6, GEN(M)} gives {1, 2, 4, 5, 6, GEN(M)}

    c. if both lists contain a value GEN(M) (the same M in both lists): take the intersection of the lists (that is the intersection of explicit values and the GEN(M) value) plus all the explicit values acceptable for module M, e.g.

        assuming that 1 and 2 are acceptable for module M and 3 and 4 are not
        {1, 3, 5, 6, GEN(M)} and {2, 4, 5, 6, GEN(M)} gives {1, 2, 5, 6, GEN(M)}

    d. if both list contain a value of the form GEN(X), say GEN(M) in the first list and GEN(N) in the second list, with module M different from module N, then: take the intersection of explicit values, the values of the first list acceptable for module N, the values of the second list acceptable for module M, and the special value GEN(M,N), e.g.

        assuming that 1 is acceptable for module M and 3 is not, and
        assuming that 2 is acceptable for module N and 4 is not
        {1, 3, 5, 6, GEN(N)} and {2, 4, 5, 6, GEN(M)} gives {1, 2, 5, 6, GEN(M,N)}

3. In the above rules, each time we have a value of the form GEN(M1,M2,M3,...) instead of a value of the form GEN(X), then proceed as described replacing "acceptable for module X" by "acceptable at the same time for module M1, and module M2, and module M3,...".

When slots have their values unified in this way, the slot rating and the slot weight are updated according to the following rule: always take the maximum rating and weight for any value. After unification, the rating of the whole slot is updated as the maximum of the ratings of the possible values, and the hypothesis rating is also updated according to the rules of hypotheses rating.

4.3.4 Implementation of Hypotheses

Each node of the tree of hypotheses is a data structure with the following fields:

- GEN is the list of module calls wanting to generate new slots for this hypothesis.
- EVAL is the list of modules calls wanting to evaluate values in slots defined in this hypothesis.
- NAME is the name of the hypothesis, referred to also as the context
- PARENT is the name of the parent hypothesis in the tree structure.
- EXTENSIONS is the list of the immediate daughters of the hypothesis.
- SEGMENT is the name of the segment the current hypothesis belongs to.
- FAMILY is the list of ancestors in the hypothesis tree.
- RATING is the overall rating (belief in the correctness of the interpretation) of the hypothesis. It is computed as the minimum of all the slot values ratings defined within this hypothesis.
- FACTS is the list of object slot values (structure described in section 3.2) added for this hypothesis.

The splitting of a hypothesis into several ones in order to account for different possible values for the same slot of an object is an action decided by the module working on these data, when it needs the values to be separated and recorded in different hypotheses. The function BBSplitHypothesis performs this task:

(**BBSplitHypothesis** CONTEXT &rest LIST-OF-LISTS-OF-SLOTNAME-OBJECT-VALUE)

- this function creates daughter hypotheses of the hypothesis referred to as CONTEXT, one new hypothesis for each set of values specified in LIST-OF-LISTS-OF-SLOTNAME-OBJECT-VALUE
- CONTEXT is the name of the hypothesis to be split.
- LIST-OF-LISTS-OF-SLOTNAME-OBJECT-VALUE is a list of lists of slot values for various objects. Each list of slot values has the form ((SLOT-NAME-1 OBJECT-ID-A VALUES-A1) (SLOT-NAME-2 OBJECT-ID-B VALUES-B2) ...)
  where VALUES-XY is as usual the list of all possible values (with rating and weight) for the SLOT Y of OBJECT X within the new hypothesis to be created.
  As many hypotheses as such lists of slot values are created, as daughters of the current hypothesis CONTEXT. LIST-OF-LISTS-OF-SLOTNAME-OBJECT-VALUE gives the distribution of possible values of different slots in the new hypotheses to be created.
- Example:

```
(BBSplitHypothesis 'HYP-23
   '(((SYN e1 ((NP1 10 10))) (SEM e1 ((SNP11 8 8) (SNP12 4 8))))
     ((SYN e1 ((NP2 10 10))) (SEM e1 ((SNP24 10 10))))
    ))
```

4.4 Ratings

Ratings are always difficult to define, but often very useful. In the present system, we need ratings to be able to choose the right interpretation of a sentence, or the most plausible one in the context of the current discourse. Many modules contribute to the interpretation of a sentence, providing different level of analysis, but usually none of these analyses, considered separately, is sufficient to fully accept or reject an interpretation. Ratings and weights are thus used to estimate the relative contribution of each module.

In the architecture of the Discourse System, interpretations are represented by hypotheses. Each hypothesis has a rating and a weight. This rating is basically used by the BB Monitor to choose among all the available hypotheses which one to consider next and to analyze further. In this selection procedure, all leaf node hypotheses compete, from all the different segments of the BB. Hypotheses ratings have a global meaning.

We want the rating to be a measure of the plausibility that this hypothesis represents the right interpretation of the current sentence within the current discourse. The weight gives an estimation of the confidence in the rating. Since several modules may evaluate the plausibility that a slot has a certain value, the weights give a method of combining multiple ratings for the same value. In particular, each module returns a rating Ri and a confidence weight Wi. The combined rating for the value is simply:

$$(\text{SUM } Ri * Wi) / (\text{SUM } Wi)$$

To allow incremental updates to this score, the weight of a value is simply the sum of the individual weights. Thus if a slot value currently has a value R with weight W, adding a new rating of r,w from a module updates the overall rating to (W*R + w*r)/(W+w) and the weight to (W+w). The ratings of a set of values (which we call the rating of a slot) is simply the maximum of the rating of the individual values. Since multiple values are considered to be disjunctions, the maximum rating is appropriate as the disjunctive should be at least as accurate as its best individual disjunct.

The meaning of a hypothesis reflects how well the data collected so far and processed by the different modules fits into the interpretation represented by the hypothesis. If all the knowledge included and applied by the different modules is complete and consistent enough, then each slot in the hypothesis will be highly rated. If a hypothesis includes a slot value with a poor rating, then there is something wrong in the current interpretation as a whole, i.e. some module could not find a reasonable value for that slot that was consistent with the rest of the hypothesis. The practical way of computing a hypothesis rating we thus use is to take the lowest rating of all the facts (slot values) which are part of this hypothesis. In this computation, only ratings whose weight is superior to a certain threshold to be defined, are taken into account. Thus, the ratings of hypotheses are a measure of the coherence of all the facts which hold in this hypothesis. This definition allows us to compute a rating with a global meaning from the local ratings of slot values. A summary of the rating calculations is shown in figure 4.2.

There may be some problems with this view. Poor ratings may be the result of an incomplete model of our linguistic and conceptual domain, unable to handle cases with some apparent contradictions or inconsistencies. Also there is no reason to believe that a model of the domain has to be consistent from all respects. We can only expect that the weights associated with facts ratings are such that no contradictory highly rated decisions will occur. Decisions which are the result of weak linguistic or conceptual constraint should have a low weight, so that when overcome by stronger constraints, there is still an acceptable interpretation to include all these facts. We of course need a mechanism to be able not to take into account low ratings when they are associated with a low weight.

As already presented, the rating of slot values is computed as the maximum of the ratings of all the possible values considered in the current hypothesis. In this computation too we may define a threshold to consider only values with a weight high enough to be significant. The relationships between the ratings is summarized in figure 4.2.

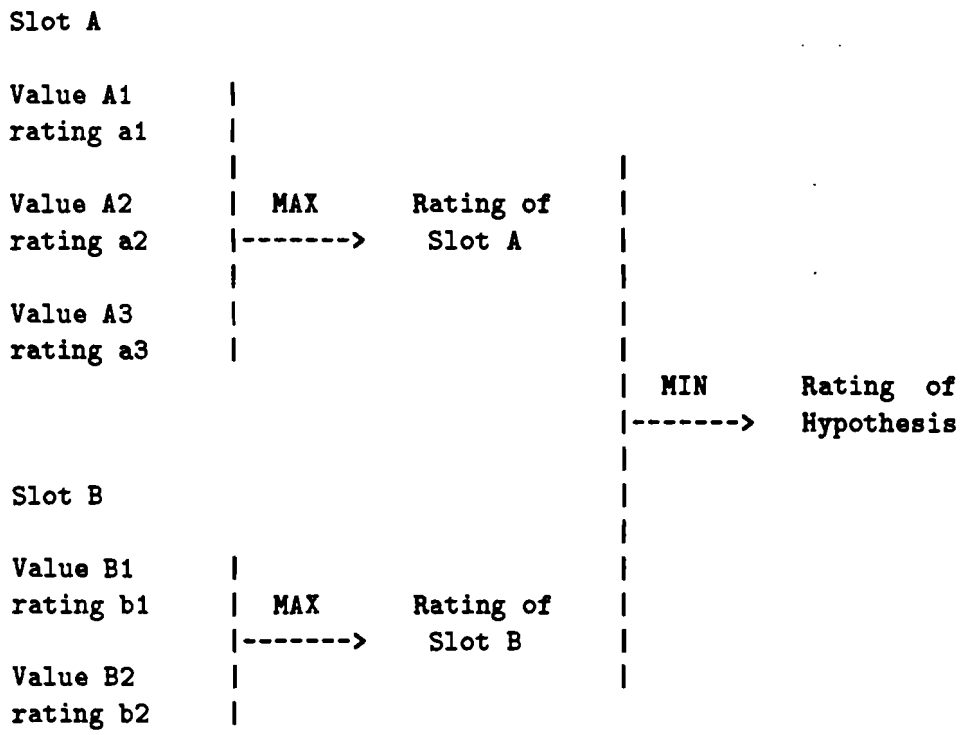

Figure 4.2: Computation of Ratings

4.5 Pattern Matching

Pattern Matching is the main mechanism used by the BB Monitor when new information is posted to the BB. In order to act on data, each module declares patterns to specify the kind of data they are looking for. The BB Monitor records patterns and response frames in a Patterns Table. When a new fact is posted to the BB, the Monitor matches the fact against all appropriate patterns in the Patterns Table. The Patterns Table is actually divided by object types. When the pattern matching succeeds, the response frame is used to construct an executable module call with the current data. This call is stored in the hypothesis associated with the posted data. These calls are executed when the containing hypothesis (or an extension of the hypothesis) is rated highly enough to be further analyzed.

4.5.0.1 Patterns Declarations

Patterns are used by the modules to declare the kind of data they are looking for. The declaration must include whether the resulting module call will only evaluate or also generate slot values. In the second case, the pairs type/slot which could be affected must be specified. When the same module is able to evaluate and to generate slot values, these two modes are clearly separated and different pattern declarations must be used for each mode.

The function BBDeclarePattern is used to declare patterns. Its syntax is the following:

(**BBDeclarePattern** MODE PATTERN FUNCTION-NAME &optional CONTEXT)

- MODE specified the intended behavior of the module. It can be:
  'EVALUATOR or 'GENERATOR

  A module called on a set of data can run on two modes: the "generator" mode and the "evaluator" mode. In the "evaluator" mode, a module can only modify the rating of slot values. In the "generator" mode, a module can assert or add values for slots. The mode must always be specified when the module is called.

  This information is primarily used as a heuristic: "evaluator" calls are executed before "generator" ones.
- PATTERN specifies the features of the object the module is looking for (see the description of the syntax of patterns)
- FUNCTION-NAME is the name of the function to be used in the return call when an object matched the pattern. This function has two arguments: the object found and the context the object was found in. These functions are the functions defined by each module to perform its task.
- an optional CONTEXT may be specified to limit the search.

Pattern matching may also be considered to retrieve data from the BB, using the function BBMatchObjects:

(**BBMatchObjects** PATTERN CONTEXT)

- This function returns the list of the IDs of the objects which match PATTERN in the hypothesis CONTEXT. The syntax of patterns is specified in next paragraph.

4.5.1 The syntax of Patterns

The general syntax of patterns is the following:
(TYPE ID-VAR &rest SLOT-VALUES)

- TYPE is the type of the objects looked for. It can also be a variable or a list of types.
- ID-VAR is a variable which stands for the object which potentially matches the pattern. Any symbol beginning by a question mark is understood to be a variable.
- SLOT-VALUES is a list of slot restrictions of the form:
  (SLOT-NAME RESTRICTION)
  where RESTRICTION may be:
  - a value (it must be quoted)
  - a pattern
  - a variable (beginning with a question mark like ?X) to indicate that the slot is set to a value without giving any further specification
  - a list explicitly beginning by OR to allow multiple choice
  - NIL to indicate that the object should not have any defined value for this slot
- EXAMPLES:
  (S ?x (TENSE 'PAST) (SEM NIL)) will match any object of type S (sentence) which has its TENSE slot set to PAST and does not have its SEM slot defined.
  (S ?x (SUBJ 'np23)) will match any object of type S with its slot SUBJ set to object np23.
  ((NP S) ?x (NUM (OR '3s '3p))) will match any object of type NP or S with a NUM slot set to 3s or 3p.
  (INFO-EVENT ?x) will match any object of type INFO-EVENT.
  (?type ?id (SYN (NP ?np))) will match any object which has its SYN slot set to an object of type NP.
  (ENTRY ?x (SYN (NP ?np)) (SEM nil)) will match any object of type ENTRY whose value for the slot SYN is an object of type NP and whose SEM slot is not set.
  (INFO-OBJ ?i (ABOUT (BOOK ?b (LANGUAGE 'French)))) will match all the objects of type INFO-OBJ with a slot ABOUT set to an object of type BOOK with the slot LANGUAGE set to the value: French.

In the matching process, if a slot restriction of a pattern is itself a pattern, the matching algorithm is recursively applied to check if this pattern matches the corresponding value in the object the first pattern is matched against.

4.6 The Blackboard Monitor

The Monitor is in charge of the book keeping in the BB, manages the Table of Patterns and controls the execution of module calls.

The Table of Patterns the Monitor keeps is a list of the stimulus and response frames, that when matched, trigger the creation of executable module calls on the triggering data. All facts of the BB (slot values) are always accessible to all modules, but the control of the system allows modules only one chance to match on any given new fact posted to the BB.

When a new fact matches a pattern declared by a module, a module call is built and stored in a run queue to be later executed. The system contains many run queues.

A BB keeps a global run-queue that is invoked if there is not any more module calls that can be executed from any hypothesis currently active in the BB. This global run queue probably will not contain any module calls but more general actions. these could include changes to the overall control strategy such as lowering threshold or changing default values. Other actions can include reading new input or other global control procedures, like erasing the current BB to analyze the next sentence.

In addition to the global run-queue, each hypothesis node in the BB contains two run queues. One contains only module calls in mode "evaluator" (the EVAL run queue) and the other contains module calls in mode "generator" (the GEN run queue). The EVAL run queue is always executed first and only when it is empty, the GEN run queue is considered.

The GEN and EVAL fields are also explicitly copied after they have been updated: module calls must always refer to a leaf hypothesis node when they are executed. No data is copied from the FACTS field. This information is always implicitly available through inheritance relations. When looking for a slot value, the tree of hypotheses is searched from the leaf nodes up to the root. Though slot values are not modified, they may be further specified, restricted or have their rating updated in the nodes the most distant from the root. In this way, only the most recent information available is considered.

The main task of the Monitor is to pick a hypothesis and execute the module calls stored in its run queues. The monitor selects a hypothesis by looking through all the leaf hypothesis nodes, and chooses the hypothesis with the highest rating multiplied by the weight. When a hypothesis is chosen, the monitor executes its module calls, beginning with the EVAL queue and then the GEN queue.

At each step, the selection of the most plausible hypothesis is considered again. After each module call is completed, the monitor goes again through the selection process, since any module call may result in a change in the rating of the current hypothesis. There is no defined policy to choose among hypotheses with the same rating, so equally rated hypotheses are chosen at random.

Module calls are inherited (elevated) by later searching the hypothesis tree. When a hypothesis has the highest rating, then all of the module calls in that hypothesis and its ancestors are scheduled. Only those module calls actually executed are marked as such in the hypothesis structure.

There are some differences in the way the monitor handles information and runs the system when the data are kept on the CHART and the modules involved are the lexical, syntactic and semantic modules. The syntactic and semantic modules may keep a private working space and not post all the data they produce to the BB. In addition, the separation of the CHART from the other segments of the BB results in a separation between two different periods in the analysis of a sentence. First the lexical, syntactic and semantic modules are triggered, but as they post information only to the CHART, no other modules can be triggered. At this point, the monitor has to initiate the second period of the analysis of the sentence, and to do so, it has to duplicate part of the data posted to the CHART and to re-post them to all the active segments. This particular task may include the creation of several hypotheses to account for ambiguities at the syntactic/semantic level.

Chapter 5

A SUMMARY OF INTERFACE FUNCTIONS

5.1 Pattern declaration

(BBDeclarePattern MODE PATTERN FUNCTION-NAME &optional CONTEXT)

Declares PATTERN to the blackboard. If an object matches the pattern a call to FUNCTION-NAME with the object and the context the object was found in as arguments is added to the Run Queue of CONTEXT corresponding to MODE ("generator" or "evaluator").
Example: `(BBDeclarePattern 'TENSE 'GENERATOR '(SYN S ?x (TENSE 'Past)))`

5.2 Defining and retrieving information on the BB

(BBDeclareObject OBJECT-NAME TYPE)

Declares a new object with identifier OBJECT-NAME to be of type TYPE.
Example: `(BBDeclareObject 'np234 'NP)`

(BBDefineSlotValue OBJECT-ID SLOT L-VALUES CONTEXT)

Adds L-VALUES to the list of possible values for the slot SLOT of object OBJECT-ID in the hypothesis CONTEXT.
Example:

```
(BBDefineSlotValue 'pro13 'REFERENT
                   '((NP37 8 10)(NP49 4 10)) 'HYP4)
```

(**BBDefineObject** OBJECT-ID TYPE CONTEXT &rest SLOT-VALUES)

Declares a new object OBJECT-ID to be of type TYPE and attributes the slot values specified in SLOT-VALUES for the hypothesis CONTEXT.
Example:

```
(BBDefineObject 's21 'S 'HYP7
                '(MOOD decl) '(SUBJECT NP12)
                '(MAINV want) '(COMPL s7))
```

(**BBGetObject** OBJECT-ID CONTEXT)

Retrieves the list of the slot/value pairs known in CONTEXT for the object OBJECT-ID. Actually returns the structure:
(TYPE OBJECT-ID (SLOT1 VALUE1) (SLOT2 VALUE2) ...)
Example: `(BBGetObject 'np45 'HYP16)`

(**BBGetSlotValue** OBJECT-ID SLOT CONTEXT)

Returns the list of possible values for the slot SLOT of object OBJECT-ID in the hypothesis CONTEXT, with the rating and weight of each value:
((VALUE1 RATING1 WEIGHT1) (VALUE2 RATING2 WEIGHT2) ...)
Example: `(BBGetSlotValue 's8 'TENSE 'HYP2)`

(**BBMatchObjects** PATTERN CONTEXT)

Returns the names of the objects which match PATTERN in CONTEXT.
Example: `(BBMatchObjects '(SYN ?x NP (NUM 'plural)) 'HYP39)`

(**BBGetType** ID)

Returns the type of an object given its ID.
Example: `(BBGetType 'NP345)`

5.3 Permanent Data

(**BBDefinePermanentFact** FACT-NAME FACT CONTEXT)

Records permanently the value FACT under the label FACT-NAME in the segment associated with CONTEXT (a hypothesis or a segment name).
Example: `(BBDefinePermanentFact 'TOPIC 'CRASH-29 'SEG3)`

(**BBGetPermanentFact** FACT-NAME CONTEXT)

Retrieves the value of the permanent fact FACT-NAME in the segment associated with CONTEXT (a hypothesis or a segment name).
Example: `(BBGetPermanentFact 'TOPIC 'SEG2)`

5.4 Splitting hypotheses

(BBSplitHypothesis CONTEXT &rest LIST-OF-LISTS-OF-SLOTNAME-OBJECT-VALUE)

Splits the hypothesis CONTEXT in as many hypotheses as lists of list of values defined in LIST-OF-LISTS-OF-SLOTNAME-OBJECT-VALUE.
Example:

```
(BBSplitHypothesis 'HYP41 '((REFERENT pro13 ((NP37 8 10))))
                          '((REFERENT pro13 (((NP49 4 10))))))
```

5.5 Handling hypotheses and segments

(BBGetHypothesis HYP-NAME)

Returns the real structure describing the hypothesis whose name is HYP-NAME.
Example: `(BBGetHypothesis 'HYP34)`

(BBGetPermanentHypothesis PERM-HYP-NAME)

Returns the real structure describing the permanent hypothesis whose name is PERM-HYP-NAME.
Example: `(BBGetPermanentHypothesis 'PERMHYP91)`

(BBSegmentList)
(BBActiveSegments)
(BBLeafActiveSegments)

These functions return respectively: the list of all the segment names, the list of active segments and the list of leaf active segments.

5.6 Information on speakers

(BBGetSpeaker)
(BBGetHearer)

Return the speaker or the hearer (a Lisp symbol).

(BBGetSpeaker-RHET)
(BBGetHearer-RHET)

Return the speaker or the hearer as a RHET object.

(BBSetSpeaker NAME)
(BBSetHearer NAME)

Set the speaker or the hearer. NAME must be a Lisp symbol.

(BBExchangeSpeakers)

Exchanges the roles of the speaker and the hearer.

Chapter 6

DESCRIPTION OF MODULES

6.1 Lexical Module

The function of the lexical analysis module is to accept and identify words (strings). The words are from utterances (sentences) between a speaker and a hearer in a dialog (discourse). Each string is to be analyzed and the module is to return information from the derived/inflected form (the actual input), suffix information and the root dictionary entry.

The module's algorithms are of an *ad hoc* design but were in part motivated by an early draft of work by Martin Kay [1989]. The general notion is to build a working (prototype) program and to explore lexical and morphological issues thru the use of the program. This will enable the Discourse System (DS) Project to have a viable and working version at all times. The modules coverage and robustness will be incrementally improved. The lexical and morphological methodologies employed will evolve over time. The role of a lexical analyzer in a discourse understanding system may also be analyzed through the use of this program and it may well be that the role of the program will change over time.

6.1.1 Forms Table

The direction that the processing takes depends on if the initial input string is found in a table of forms (irregular entries). This is the initial course of processing upon reading from the input stream.

If an input string is found in the forms table, the associated information is returned. This information is augmented with positional and context information, for DS use, and currently with a pointer to the 'root' entry for this word. Seen here as the variable 'fly-entry.

Example:

Input word "Flew"

```
("FLEW" (VERB (FORM . PAST) (ROOT . "FLY"))
        'fly-entry
        (POSN 8) (UTTER 1)
        :CHART)
```

The table contains word forms that can not be morphologically analyzed for their root (stem plus derivations/inflection). A successful look-up returns the explicitly stored (morphologically embedded) information about that word. This usually includes the tense,

number, sentence positional information and syntactic category and the dictionary root entry. In a fuller development this latter information may serve as a pointer into a hierarchy of lexical information. This might allow more detailed information to be extracted from a more general encoding of information.

6.1.2 Stems and Suffixes

If no entry is found in the forms table then morphological decomposition begins. Currently, only one level of analysis is performed. Recursive extensions to this should be pursued. The processing begins with the identification of legal suffixes. The lists of legal suffix strings are contained in global variables for each of 1, 2, 3 and 4 character length suffixes. The zero length suffix corresponds to a dictionary root entry equivalent to the current input string. There is no inherent limit to suffix length but it is constrained by the input string length. This could become a heuristic for a more complex module attempting to run in real time, that is, not to be exhaustive in the search.

Suffixes are created by stripping off subsequences of characters from the string . Each potential suffix is checked against the legal list and a 'suffix-list' of legal suffixes is created. Currently this list always contains five entries (the number of defined legal suffix lengths plus the empty string suffix). Each suffix-list entry, along with the zero length suffix, may be nil. This indicates that no legal suffix of that length was found. Each non-null suffix and its associated stem (input string less suffix) is mapped to a look-up routine. For each length suffix and for each specific suffix certain transformations on the stem may generate more than one actual look-up in the dictionary.

6.1.3 Rules Example

There is a separate rule for each suffix length. Within each rule are cases for each identified legal suffix; currently not complete. Within each case there may be any number of specific calls to search the dictionary for root entries based on the current stem. The rule below shows an example of:

- Remove a consonant that was doubled to form the past tense form. An example is *"manned"* from *"man"*.
- Searching the dictionary for the stem, as is, OR searching for the the stem with an added vowel.

```
((string= "ED" suffix)
            (if (dbl-consonant-p root)
                (look-up (del-last-char root) suffix)
                (or (look-up root suffix)
                    (look-up-with-added-vowels root suffix))))
```

This excerpt of the rule for length 2 character endings indicates that the stem (root) will be transformed by deleting the last character if the last two characters form a double constant subsequence. If this fails to find an entry then the stem will be looked up as is and finally with added vowels; if the simple stem is not found as an entry. Searching for the stem plus an added vowel stops on the first successful look-up.

6.1.4 Suffix Semantics

After one particular entry has been found that entry is filtered by the associated suffix. The attempt is to map certain semantics with each suffix. This allows, for example, only the noun portion of each entry to be returned for possessive ("'s" suffix) forms. This is the process that dynamically creates the return form that is explicitly stated for each entry in the forms table.

The area of suffix semantics seems a good research area, especially combined with hierarchical notions. Various hierarchies of semantics, rules and word types might be able to return significant information from morphological analysis.

Further, the module, operating in a discourse understanding system may be able to use additional available information. The presence of certain classes of words or forms of words in some positional relation may be able to trigger certain key-word or phrase searches. These words/phrases may be semantically void linguistically, as "in any event", but carry much discourse information. We hypothesis that there may be lexical entries such as "the stock market crash of 1929" which may be impossible to parse without lexical support and additional control in parsing.

6.1.5 Example

The following is a sample pattern posted to the Blackboard, and a sample output of the lexical module.

```
(BBDeclarePattern 'GENERATOR
                  '(SENTENCE %s (STRING %y))
                  'Lexical-Analysis
                  :CHART)

(LEX w1  (WORD "I") (ROOT "I") (CAT PRO) (NUM 1s))
         (ENTRY ew1 (SYN w1) (BEG 1) (END 2))
```

This Lexical entry would be posted in the following manner.

```
(BBDeclareObject 'W1 'LEX)

(BBDefineSlotValue 'W1 'WORD '(("I" 10 10)) CONTEXT)
(BBDefineSlotValue 'W1 'ROOT '(("I" 10 10)) CONTEXT)
(BBDefineSlotValue 'W1 'CAT '((PRO 10 10)) CONTEXT)
(BBDefineSlotValue 'W1 'NUM '((|1S| 10 10)) CONTEXT)

(BBDeclareObject 'EW1 'ENTRY)

(BBDefineSlotValue 'EW1 'SYN '((W1 10 10)) CONTEXT)
(BBDefineSlotValue 'EW1 'BEG '((1 10 10)) CONTEXT)
(BBDefineSlotValue 'EW1 'END '((2 10 10)) CONTEXT)
```

6.2 Syntactic Module

The Syntactic module is not yet implemented. It is currently simulated for the test input sentences. There is nothing special about the architecture or the interactions that requires anything more than a chart or bottom-up parser. As is the case with all modules, the requirement is that be able to interact with the Blackboard in terms of I/O and functionality. No restraints are placed on its internal workings.

The following is an example of a pattern declaration by the syntactic processing module. The pattern of interest is any Entry on the chart that has a Syntactic slot. In this example the Entry "the stock market crash of 1929" is found and identified as an NP.

```
(BBDeclarePattern 'GENERATOR '(ENTRY %x (SYN %s)) 'Syn-Parse :CHART)

(BBDeclareObject 'E7 'ENTRY)
(BBDefineSlotValue 'E7 'SYN '((NP3 10 10)) CONTEXT)
(BBDefineSlotValue 'E7 'BEG '((9 10 10)) CONTEXT)
(BBDefineSlotValue 'E7 'END '((15 10 10)) CONTEXT)

(BBDeclareObject 'NP3 'NP)
(BBDefineSlotValue 'NP3 'DET '((EW9 10 10)) CONTEXT)
(BBDefineSlotValue 'NP3 'PREMODS '(((EW10 EW11) 10 10)) CONTEXT)
(BBDefineSlotValue 'NP3 'HEAD '((EW12 10 10)) CONTEXT)
(BBDefineSlotValue 'NP3 'MODS '(((E8) 10 10)) CONTEXT)
```

6.3 Semantic Module

6.3.1 The Semantic Interpretation Module

The *Semantic interpreter* transforms the syntax into a logical form, which is quasi-independent from the surface format of the input. The logical form captures the functional relations of syntactic constituents, and is based in part, especially for verb phrases, on thematic roles. The format is roughly that of [Allen, 1987], as described in Chapter 7. Each interpretation is a blackboard object, where the *type* is the *head interpretation* of the syntactic object, each slot-name represents a role, property or attribute of the object, and the fillers are other semantic objects. As an example (with ratings and weights omitted):

```
Input:
     entry22: ''the man at the bank''

Syntactic interpretation: np23
     np23  :  (np (DET the)
                  (HEAD man)
                  (MODS (pp (PREP at)
                            (pobj (np (DET the)
                                      (HEAD bank))))))

Semantic Interpretation: sem24
     sem24 : (man (SPECIFIER def sing)
                  (AT_LOC sem25))
```

```
sem25 : (bank (SPECIFIER def sing))
```

The knowledge for semantic interpretation is stored in two unconnected hierarchies. The *lexical hierarchy* stores language specific knowledge, such as the thematic roles of subcategorized for constituents. The *conceptual hierarchy* stores information related to the meanings of the objects. This hierarchy contains general world knowledge, independent of how such knowledge might be expressed in any language utterance.

In the discourse system, as currently conceived, the lexical hierarchy is strictly a part of the semantic interpreter, and the conceptual hierarchy is contained strictly within the conceptual interpreter/reference module. One consequence of this is that selectional restrictions on thematic role assignments (eg AGENT must be animate) can not be done by the semantic interpreter. A sentence like "the chair ate my wallet" must be equally acceptable to the semantic interpreter as "the man ate an orange." It is only the conceptual interpreter, which has access to the properties of chairs, which can reject the first sentence (or find a metaphorical meaning for eat).

6.3.2 A close up look at the Semantic Interpreter

The interpretation of a phrase is done in a three step process. The first stage is *Head Analysis*, which finds all acceptable head-interpretations. A head-interpretation consists of the type of a semantic object. There is a distinct head-interpretation for each different surface sense of the head. Surface senses differ only if they have different subcategorization requirements. Thus *explore* has one common sense in *I explored the forest* and *I explored this new idea for adjective interpretation*, but *bomb* has two different senses in *The Allies bombed the Germans* and *Our show bombed last night.*

In the current implementation only the syntactic head word is checked in the head analysis but there is the potential for more complex forms of head interpretation to be incorporated. If the combination of a head word with one or more of its modifiers yields a different type, the compound can yield a different head. There is room here for more complex interaction with the lexicon. We may not want to represent compounds as single lexical items in the syntax, but may want to treat the semantics as such. Particular decisions will depend on the format for the lexicon and syntactic parser, which have not yet been worked on to any great extent.

The second stage is *Inner-case Analysis.* First, each head-interpretation is used as a pointer into the lexical hierarchy. The hierarchy stores two kinds of information, *subcategorization frames* and *interpretation rules. subcategorization frames* are a template for the final interpretation of a given type. They contain lists of which slots are necessary and which slots are prohibited for a given sense. As an example, the frame for the verb *buy* might look something like:

```
( NEEDED-SLOTS theme
  PROHIBITED-SLOTS to-poss )
```

This signifies that the THEME roll must be present in the surface sentence, and that no TO-POSSESSOR may be present (the conceptual function of the TO-POSSESSOR is filled by the AGENT). Thus sentences like **I bought the book to Mary* would be blocked from interpretation, as would any in which no constituent could be found to fill the THEME slot.

An *interpretation rule* is roughly equivalent in function to the rules given in [Allen, 1987], with a few format changes for computational efficiency. Rules have been localized, so that each rule consists of a list of predicates which must be true of the input for the rule

to apply (so far this is restricted to two types, patterns which must be in the input, and patterns which may not be), and functions for determining the constituent accounted for by the rule, and the slot and filler provided by the rule. A rule used for interpreting the subject of an active sentence might look like:

```
( MATCH-PATTERNS ( (S (SUBJ +)) )
  DON'T-MATCH-PATTERNS ( (S (VOICE PASSIVE)) )
  SLOT-NAME AGENT
  SLOT-VALUE (%SEM %0)
  MATCHED-SYNTAX %0 )
```

This rule matches a non-passive sentence that contains a subject. If the rule matches, it fills the AGENT role with the semantic interpretation of the subject (%0 means the first match) and accounts for this match.

Once the hierarchy has been searched, an attempt is made to fill an interpretation for each frame. Each rule is tested on the input, succeeding only if its conditions are met and the goal role is not excluded by the frame. Merging is also guided by the principles that each syntactic constituent may only be accounted for by one semantic interpretation, and, for certain roles, only one of them may appear per sentence. For example, the sample rule above could be used in any frame which does not prohibit the agent slot, but could not be used in combination with another rule which either 1) matches the subject, or 2) fills the AGENT slot. It also could not be used in any completed frame where there is no other way to get a NEEDED-SLOT than to use a conflicting rule.

The third phase is the modifier analysis. Here all remaining modifiers, which are not linked to the interpretation of the head, but have the same interpretation wherever they appear are interpreted. These rules come from separate hierarchies (or tables) which are indexed by the particular modifiers (or their heads for compound modifiers like prepositional phrases). The modifier rules have the same format as the rules described above which are indexed by the head interpretations.

The final result is a list of all completed frames from all head interpretations which meet the following two constraints: 1) all of their must-match roles have been filled 2) each constituent in the syntax has been given an interpretation in the frame.

6.3.3 Example

The following are examples of patterns declared by the Semantic module that will match against entries that have a syntactic slot with a value of type S or NP and that have no SEM slot defined.

```
(BBDeclarePattern 'GENERATOR '(ENTRY %x (SYN (NP %z)) (SEM NIL))
                  'Sem-Interpret  :CHART)

(BBDeclarePattern 'GENERATOR '(ENTRY %x (SYN (S %z)) (SEM NIL))
          'Sem-Interpret :CHART)
```

Examples of output from matches to the of patterns are given below. The first is in response to the posting of the SYN slot value NP4 and the second set of declarations is in response to the posting of the syntactic object S1.

```
(BBDeclareObject 'SEMNP4 'DATE)
(BBDefineSlotValue 'SEMNP4 'SPEC '((NAME 10 10)) CONTEXT)
(BBDefineSlotValue 'SEMNP4 'VALUE '((1929 10 10)) CONTEXT)
(BBDefineSlotValue 'E9 'SEM '((SEMNP4 10 10)) CONTEXT)


(BBDeclareObject 'SEMS1 'BE-INTERESTED)
(BBDefineSlotValue 'SEMS1 'SPEC '((PRESENT 10 10)) CONTEXT)
(BBDefineSlotValue 'SEMS1 'EXPERIENCER '((SEMNP1 10 10)) CONTEXT)
(BBDefineSlotValue 'SEMS1 'THEME '((SEMS2 10 10)) CONTEXT)
(BBDefineSlotValue 'E1 'SEM '((SEMS1 10 10)) CONTEXT)
```

6.4 Reference Module

6.4.1 Introduction

The reference module builds conceptual representations of sentences from their logical forms in a RHET environment. This process consists of two main subprocesses: the conceptual interpretation of expressions and the assignment of objects to the expressions to which they refer. The conceptual interpreter is similar to the semantic interpreter in that it is a recursive process which builds conceptual representations of expressions out of representations of their parts. The structures of conceptual representations usually parallel their corresponding logical forms. The treatment of rolenouns and default values are exceptions. The process of building conceptual representations relies on the reference resolution process to determine the objects to which anaphoric expressions refer, and to create new objects when it is appropriate.

The module employs the blackboard structure to store and retrieve information. The reference module is called by the blackboard when a sentence with a semantic interpretation is posted. It uses the syntactic and semantic information of the sentence in its subsequent processing. In addition, it makes use of the blackboard to maintain a history list for each hypothesis which it uses to match conceptual entities corresponding to the sentence to objects in the representation of the world thus far constructed. (At a later stage, it will maintain and make use of the focus as well.) The module posts conceptual representations with ratings to the blackboard and consequently affects ratings for hypotheses and segmentations in the overall discourse understanding process.

The reference module also stores three types of information independently of the information stored on the blackboard. It contains a list of conceptual interpretation rules, each corresponding to the mapping of a logical form to a complex concept, i.e., a concept whose representation contains slots and fillers. It also contains a data base of concepts arranged hierarchically in terms of their meanings which is used in the construction of conceptual representations and in the enforcement of the type restrictions of the conceptual interpretation rules used in this construction. Finally, there is a large data structure containing heuristics for determining reference. Examples of each of these are given below.

6.4.2 The Hierarchy - Theoretical Issues

The organization of the type hierarchy reflects a simple theory of meaning and reasoning. It assumes that reasoning, to the extent that it relies on the hierarchy to retrieve conceptual information, is a process based solely on the meaning of its concepts and not on structural properties of noun phrases and sentences (except to the degree that these properties are

reflected in their meaning). By structural properties, we mean both the syntactic properties of a parse and the general semantic properties such as *X contains a theme.* It seems appropriate that the conceptual interpreter, which implicitly reasons about the concepts associated with the words it is processing, be guided by constraints pertaining to the meanings associated with the words it is interpreting, and not their structural properties. That is, words with similar meanings should be processed in similar ways, regardless of the dissimilarities they may have in their logical forms. The meaning of a concept, as determined by the hierarchy, is reflected only in its type, supertypes and subtypes, its roles and their type constraints, and the axioms which correspond to it. The sentence in which a word is used can affect which interpretation of the word is arrived at by enforcing type constraints associated with the complex concepts and axioms associated with the words of the sentence.

Consider the type hierarchy as shown in Figure 6.1.

This hierarchy requires that *date* in the sentence *The date of the book is 1988.* refer to a concept which implies a point in time and not a social event.

6.4.3 The Conceptual Interpretation Algorithm and Reference

The conceptual interpreter modifies the system's representation of the world by processing logical forms of sentences of the dialog. The main two procedures involved in this process are (1) the application of appropriate rules to the logical form in order to construct instances with the appropriate roles, and (2) calls to the reference determiner to identify instances which have previously been created which may fill roles in instances under construction.

The interpreter applies rules in a variety of ways, reflecting the various meaning structures concepts can have. In the simplest case of a lexical entry that corresponds to a concept which is represented in the hierarchy as a *simple* type, it either creates an instance of this type or retrieves an appropriate instance from a call to the reference determiner. In the most typical case of a lexical entry

corresponding to a concept which is represented in the hierarchy as a *complex* type, it does a simple mapping between slot-fillers in the logical form and roles in the conceptual representation. It assigns as the role of an instance of a concept, the instance associated with the contents of the corresponding logical form slot. Under many circumstances, this instance is constructed on the spot (for example, if the logical form slot is filled by an indefinite noun phrase). More often, however, it is retrieved by complex operations of the reference determiner on the history list. Finally, there are more complex techniques of constructing a representation that involve mapping logical forms to instances with default role values and mapping logical forms with no slots to instances with several roles. These more involved techniques occur when processing logical forms containing role nouns and nominalized verbs.

The Interpretation Algorithm

The current implementation allows for the following types of interpretation:

- simple type interpretation - lexical entries correspond to simple types.

```
    (ref-interpret '((indef sing) b1 t-book))
→ [T-BOOK4231]
```

- normal structured interpretation - lexical entries correspond to a complex type such that there is a one to one mapping between cases and roles (with the option of default values in any unaccounted for roles). Simple structured nouns (i.e., not rolenouns) and verbs are interpreted this way.

```
(ref-interpret '(pres w1 write (agent (pro y1 t-you))
```

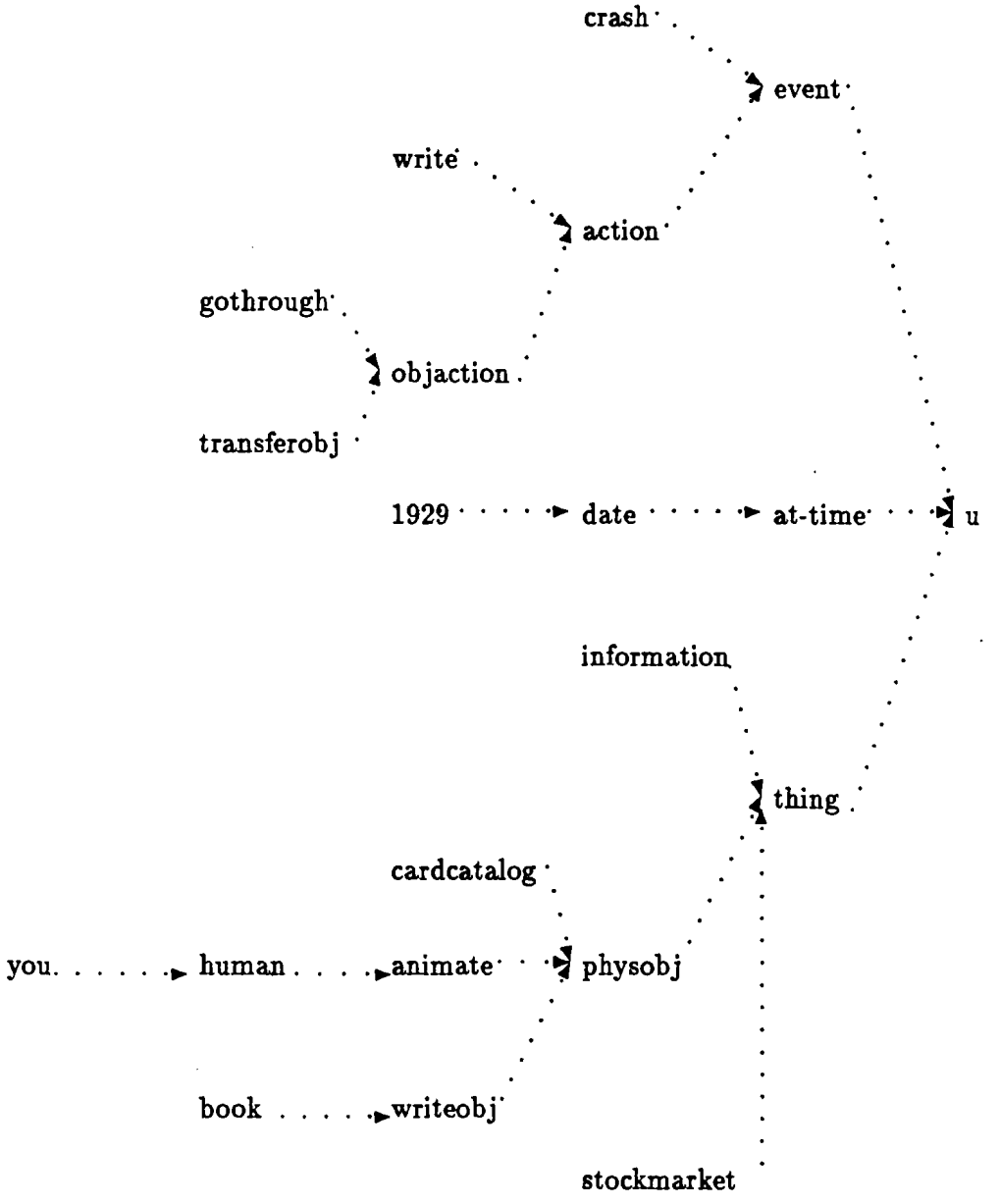

Figure 6.1: Hierarchy of Types (only relevant types shown)

(theme ((indef sing) b1 t-book))))
→ [WRITE2345]
[F-ACTOR[WRITE2345]] = [USER1]
[F-WRITE-OBJ[WRITE2345]] = [T-BOOK1234]

(ref-interpret '(pres s1 sell (agent (pro y1 you))
(to-poss ((def sing) b1 boy))
(theme ((indef sing) b1 book))))
→ [SELL2222]
[F-TRANSFERER[SELL2222]] = [USER1]
[F-RECEIVER[SELL2222]] = [T-BOY4321]
[F-OBJECT[SELL2222]] = [T-BOOK4567]
[F-EXCHANGEOBJ[SELL222]] = [MONEY1234]

- nominalized verb interpretation - lexical entries which are nominalized verbs are interpreted as if they were verbs (normal structured interpretation).

(ref-interpret '((def sing) c1 crash (theme ((def sing) s1 stock-market))))
→ [CRASH8888]
[F-MARKET[CRASH8888]] = [T-STOCK-MARKET1111]

- role noun interpretation - an instance is located on the history list or created which has a role whose value corresponds to the lexical entry of a role noun.

(ref-interpret '((def sing) a1 author))
→ [T-HUMAN1234]
(where for some object, say [T-WRITTEN-OBJ3333],
[F-AUTHOR [T-WRITTEN-OBJ333]] = [T-HUMAN1234])

These interpretation processes are guided by a general interpretation algorithm which requires that each lexical entry found in the logical form be associated with a type. For simplicity, the type name for a simple entry (entry having no corresponding slots and fillers in its logical form) is the same word as the entry itself. Verbs and other complex types tend not to follow this convention. For example, the type associated with *book* is *book*, whereas the type associated with *buy* and *sell* is *transferobj*.

The type information for a complex type is recorded in its corresponding rule(s). Each lexical entry corresponding to a complex conceptual representation must be associated with a list of one or more such rules. Each rule has:

- a STYLE (normal or role-noun)
- a TYPE which represents the concept corresponding to the lexical entry
- a list of pairs of logical form case names and their corresponding role names called CASE-ROLE-PAIRS
- a list of pairs which vary in nature according to the style of the rule called SPECIAL-ROLES

The rules used to arrive at the interpretations at the beginning of this section are the following:

1. WRITE: (normal write ((r-actor agent) (r-write-obj theme)) ())
2. SELL: (normal transferobj ((r-transferer agent) (r-object theme) (r-receiver to-poss)) ((r-exchangeobj *ref-money*)))
3. CRASH: (normal crash ((r-market theme) (r-time at-time)) ())
4. AUTHOR: (rolenoun t-writtenobj () ((r-author t-human)))

The interpretation algorithm is as follows:

1. If the logical form being interpreted is simple (has no slots and fillers), then the reference determiner is called to find or create an appropriate instance *I* to which the expression will be assigned.

   Add *I* to the history list.

   Return *I*. If the logical form is complex, then:

2. Look up the rule corresponding to the head of the logical form.
3. If the style of the rule corresponding to the logical form is NORMAL, then (a) create an instance *I* of type TYPE, (b) assign to each role of *I* listed in CASE-ROLE-PAIRS, the *conceptual interpretation* of the value of the corresponding slot in the logical form, and (c) assign to each role of *I* listed in SPECIAL-ROLES, the predefined instance it is paired with. (See Special Aspects...)

   Add *I* to the history list.

   Return *I*.

4. If the style of the rule corresponding to the logical form is ROLENOUN, then locate on the history list the most recent instance *I* of type TYPE. (If *I* is null, define an instance *J* of type TYPE, and add it to the history list.)

   Assign as the interpretation of the logical form the value of the role of *I* (or of *J* if *I* is null) which corresponds to the LF. (Call it *R*.)

   Add *R* to the history list.

   Return *R*.

The Reference Determiner

The reference determiner has the job of assigning to each event and noun phrase of the logical form of the dialog a particular object. In doing so, it must successfully account for anaphoric reference as well as non-anaphoric reference. Non-anaphoric reference that is considered in this work includes reference to objects which are default values for events, those which serve as parameters in the plans of the speaker and the hearer (currently unimplemented), and those which for the speaker are objects of world knowledge. The reference module assumes that the system contains no world knowledge aside from the knowledge acquired through the current dialogue, the set of possible plans it is programmed to recognize, and the type hierarchy and rules. There is no data bank of names, places, and events, for example.

The algorithm for reference resolution is based on an algorithm proposed by Elaine Rich and Susann LuperFoy [Rich and Luperfoy, 1988] which resolves pronomal anaphors. It has been extended to resolve definite descriptions as well. Eventually, it should resolve event descriptions and indexicals referring to events. It works under the assumption that no one theory successfully accounts for all of reference. Rather, several partial theories coded as separate modules are used (ideally in parallel) and the final outcome is the result of combining their individual results.

Currently, there are only two modules implemented:

- *REF-RECENCY* - proposes as possible antecedents all of the items on the history list, giving more recent items higher ratings.
- *REF-TYPE-CONSTR* - evaluates all items proposed, significantly reducing ratings of those items with types incompatible with type of the expression whose antecedent is sought.

Other appropriate modules which have not been implemented include the following:

- *REF-FOCUS* - evaluates all items proposed, significantly increasing the rating of that item which is the focus of the sentence. (Unimplemented)
- *PLANS* - proposes items that play roles in the current plans being enacted. (Unimplemented)
- *NUMBER* and *GENDER* - evaluate all items proposed and give low evaluation scores to candidates whose number and gender are not compatible with the anaphor. (Unimplemented)

The task of coordinating all the modules to arrive at a list of possible objects and associated scores for a particular semantic item is the job of the handler. The handler chooses a subset of the modules determined by the item's category type. It calls each of these modules in turn and collects proposals. After receiving a set of proposals from a module, each of the other modules are called to evaluate each proposal of the set. The end result is a numerical ranking of all of the proposals of all of the modules. The handler then invokes a procedure which determines whether there is enough evidence (whether the numerical ranking is high enough) to assign the highest ranked proposal as the object referred to by the item. If not, the handler simply returns nil.

The reference module works according to the following algorithm:

For each item on the blackboard matching either PAT1 or PAT2 (defined below), if the item is a noun phrase:

1. If it is *I* or *YOU*, then consult the blackboard to determine who is the speaker is, and assign the noun phrase the appropriate object: *USER* or *SYSTEM*.
2. If it is an indefinite noun phrase, create an object to correspond to it from its logical form.
3. If it is a definite noun phrase, call the handler to find the best object the modules can suggest. If this object does not have a high enough score associated with it, then create an object to correspond to the noun phrase from its logical form. Otherwise, assign this object to the noun phrase.
4. If it is a pronoun, call the handler to find the best object the modules can suggest. If this object does not have a high enough score associated with it, then assign :NULL as the antecedent and lower the rank of the current hypothesis. Note that the current algorithm for the reference determiner is deterministic. That is, it does not return multiple references for a single expression. Furthermore, it does not return an object if it is considered unacceptable, i.e., it has a rating below the acceptability threshold.

In the case where the item matching PAT1 or PAT2 is a sentence, the reference module, as it is currently implemented, simply returns a new event object. Eventually, expressions

such as *THIS*, *THAT*, and *THE CRASHING OF THE STOCK MARKET* will be assigned appropriate event objects by the modules as well.

Interface with the Blackboard System

The interpreter processes only logical forms posted to the blackboard that correspond to the full sentences spoken in the dialog. No embedded expressions are processed separately. However, it posts to the blackboard a list of possible conceptual interpretations for every embedded logical form corresponding to a noun phrase or sentence within the outermost structure that it is processing. If two or more interpretations are posted for a given logical form, the blackboard system is required to create a corresponding number of hypotheses, each with its own history list (and later, focus). Currently, more than one interpretation is arrived at only if the logical form contains ambiguous words. Since the reference determiner is deterministic, ambiguity of reference is not acknowledged and does not lead to separate hypotheses. The reference algorithm is set up, however, to be easily modified to return more than one object subject. For example, if several objects receive a high rating as the antecedent of an expression, then the algorithm can be modified to return them all to the interpreter, which in turn could return a conceptual interpretation corresponding to each of them, causing the blackboard system to create several hypotheses.

In addition to causing the creation of new hypotheses, the reference module can lower the ratings of current hypotheses. Currently, the only reasons it ever has to lower a rating is (1) the sentence is meaningless (type restrictions are violated), and (2) there is no object to which a particular pronoun can refer.

6.4.4 Example

The following is an example of a pattern declaration by the reference analysis module. In the first example, the pattern of interest is any entry whose syntax is a sentence structure and whose semantic slot is defined. In the second example, the pattern of interest is any entry whose syntax is a noun phrase structure and whose semantic slot is defined.

```
(BBDeclarePattern 'generator
                  '(entry %e (Syn (S %sent (MOOD (OR 'decl 'inter))))
                             (Sem %ss))
                  'Reference-Analysis)

(BBDeclarePattern 'generator
                  '(entry %e (Syn (NP %np)) (Sem %ss))
                  'Reference-Analysis)
```

The following is an example of output:

```
(BBDefineSlotValue 'E1 'REF (([INTEREST-EVENT-1] 8 8)) Context)

(BBDefineSlotValue 'E7 'REF (([CRASH29] 10 10)) Context)
```

where [INTEREST-EVENT-1] and [CRASH29] are RHET objects which stand for the event "I am interested in..." and the object "the stock market crash of 1929".

6.5 Speech Acts Module

This module generates speech act interpretations of utterances, by the method of [Hinkelman and Allen, 1989]. Very briefly, the goal of speech act interpretation is to infer the intentions a speaker has in making a particular utterance. For instance, the agent may intend to request, promise, greet, or perform some other action that can be done with words. The module uses lexical, syntactic, logical form, and reference analyses to provide clues to these interpretations. It then generates action descriptions for the utterance, interpreting it as a set of possible speech acts. This module will be referred to as the linguistic module, because it uses a compositional process similar to parsing to generate the speech act interpretations. It requires for operation both a set of interpretation rules and a set of action definitions.

The module is invoked by the blackboard on only a very simple pattern of input, due to restrictions in the discourse system's pattern specification language. The pattern specifies any utterance or part, with syntactic unit S (clause), which has already been assigned some logical form and knowledge base structure. It is shown below.

```
(BBDeclarePattern 'GENERATOR
  '(ENTRY %id (SYN (S %s) (SEM %sem) (REF %ref)))
  'SSA)
```

Some interpretations can be done with much less information, but requiring SEM and REF values ensures availability of all information that might be needed. When it is invoked, the module translates the discourse system's representation of the structure into an internal representation. This representation can then be matched against the speech act interpretation rules.

The module's internal representation of utterances is a LISP list, consisting of a category symbol followed by any number of slot/filler pairs. A category is just the TYPE value from a discourse system tuple, whether a syntactic category or feature like NP, a logical form class like Capable, or a knowledge base type like inform-act. A slot/filler pair is also a list, consisting of a slotname followed by a word or some other value, or a (category (slot filler) ...) structure. Names of Rhet objects have square brackets. Here is the sentence "Can you go to the store?":

```
(setq s1 '(s  (mood  y-n-q)
              (voice  act)
              (subj (np  (pro  you)
                         (sem  h1)
                         (ref  [H])))
              (auxs  can)
              (main-v  go)
              (tense  pres)
              (mods  (pp (prep  to)
                         (pobj  (np (det  the)
                                    (head  store)
                                    (sem  (STORE (id sto1)
                                                 (num  1)
                                                 (gen  n)))
                                    (ref  [store7])) )))
              (sem (CAPABLE (time  pres)
                            (agent  h1)
                            (theme  (GO (to-loc sto1)))))
              (ref  [able1])))
```

The linguistic module is based on a simple bottom-up parser. The parser takes as input a sentence representation, as shown above, and a set of rules. Here is a small set of rules:

```
((\$) (T-SPEECHACT (R-AGENT [S])))

((\$) (T-SPEECHACT (R-HEARER [H])))

; please signals a request
((\$ (ADV PLEASE))
 (T-REQUEST (R-OBJECT (V OBJ REF)) )
)


; "can you....?" may be a request or some other act
((S (AUXS /MODALS) (MOOD Y-N-Q)
    (VOICE ACT) (SUBJ (NP (PRO YOU)))
 )
 (T-REQUEST (R-OBJECT (V OBJ REF)))
 (T-SPEECHACT)
)

; a yes-no question may be a yes-no question or some other act
((S (MOOD Y-N-Q))
 (T-ASK (R-PROP (V REF)))
 (T-SPEECHACT)
)

(setq /MODALS '(CAN COULD WOULD WILL MIGHT)
```

The rules use the internal representation discussed above, with a few extensions. They are lists, containing a left hand side followed by all of the possible interpretations. The left hand side of the first two rules consist simply of $, a wildcard matching any category. They thus match any syntactic unit, and simply fill in the speaker and hearer. [S] and [H] should already be bound using the BB functions retrieving speaker and hearer.

The third rule says that any syntactic unit to which the adverb "please" is attached is a request. The requested object is found in the object role of the sentence's knowledge base interpretation, using the value function. All lists beginning with V are interpreted as the value function: the remainder of the list is a series of slot names, which the value function uses to retrieve information from the utterance. It simply uses the slot names (from right to left!) to trace down into the utterance's structure, and returns the contents of the deepest one, filling out the new structure being created.

The fourth and fifth patterns are more complicated, and have more than one speech act interpretation. For a given domain, such rules could be weighted according to their predictive power, and the output given corresponding weights on the blackboard. Atoms beginning with / are names of disjunctive lists, so rule four matches sentences containing any of the five modal auxiliaries listed in /MODALS.

For a rule to match a structure, the category of the left hand side must be the category of the structure. Each of the slots in the left hand side must be present in structure too, with the same value. The structure may have extra, unmatched slots. We have already discussed the role of wildcards and disjunction. The utterance structure given above matches the wildcard in rules one and two, yielding the first two interpretations below. It does not have an ADV slot containing "please", so it fails to match the third rule. It has the correct

category and mood for the fourth rule, one of the list of modal verbs, and a subject that matches recursively. This match yields two possible interpretations, with the object of the request the object of [able1], [go881]. The final match is a simple match on sentence mood.

```
((T-SPEECHACT (R-AGENT [S])))

((T-SPEECHACT (R-HEARER [H])))

((T-REQUEST (R-OBJECT [go881]))
 (T-SPEECHACT))

((T-ASK [able1])
 (T-SPEECHACT))
```

Those are the four sets of top-level matches for our example sentence and rule set. However, the pattern-matcher is based on a bottom-up parser and therefore performs this whole process at each level. The parser descends the sentence representation recursively, then backs out, attempting to apply the rule set at each level. If a rule matches, the corresponding interpretation is generated. This partial interpretation is merged with any others before backing out to the next level, and the corresponding action instance in the knowledge base is created. The merge operation is to take the cross product of the sets, and combine the interpretations in the resulting sets. The combining is unification or graph matching, in which categories intersect, slots union, and slot values intersect. For the rules and utterance given above, these are the interpretations:

```
(T-REQUEST (R-AGENT [S])
           (R-HEARER [H])
           (R-OBJECT [go881]))

(T-ASK (R-PROP [able1])
       (R-AGENT [S])
       (R-HEARER [H]))

(T-SPEECHACT (R-AGENT [S])
             (R-HEARER [H]))
```

The set containing both Request and Ask interpretations cannot undergo combination, and is eliminated.

The linguistic module has output both to the knowledge representation system and to the blackboard. Each interpretation that is generated by the merge is translated into the knowledge representation, creating an action instance of the appropriate speech act type with variable bindings. The names of these knowledge base interpretations (with equal weights) for a structure is returned to the blackboard in the SAs slot.

```
(BBDefineSlotValue utterance 'SAs value-list context)
```

The world knowledge base is managed by the Rhetorical knowledge representation system [Miller and Allen, 1989]. Rhet is a Horne-clause theorem prover supporting not only forward and backward chaining, but also advanced features such as structured types, reasoning about equality, various proof modes, the Tempos time reasoner, and a hierarchy of belief spaces. The discourse system is loadable in Rhet, making Rhet's programmatic interface available to the modules with no restrictions. The linguistic module shares the knowledge base with implicature, reference, plan reasoning and other high-level modules,

and all of these modules make demands on the plan hierarchy stored there. Rhet objects may be placed on the blackboard but are uninterpreted there, and vice-versa.

Speech act interpretations are created in individual hypothesis spaces in Rhet, where retraction is cheap and further reasoning in an inherited context can be done. The knowledge base contains the type definitions for the speech acts. They use the structured type system, which provides inheritance of roles and initializations. The initializations serve associate values with certain functions of the type, allowing (as yet uninterpreted) sets of propositions to be identified as preconditions, effects, etc. of any action of this type. The definitions supplied by the speech acts module assume that the reference module has already defined a type T-Plan, of which speech acts are children.

```
(Define-Subtype 'T-Language 'T-U)
(Define-Instance [English] 'T-Language)

(Define-Subtype 'T-SpeechAct 'T-Plan
        :Roles '((R-hearer T-Human) (R-Language T-Language))
        :Initializations '([Set-Function-Value [F-Language ?self] [English]]
                           [Set-Function-Value [F-Preconditions ?self]
                                ([Listening [F-Hearer ?self]]
                                 [Noise-Free-Line])]
                           [Set-Function-Value [F-CONSTRAINTS ?self]
                                ([Speaks [F-Agent ?self] [F-Language ?self]]
                                 [Speaks [F-Hearer ?self] [F-Language ?self]])]))

(Define-Functional-Subtype 'T-Request 'T-SpeechAct
        :Roles '((R-object T-Plan))
        :Initializations '([Set-Function-Value [F-EFFECTS ?self]
                                ([Exec [F-Hearer ?self] [F-Object ?self]] ) ]
                           [Set-Function-Value [F-Constraints ?self]
                                ([Able [F-Hearer ?self] [F-Object ?self]] ) ] ))
```

A SpeechAct has the inherited role Agent as well as two of its own. It requires the speaker and hearer to share a specific language, and for the hearer to be listening. A Request more specifically has a requested object, which is an action. A constraint is that the hearer is able to do the action, and the effect is that they actually do. The semantics of these operators is discussed more thoroughly in [Hinkelman, 1989] These structures are used by other high-level modules to determine the agent's intentions and keep track of what happened in the conversation.

6.6 Tense and Aspect Module

6.6.1 General Overview

Temporal information contributes valuable knowledge when processing discourse. Discourse has inherent structure, and temporal reasoning can do much towards discovering this structure. Allen [1987] discusses the idea of 'discourse segments' to represent the structure of discourse. Bonnie Webber has convincingly shown that the temporal information given by tense provides support for the segment theory. Aspect, both grammatical and lexical, also contributes valuable information as do temporal adverbials. The temporal relations between events, determined from these temporal clues, are an important part of the structure of discourse, which reflects the underlying structure of events.

The main purpose of the temporal module is to decide which segments can felicitously be continued and which cannot by analyzing various temporal clues, including the tense and aspect of each clause, as well as temporal adverbials. Certain combinations of tenses allow the current segment to be continued, while others do not. [Allen, 1987] Similarly, temporal discontinuities can arise from some combinations of aspect. These discontinuities indicate a change in segment. Grammatical aspect (e.g. the progressive and perfect) interacts with both tense and lexical aspect.

The temporal module also builds a structure, which reflects the structure of events and situations described in the discourse. This structure is based on Bonnie Webber's Event Situation Structure (ESS). It temporally locates events to each other. However, temporal relations alone are not enough to make a discourse cohesive, and thus the structure also encodes other relations. In fact, strictly temporal relations are a by-product of these other relations, described by Moens and Steedman [Moens and Steedman, 1988] as 'contingency' relations. Temporal reasoning plays a very important part in computing these relations.

The traditional view of a linear 'time-line' is thus augmented with a more complex ontology based on such notions as causation and consequence, and the simple notion of events is replaced by a tri-partite structure. The event structures of Webber and our system are based on this more complex ontology. To construct the ESS, the listener must find the correct place in the evolving structure for each new clause. The temporal clues discussed above play a substantial role in this process, including in particular the anaphoric nature of tense. When the entire discourse has been processed, the ESS represents the listener's conception of the speaker's view of the world.

The information obtained from the tense and lexical aspect as well as these contingency relations allows a disjunction of Allen's [1983] primitive temporal relations to be computed. These thirteen relations describe how two events are temporally located in exact terms, whereas Webber's terminology is only approximate.

Temporal reasoning relies fairly heavily on world knowledge. For instance, which contingency relation holds between events is a function of world knowledge as well as of temporal information. In our system, the crucial world knowledge is requested from the user. That is, queries are issued and the user must directly input the appropriate information.

6.6.2 Background

Reichenbach

Reichenbach [1947] proposed that tense actually consists of three times - event time (ET), speech time (ST), and reference time (RT), and this has for the most part been widely accepted in the literature on temporal reasoning. The ordering of these times is uniquely determined by the tense of a sentence. A tensed clause then, provides two types of information: a description of an event or situation, and a particular configuration of ET, ST, and RT.

Vendler

Vendler [1967] established four aspectual categories. He argued that verbs could be classified as one of these four types, and presented various tests for doing so. For example, one distinguishing characteristic is that time tends to 'move forward' from one sentence to the next if the second sentence contains an accomplishment or achievement predicate, but does not usually move if the sentence contains a stative or an activity predicate.

A Contingency Ontology

Marc Moens and Mark Steedman [1988] suggest that instead of a linear model of time, we actually need a more complex ontology which depends on such concepts as causation, consequence, and 'contingency'. Contingency is not strictly temporal nor causal, although it is related to both of these. It is similar to enablement, although it is intransitive. They introduce the concept of a 'nucleus', which is defined as a culmination or goal event, along

with an associated preparatory process and consequent state. This tri-partite structure of events yields three contingency relations that may exist between events: preparatory, identity, and consequence relations.

Moens and Steedman change Vendler's terminology, but keep his main ideas concerning the classification of verbs. They distinguish events along two dimensions: the first is punctuality versus temporal extension, and the second is whether or not the event is associated with a consequent state or not. This two-way division of events yields four aspectual classes: (1) Culminations, which are considered punctual, and are associated with a consequent state. (2) Points are also punctual, but are not associated with any consequent state. (3) Processes are extended in time and are not accompanied by any consequent state. Finally, (4) Culminated Processes are also extended in time, and do result in a consequent state. These events, which have definite endpoints, are contrasted with states, which extend indefinitely.

They argue that linguistic devices, such as tense, aspect, and adverbials transform, in a systematic way, an entity which is *typically* classified as one aspectual type into another by "coercing" the input (the lexical aspect of the verb) as necessary. These transitions also result from semantic and pragmatic factors.

Webber

Webber suggests that the main task of the listener is to construct an Event Situation Structure (ESS); to do this each clause must be placed in the evolving structure. Each tensed clause is located by its relation to other clauses. The anaphoric nature of tense provides the desired relationships. Tense is interpreted with respect to a temporal referent, either one that has been previously established or one that the clause itself establishes. This temporal referent takes the form of another event in the discourse, and thus tense has the effect of temporally relating events. A tensed clause may either evoke and specify an entity, or simply specify one if it establishes the referent. The ontology of events proposed by Moens and Steedman [1988] provides ways in which a tensed clause can be related to another event/situation; namely, the contingency relations. Each contingency relation also yields a temporal relation. The preparatory relation implies that the event referred to by the tensed clause is before the event/situation that it is related to; the consequence relation implies that the current event occurs afterwards, and the identity relation implies that the two events or situations are cotemporal. Reichenbach's [1947] RT (reference time) is the part of the tensed clause that anaphorically corresponds to an entity whose temporal location is given by its ET (event time).

Bonnie Webber [1988] proposed the Temporal Focus (TF) to model the movement of time through the discourse. The TF is associated with the event that is currently being focussed on temporally. This is the event that is most likely to be anaphorically referenced by the current tensed clause. The TF can and does change throughout the course of the discourse, depending on the relations that hold between events and on the tense of the clauses involved. If the identity relation holds the TF does not move. The consequence relation models the natural forward movement of time, and thus the TF will move forward to become associated with the current clause. The preparatory relation implies that time is moving backwards; thus the TF will move backwards and a new segment will be started, indicating a temporal discontinuity.

Allen

Allen [1984] proposes a temporal logic based on a set of mutually exclusive temporal primitives. These are: BEFORE, EQUAL, MEETS, OVERLAPS, DURING, STARTS, and FINISHES. Together with their inverses, these yield thirteen distinct relationships. These describe the relationship that holds between two temporal intervals. It is unusual to have temporal relationships explicitly given in a discourse, but this scheme allows the ambiguity to be expressed by using a disjunction of the possible primitive

6.6.3 The Temporal Module

The temporal module relies heavily on the ideas of Webber [1988] and Moens and Steedman [1988]. It is assumed that events or situations have a tri-partite structure and are related to each other via one of the contingency relations. Events/situations are classified as states, points, processes, culminations, or culminated processes. It is important to note that events are classified as instantaneous as well as extended in time. Thus, the temporal primitive associated with events and situations is the instant and not the interval. There is a point to distinguishing events along the dimension of atomicity. Even though one can always break an event into smaller pieces, there are certain events that we intuitively consider instantaneous, and our reasoning and our ontology reflects that. From the contingency relations, and the temporal information gained from tense, aspect, and temporal adverbials, a disjunction of Allen's [Allen, 1984] temporal primitives is computed. Both the contingency relation and the temporal relation(s) are encoded in the event structure.

The idea of a temporal focus (TF) is adopted from Webber's model also. The TF, the temporal analogue of Grosz and Sidner's discourse focus (DF), models the movement of time through a discourse. Each proposed segment has a TF, which is always associated with the event on which the listener is currently focussed. This event changes as the discourse is processed and as the movement of time through the discourse is discovered. As the discourse moves to talk about events that are consequences of the current TF, time and thus the TF move forward. Similarly, time and the TF move backwards as preparatory events are discussed. The TF does not change if the current event is related via the identity relation.

Unlike Webber's model, all events in this last category are candidates for future anaphoric relations. That is, if the current tensed clause is not anaphorically related to the TF, other events that are related to the TF via the identity relation are examined. These events are indirectly related to the TF. If none of these provide temporal referents, then a new segment is started.

Input

The temporal module examines tense, aspect, and temporal adverbials in order to compute ratings concerning which segment, if any, the current clause should continue. The aspectual perspective of the clause is computed by looking at the lexical aspect and any linguistic clues present, including grammatical aspect and which type of adverbials are present. The lexical aspect becomes the aspectual perspective by default, but the linguistic clues may coerce it into something else. For instance, the progressive must take a process as its argument, while the perfective forces its input to be a culmination. Similarly, durational adverbials may only be used with processes or culminated processes and rate adverbials cannot be used with states or points.

The transitions between aspects are in part controlled by world knowledge. Currently, this world knowledge is obtained by inputting the lexical aspect "by hand". That is, it is not looked up in the lexicon, but rather directly input into the temporal module. From this point on, what is referred to simply as 'aspect' is actually the aspectual perspective of a clause.

For now, the module queries the user to see which contingency relation(s) hold between the clause and each segment that has been proposed at the time the clause is processed. If there is not one between the TF and the current event, then the other events indirectly related to the TF are tried in the reverse order that they appear in the discourse. This information is then considered, along with the interaction of tense and aspect and a rating is computed, reflecting the likelihood that the current clause can felicitously continue each segment. An event structure that reflects the structure of the discourse is built as well.

Ratings

The temporal module computes a rating, indicating how likely it is that the current event or situation should be incorporated into each hypothesized segment. It examines the

TF and if necessary, any associated events of each segment in turn, and computes a rating determined by the interaction of the tenses, aspects, temporal relations, and contingency relations involved. A confidence level is also computed, resulting in a two-level rating scheme. The confidence level, as expected, reflects how strongly the module believes in its rating. Some temporal information is highly indicative, while other information is not as reliable in the absence of any supporting (nontemporal) information. For instance, while usually switching from the present tense to the past tense implies a segment change, there are many instances when it does not, especially in a non-narrative domain such as ours. Thus, the confidence level associated with this rating would be relatively low.

The algorithm used by the module is the following:

For each segment:

1. Check the contingency relation between the current event and the TF. If none exist, go to step 2. If it is a preparatory relation, output a low rating with a high confidence level. Else, go to step 3.

2. Check (in reverse order of appearance in discourse) the events indirectly associated with the TF for a contingency relation with the current event. If none exist, output a low rating with a high confidence level. Otherwise, mark the first one with which a relation does hold. If it is a preparatory relation, output a low rating with a high confidence level.

3. At this point, either an identity or a consequence relation holds between the current event and one of the events associated with the TF. Check the temporal relations between these two events. If the disjunction is null, output a low rating with a medium-high confidence level.

4. We now look at the tense sequence of the segment and at the aspects. These each give a separate rating, and we take the average. Similarly, we take the average of the confidence ratings.

The tense sequence of each proposed segment is analyzed and any possible temporal discontinuities are recorded. For instance, switching from simple past to repeated use (more than one) of the past perfect implies that time is moving backwards, and thus a segment change is suggested.

The aspect sequence, and the appropriate tenses, are also analyzed and again, any temporal discontinuities are recorded. Tense is taken into account here because the interactions yield more information than just looking at aspect alone. For instance, if the event associated with the TF is a process and the tense switches from past (at the TF) to present (at the current clause), a segment change is suggested. The aspect implies that time moves backwards or stays where it is, while the tense implies that it is moving forward. The confidence levels associated with these ratings are not very high, and are even lower if the event we are trying to anaphorically relate our tensed clause to is only indirectly associated with the TF.

The Event Structure

The temporal module builds an event structure very similar to Webber's ESS. The structure is built as the discourse is processed, with each event/situation fit into its appropriate place. Once all the other modules have returned their results and it has been decided which segment, if any, the event will be attached to, the temporal module incorporates the event into the appropriate segment. Or, alternatively, it starts a new segment with the event. It identifies the event or situation to which the current tensed clause is anaphorically related. This is usually the event directly associated with the TF, but it may be indirectly associated, as mentioned above. The event is then inserted into the structure by attaching it to this event along with the appropriate contingency and temporal relations.

Webber uses a simple ("before", "after", or "at-the-same-time") temporal scheme in her ESS. Ours is a disjunction of Allen's [1984] thirteen primitives and thus is much more exact. We also explicitly encode the contingency relation. Since we allow for instantaneous events as well as those that are extended in time some constraints must be employed to generalize Allen's scheme. OVERLAP requires that both times be extended, while the second time period of DURING, STARTS, and FINISHES must be extended. EQUAL requires that both time periods be instantaneous or both hold over time intervals.

The completed event structure reflects the underlying structure of events by encoding the relations that exist both between the events within a segment and between the segments themselves. The events within a segment are fully connected in that each event is related to another event in the structure by both a contingency relation and a temporal relation. If the temporal relation between two events that are not contingently related is known, this can be encoded in the event structure. Since the two events will necessarily be in different segments, this has the effect of temporally relating entire segments. Segments may also be connected by a preparatory relation. This indicates that one segment is actually a sub-segment of the other.

The movement of time can be extracted from the event structure by tracing through it to see how the events are related.

States

Moens and Steedman believe that the contingency relations do not apply to states, as it is based on an ontology of *events*. However, assigning states aspectual classes is helpful and can contribute temporal information when processing discourse in much the same way that classifying event verbs does. In the temporal module, states are distinguished by whether or not they are associated with any obvious consequences. These can be further differentiated into states that will reach an end point or goal in the normal course of events (telic), and states that will not. Thus, any state associated with consequences can felicitously have other events/situations anaphorically related to it via the consequent relation. It can be assumed that telic states do not hold at any time located past its end point, and, barring any unusual circumstances that must be explicitly mentioned, its goal or end point will be reached. States are also differentiated along the dimension of required resources, as Nakhimovsky [1988] suggests. States can also be transformed to another aspect, as events are, by the presence of certain linguistic clues. This classification of states seems to be especially important for our domain, where a great many sentences are aspectually classified as states.

6.6.4 Future Work

The temporal module of the discourse system is an evolving entity. There are many additions planned. Temporal adverbials provide more information than is currently extracted. Any time an adverbial is used to connect a main and modifying clause the information should be used. These could be rate, frequency, or durational adverbs. Also, many connectives can at least narrow down the possible contingency and temporal relations, even if they cannot uniquely identify which one holds.

Another useful piece of information is whether or not actual time [Passonneau, 1988] is associated with an event. If it is not, there is no reason to put the event into the event structure. Events and situations that are mentioned later in the discourse cannot temporally reference events that do not occupy actual time, and thus there is no reason to have them in the event structure. Certainly any temporal information that can be extracted should be, but they do not belong in the actual event structure.

Nakhimovsky [1988] proposes that there are three kinds of world knowledge which we possess and which facilitate our ability to understand discourse. We have already incorporated compositional and aspectual knowledge into the module, but durational knowledge still needs to be added. Any known durational information and knowledge concerning time

scales will be put into the lexicon so the temporal reasoner can access it. If in the discourse a switch is made to an event that differs by more than one time scale unit, a temporal discontinuity is indicated. This knowledge will allow us to reason in a more useful manner about if, how, and why an event ended - useful pieces of information. Related to durational informations is the knowledge concerning an event's resources. This can determine how fast an event or situations is likely to progress as well as whether an event has a built in end point that will be reached in the natural course of things. As much specific domain knowledge available should also be put into the lexicon.

Perhaps the most important extension to the implementation of this module is integrating temporal reasoning with planning. While the temporal relations are quite important, they alone are not enough to make a discourse cohesive. This is indicated by the importance of the contingency relations. Temporal reasoning has depended heavily on what has been referred to as world knowledge to fill some of its holes. Currently, this limitation of temporal reasoning is overcome by asking the user various questions and therefore directly inputting the necessary knowledge. Planning provides the information needed to tell how the events are related in ways other than temporally. Planning systems have traditionally provided 'drough outlines' of events and activities, the execution of which realizes some goal. Planning systems can provide expectations of events to come. What they have been unable to do is go 'backwards in time'. Temporal reasoning is important here and can interact beneficially with planning. Temporal reasoning can provide the knowledge that tells where in the plan one is, and where to look for the next event.

6.6.5 An Example

Here is an example of a pattern declared by the tense and aspect module:

```
(BBDeclarePattern 'GENERATOR
                  '(S %id (TENSE %tense) (ASPECT %aspect))
                  'temp-call)
```

When the first sentence is analyzed, the following facts are posted:

```
(ASPECT S1 ((:STATE 10 10)))
(TENSE  S1 ((:PRES 10 10)))
```

so that the object S1 of type S (sentence) matches the previous pattern. The module call:

```
(TEMP-CALL S1 HYP4711)
```

is then scheduled and later executed. The tense and aspect module sends to the blackboard the following facts, as result of it processing:

```
(BBDefineSlotValue 'S1 'ASP-PERSP '((:STATE 10 10)) 'HYP4711)
(BBDefineSlotValue 'S1 'CONTREL '((:ID 8 8)) 'HYP4711)
(BBDefineSlotValue 'S1 'TEMPRELS '((NIL 8 8)) 'HYP4711)
```

Chapter 7

DETAILED TRACE OF AN ANALYSIS

This section present a partial but more detailed trace of the processing of sentence (3) on page 8.

The lexical module takes this sentence as input and returns information about each word. In this case some 14 lexical objects are created along with 14 ENTRIES. Each ENTRY has a syntactic slot whose value is the corresponding lexical object.

For the first word, "I" an object (W1) of type "lexical" is declared to the chart.

```
(TYPE LEXICAL W1)
```

W1 is further defined to have the following slot/value pairs.

```
(W1 (WORD "I")
    (ROOT "I")
    (CAT PRON)
    (NUM 1S))
```

An object (EW1) of type ENTRY is then declared

```
(TYPE ENTRY EW1)
```

and defined to have a syntactic slot with value W1.

```
(EW1 (SYN W1)
     (BEG 1)
     (END 2))
```

and beginning before the first word and ending before the second word of the sentence. Later, as larger syntactic and semantic objects are constructed on the chart, entries will span several words.

Here the declaring (or posting) of a syntactic slot for EW1 triggers the pattern matching mechanism to schedule a call to the syntactic module. The function call

```
(SYN-PARSE EW1 CHART)
```

is added to the run-queue of the chart. Recall that the chart is just a specially named hypothesis.

When this call is executed the syntactic module created two new objects, NP1 and E2. Each object has one slot defined by the syntactic module during this execution.

```
(TYPE NP NP1)
(NP1 (HEAD EW1))    The head of this noun phrase is the syntactic
                          structure EW1. In this case a simple pronoun.
(TYPE ENTRY E2)
(E2 (SYN NP1))
```

The function call

```
(SEM-INTERPRET E2 CHART)
```

is now added to the run-queue of the chart. This is the result of matching an entry whose syntactic slot value is of type NP.

When the scheduler executes this function call, the semantic module declares a new object and a slot/value pair for it.

```
(TYPE SEMANTIC SEMNP1)
(SEMNP1 (SPEC I))
```

A semantic slot/value is now added to E2, which then looks like this.

```
(E2 (TYPE ENTRY)
    (SYN NP1)
    (SEM SEMNP1)
    (BEG 1)
    (END 2))
```

The object E2 is now an ENTRY, of length one word, that has both a syntactic and a semantic interpretation. The value of NP1 and SEMNP1, and ultimately of W1, are stored as slot/value pairs in the hypothesis named CHART. It is the task of the Blackboard monitor to be able to assemble the distributed values of an object and return that object on demand.

Snapshot of the blackboard entries pertaining to E2.

```
(E2 (TYPE ENTRY)
    (SYN NP1)
    (SEM SEMNP1)
    (BEG 1)
    (END 2))
(SEMNP1 (TYPE SEMANTIC)
```

```
        (SPEC ("I" EW1)
              (BEG 1)
              (END 2))
(NP1 (TYPE NP)
     (HEAD EW1)
     (BEG 1)
     (END 2)
(EW1 (TYPE ENTRY)
     (SYN W1)
     (BEG 1)
     (END 2))
(W1 (TYPE LEXICAL)
    (WORD "I")
    (ROOT "I")
    (CAT PRON)
    (NUM 1S)
    (BEG 1)
    (END 2))
```

The ENTRY structures are used to combine and organize basic syntactic and semantic structures (objects) into ever larger and more complex structures (objects).

As the processing of the example sentence continues, each lexical structure (word) will have a corresponding ENTRY created. Each entry will contain only a syntactic slot, since no semantic interpretation of individual words will be found. The exception will be "1929" the last word of the sentence. A new noun phrase will be created after processing similar to the above example. The chart will now contain the following additional information.

```
(E9 (TYPE ENTRY)
    (SYN NP2)
    (SEM SEMNP2)
    (BEG 13)
    (END 14))
(SEMNP2 (TYPE SEMANTIC)
        (SPEC NAME)
        (VALUE 1929)
        (BEG 14)
        (END 15))
(NP2 (TYPE NP)
     (NAME EW14)
     (BEG 14)
         (END 15))
```

...and similarly the data for EW14 and W14.

Now the execution of the call generated by the posting of the syntactic slot (SYN NP) for E9 will result in the syntactic module combining previous syntactic structures into larger structures. In this case E9 (the ENTRY whose syntactic structure is the NP "1929") and EW13 (the ENTRY whose syntactic structure is the lexical item W13 .."of").

```
(PP13 (TYPE PP)
      (PREP EW13)
      (POBJ E9)
```

```
          (SYN E18)
          (BEG 13)
          (END 15))
     (E18 (TYPE ENTRY)
          (SYN PP13)
          (BEG 13)
          (END 15))
...etc
```

The order of processing of any of the calls on the runqueue is asynchronous. The syntactic processing must take this into account and be able to construct syntactic structures in no particular order. The processing is bottom-up since the processing modules never propose objects but rather propose interpretations of objects that already exist.

Below is the representation of the noun phrase "some information about the stock market crash of 1929".

```
(E36 (TYPE ENTRY)
     (SYN NP22)
     (SEM SEMNP22)
     (BEG 6)
     (END 15)
(NP22 (TYPE NP)
      (DET EW6)
      (HEAD EW7)
      (MODS E35)
(SEMNP22 (SPEC SOME)
         (THEME SEMNP18)
(E35 ("about the stock market crash of 1929"))
..
(SEMNP18 ("the stock market crash of 1929"))
....etc etc to the terminal word entries.
```

The quoted strings for E35 and SEMNP18 should be taken to mean all the data that would be posted to the CHART in the construction and interpretation of those objects. All other intermediate objects that contribute to the ENTRY E36 have been omitted.

Processing at this point now begins to generate objects of type SENTENCE. This comes about from the syntactic modules combining of verbs with the previous declared phrases. Two final examples are given, first of the NP entry E35 and the verb getting and then a snapshot of the final representation of the entire sentence on the CHART.

```
(ES42 (TYPE SENTENCE)
      (MOOD "ING")
      (SUBJ E2 )...(the pronoun "I")
      (MAINV EW5)..(the verb "getting")
      (OBJ E35))...(the phrases "some information...")
```

The object ES42 is then incorporated into PP22, the prepositional phrase "in getting some information about...". The final representation of the sentence is then E50.

```
(E50 (TYPE ENTRY)
             (SYN S1)
             (SEM SEMS1)
             (BEG 1)
             (END 15))
        (S1 (TYPE SENTENCE)
            (MOOD DECLARATIVE)
            (VOICE PASSIVE)
            (TENSE PRESENT)
            (ASPECT STATE)
            (AUX EW2)....(the verb "am")
            (MAINV EW3)..(the verb "interested")
            (COMPL E38)..( the pp "in getting some..")
        (SEMS1 (TYPE SEMANTIC)
               (SPEC PRESENT)
               (EXPERIENCER SEMNP1)
               (THEME SEMNP25))..(the np "some information...")
```

All information about S1, our example sentence, can be found by decomposing the object E50. Below is the obvious embedded representation. Note that only the S1 - EW2 - W1 composition and the SEMS1 - SEMNP1 composition are fully represented. This representation is only illustrative and not how information is actually stored or passed to processing modules.

```
(E50 (TYPE ENTRY)
     (SYN (S1 (TYPE SENTENCE)
              (MOOD DECLARATIVE)
              (VOICE PASSIVE)
              (TENSE PRESENT)
              (ASPECT STATE)
              (AUX (EW2 (TYPE ENTRY)
                        (SYN (W1 (TYPE LEX)
                                 (WORD "am")
                                 (ROOT "be")
                                 (CAT AUX)
                                 (NUM 1S)
                                 (TENSE PRES))))
              (MAINV EW3)  ;; the verb "interested"
              (COMPL E38)  ;; the pp "in getting some.."
     (SEM (SEMS1 (TYPE SEMANTIC)
                 (SPEC PRESENT)
                 (EXPERIENCER (SEMNP1 (TYPE SEMANTIC)
                                      (SPEC I)
                                      (BEG 1)
                                      (END 2)))
                 (THEME SEMNP25))))
```

The slot/vaue pairs labeled TENSE and ASPECT for object S1 are supplied by the temporal module. This then concludes the syntactic and semantic (CHART) processing of the sentence. The higher level objects (entry, syn , sem, sentence and np) are now copied to a completely public part of the blackboard architecture and completely asynchronous processing continues at the discourse level.

The reposting of these objects allows the stimulus patterns posted by the various modules (reference, plans, speech-act etc) to be matched against. This creates more function calls for the hypothesis' run-queues. Recall that each hypothesis is processed in the context of the segment to which it might attach.

Consider the execution of the top call from the following run-queue. The run-queue for the hypothesis associated with the current segment, assume this to be the only active segment, is as follows.

```
Scheduled Calls of Hypothesis HYP4711
     (REFERENCE E4 HYP4711)
     (SSA E4 HYP4711)
     (TEMP-CALL S1 HYP4711)
     (REFERENCE E1 HYP4711)
     (SSA E1 HYP4711)
```

The reference module, in this case finds two possible referents for ES42. The score of (9 9) indicates that this referent is preferred over of the slot value scored (4 8). After the creation of the two new hypothesis, they differ only by this slot value. Each of the new hypothesis' inherits all the objects of HYP4711 and the each inherits the run-queue of HYP4711. The following shows the informational display during this processing.

```
Reference Call: Trying to split hypothesis
Splitting Hypothesis HYP4711
     ((D-REF ES42 ((D-REF-ES42-ONE 9 9))))
     ((D-REF ES42 ((D-REF-ES42-TWO 4 8))))

Creating NEW HYPOTHESIS  HYP4712
in Segment SEG4402  as daughter of Hypothesis HYP4711

Defining Slot Value (ES42 (D-REF D-REF-ES42-ONE ))

Creating NEW HYPOTHESIS  HYP4713
in Segment SEG4402  as daughter of Hypothesis HYP4711

Defining Slot Value (ES42 (D-REF D-REF-ES42-TWO ))


List of ALL Scheduled CALLS in ALL the Leaf Hypotheses

Scheduled Calls of Hypothesis HYP4713
     (SSA E4 HYP4711)
     (TEMP-CALL S1 HYP4711)
     (REFERENCE E1 HYP4711)
     (SSA E1 HYP4711)

Scheduled Calls of Hypothesis HYP4712
     (SSA E4 HYP4711)
     (TEMP-CALL S1 HYP4711)
     (REFERENCE E1 HYP4711)
     (SSA E1 HYP4711)
```

Although these calls on the two run-queues have HYP4711 as arguments, when these calls are actually executed the named hypothesis will be updated to HYP4712 or HYP4713

as appropriate. Recall that the highest rated leaf hypothesis is selected for further processing and then a particular call is selected from the run-queue of that hypothesis. Further processing of HYP4712, for example, could lead to more splitting of the hypothesis. Again when the calls on the run-queue that refer to HYP4711 are executed they named hypothesis that is passed to the module is the most current (leaf) hypothesis. This insures that all information posted since the call was placed on the original run-queue will be available to the processing module.

In this particular example, HYP4712 is the Hypothesis that gets processed further. The expansion by the call to the Reference module leads to another splitting of Hypothesis due to two possible definite references. The calls in HYP4750 are selected for execution since the referents yield a highly rated Hypothesis. This is finally resolved by the Speech Acts module finding a speech-act interpretation for object E4. The BB Monitor is now able to select a complete interpretation for acceptance. The trace of this activity follows. The other executing calls are suppressed here for they do not add information but only effects the ratings.

```
Splitting Hypothesis HYP4712
        ((D-REF S1 ((D-REF-S1-ONE 9 9))))
        ((D-REF S1 ((D-REF-S1-TWO 4 8))))


Creating NEW HYPOTHESIS  HYP4749
in Segment SEG4402  as daughter of Hypothesis HYP4712

Defining Slot Value (D-REF S1 ((D-REF-S1-ONE 9 9)))

Creating NEW HYPOTHESIS  HYP4750
in Segment SEG4402  as daughter of Hypothesis HYP4712

Defining Slot Value (D-REF S1 ((D-REF-S1-TWO 4 8)))


List of ALL Scheduled CALLS in ALL the Leaf Hypotheses

Scheduled Calls of Hypothesis HYP4750
     (SSA E1 HYP4711)
     (IMPL E4 HYP4713)

Scheduled Calls of Hypothesis HYP4749
     (SSA E1 HYP4711)
     (IMPL E4 HYP4713)

No Calls Scheduled for Hypothesis HYP4748
                       Hypothesis HYP4747
                       Hypothesis HYP4404


Executing Module CALL (IMPL E4 HYP4750)

Defining Slot Value (SA E4 (([INFORM8] 9 10)))


Selecting the Hypothesis HYP4750  (rating 125)
Leaf Hypotheses: (HYP4750 HYP4749 HYP4748 HYP4747 HYP4404)
```

```
GLOBAL STATUS (before HYP4750 is grounded to SEG4402 via PERM-HYP4403)

Number of Hypotheses: 8
List of Hypotheses: (HYP4750 HYP4749 HYP4748 HYP4747 HYP4713 HYP4712 HYP4711 HYP 4404)
Number of Leaf Hypotheses: 5
List of Leaf Hypotheses: (HYP4750 HYP4749 HYP4748 HYP4747 HYP4404)
Number of Segments: 2
List of Segments: (SEGMENT0 SEG4402)
List of Active Segments: (SEGMENT0 SEG4402)
List of Active Leaf Segments: (SEG4402)
List of Permanent Hypotheses: (PERM-HYP4401 PERM-HYP4403)
Number of Patterns: 8
Number of Facts: 358
```

This ends the processing of the utterance and the Black Board is now cleared of all non permanent hypothesis structures.

Bibliography

[Allen, 1984] J. F. Allen, "Towards a general theory of time and action," *Artificial Intelligence*, 23(2):123–154, 1984.

[Allen and Koomen, 1983] James F. Allen and Johannes A. Koomen, "Planning Using a Temporal World Model," In *Proceedings 8th IJCAI*, pages 741–747, 1983.

[Allen, 1987] J.F. Allen, *Natural Language Understanding*, Benjamin Cummings Publishing Co., Inc, 1987.

[Allen and Perrault, 1980] J.F. Allen and C.R. Perrault, "Analyzing intention in utterances," *Artificial Intelligence*, 15(3):143–178, 1980.

[Appelt, 1985] D.E. Appelt, *Studies in Natural Language Processing: Planning English Sentences*, Cambridge U. Press, 1985.

[Cohen and Perrault, 1979] P.R Cohen and C.R. Perrault, "Elements of a plan-based theory of speech acts," *Cognitive Science*, 3:177–212, 1979.

[Cullingford, 1981] R. Cullingford, *Inside Computer Understanding (ed. R. Shank and C. Riesbeck)*, chapter SAM, pages 75–119, Lawrence Erlbaum, 1981.

[Grosz, 1977] B.J Grosz, "The representation and use of focus in a system for understanding dialogs," In *Proc. IJCA*, pages 67–76, 1977.

[Grosz *et al.*, 1983] B.J. Grosz, A.K. Joshi, and S. Weinstein, "Providing a unified account of definite noun phrases in discourse," In *Proc. ACL*, pages 44–50, 1983.

[Grosz and Sidner, 1986] B.J. Grosz and C. Sidner, "Attention, intention, and the structure of discourse," *Computational Linguistics*, 12(3), 1986.

[Hinkelman, 1989] E Hinkelman, *Linquistic and Pragmatic Constraints on Utterance Interpretation*, PhD thesis, Univ. Rochester, 1989.

[Hinkelman and Allen, 1989] E. Hinkelman and J. Allen, "Two constraints on speech act ambiguity," In *Proc. ACL*, pages 212–219, 1989.

[Kaplan, 1973] R. M. Kaplan, *Natural Language Processing (ed R. Rustin)*, Algorithmics Press, 1973.

[Kay, 1989] M. Kay, *Natural Language and Symbolic Computing*, Morgan-Kaufman, to be published 1989.

[Lesser and Erman, 1977] V. Lesser and C Erman, "A Retrospective View of the Hearsay-II Architecture," In *Proc. Fifth International Joint Conference on Artificial Inteligence*, pages 790–800, 1977.

[Litman and Allen, 1987] D.J. Litman and J.F. Allen, "A plan recognition model for subdialogues in conversations," *Cognitive Science*, 11(2):163–200, 1987.

[Miller and Allen, 1989] B. Miller and J Allen, "The Rhetorical Knowledge Representation System: A User's Manual," Technical Report TR238, Department of Computer Science, University of Rochester, March 1989.

[Moens and Steedman, 1988] M. Moens and M. Steedman, "Temporal Ontology and Temporal Reference," *Computational Linguistics*, 14(2):15–28, 1988.

[Nakhimovsky, 1988] Nakhimovsky, "Aspect, Aspectual Class and the Temporal Structure of Narrative," *Computational Linguistics*, 14(2):29–43, 1988.

[Passonneau, 1988] R. J. Passonneau, "A Computational Model of the Semantics of Tense and Aspect," *Computational Linguistics*, 14(2):61–73, 1988.

[Reichenbach, 1947] H. Reichenbach, *Elements of Symbolic Logic*, Macmillan and Company, 1947.

[Reichman, 1978] R. Reichman, "Conversational Coherence," *Cognitive Science*, 2(4):283–328, 1978.

[Rich and Luperfoy, 1988] E. Rich and S. Luperfoy, "An Architecture for Anaphora Resolution," In *Second Conference on Applied Natural Language Processing*, pages 18–24, 1988.

[Schank, 1975] R.C. Schank, editor, *Conceptual Information Processing*, North-Holland, 1975.

[Sidner, 1983] C. Sidner, *Computational Models of Discourse (eds M. Brady and R.C. Berwick)*, MIT Press, 1983.

[Sussman and McDermott, 1972] G.J. Sussman and D. McDermott, "From Planner to Conniver - a genetic approach," In *Proceedings of the Fall Joint Computer Conference*, 1972.

[Vendler, 1967] Z. Vendler, "Verbs and Times," *Linguistics and Philosophy*, 9:97–121, 1967.

[Webber, 1988] B.L. Webber, "Tense as Discourse Anaphor," *Computational Linguistics*, 14(2):61–73, 1988.

[Wilensky, 1983] R. Wilensky, *Planning and Understanding*, Addison-Wesley, 1983.

[Winograd, 1983] T Winograd, *Language as a Cognitive Process. Vol. 1: Syntax*, Addison-Wesley, 1983.

Appendix A

TRANSCRIPTS OF ACTUAL TEXT

A.1 Introduction

The application domain we have been focusing on consists of dialogues between a library operator and a student looking for books and articles in order to be able to write a paper on a subject assigned to him/her. The library operator has access to a computerized library database to help the student.

A few experimental dialogues were collected to serve as a basis for the discourse project. People taking part in the experiment were given the following instructions:

> The purpose of this experiment is to collect dialogues for subsequent research. Your role in the experiment is to engage in a dialogue with a data base operator who works with Chester - a library data base. The goal throughout your dialogue should be to collect enough references to write an assigned paper.

along with technical instructions to do the experiment. It must be noted that the people involved in the experiment knew that the purpose of the experiment is the study of dialogues. However, the experiment was realistic enough to consider that this prior knowledge did not have a decisive effect on the structure and the content of the dialogues.

A communication protocol was also given: the users had to type the sequence -o- after each turn. This signalled to the data base operator that the user had finished typing and was waiting for a reply. The data base operator was using the same protocol.

Transcripts of each of the six dialogues which have been collected this way are given in the next sections.

A.2 Dialogue 1

Subject: *What were considered the major causes of the stock market crash in 1929? What changes in the workings of the market and in U. S. economic policies resulted in order to prevent a similar disaster in the future? Your paper should be approximately 30 pages.*

```
O: This is the data base operator. (1)
  What can I do for you? -o- (2)

U: I'm interested in getting some information about the stock
market crash of 1929? (3)
 Specifically I want to know ... causes and safegards.-o- (4)

O: Okay,(5)
    lets see. (6)
I'm checking under stocks.-o-  (6)

U: Could you look under crash of '29 or something to find sources that
discuss the crash?-o- (7)

O: Okay, lets see. (8)
I'm checking under stocks. -o- (9)

U: Sure. (10)
 Lets try that. (11)

O: There's nothing listed under crash of '29, or even crash. (12)
  Lets see what's under stocks -U.s. (13)
...  That doesn't look good either. -o- (14)

U: How about under the Stock market? -o- (15)

O: No! (16)  This is surprising. (17)
 Do you think Economy -U.s. might be good? -o- (18)

U: Hmmm, Its worth a shot. (19)
  If you can find anything related to the Economy of the twenties
  that would be good. (20)
  Also you might look at banking ?-o- (21)

O:There we go. (22)
  I have economics 1900 - 1934. (23)
  I'm now looking at some specifics. (24)

U: Ahhh, Could you look under FSLIC (Fedral Savings Loan Insurance ?) (25)
 That was established to prevent further crashes... (26)
 That might be useful. -o- (27)

O: First,(28)
        let me give you a few books that look useful. (29)
First, (30)
        I have Economic Basis of Tax Refor, by H.G. Brown, (31)

I also have something which may or may not be useful:
Industrial Crisis, its causes and...by Edie. -o- (32)

U: I definitely would like to see the first one. (33)
  The second is at least worth exploring. (34)
  Anything on the FSLIC? (35)
-o-
```

```
O: Okay, (36)
       that sounds good. (37)
   One moment. (38)
...
Yes, (39)
 I have a book called Proposals for reform of the Deposit Insurance System. (40)
  The author must be American Enterprise Institute. (41)
 Just in case, here is the call number: (42)
KF 1023.p123 -o- (43)

U: Is that Rhees or Carlson? (44)

O: That is in Rhees. -o- (45)
U: Also could I get the call number & locations of the other books?-o- (46)
O: Well, I'll have to look them up again. (47)
  Do you have time to wait a few minutes.-o- (48)

U: Sure-o- (49)

O: O.k. (50)
 Why don't you give me the titles it will go more quickly. (51)
 I don't have them written down. -o- (52)

U: Yes, (53)
          the first is economics 1900-1934, (54)
          the second is industrial crisis-o- (55)

O: Ok, The call Number is HJ 2305/b23 (56)
It is in Rhees ... (57)
The second is HJ 463.h143. (58)
 It is also in RHEES.-o- (59)

U: Ok, (60)
    that should get me going. (61)
   Thanks for the help.-o- (62)

O: You're welcome. (63)
  Good Luck on your project.-o- (64)
```

A.3 Dialogue 2

Subject: *Why do we dream? Give a multidisciplinary historical account. You might want to include theories and explanations from religion, psychology, neuroscience, and computer science. You paper should be approximately thirty pages.*

```
O: May I help you? -o- (1)

U: Yes, what do you know about dreams? -O- (2)

O: Well, do you need general information or something specific?-o- (3)
```

U: Lets start with on theories on why we dream! (4)
Where do they come from and why? -0- (5)

0: Okay. (6)
If you wait just a moment, I'll see what we have in the card catalog on dreams. (7)
Hold on. (8)

There are all kinds of books on dreams. (9)
Are you interested in dreams in art or poetry, or psychoanalysis or dreams, or what? -o- (10)

U: Give me theories about dream (ie where do they come from). (11)
These theories could come from religion, psychology, neuroscience, and computer science! -0- (12)

0: Okay, first there is a book which might help called The Dream in Psychoanalis, by L. Altman. (13)
The number is BF1078.A46d 1975. (14)
Do you think this will help? -o- (15)

U: yes -0- (16)

0: I also have a book called Sleap and Dreaming: Origins, Nature, and Functions, by D. B. Cohen. (16)
It's number is QP425.C56 195$ $79. -o- (17)

U: This sounds even better -0- (18)

0: Are you interested in something called the grammar of dreams? -o- (19)

U: What does this book entail? -0- (20)

0: Well, It isn't cross listed in any other subject category (21)
so I can't tell you any more about what it is. (22)
It is dated 1978. -o- (23)

U: Well it can't hurt to look at it, what is the call number? -0- (24)

0: BF1078.F62 -o- (25)

U: Are you supposed to be a computer? -0- (26)

0: No. I am the operator for the data base. Why? (27)

U: I was just wondering if you could give me access to the database (28)
so that I could get lists of books in this field, (29)
instead of going through one by one? -0- (30)

0: Well, that is not possible yet, (31)
but in the next month or two the data base will become public domain. (32)
Shall I continue the search on dreams, (33)
or would you will that be enough for now? -o- (34)

U: Since this is a thirty page paper due in less than 5 minutes, (35)

```
   can you give me more books in the BF1078 range (36)
   since this seems to be in the correct area! -O- (37)

O: Okay.  (38)
   Let me check. (39)

   I have a book called Dreams, What They Are and How They Are Caused.  (40)
   I'm sure this one interests you.  (41)
   It is by Leadbeater (42)
   and the number is BF 1099.L43d.  -o- (43)

U: You are a good interpreter, yes I would like this book.  (44)

O: Are there - (45)

U: I guess that will be all for now, thanks -O- (46)

O: Okay. Good luck.  Goodbye. (47)
```

A.4 Dialogue 3

Subject: *It is said that all wine grapes in France contain strains of American wine grapes. Give a detailed account of how this came to be. Is the same true of wine grapes from other European countries? Your paper should be about 20 pages.*

```
U: Hello -o- (1)

O: This is your data base operator.  (2)
   What can I do for you? -o- (3)

U: I am looking for information about French and European wine
     grapes.-o- (4)

O: Okay.  Just a minute please. (5)

   I have several entries under wine and wine making. (6)
   Do you think these will be helpful? -o-  (7)

U: Not wine making.  (8)
   And wine perhaps is too general.  (9)
   Are they many entries under wine ?  -o- (10)

O: Yes, but they are listed by country as well as history. -o- (11)

U: History of French wine may be fine.  (12)
   Can you select all the entries on this subject? -o- (13)

(Interrupt message on the screen:
  NFS server castor.cs.rochester.edu not responding still trying
  NFS server castor.cs.rochester.edu ok)
```

O: There seems to be something odd going on with the NFS server. (14)
Can you still read me? -o-(15)

U: Yes, I think so, what you say seems coherent !! -o- (16)

O: Good. (17)
I will check. (18)
Well, here is a book that might be useful. (18)
It is made up of a selection of addresses, essays and lectures
called Wine, Celebration, and ceremony. (19)
It's listed under History. (20)
It is by G. Johnson, (21)
and the call # is TP549.w561 1985. (22)

Also, there is another entry under history: (23)
The story of the Vine by E. R. E. Emerson. (24)
The number is TP549 .E53s. (25)
Do you think these will be helpful? -o- (26)

U: I am afraid not. (27)
What I am really interested in is wine grapes. (28)

O: Okay, well let me look under - (29)

U: When has the second book been published? -o- (30)

O: It was published in 1902. -o- (31)

U: That is ancient history! (32)
But I may find some useful information in it. (33)
Is "wine making" a keyword entry? -o- (35)

O: Yes. (36)
Shall I look into the listings of Wine Making under specific
countries? -o- (37)

U: Yes. (38)
Try with France. (39)
If we don't find anything, (40)
we'll extend the search. -o- (41)

O: Okay. (42)
Can you read French? -o- (43)

U: Yes, I can. -o- (42)

O: I don't, (43)
so I can't tell you what's in the book, (44)
but there is one called Un Drame Economicque; Les Delimitations: Le
Passe l'Avenir, by E. Clementel. (45)
What do you think? -o- (46)

U: What a strange title! (47)
If it really speaks about wine, (48)
it may be of great help. (49)

Are there more details on this book, perhaps? (50)

O: Sure just a minute. (51)
The book was written in 1914. (52)
It is also listed under Champagne and Competition, Unfair. -o- (53)

U: Ok. (54)
Is there something listed under grapes? -o- (55)

O: Yes. (56)
What is it exactly you need to know about grapes? (57)
There are lots of listings.-o- (58)

U: I need to find something about grapes that have been destroyed in France (59)
and grapes that have been sent from America to Europe -o- (60)

O: Well, I have a few books on Diseases and pests. (61)
Could that be relevant? -o- (62)

U: Yes. (63)
Sure. -o- (64)

O: Okay... (65)
Well this might not get us far. (66)
There is a book on the fungus diseases of the grape vine. (67)
The other is on diseases of NY grapes. (68)
What do you think? -o- (69)

U: I think they are too specific. (70)
Are there entries listed under grapes and history, limited to France or Italy or Spain. -o- (71)

O: I think that the entries under grape won't be useful. (72)
I am going to look back at Winemaking - France. (73)
I think there was an option there which I haven't tried yet. -o- (74)

U: I mainly need historical information about grapes. -o- (75)

O: Okay. (76)
Here is something. (77)
In french, its called: Un grand Commerce d'Importation: Les vins de France aux anciens Pays. -o- (78)

U: That's fine. (79)

O: The call number is HD9382.5 .c91. (80)

U: Under which entries is it listed? -o- (81)

O: It is under wine making - Europe. (82)
There may be some historical books there. (?) -o- (83)

U: Did you already try under grapes and history - France or Europe ? -o- (84)

O: Yes, I'm afraid the selections listed under grapes were limited. -o- (85)

U: Ok, so let's try under winemaking - Europe -o- (86)

O: Fine. (87)
There is a book called Studies in the medieval wine trade, ed.
by E. Veale. (88)
It is also listed under England - Gascone - History. (89)
Otherwise, I think were are just out of luck.! -o- (90)

U: Can you provide information about magazines an journals ? -o- (91)

O: No. (92)
You have to call another data base for that. (93)
I have one last idea though. (94)
I am going to give you the book listed under wines. (95)
I think it may at least mention the subject you are interested in. (96)
It may give you some hints for further searching as well. (97)

I'm sorry. (98)
The data base is not responding. (99)
I think we will have to end our session. (100)
I'm sorry I could not be of further help. (101)
My suggestion is to try a more complete data base. -o- (102)

U: Ok, thanks. (103)
I have noted the first references you gave me. (104)
They also may be of some help. -o- (105)
Goodbye. (106)

A.5 Dialogue 4

Subject: *Compare and contrast the mating habits of whales with those of fish. Your paper should be approximately 15 pages.*

O: Hello, this is your data base operator. (1)
What can I help you with? -o- (2)

U: I'm researching a paper on the mating practices of whales and fish. (3)
What kinds of sources do you have available?-o- (4)

O: Just a minute and I'll check. (5)
Well, I've just looked up whales in the data base (6)
and there are several books. (7)
None, however, are specifically on mating habits. -o- (8)

U: Actually, by my previous question I wanted to know if you had
information only on books, (9)
or on periodical articles also, etc.-o- (10)

O: I'm sorry. (11)

I misunderstood the question. -o- (12)

U: OK. (13)
Can you tell me the names of the books you found, (14)
or are there more than just a few?-o- (15)

O: There are 13 (16)
but I'll tell you all those that don't look totally irrelevant. (17)
I don't know all that much about whales maybe you can help me. (18)
There is a book called Conservation and Management of Whales. (19)
Do you think it sounds relevant?-o- (20)

U: No, probably not. (21)
Do any of the titles mention behavior, or something like that? -o- (22)

O: Yes. (23)
There is a listing under Whales -- Behaviour (24)
but the book is abbreviated as trad. English. (25)
Does it still interest you? (26)
The title is nothing that looks like English to me.-o- (27)

U: Sounds weird. (28)
I wonder what "trad. English" means. (29)
Well, why don't you give it to me anyway.-o- (30)

O: I'm sorry. (31)
On further examination, I have discovered that it is translated
TO English from Swedish. (32)
The book is mostly about whale intelligence, (33)
but it might contain some useful info, (34)
so I'll give it to you. (35)
It is called (in English) Smarter Than Man? by Karl-Erik Fichtelius. (36)
The call # is: QL789,4,M3 -o- (37)

U: And there were no others specifically on behavior?-o- (38)

O: I have one other book listed under Whale behaior - Congresses. (39)
Do you know what congresses are?-o- (40)

U: I guess they are some sort of conference, (41)
in which case it could be useful.-o- (42)

O: Okay, (43)
Well it is called Communication and Behavior of Whales ed. by
R. Payne. (44)
The call # is QL737.C4 1983. -o- (45)

U: And why don't you give me one or two general books on whales. (46)
Just pick random ones, (47)
if none look more promising than the others.-o- (48)

O: I found a great source. (49)
Its called The Whale Problem; a Status Report, by the International (50)
Conference on the Biology of Whales.
The call # is QL737,c4 1971. (51)

Carlson Library. -o- (52)

U: Well, that should give me a start on the whale side. (53)
I think fish are probably a lot less interesting, (54)
but I suppose we should try them. (55)
Or maybe it would be more productive to look under something like mating or sex or something. (56)
I think that could result in to many references on other than what I'm interested in finding out about at this moment though. (57)
Let's try fish.-o- (58)

O: Just a minute. (59)
I'll check... (60)
I have lots of sources on commercial fish farming and hatcheries and the like. (61)
Will that be useful? -o- (62)

U: I'm not really sure, (63)
but I'd like to look at one and then decide. (64)

O: Okay. (65)

U: I also know that some fish are egg-layers and some live-bearers, (66)
which should figure into my paper somehow. -o- (67)

O: How about fish hatching and fish catching by R. Roosevelt? -o- (68)

U: Sounds thrilling.-o- (69)

O: Okay, (70)
the number is SH151.R34f. (71)
I also have a book on artificial propagation. (72)
Could that be relevant?-o- (73)

U: No, I don't think so. (74)
Do you have anything relating specifically to biology? -o- (75)

O: I'll check. (76)
Strange as it may sound, (77)
there is no subject heading called FISH. (78)
I do have a book called the Evolution of Fish Species Flocks, ed. by A Echelle.-o- (79)

U: That doesn't sound useful. (80)
There's a word for the study of fish -- (81)
ichthyology, or some variant of that spelling?-o- (82)

O: Good idea. (83)
How about a nice introductory book. (84)
It's called Fishes: An Intro. to Ichthyology. by P. Moyle. -o- (85)

U: Awfully convenient of them not to list it under fish. (86)
The call number?-o- (87)

O: QL615 .M64. -o- (88)

U: Thanks. (89)
That should give me enough to start on. (90)
Thanks for all your help.-o- (91)

O: Your welcome. (92)
Good luck on your project. (93)
Bye. -o- (94)

U: Is there an address to which I can send my comments on this service?-o- (95)

O: Certainly. (96)
We appreciate any comments you might have. (97)
The address is:

Chester
University of Rochester
Rochester NY 14627 (98)

Anything else? -o- (99)

U: When do you get off work?-o- (100)

O: I work 24 hours a day. (101)
Will that be all? -o- (102)

U: Well, I was hoping not, (103)
but I guess so. (104)
Bye.-o- (105)

A.6 Dialogue 5

Subject: *Why do we dream? Give a multidisciplinary historical account. You might want to include theories and explanations from religion, psychology, neuroscience, and computer science. You paper should be approximately thirty pages.*

O: Hello. (1)
This is your data processor.(2)
What can I help you with? (3)

U: The history of dreams. (4)
Specifically a multidisiplinary account, (5)
stressing religion and philosophy. (6)

O: Okay. Just a moment please. (7)

Well, so far, I've come up with a book called Dreams, illusion, and other Realities. (8)
Does this interest you or shall I keep looking for a while? (9)

U: Continue looking for books on dreams as illusions.(10)

I would also like some information on Jung's theories of dreams.(11)

O: Okay. I have a book called Clinical uses of dreams about Jungian Dream Interpretation. (12)
How's that? (13)

U: Very good. (14)
Do you have anything contrasting and comparing Jung with Freud? (15)

O: Perhaps. (16)
First, let me give you the call number of the other Jung book. (17)
It is BF1078.H263. (18)
Now I'll see what else I can find. (19)

There is one relating to Buddism,(20)
and one relating to Jainism. (21)
Are you interested in these (22)
or shall I persue the Freud and Jung comparison.(23)

U: I'm interested in the one on Buddism, but you should continue your other search. (24)

O: The Buddist book on dreams is called Dream-symbolism in the sramanic tradition: two psycholanalytical... (25)
Its call number is BL1375.D73. (26)
Now, back to Freud and Jung. (27)

U: Actually you can cancel the Freud and Jung and start a search in neuroscience. (28)
I would specifically like to seem some book relating dreams to certain areas of the brain. (29)

O: Okay. (30)

Well, here is one book. (31)
It doesn't mention the science of dreams (32)
- I mean it doesn't mention neuroscience - (33)
but is is called the Science of Dreams. (34)
It's probably worth a try. (35)
The call num is BF1078.D5NFS3. (36)
Do you think it is worth looking under "Neuroscience"? (37)
I haven't yet finished going through the Dream entries yet. (38)

U: Can you give me a brief description of the Science of Dreams? (39)

O: Unfortunately, the listing is very brief. (40)
It was written in 1962, (41)
it has some illustrations, (42)
and the author is Edwin Diamond. (43)
That is all I can tell you. (44)

U: Does your search for other texts on neuroscience and dreams look hopeful? (45)

O: Well, I think it is worth looking through the individual books on dreams

for those that look scientific. (46)
Then, we <can> try another approach if it doesn't work. (47)

U: While you're doing that you can also see if any of the books mention computer science and dreams. (48)
That might be usefull as well. (49)

O: Okay. One minute. (50)

U: Can you tell me how many books I have to date? (51)

O: I believe you have three. (52)
I just found a book called Nightmare: biological and Psychological, (53)
and another book called Mecanisme cerebral de lo pensee. (54)
It looks appropriate, but french. (55)
Any of these seem worthwhile? (56)

U: Yes, keep the french one. (57)
Even if I don't use it, (58)
it will still look impressive in my bibliography.(59)
I think 4 books should do for 30 pages. (60)
Thank you. (61)

O: Okay. (62)
The French books call number is BL8233.s. (63)
I hope this information has been useful. (64)
Goodbye. (65)

A.7 Dialogue 6

Subject: *Philosophers, linguists, psychologists, and computer scientists offer a wide range of accounts of the semantics of natural language. They refer to sentences, sentence uses, speakers intentions, culturally determined stereotypes, speakers' beliefs, etc. Summarize five well-known accounts. Are any of them compatible? Your paper should be about 30 pages.*

U: Hello. (1)

O: This is the data base operator. (2)
What can I help you with? (3)

U: I would like information on the semantics of natural language. (4)

O: Okay. (5)

I have listings under Philosophy and Psychology. (6)
Nothing under Natural Language. (7)
Are one of these more relevant? (8)

U: Please search psychology and human language use. -o- (9)

O: Okay. One minute please. (10)

There is a book in german listed underf Seamntics - Psychological. (11)
Would that interest you? (12)

U: No, I am only interested in books written in English. (13)

O: Okay. (14)

How about a book like Studies in Syntax and Semantics? (15)

U: How current is the book, and who are the authors? (16)

O: Actually, the book is quite old: 1969. (17)
It is a selection of papers edited by F. Kiefer. (18)

U: Are there any current texts on "Psycholinguistics," from the past 10 years? (19)

O: There are over 200, (20)
most of which were published within the past 10 years. (21)
I can randomly pick some out for you (22)
and describe them if you like. (23)

U: No. (24)
Could you search the 200 books on "psycholinguistics" (25)
and return only books in English which deal with the subject of semantics. (26)

O: I'll give it a try. (27)

I am assuming that books with titles that mention words in the mind and cognitive aspects of language are relevant. (28)
For example here is a book called "Words in the mind: and introduction to the mental lexicon." (29)
Is this the sort of thing you're interested in? (30)

U: That sounds too specific, (31)
how many books does a search for the keywords psycholinguistics and semantics result in? (32)

O: Unfortunately, the way this data bank works is by one keyword at a time. (33)
We are currently trying to change that, (34)
but we haven't yet. (35)
I could give you a selection of current essays on semantics, and a selection of current essays on psycholinguistics. (36)
That might be a good start, (37)
and the individual essays will each have bibliographies that could assist you in a further search. (38)

U: How large is the selection of current essays? (39)
There could be hundreds and that would be too much to read for a 30 page paper. (40)
Are there any current critical reviews of the literature in the field of semantics? (41)

O: I will look. (42)
But first, I just discovered a book called :Semantics of
Natural Language by Donald Davidson. (43)
It is old (1977) but looks relevant. (44)
Would you like the call number? (45)

U: yes. (46)

O: It is P106.S39. (47)
Instead of looking at critical reviews, (48)
I will tell you of some other books that look good. (49)
I must have missed this section of the data base before. (50)

U: What section of the database? (51)
At this point the easiest way to get information on this topic is (52)
to go to PsychLit (53)
and search for current reviews of semantics (54)
using two or three relevant keywords (55)
but I guess that's not the problem here (56)

O: This section is called Addresses, Essays, and Collected Works under
Semantics. (57)
There is a book called Meaning Reference and Necessity,(58)
another called Ethnolinguistics, (59)
and another called Approaching Vagueness.(60)

U: How large are these sections of the database (61)
- total # of books? (62)

O: There are a total of 20 entries, (63)
not all of which look relevant. (64)
There is one that seems especially relevant to your study. (65)
It is called "Semantic; an interdisciplinary reader in philosophy". (66)

U: That sounds fine. (67)
Actually a listing of all twenty entries would be all I need. (68)

O: Okay, Instead of writing them to you, (69)
I will mail them. (70)
It should only take a day through the university mail system. (71)
By the way, the last book was called "An interdisciplinary reader in
philosophy, linguistics and psychology. (72)
In case you want to look for the book today, (73)
the call number is B840./s82. (74)

U: Is there any particular reason you are sending them through the
university mail? (75)
It makes it difficult to do research now. (76)

O: Well, if you like, I can give you the numbers of the books now. (77)
I thought that you might not want to take the time. (78)
It will take about 5 minutes to look them up individually. (79)
(This is another problem we are currently working on.) (80)
Shall I get the numbers now? (81)

U: Actually you were right, that would take too much time. (82)
Let me tell you what I was thinking of doing with the listing.(83)
I was going to manually scan the listing so that (84)
I could search along two or three key phrases in the titles (85)
so that I could narrow the field of relevant books to just a few. (86)

O: Oh. That sounds clever. (87)
I'm sorry our services aren't yet up to maximal service. (88)
But they will be soon. (89)

U: Thanks, you've been very helpful. (90)