NATURAL LANGUAGE PROCESSING AND ADVANCED INFORMATION MANAGEMENT

James E. Hoard

Boeing Advanced Technology Center
P.O. Box 24346, MS 7L-64
Seattle, Washington 98124

1.0 Introduction. It is widely recognized that the development of sophisticated natural language interfaces (NLIs) will have great potential for users of complex information management systems. Such NLIs would largely alleviate the need to learn and use complicated access protocols. More importantly, such NLIs would also alleviate the need first to learn and then spontaneously to recall the names of hundreds, even thousands, of table and field names and their interconnections within a given information management system. Indeed, if one seriously entertains the notion of integrating several complex information systems into a larger whole, adding, say, application software packages for data analysis (numeric, symbolic, or both), the use of NLIs becomes almost mandatory. Note, too, that, in principle, there is no reason that the information management systems themselves need be of the same type. One can easily envision the need and desire to integrate the information in relational (and other kinds of structured) databases with the information in such diverse sources as CAD/CAM systems and text documents.

Granted that the user of an integrated system can expect to issue requests for action through an NLI, a fair question is this: What is the glue that can bind everything together behind the interface, the means by which just the right information is extracted from among a number of different information sources, put into the correct form and given to the appropriate application software packages, and then correctly processed by the application software? I advance the thesis that, in the long run, the glue must be natural language, that there is no other means for us to describe and understand the world and what goes on in it, and that there is no alternative means by which computers can be expected to do so. To be sure, one can write special procedures that directly integrate selected information sources with selected application packages (bypassing understanding, as it were), but I maintain that all such integrations must forever be ad hoc endeavors which are neither general, flexible, nor extensible.

Section 2 discusses the concept of Advanced Information Management (AIM)---the attributes it must have and the capabilities it will require. Section 3 presents the basic requirements for an adequate natural language processing (NLP) system and discusses some of the work we are doing at the Boeing Advanced Technology Center towards achieving robust natural language understanding. Section 4 presents a very high-level AIM architecture. Section 5 offers a brief summary and some concluding thoughts on how to go about developing a simple, but general, AIM system.

2.0 Advanced Information Management. It is commonplace to note that software application packages typically produce and process data in a particular format and, moreover, that they run only on a particular kind of computer using a particular operating system. Integrating, in a principled way, application software packages that were not originally designed to work together will require advanced information management. By advanced information management I mean the ability:

1) to query (or access) diverse information sources through a multi-modal, integrated user interface
2) to have the system access information automatically across heterogeneous computers, operating systems, and networking systems
3) to have the system perform automatically certain numeric and symbolic calculations, implied by the query, over the accessed information
4) to have the system present automatically to the user the results of the calculations in a meaningful and serviceable way.

2.1 Accessing Diverse Information Sources. Information sources, which either are or can be put in electronic form, range over a wide spectrum of types: text documents, drawings, schematics, photographs, movies, printed and recorded music, flat-file databases, hierarchical databases, relational databases, semantic-net knowledge bases, and so forth.

Clearly, fashioning an electronic card catalog that offers unique addresses and descriptions for (potentially) billions of information sources, complete with all relevant cross-references, is not a trivial task. Nonetheless, the complex addressing scheme that Theodor Nelson and his associates have developed to access the universe of information sources--Nelson refers to this as the "docuverse"--seems to be fully adequate to the task [1]. In Nelson's scheme, addresses are divided into four sections: Server, User, Document, and Contents. The Server section indicates where the material is located, physically or logically. The User section indicates the owner and other control and security information. The Document section provides the logical entity (i.e. name, version, etc.) under which the information is stored. The Contents section describes the material and includes in hypertext fashion all the links to directly related material.

The docuverse as a whole will not be catalogued and made accessible anytime soon. Indeed, issues of privacy, public safety, and national security suggest that access to everything is, in any event, undesirable. Within a number of spheres, however, a well- and consistently-catalogued docuverse is highly desirable. Examples among docuverses that could be publicly available include those for various academic disciplines, unclassified government scientific programs, and court proceedings. Proprietary docuverses will include those of many government agencies and of virtually all individual manufacturing and service industries.

For an AIM system to make use of the items in any particular docuverse, a simpleminded Contents section will not do. In order for an AIM system to access and make use of a docuverse of diverse information types, the Contents section of the address scheme must contain links to both a Structural Description (SD) of the material and a Real-world Interpretation (RWI) of the structural description. To see why we need both of these, consider as a document some instance of a relational database. (At this level of granularity, we can consider the set of files to be a hypertext document.) Then the SD indicates 1) that the document was created by using a particular database system, say, ORACLE, and 2) which files are data files and which files give the complete database definition of this particular instance of ORACLE. Now, in and of itself, a database definition has no intrinsic meaning, since, in general, table and field labels are not self-explanatory. Although mnemonic labels obviously help those familiar with a particular database to remember what it contains, table and field labels could as well be arbitrarily chosen numbers (say, tables 1 through 157 and fields 1 through 905) for all the help they afford the integration task. Only by providing a complete RWI for a structural description can database definitions be interpreted by machines (and, by people) unfamiliar with the design and contents.

It is easy to see that having a RWI of the information in a database is crucial to integrating the information with information from other sources. For instance, given a field label such as "DATE", it is natural to ask what it is that "DATE" is the date of and to inquire about the interpretation of the internal structure, if any, of the values in the "DATE" field. Compare "760704", "040776", "04-07-76", "070476", "07-04-76", "04/07/76", and "07/04/76" as possible ways to express "July 4, 1776".

There are many other document types beside databases that require explicit RWIs. Among them are CAD/CAM documents. Given that a document is a drawing created using some CAD/CAM system, say, CATIA, then the SD indicates that it is indeed a CATIA drawing and gives descriptions of its files. The RWI that parallels the SD provides the basis for integrating the information in the drawing with the information in other documents of other types.

The RWIs of databases, CAD/CAM documents, and many other materials in a docuverse will have to be given in some natural language (English is the obvious choice for English speakers) for one simple reason: Natural language is is the only universal language people have, the only kind of language there is for describing anything at all, whether it be a description of some state of affairs, real or imagined, or of some action we wish to take. This means that natural language offers the only way we can realistically hope to integrate a large number of distinct information sources in an automated fashion.

To see what an RWI looks like, consider a database of geographical information. Suppose the SD reveals that we have a table, labeled GEOGRAPHICAL-DATA, which contains, among others, fields labeled PLACE and ALTITUDE. The RWI of these labels might be as follows:

> ALTITUDE is the altitude of PLACE in feet above mean sea level; PLACE is the name of a geographic feature (mountain, lake, city, airport, ...); GEOGRAPHICAL-DATA contains the names and geophysical attributes (see ALTITUDE, LOCATION, AREA, ...) of a number of geographic features (see PLACE) of North America.

If we have a natural language understanding system that can interpret the RWI and put it into machine-usable form, then we have, in principle, a way of integrating information from this information source with that of any other.

Unlike databases and CAD/CAM systems, text documents contain, by and large, their own RWIs by virtue of being text documents. I say "by and large", since text information that is presented in tables, indexes, tables of content, chapter and section headings, headers, and the like will require RWIs unless they conform to default formats known to the AIM system. In addition, texts often contain pictorial and graphic insertions that are not, in general, self-explanatory and, for which, RWIs will be required.

In sum, using Nelson's docuverse addressing scheme, within a given docuverse, an AIM system will automatically assign document addresses to all new items that are created and will automatically update the addresses as documents are revised. There is, however, one crucial point: It will be the responsibility of the creators and revisers of documents to provide and maintain the integrity of the RWIs in the Contents section of the addresses, since there is no way that any AIM system can read minds.

2.2 Executable Documents. Many of the items in the docuverse are not static, run-of-the-mill materials, i.e. unformatted text, graphics, database files, or whatever. They are, in fact, executable programs, materials that from a docuverse perspective can be viewed as Executable Documents (EDs). Such programs run the gamut from the simplest COBOL or C program to massive expert systems and FORTRAN programs. Since the docuverse address scheme allows us to link documents at will, we can link together compiled code, source code, and descriptive material in hypertext fashion. Now, if, in addition, we can prepare and link to an executable document an Input-Output Document (IOD), a document specifying a program's input and output requirements and behavior, and an RWI describing the IOD, we can entertain the notion of integrating data and programs that were not originally designed to work together. Moreover, we can hope to do so with methods that are not ad hoc and do not rely on brute force.

In particular, we could avoid brute force, "one-off", non-general procedures that map words like "high" and "height" directly onto a database field like ALTITUDE. Rather, we would invoke general natural language processing procedures to relate the word "altitude", taken from the RWI of ALTITUDE, to vertical elevation and height. Since the RWI also tells us that the altitude is expressed in feet, an AIM system can also report the units along with the value. In fact, if we have another database, say, a geographic database for Europe, and we have its SD and RWI, we can use our general methods to find out the altitude of the Matterhorn. Suppose now that we wish to know which mountain is higher, and by how much. If the value for the height of the Matterhorn is expressed in meters--and an AIM system will know that it is from the RWI--the AIM system will invoke an ED for converting units, do the conversion, compare the altitudes, and report the results in feet or meters, as we desire. (Not having an AIM system with geographic databases in its docuverse, I had to do all the work myself. As it happens, the Matterhorn is about 112 meters, or 369 feet, higher than Mt. Rainier.)

2.3 Heterogeneous Computer Systems. Obviously, to access very much at all of the docuverse, an AIM system will immediately be confronted with the fact that materials tend to reside on different computers with different operating systems. To make matters worse, the threads that stitch computers together come in different weights and colors, e.g. the Network File System (NFS) and the Transmission Control Protocol/Internet Protocol (TCP/IP). While a successful AIM system will require a number of IODs and RWIs in order to move information round and about, I will not pursue the matter here.

2.4 Implied Calculations. Every interesting use of an AIM system requires that the system have the ability to do implied calculations. Some of the queries one might direct at an AIM system will be of a simple sort. For example, one might inquire: "What is the height of Mt. Rainier?" Given access to a geographic database, we surmise that the system will simply look up the answer in a table of mountain names and heights and report the value to us, viz. "14,411 feet". Suppose the query were framed with less specificity, say, "What is the tallest mountain in Washington, and how tall is it?". On the assumption that the database tables contain information correlating mountain and state names, the system is now required to perform some very simple implied calculations, both numeric (comparing heights of mountains) and symbolic (distinguishing mountains in Washington from those in other locales).

In principle, queries containing implied numeric and symbolic calculations can be as elaborate and as oblique as we like. For example, it would be quite reasonable for a geologist to ask: "How much did the North American Plate move with respect to the Pacific Plate in 1988?" [2]. The first task for the AIM system is to understand the query. This requires symbolic processing of a high order and is the topic of section 3. Suppose for the sake of argument that the system succeeds

in understanding the query. Then, if data files are available for 1988 that give range values from satellites to ground stations on each plate as well as the angles between the ground stations and the satellites, fetching the data and performing some numeric calculations involving the sine of this and/or the cosine of that will yield the answer the geologist seeks.

It should not be supposed that the worth of natural language symbolic processing is confined to natural language interfaces--or even, as I have asserted, to gluing together diverse computer packages. On the contrary, the vast majority of the world's information sources are text documents, as a trip to any library, perhaps even to one's study, will easily reveal. A real AIM system will answer queries that require symbolic calculations over the text data that is included in the docuverse. For example, suppose we have a biography of Abraham Lincoln in electronic form. If so, it should be quite reasonable to ask such questions as: "Did Lincoln ever visit Richmond, Virginia?", "How many terms did Lincoln serve in Congress?", and "Who killed Abraham Lincoln?". In general, the answers will not be found by simplistic pattern matching techniques. It would probably be difficult indeed to find a biography of Lincoln that contained the sentence "John Wilkes Booth killed Abraham Lincoln," or even one that stated "It was John Wilkes Booth who killed Abraham Lincoln." Instead, we have all, I expect, read the narrative accounts of John Wilkes Booth sneaking up a back stairway in Ford's Theater, entering Lincoln's box, shooting Lincoln, leaping to the stage and breaking his leg, making his escape, and of Lincoln being carried across the street and dying early the next morning. In short, answering even such apparently simple queries such as "Who killed Abraham Lincoln" will require sophisticated natural language processing that can comprehend narratives (understand each of the sentences and the interconnections among them) and is able to use semantic and real-world knowledge to draw conclusions (e.g. "X kill Y" means "X cause Y to die"; "If X does Z, and Z causes Y to die, then X kill Y") .

Clearly, there will be limits on the implied calculations that even a mature AIM system could be expected to perform automatically. Basically, we expect such a system to be able to do what an able human assistant could do. Thus, it would obviously be futile to ask: "Is Fermat's last theorem true?" or "Will the universe ultimately collapse?". At best, the system would respond with "No one knows.", or something of the sort. On the other hand, there is a broad spectrum of relatively mundane tasks that an AIM system ought readily to perform. Among these are reading vast numbers of text documents for content and applying a variety of EDs to data gathered from many different information sources.

2.5 The Interface. The interface of an AIM system will need to be very capable. Issues involving user modeling, avoiding cognitive overload, and the appropriateness of input and presentation techniques are beyond the scope of this paper. Nevertheless, a few remarks on input and output are required.

First, a sophisticated natural language interface is a necessary part of any AIM system. While I have nothing against icons and desktop metaphors, there are obvious limitations to iconic interfaces--viz., they are virtually useless for formulating syntactically complex commands. While command languages do not suffer from that limitation, they suffer from two other undesirable attributes: 1) they tend to be hard to learn and use, and, more fundamentally, 2) each of them is artificial and, hence, has no intrinsic semantic interpretation. For example, on a SUN workstation, you can invoke a command line to put a representation of a clock on the screen at the location of your choice. The command line is best described as arcane and can be compared to an AIM system command, which we might frame as "Put a clock in the upper right-hand corner of the screen". In a multi-modal interface we could frame the request as "Put a clock there", with an appropriate gesture to indicate where "there" is.

Second, the presentation component of an AIM system must evaluate the characteristics of the information that is accessed, presenting it to the user in a comprehensible and appropriate manner. The characteristics of interest include the number of elements and whether they are numbers, text, or graphics. A smart presentation manager will be able to make many correct choices on its own and adjudicate the form of the presentation with the user, suggesting alternatives on a case by case basis, as required. Suppose, for instance, that, not realizing the extent of the work done on the Athapaskan languages, I asked my AIM system for a listing of all the linguistic work done on this family of languages. The presentation manager should inform me that it has found many hundreds of items to include in an Athapaskan linguistics bibliography and give me the opportunity 1) to narrow my query or 2) to group the items in some convenient way and put them into a file for browsing. Similarly, an AIM system should make sensible choices in deciding how a set of data values is best presented graphically, deciding from the nature of the data to use, say, a bar chart, histogram, or scattergram.

2.6 AIM Requirements. The requirements of an AIM system can be summarized as follows: We need a universal addressing scheme, such as the one Nelson has devised, to keep track of the things in a docuverse. We need an adequate NLP system, and we need it both for the interface and for interpreting what static documents contain and what EDs can do.

We will need what appears to the user to be an "intelligent agent" that can use natural language descriptions and instructions to carry out our wishes. Figure 1 shows the overall AIM system. Note that as far as an AIM system is concerned, there are only three kinds of things in the docuverse: stand-alone EDs, e.g., application programs like statistics packages and grammar and style checkers; stand-alone static documents, e.g. text documents; and application packages that consist of static documents and the associated EDs that are used to manipulate the data in the static documents. To make use of an application package, an AIM system will query the RWIs and SDs of the static documents and the associated ED as information sources, then use the application package ED to access the data in the static documents.

3.0 An Adequate Natural Language Processing System. In our work on natural language processing at Boeing, we base our approach on three fundamental premises: 1) all linguistic forms have meaning--there are no meaningless linguistic elements; 2) linguistic and real world knowledge can be described in discrete syntax, semantics, and pragmatics modules; and 3) natural language processing must interleave linguistic and real world knowledge.

The first premise is easy to justify. Consider the pitfalls in ignoring the difference between infinitive and gerund constructions revealed by such pairs as: "They stopped to search for survivors" and They stopped searching for survivors". In the first example, the infinitive gives the reason for stopping; i.e., semantically, "to search for survivors" is a 'purposive' modifier of the intransitive verb "stop". In the second example, the gerund phrase "searching for survivors" is the direct object of the transitive verb "stop"; semantically, it is the 'range' complement of "stop", a complement that states the activity that was stopped. Similarly, consider the pair of examples "Reagan sent a message to Iran" and "Reagan sent Iran a message". For the first sentence of this pair, Reagan's message could have been sent anywhere in Iran, although it could have been sent to the government of Iran. The second sentence states unambiguously that the message was sent to the government of Iran. This pair of sentences underscores the need to take account of subtle meaning differences occasioned by seemingly innocuous differences in word order and the presence or absence of words like "to".

The second premise stems from the observation that whatever the subject matter, the English language is, minor dialect differences aside, everywhere the same and that the work of developing general syntax, semantics, and pragmatics modules need be done only once. When such a system is extended to a new domain, only the knowledge specific to that domain need be added to extend the system's coverage. The conceptual distinction between syntax, semantics, and pragmatics embodied in the system allows us to develop each module separately and incrementally--we can make progress in one module while we resolve problems in another. Moreover, we can keep declarative knowledge, say, the rules of English syntax, quite separate from the procedural knowledge required for an efficient syntactic parsing algorithm. Thus, we can expand and revise the syntax rules without affecting the parser, and conversely, we can improve the parsing algorithm without affecting the syntax rules. Indeed, extending our syntactic rules to cover telegraphic speech (e.g."Having wonderful time. Wish you were here.") was accomplished by adding a few rules to the grammar for ordinary text, without making any change to the existing grammar or to the parser.

Our third premise is an explicit acknowledgment of the inherent massive ambiguity of natural language, ambiguity that is at once its strength--for it allows us to say whatever we wish with decidedly finite means--and its weakness--for it is all to easy to misinterpret what one hears or reads and to be misinterpreted in turn. Out of context, even relatively simple sentences can be very ambiguous. "The girl saw the boy on the hill with a telescope" is a typical example. Either the boy or the girl or both is on the hill, and either the boy or the girl or the hill has a telescope, which the girl may or may not have used to see the boy. It seems to me that only by interleaving the application of linguistic and real-world knowledge word by word and phrase by phrase during text (or speech) processing can we hope to reduce to tractability the amount of computation required for disambiguation. In any event, a system that interleaves syntactic, semantic, and pragmatic knowledge, making disambiguation decisions as soon as possible in every instance, can converge on the best interpretation of a text in some given context much more quickly than can any system that mechanically constructs all possible syntactic parses, then constructs all the possible semantic interpretations of all the possible parses, and then decides upon the best text interpretation from among all the possible interpretations.

3.1 The Sapir Natural Language Processing System. In our efforts to develop the Sapir natural language processing system (named in honor of Edward Sapir, the eminent linguist of the first half of the 20th century) we have endeavored to utilize the best current knowledge about the properties of natural language in general and of English in particular. As shown in Figure 2, we are implementing our linguistics-based approach in four principal modules: token and layout analyzer, syntactic analyzer, semantic analyzer, and pragmatic analyzer. The components of the system are designed to interact cooperatively through the message workspace.

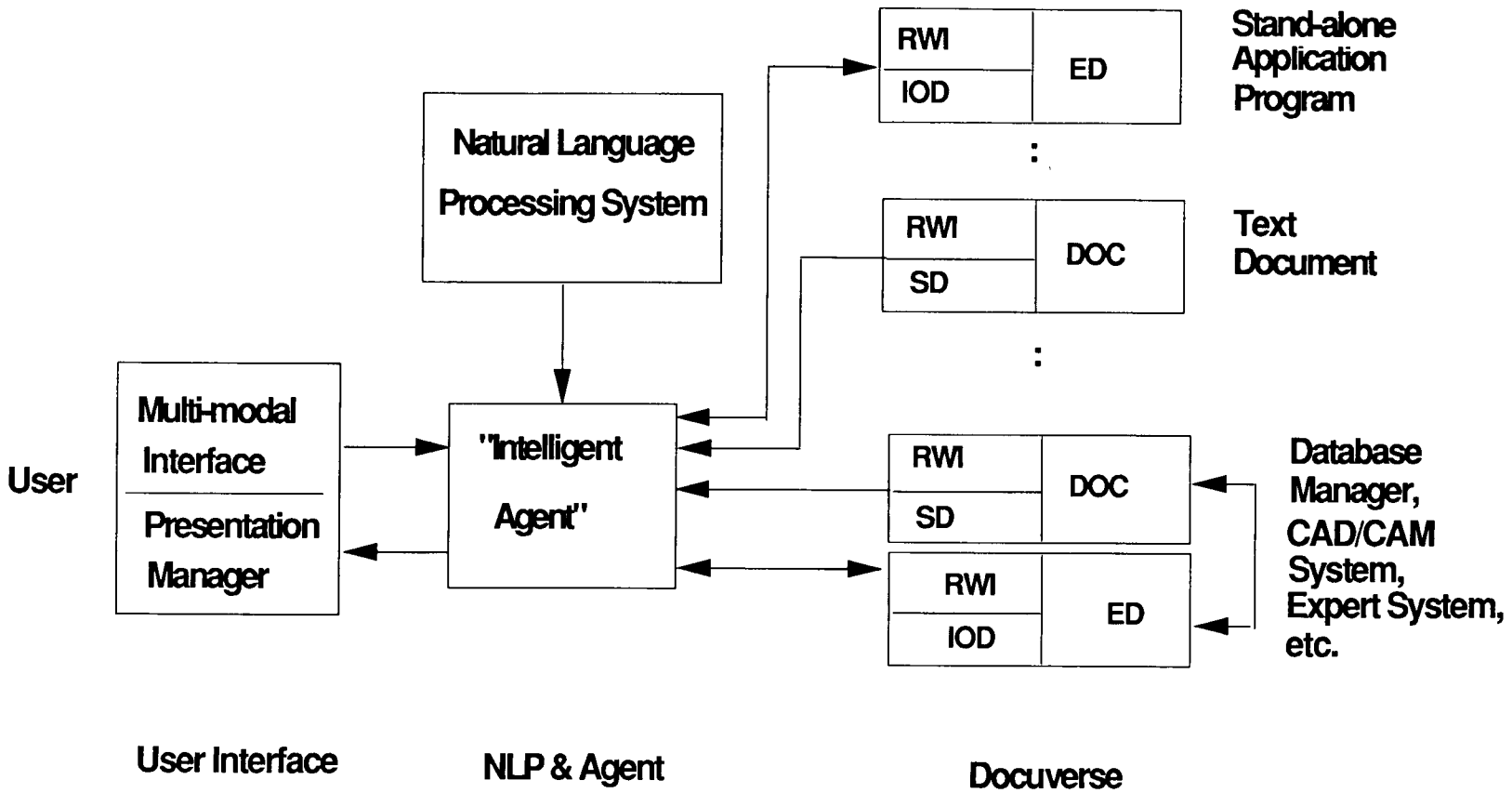

Figure 1. Overall AIM System

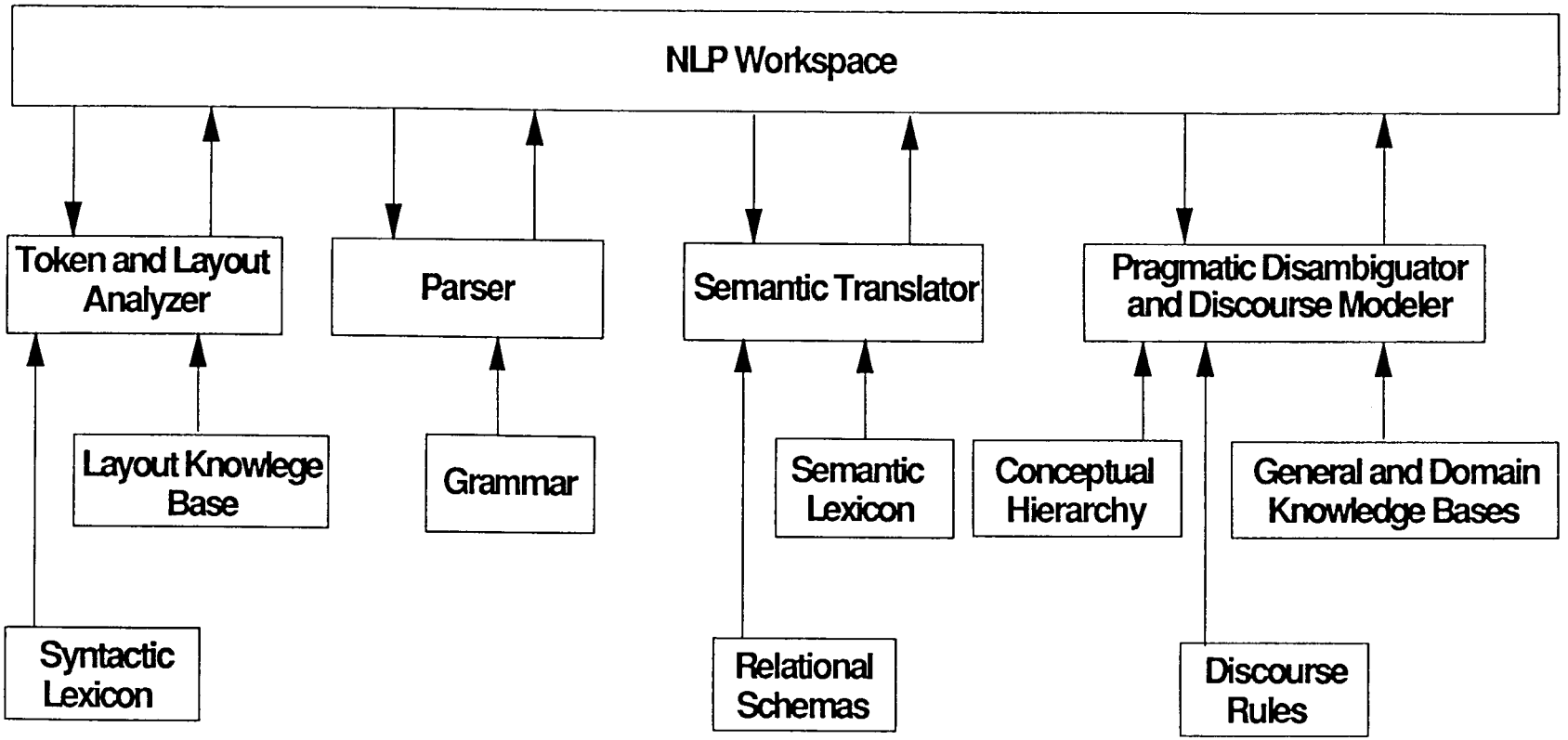

Figure 2. Sapir System Architecture

The token and layout analyzer initiates the text analysis process by recognizing each word in the input string in turn and posting a word packet (the word, its position, and its syntactic characterizations) in the message workspace. The parser and grammar, working bottom-up and left-to-right, use the word packets to produce syntactic constituents, each of which is posted into the message workspace as soon as it is constructed. The semantics module takes each of the syntactic constituents in the order in which they appear, provides all possible possible semantic translations, and posts the semantic structure in the message workspace, linking each interpretation with its syntactic counterpart. The pragmatics module takes each syntactic-semantic structure in turn and rates it for pragmatic likelihood. Those syntactic-semantic structures that are deemed acceptable are used for further processing by the syntax and semantics modules; those deemed unacceptable are held aside and, hence, are not carried forward as potential sub-constituents of larger syntactic- semantic structures. Viable syntactic-semantic structures posted in the message workspace are available to the discourse modeling component to analyze as potential discourse constituents. Thus, our system architecture deals forthrightly with the inherent massive ambiguity of natural language, eliminating unlikely interpretations of a text contextually as soon as possible during the analysis process and reducing to a minimum the number of structures that must be carried downstream, at great computational expense, during the course of the text understanding process.

We are currently evaluating our interleaved architecture in a blackboard environment, using Boeing's Erasmus Blackboard, which is configurable in a variety of ways [3]. The advantages of a blackboard implementation are fourfold: 1) we can run the modules in distributed fashion across several computers; 2) the system modules can be implemented in different languages; 3) we can experiment with interleaving (and parallelizing) the computations; and 4) we can experiment with opportunistic problem solving techniques which would allow our system to dynamically decide where to focus its attention.

3.1.1 The Token and Layout Analyzer. The token and layout module processes ASCII files and infers typographical, structural, and lexical information from the printed form of documents. The current output of the layout analysis subsystem of this module is a demarcation of sentence boundaries. When it is mature, the layout analyzer will recognize and characterize text constituents such as sentences, paragraphs, and sections, and will model information about text structure in concert with the discourse analyzer. The token analysis subsystem of this module uses finite-state recognizers to identify character strings as tokens and converts the tokens into lexical items. Ambiguities are passed up to a token parser that resolves them on the basis of the larger textual context. The token parser can postulate both single and multi-word lexical items from token strings, and, conversely, it can map the substrings of a single token into individual lexical items. At present, we have a lexicon with syntactic information on over 11,000 words. Even though the number of words in our lexicon is increasing rapidly, the lexicon will never be complete, since previously unencountered proper names will always appear in new text material, and new words, especially single- and multi-word proper names, are, in any event, constantly being added to the language. Thus, at the user's discretion our present token analyzer can default an unrecognized token to the category proper noun or ask the user to identify it interactively.

3.1.2 The Syntax Module. The syntax module provides a domain- independent description of English syntax. Our approach is based on Generalized Phrase Structure Grammar (GPSG) [4], a high-level formalism in which each grammar rule is equivalent to a large number of traditional context-free phrase structure rules. Without this formalism, it would be necessary to write tens of thousands of phrase structure rules. GPSG is especially well suited to natural language processing since it allows parsing to proceed in a context-free, bottom-up manner. Our syntax module, which provides very broad coverage of English, consists of approximately 300 grammar rules and an efficient parser [5] that we believe is one of the fastest in existence. We have developed special parsing methods that make it unnecessary to convert the high-level GPSG-like grammar into its expanded form, making it possible to develop the grammar interactively, an indispensable aid for iteration, testing, and validation of possible rules.

3.1.3 The Semantics Module. The semantics module provides semantic translations of syntactic constituents. We have designed and built a prototype semantic translation program which takes in turn each syntactic constituent constructed by the parser and produces a semantic interpretation of a sentence. The module generates semantic structures by pairing semantic translation functions associated with each lexical element with every syntactic configuration in which the lexical item can be a leaf node. The initial semantic translation of a sentence is then the semantic structure built compositionally from the individual syntactic configurations and lexical elements that are its constituents. The semantic representations are expressed in a formalism, called relational logic, that we are developing for natural language semantics. The formalism uses a special vocabulary of relational operators that link lexical items into semantic structures. Relational logic structures make explicit both modifier-head and predicate-complement bindings and can be decomposed algorithmically to derive both lexical and predicational inferences. Lexical inferences are made by consulting basic definitions, expressed as relational logic structures, and invoking general substitution procedures.

The relational logic representation we have developed offers a holistic view of semantic structures, explicitly giving the cognitively salient relation (or function) that obtains between the linked elements of the semantic structures. While the cognitive view of language it reflects has its origins in antiquity and traditional grammar, the recent precursors of relational logic include writings as diverse as Fillmore [6], Chafe [7], Quirk et al. [8], and Hudson [9].

Relational logic is very much in the spirit of what is now termed cognitive linguistics. The theoretical underpinnings of cognitive linguistics have been carefully set forth by Langacker [10]. The essential notions are 1) the distinction between autonomous and dependent elements and 2) the asymmetry between these two kinds of elements (called A/D asymmetry). Autonomous linguistic elements are those that require no elaboration to be semantically complete. The most prototypical autonomous elements are the nouns. Dependent elements are those that cannot stand alone, that obligatorily require elaboration. The prototypical dependent elements include verbs and auxiliary verbs, articles, adjectives, adverbs, and prepositions. The categorizations of linguistic elements as autonomous and dependent is not rigid. A particular linguistic element can figure autonomously in one grammatical construction and dependently in another. A grammatical construction typically has at least one element that functions dependently and one that functions autonomously; i.e., A/D asymmetry is an observed property of the vast majority of grammatical constructions. Langacker also gives adequate characterizations of the notions modifier, head, predicate, and complement--characterizations that are indispensable for adequate and accurate semantic analysis, and which we have adopted in our system of relational logic.

Figure 3 shows the relational logic semantic graph for the sentence, "How much did the North American Plate move with respect to the Pacific Plate in 1988". The query is one that a geologist might ask of NASA's Crustal Dynamics Data Information System, as described in [2]. The nodes of the graph are the linguistic elements and the arc labels are the relations. Dependent elements are placed above autonomous elements: thus modifiers are shown above the heads they modify, and complements are shown below their corresponding predicates. The relations are drawn from a small, finite set and serve to indicate the overall cognitive (or conceptual) meaning of the sentence.

The semantic ambiguity problem for a natural language processing system is twofold--first, to have a means of representing functional ambiguity, and second, to provide a way of choosing from among the semantic alternatives. Relational logic provides a quite natural way of representing functional ambiguity. For example, the prepositions "of", "by", and "for" have a number of the relational meanings, as shown in Figure 4. We view the semantic relation between the preposition and its first argument as holding between the element that typically precedes the preposition in syntactic configurations and the entire nominal phrase that follows the preposition. Thus, in "a test of skill" the "test" is "with respect to" "skill"; and, in the "most of today", "most" is the "extent of" "today". While "of" seems to have little inherent lexical content, the prepositions "by" and "for" have intrinsic meaning, i.e. they have different senses, in each of their uses. For instance, in a locative use, "by", as in "by the window", means roughly "near", and "for" in its directional use, as in "leave for London" means roughly "towards". In a temporal use, "by", as in "by Tuesday", means roughly "not later than", and durational "for", as in "sleep for hours" means roughly "during a time interval of". The various senses of a word are also expressed by relational logic structures. This provides a basis for deriving natural language inferences among words and sentences, as discussed in [11].

Relational logic also provides an efficient means for choosing among alternative semantic structures. Consider the sentence, "John thought for days of the incident by the river." The main verb "think" enters into complement structures such as 'think(cog:X)' and 'think(cog:X, r:Y)', where 'cog' means "cognizer" and 'r' means "range". Since "think" is used intransitively in this example, our system selects the first of these complement structures, 'think(cog:X)'. The possible modifiers of "think" include both outer situational locatives and temporals (those that specify a location in space or time, respectively) and such inner circumstantial modifiers as 'wrt' ("with respect to"), 'man' ("manner"), and 'ext' ("extent"). (Here we follow Barwise and Perry [12] and Pollard and Sag [13] in distinguishing situations from circumstances.) Our procedures must choose among the alternative possible prepositional phrase interpretations based partly on the range complements of the prepositions and partly on the modifiers permitted by the main verb. In analyzing the prepositional phrase "for days", the range of "for", namely "days", makes it likely that the durational interpretation of "for" is intended and that the other possible interpretations of "for" are unlikely in this context. We further interpret the phrase "for days" as an inner circumstantial modifier of "think", since it does not specify a point on the spatio- temporal plane (as would, say, "in 1988" or "on Tuesday"). Next, our program considers both attachment possibilities for the phrase, "of the incident": It must be associated with either the preceding noun to form the phrase, "days of the incident", or with the verb "think" to form "think of the incident". Our system recognizes "think" plus a 'wrt' modifier as a conventional pattern. Hence, the interpretation "thought of the incident" is chosen as the more likely reading. Similarly, the phrase "by the river" can be associated either with the preceding noun to form the nominal phrase "incident by the river" or with the

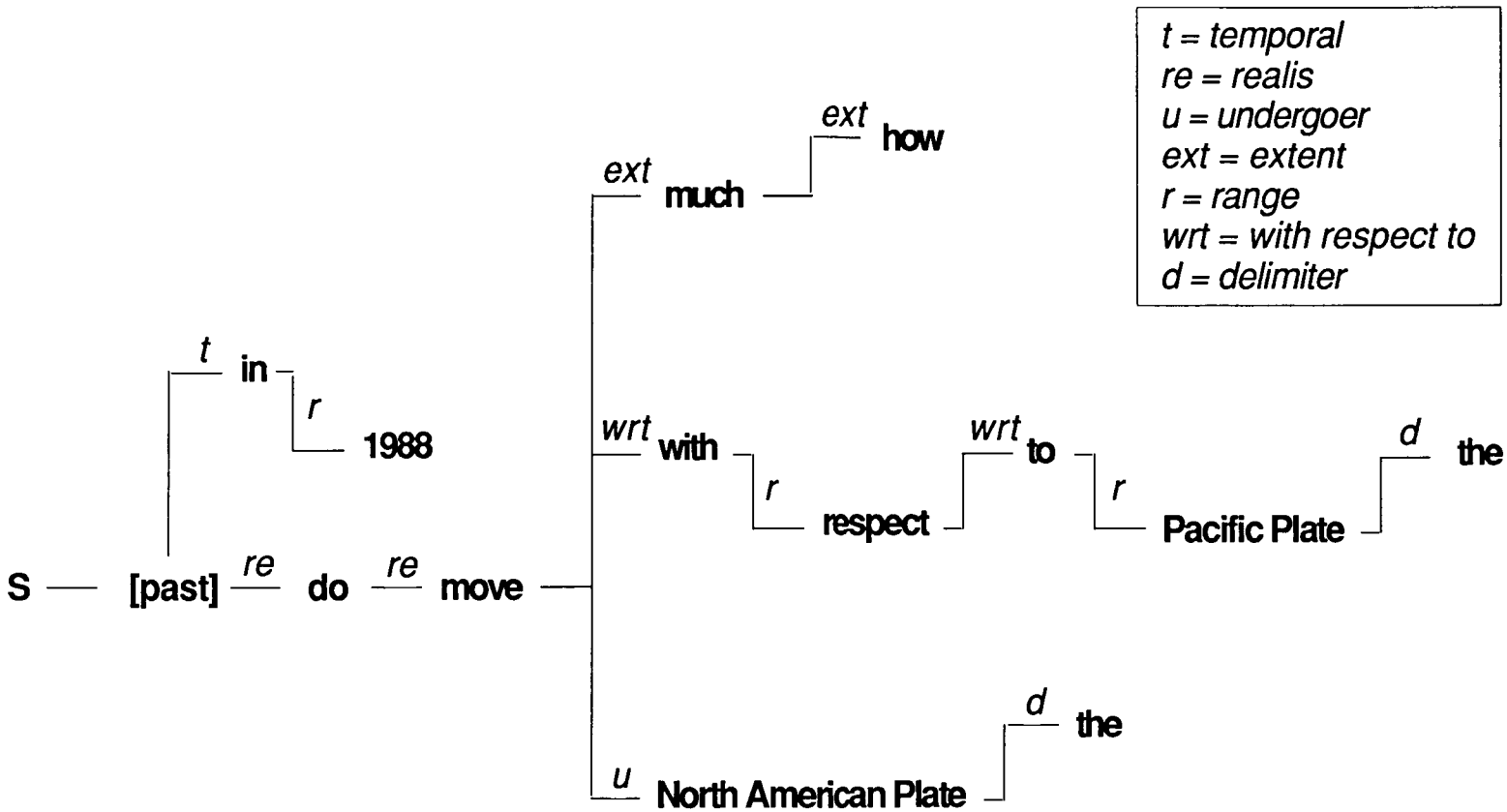

Figure 3. Semantic Structure of "How much did the North American Plate move with respect to the Pacific Plate in 1988?"

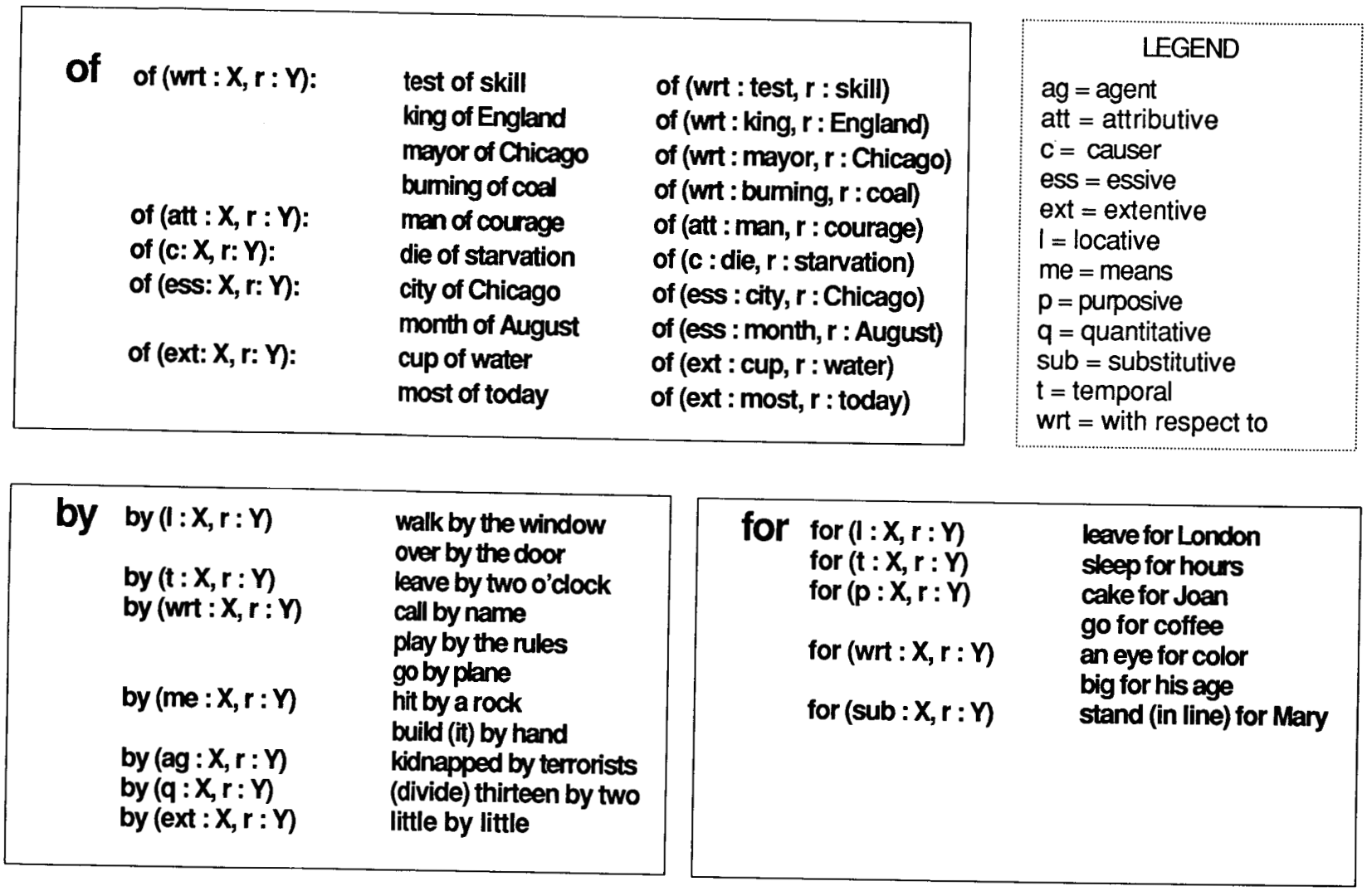

Figure 4. Functional Relations for Of, By, and For

main verb to form "thought by the river". The locative interpretation of "by" applies equally to both possibilities. Thus, the discourse model must be consulted to help make the decision: If it has knowledge of some event which occurred by a river, the nominal phrase would be sanctioned. Otherwise, the verbal attachment, as a situational modifier, will be preferred.

The foregoing example demonstrates our strategy for interpreting such constituents as prepositional phrases and for choosing among the potentially large number of alternatives. Our procedures first narrow down the possibilities to those that are most likely based on a functional analysis of the words. Then, reasoning and discourse considerations select the best choice from among the likely candidates. Relational logic serves as an effective representational formalism both for expressing functional ambiguity and as a basis for performing intrasentential disambiguation.

3.1.4 The Pragmatics Module. The pragmatics module has two principal functions. First, it provides the means to select (disambiguate) in context the most likely semantic structures from among those provided by the semantics module. Second, the pragmatics module builds and maintains a discourse model, interpreting the text as a coherent structure having causal, temporal, and spatial connections among its sentences. To do this, the pragmatics module must, among other tasks, establish different levels of event description in a sentence; establish discourse segments; resolve pronominal and definite references; and determine the most likely scope of quantifiers.

Our strategy for disambiguation initially uses a category hierarchy, matching categories against expectations to assess the likelihood of the hypothesized semantic relations among the lexical items in a given semantic structure. If several viable candidate structures remain, a reasoning system brings discourse and encyclopedic knowledge to bear to select the most likely meaning. For example, the transitive relational frames for "teach" are "teach(c:X, g:Y)" and "teach(c:X, r:Y)", where "c" is the "causer", "g" is the "goal", and "r" is the "range" complement of "teach". "John taught Bill" and "John taught algebra" illustrate the two possibilities. Ambiguous semantic translations arise, however, since it is not possible to tell a priori if the direct object of "teach" is the goal or range complement. An example is "John taught Wittgenstein", where we cannot be sure without considering the context whether John taught Wittgenstein's philosophy or Wittgenstein himself. In such cases we must appeal to the reasoner and the discourse model.

Our work on the pragmatics module is being undertaken jointly with Professor Lenhart Schubert of the University of Alberta and several of his students. The semantic net system that Schubert and de Haan have developed has a number of innovative features that enhance the efficiency of inference [14]. Without such innovative strategies, a natural language processing system would be overwhelmed by the large number of propositions in its knowledge base. The reasoning system uses an extended first-order logic that provides for event variables. The primary inferencer, called ECoLogic, is supplemented by a number of specialist reasoners [15] that reason about such things as anaphora and definite reference, quantifier scoping [16], temporal relations [17], color relations, and set enumeration.

The anaphora and definite reference program, for example, uses heuristics derived from Reinhart [18] and Bosch [19]. The program considers coreference possibilities among six different types of input phrases (or terms): pronouns, proper names, generic noun phrases, definite noun phrases, indefinite noun phrases, and quantified noun phrases. In addition, verbs are assigned "episodic constants" which can be the reference of pronouns and definite noun phrases (NPs). An example discourse that is handled by the program is: "John kissed Mary on the forehead. It embarrassed the young woman." The program assigns a numerical weight to each coreference possibility (a pairing of terms that can refer to the same object). It then lists the readings in order of preference, based on the combined weights derived from the heuristics. For the simple discourse above, the highest average weight is assigned to the reading in which the definite NP "the young woman" is coreferential with the definite NP "Mary", and the pronoun "it" is coreferential with the episode (the event of John's kissing Mary on the forehead) associated with the first sentence.

Our approach to discourse modeling uses as a starting point the observations on discourse structure offered by Reichman [20] and Webber [21]. A discourse is a text which coheres in a real-world context, semantically and pragmatically. The task of discourse modeling is to establish a coherent interpretation of a text. Our strategy is to develop disambiguation algorithms utilizing a number of specialist subsystems which place in context the relational logical structures generated by the semantics module. These subsystems will 1) decide what the discourse focus is at any given point in a text, tracking topic and focus shifts to establish discourse segments; 2) generate and maintain sets of potential referents both for entities mentioned explicitly in the text and for entities implicit in the text and derived by inferencing; and 3) discover and dynamically maintain a model of the discourse. We are concentrating our efforts initially on a few of the most common discourse devices, exploring ways of utilizing adverbial discourse cues (e.g. "for example", "thus", "first", and "on the

other hand") as indicators of discourse structure and overt text format markers such as paragraphing, section numbers and titles, and the like.

4.0 An AIM Architecture. The obvious candidate for an "intelligent agent", as discussed in 2.6, is an Object-Oriented Database (OODB) or Object Server [22]. OODBs have the desirable property of being able to deal with documents, both static documents and EDs, as "objects". OODB objects are put into "classes", which define the kinds of operations, including any number of special "methods", that can be performed on them. To invoke any of an object's methods, one sends it a "message".

To integrate diverse documents in an AIM system requires a class of executable documents (or methods) that can be called Transform Documents (TDs). TDs use SDs and IODs to access data files (through an ED or directly), to export data to another ED, and to receive the results of computing done by an ED. In brief, a TD takes data in its original form, transforms it into the form expected by some ED, and directs the ED to use the data to compute some function or functions, which the AIM system knows the ED to be capable of because it has accessed and understood its RWI and IOD.

The TDs will be supplemented by Reasoning Specialists (RSs), in the manner of the NLP specialists, that analyze queries and docuverse RWIs to determine what actions to take in a given situation. Taking an anthropomorphic view of the matter, the RSs will do what a person would do in the same situation. For instance, if you were asked to find out whether Lincoln ever visited Richmond, Virginia, and you did not know the answer, how would you proceed? First, it would seem reasonable to seek out biographies of Lincoln. Once you had one or more of them, it would be advantageous to search in the indexes under "Richmond", "Virginia", and the like. If there were no indexes, you might look through tables of content and section headings. Finally, you could just start reading the biographies. This suggests how the RS would proceed with text searches: examine RWIs to find pertinent documents; examine the indexes of relevant titles to narrow the inquiry; if indexes are lacking, examine the tables of contents; then, if there are any, examine individual section headings; finally, if still unsuccessful, read the books. Since an AIM system can read and understand English, we suppose that it has no trouble finding out if Lincoln "visited" Richmond, even if a biography states that he "went to" Richmond and never uses the word "visit".

To do an implied numeric calculation, such as the one implied by the sentence graphed in Figure 3, an AIM system will have to analyze the query and recast it as: "What was the difference in distance between the North American Plate and the Pacific Plate at the beginning of 1988 and the end of 1988". That is, "in 1988" can be interpreted in this context as "at the beginning of 1988 and the end of 1988"; and "How much did the North American Plate move with respect to the Pacific Plate" is interpreted as "What is the difference in distance between the North American Plate and the Pacific Plate". Recasting a query is effected by RSs that use natural language definitions and synonyms to go from the query to an interpretation form that can be used by the TDs. Now, before the distances can be calculated, the AIM system must, among other things, access the plate location data. In this instance, just to find out what the relevant site names on the two plates are, an RS will end up requesting a TD (in, of course, the relational logic natural language formalism) to "List all the western sites on the North American Plate" and "List all the eastern sites on the Pacific Plate". The TD, we suppose, will transform the requests into something usable by a particular database ED. Once the site names are known, an RS can ask a TD to fetch the data values for the appropriate time period; and so forth, until the user's implied calculation is satisfied.

In short, what we require of an AIM system is that it have sufficient RSs and TDs to make effective use of the documents in its docuverse. Clearly, this is not a trivial task and must be the subject of further research. We can see, however, what the high-level architecture of an AIM system must be like. It will have the basic architecture given in Figure 5.

5.0 Conclusion. Integrating diverse information sources and application software in a principled and general manner will require a very capable AIM system. In particular, such a system will need a comprehensive addressing scheme to locate the materials in its docuverse. It will also need an NLP system of great sophistication. It seems to me that the NLP system must serve three functions. First, it provides an NLI for the users. Second, it serves as the core component that understands and makes use of the RWIs contained in the docuverse. Third, it enables RSs to arrive at conclusions that can be transformed into procedures that will satisfy the users' requests. The best candidate for an "intelligent agent" that can satisfactorily make use of RSs and TDs appears to be an OODB. OODBs have, apparently, an inherent capacity to use the large numbers of RSs and TDs that will be required by an AIM system and an inherent capacity to use them in an effective way.

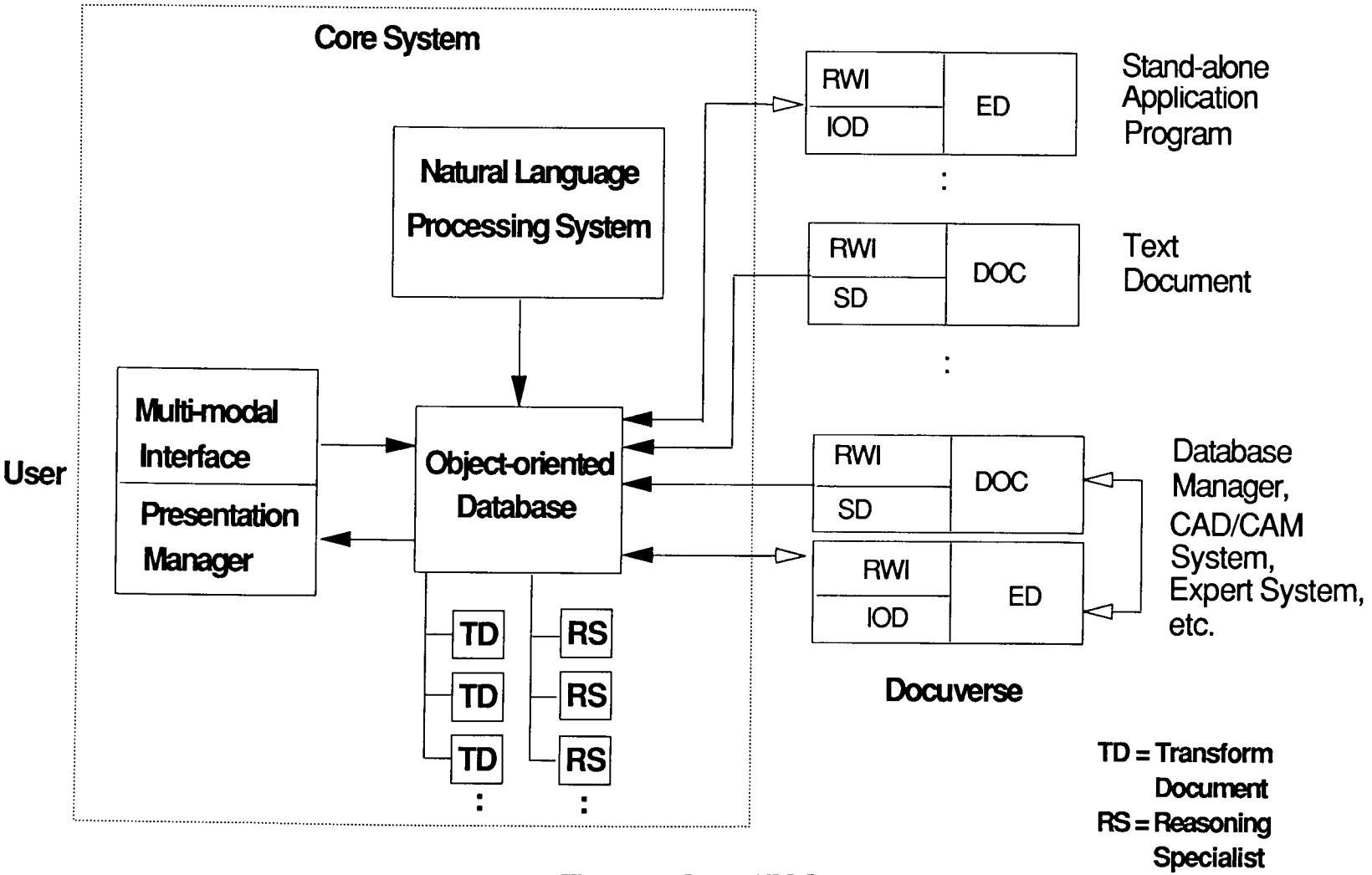

Figure 5. Core AIM System

Bibliography

1. Nelson, T. 1988. Managing immense storage. BYTE, 13,no.1.225-238.
2. Short, N., W. Campbell, L. Roelofs, and S. Wattawa. 1987. The crustal dynamics intelligent user interface anthology. NASA technical memorandum 100693.
3. Baum, L. et al. 1987. The Erasmus system. Presented at the AAAI Blackboard Systems Workshop, Seattle, July, 1987.
4. Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. Generalized phrase structure grammar. Oxford: Basil Blackwell and Cambridge, MA: Harvard U. Press.
5. Harrison, P. 1988. A New Algorithm for parsing generalized phrase structure grammars. Ph.D. dissertation, U. Washington, Seattle.
6. Fillmore, C. 1968. The case for case. In E. Bach and R. Harms, eds., Universals in linguistic theory. New York: Holt, Rinehart, and Winston.
7. Chafe, W. 1970. Meaning and the structure of language. Chicago: U. Chicago Press.
8. Quirk, R., S. Greenbaum, G. Leech, and J. Svartvik. 1985. A comprehensive grammar of the English language. New York: Longman.
9. Hudson, R. 1984. Word grammar. Oxford: Basil Blackwell.
10. Langacker, R. 1987. Foundations of cognitive grammar; volume 1, theoretical prerequisites. Stanford: Stanford U. Press.
11. Hoard, J. 1986. Drawing natural language inferences. Presented at the 61st annual meeting of the Linguistic Society of America, New York, December, 1986.
12. Barwise, J. and J. Perry. 1983. Situations and attitudes. Cambridge, MA: MIT Press.
13. Pollard, C. and I. Sag. 1987. An information-based syntax and semantics. Stanford: Center for the Study of Language and Information.
14. de Haan, J. and L. Schubert. 1986. Inference in a topically organized semantic net. Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86), 334-339.
15. Miller, S. and L. Schubert. 1988. Using specialists to accelerate general reasoning. Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88), 161-165.
16. Hurum, S. 1988. Handling scope ambiguities in English. Proceedings of the Second Conference on Applied Natural Language Processing, 58- 65. Austin, TX.
17. Miller, S. and L. Schubert. 1988. Time revisited. Proceedings of the Seventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence, 39-45. Edmonton, Alberta, June, 1988.
18. Reinhart, T. 1983. Anaphora and semantic interpretation. Chicago: U. Chicago Press.
19. Bosch, P. 1983. Agreement and anaphora: A study of the roles of pronouns in syntax and discourse. New York: Academic Press.
20. Reichman, R. 1985. Getting computers to talk like you and me. Cambridge, MA: MIT Press.
21. Webber, B. 1988. Discourse deixis: Reference to discourse segments. Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics, 113-122.
22. Purdy, A., B. Schuchardt, and D. Maier. 1987. Integrating an object server with other worlds. ACM Transactions on Office Information Systems, 5:1.27-47.