COMPUTER MEDIATED COMMUNICATION

Thesis by

Rémy D. Sanouillet

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1994

(Submitted August 19, 1992)

Acknowledgments

I would like to express my gratitude towards NCR Corporation for its generous personal support during the final stages of this research. I am particularly indebted to Mike Graf and Tom Kunz for their constant help and unfaltering friendship.

I would equally like to thank David Robinson for always being on the receiving side of a question with a good answer, and Andrei Sherstyuk for allowing me to borrow his graphic talent.

This work was very much influenced by Dr. Bozena Thompson's extensive linguistic knowledge which I have tapped repeatedly.

My greatest appreciation goes to my advisor, Professor Frederick B. Thompson, for instilling in me through his perseverance, insight and dedication to this project a flame that has altered the course of my life and that I shall keep carrying forward.

Finally, I acknowledge the loving and technical support of my family and of Arlette without whom this endeavor might not have reached its culmination.

Abstract

What will the age of the telephone-computer be like? In this thesis we present an answer to this question. We will base this answer on a conceptual framework being developed by the research group in which this work has been done, namely the Caltech/NCR Project; the work of this Project is embodied in the New World of Computing System. This framework will be stated as a paradigm for human information activities.

The main contributions of this thesis are, first, an examination of the implications of this framework for the communication aspects of information processing. The second is the design of the communication aspects of the New World of Computing System, reflecting the results of this examination. The System, in its totality, provides a computer environment for the telephone-computer age.

Table of Contents

I. Introduction: The Accelerating Information Revolution

We are all very much aware that major technological and social changes are under way as a result of the rapid advances in the technologies involved in the processing of information. Now — in the last few years and stretching through the next several decades — an acceleration of this change is taking place as two main streams of these advancing technologies combine — namely, computer technology, represented by personal computers, professional workstations and digital switches, on the one hand, and telecommunications technology, represented by the digital telephone, packet switching and fiber optics, on the other. Within this relatively short period of time, we will see the emergence of the telephone-computer, a device that combines the capabilities of the workstation and the telephone into a single instrument which will rapidly become a ubiquitous part of our environment, with profound implications on almost all aspects of our lives.

We think of the telephone as the archetype of an instrument for communication. Indeed, the international telephone system makes it possible for almost any two persons, at least in the developed part of the world, to be almost instantaneously in direct voice contact. However, it is totally lacking in important communication capabilities. Books and journals and maintenance manuals communicate between

authors and recipients over time. They also communicate from one person, the author, to many persons. Pictures, drawings and video are indeed on occasion worth a thousand words. There is another important dimension to communication, namely the transformation of the form of information between the sources and recipients. With the old technologies, there has been no capability of automating this aspect of communication. Accounting sections of businesses, considered as instruments of communication among the many offices of the firm, collect data from diverse sources, transform it into entirely different forms, and communicate these to many recipients. It is these aspects of communication that will now greatly enrich communication across this vast telephone network.

Only recently have we begun to think of the computer as an instrument of communication. We find an excellent example of this new thinking in the words of Terry Winograd:

> "The prevalent view is that in AI we design 'expert systems' that can stand as surrogates for a person doing some job. From a viewpoint of human interaction we see the computer's role differently. It is not a surrogate expert, but an intermediary — a sophisticated medium of communication. A group of people (typically including both computer specialists and experts in the subject domain) build a program incorporating a formal representation of their beliefs. The computer communicates their statements to the users of the system, typically doing some combinations and rearrangements along the way. The fact that these combinations may involve complex deductive logic, heuristic rule application or statistical analysis does not alter the basic structure of communicative acts." [WINOGRAD82]

The computer has memory, can accept inputs from many sources over time, combine and transform its inputs into new forms, and provide its clients with those particular forms each of them finds most useful. Now we are putting this powerful mediating device at each station in the vast telephone net, augmenting our capability for direct and instantaneous communication with the vast informational resources, some stored over time, that this new instrument can bring to bear for our immediate, relevant needs.

What will the age of the telephone-computer be like? Daily newspapers as well as specialized journals in the fields of computing and telecommunications are filled with conjectural articles. John Sculley, president of Apple, has made a videotape, "The Knowledge Navigator," publicizing his personal speculations. One of the laboratories of Pacific Bell has its own tape, "Protonet: A Direct Line to the Future," made for the purpose of carrying their projection to higher Pacific Bell management. The answer to the question posed in the first sentence of this paragraph, however, was given by Robert W. Lucky, Director of Communications Sciences Research, Bell Laboratories, in his address as Annenberg Distinguished Lecturer, University of Southern California on January 22, 1990:

> "What are we going to do with this gigabit [of in-place telephone bandwidth]? The honest answer is we don't know."

This lack of clear, coherent perspective is reflected in a floundering of the information technology industries, in a lack of policy leadership in both industry and government, and — in the United States — in an almost disastrous legal muddle in the regulation of telecommunications. This lack of coherent vision stems from a deeper source. These are the technologies that support the information processes of ourselves as individuals and of our society as a whole. "Information processing" has

come to have too narrow a connotation to serve as a basis for understanding the revolution that is taking place. We do not share a clear paradigm that can provide us with perspective and understanding of intellectual activities. The rapidity of change in these areas has decayed the meaningfulness of the vocabulary, the words which until recently we all seemed to understand very well, indeed which we still use as if they still carry their usual force. Words such as "information", "reasoning", and "intelligence" have lost a substantial measure of their content. For example, the amount of single copy, printed "information" coming off computer printers each day exceeds by orders of magnitude the total of what all the human race together could read if that were our only activity. In what sense is it "information"? Whom does it inform?

What will the age of the telephone-computer be like? In this thesis we present an answer to this question. We will base this answer on a conceptual framework being developed by the research group in which this work has been done, namely the Caltech/NCR Project; the work of this Project is embodied in the New World of Computing System. This framework will be stated as a paradigm for human information activities.

The main contributions of this thesis are, first, an examination of the implications of this framework for the communication aspects of information processing. The second is the design of the communication aspects of the New World of Computing System, reflecting the results of this examination. The System, in its totality, provides a computer environment for the telephone-computer age.These communication aspects have been implemented by the author, as an adjunct to this thesis, as a part of this System.

II. The Background Environment of this Thesis

A. A Paradigm for Information Processing

1. THE NOTION OF SUBLANGUAGE

We human beings engage in informational activities. Informational activities include talking among ourselves, writing and reading, creating engineering drawings, taking pictures of many different aspects of the world around us, choosing a course of action, checking with others whether our plan of action has been understood. Aspects of these activities have variously been called information processing, decision making, planning, communicating. These activities include using computers and using the telephone. We are in need of a simple, understandable paradigm for the notion "intellectual activity"; one that can provide perspective to the relationship between the telephone and the computer, on the one hand, and human informational activities on the other.

If we examine the intellectual activities of a human being at a given moment in time, it is clear that one's thoughts are confined to a small conceptual domain. One deals at such a moment with only a very small part of the wide-ranging material that is generally available in one's memory. One's immediate conceptual domain contains

all sorts of entities: some that relate directly to objects out of one's sensory experience, and some that may be totally abstract. Among these objects one envisions many interrelationships relating entities over time: some accounting for special distributions, and some not related to sensory data at all. However these world models are by no means haphazardly disparate; they are comparable and related, but in ways idiosyncratic to one's limited, immediate conceptual domain. This conceptual domain, with all of its complex structure, has been referred to in the literature as the individual's conceptual model; but this is a misleading term, for one sees in one's mind's eye many possible models of the world which one evaluates and compares. In order to find a clear and concise way of characterizing these small conceptual world views, we turn to the rapidly merging fields of mathematical linguistics and theoretical computer science. To give a point of focus to our discussion, we introduce the notion of a "sublanguage".

An illustration here may be useful. After performing a surgical operation, the surgeon writes a report on what was accomplished. Such reports often end with the phrase: "The patient left the operating room in good condition." When it is pointed out that this sentence is ambiguous, the surgeon is at first puzzled, then amused to think of the patient getting up, cleaning the instruments, mopping the floor and leaving the room in good condition. In the context of the operating room, there is no ambiguity; the sublanguage brooks but one interpretation of this phrase. As a second illustration, consider two situations. In the first, a person is at the tobacconist and orders: "Filter!" In the second, the same person is staring at the oil puddle below his car's engine and knowingly mutters: "Filter." The word "filter" is not ambiguous in either situation. It is clearly not a matter of recognizing which of the several meanings

of "filter" apply, any more than a car driver chooses between the brake pedal and the accelerator when approaching a red light.

We do not think and speak in a "natural language." The notion of natural language has played a useful rôle in the linguist's development of a general understanding of human communication, in the codification and maintenance of purity of national languages and the training of language teachers. Language, in the sense used by the linguists, might more properly be thought of as an integrated family of linguistic mechanisms, often expressible as grammar rules. The phenomenon of information processing and communications, although exhibiting in a given cultural community the adherence to such syntactic forms, also has features which are better characterized, we believe, by the notion of "sublanguage," here being introduced.

> "The individual is changed through the use of language, and the language changes through its use by individuals." [WINOGRAD87-1]

When a research team, representing several disciplines, is first brought together, the sentences used by its members are long and most are for the purpose of clarifying word meanings rather than exchanging ideas. Later, their sentences become terse and depend on a great deal of tacit understanding. It is this maturing sublanguage that brings efficiency to the team's communication. When someone whose interests are the linguistics of "natural" language studies the sentences used by such an experienced team, they may appear ambiguous, ill-formed, fragmentary. However, careful attention to the semantic relationships between these utterances and the narrow subject matter to which they refer reveals that this discourse is indeed lawful and fully understandable, but in a sublanguage which in many ways is entirely their own.

In dealing with their immediate environments, people narrow their considerations by making judgments of relevance, value and effectiveness, judgments that are characteristically human. The results of these judgments take concrete form in the sublanguages they use both in communicating within their group and in their internal thought processes. A moment's introspection makes it clear that as we move about from one place to another during a busy day we change from one sublanguage to another.

The stability of a given sublanguage is found in the stability of the task we undertake. When we return again and again to a task, and to that group in which we interact in conducting that task, it is the sublanguage that codifies and externalizes our on-going considerations. As our immediate environment changes, and our attention is directed elsewhere, our sublanguage, i.e., the formal language that characterizes the conceptual environment in which our thought processes take place, shifts as well.

As an illustration, consider a particular work environment, say that of a person working as secretary for an industrial manager. One aspect of that person's environment is the typewriter. The technology of the typewriter keyboard has not changed in many years, even though a more efficient layout of the keys is known. The reasons for this stability are not hard to envision. Namely the keyboard does not change because of the strong social inertias resulting from so many people having been trained on the existing one. There are many other aspects of the typing, filing and sending of letters, reports, etc., where there are both physical and social inertias that mitigate against change in many aspects of the secretary's concern. The moment-to-moment sublanguage of such a person is constantly shifting as a needed address must be found, a letter retrieved from a file, a phone call answered. However a part of all of these sublanguages remains essentially constant — that part related to the

mechanics of typing, phoning, filing, where the physical and social inertias are high. This part is itself characterizable as a formal language. It is precisely these highly stable sublanguages that can economically be built into computer systems. Word processors are an ideal example of how the inertias of a significant part of the secretarial world can be exploited.

2. THEORETICAL FOUNDATIONS

We impose structure on the jumble of our moment-to-moment experiences so as to create order and provide perspective. Although "structure", in this usage, has been characterized in a number of equivalent ways: recursive functions, Turing machines, etc., another way, equivalent to these others, is as language, the formal definition of language now common to mathematical linguistics, philosophy of language and computer science. It is the infinitely variable expressions of language that give tangible form to our immediate view of the world and by which we share that view with others. It should be no surprise to find human language playing the central rôle in leading to an understanding of information processing. The mechanisms of language are precisely the tools we need and use to express the recursive structures we impose on our experience. It is these mechanisms of language which we share that form the basis of communication. Mechanisms, such as word order, case endings, connectives, self embedding such as the use of the relative clause, number systems, expressions of time, etc., are learned at an early age, and provide the formal ways to sew together concepts into expressions that can be widely understood.

Once the judgments of relevance and depth of focus have been made, and these judgments become tacit, subsumed into underlying semantic structure, sublanguages take on the characteristics of formal languages. This is what Church's thesis says. The

deliberations we carry out in our minds about a fixed subject matter — fixed in level of relevant detail and breadth of concerns — are "calculations". This is what Gödel, Turing, Markov, Post and Church, in quite disparate ways, were trying to characterize. Each of them characterized/defined bounds on human intelligent activity. The abstract equivalence of these separate attacks gives compelling evidence that they indeed captured the essence of human calculations within the confines of a tacitly accepted sublanguage. The class of all such formal languages constitutes the domain in which is found, for a given instant in time, the sublanguage that precisely characterizes/encapsulates our view of our world at that instant. From moment to moment, our view — our sublanguage — may change as our circumstances change. However as long as environmental restraints stay the same and our attention is sustained on a given area of concern, this same formal language continues to embody our "calculations". Indeed, our ongoing activities as human beings, even in highly stable tasks, are such as to change in evolutionary ways the sublanguage that characterizes our view. How do we change sublanguages? Although formal proof has yet to be given, it is most likely that this shift in sublanguage is not a computable function.

The concept of "sublanguage" is abstractly equivalent to the concepts of "recursive function" and "Turing machine". Thus our "sublanguage" paradigm is equivalent to Church's thesis; that is, as was stated above, that human information processing can be characterized by these formalisms. To give utmost precision to this statement, we restate this paradigm in terms of one of the above formalisms. In the usual proof that a Turing machine can mimic the calculations of any formal language, a Gödel mapping is developed from the alphabet, finite strings of these, and finite sequences of such strings into the integers. Then it is shown that each constructive

process needed in language processing can be mimicked by a Turing machine operating on the Gödel numberings of the arguments of the process. This line of reasoning can be insightfully extended by bringing in the notion of a Universal Turing machine. Suppose we have a universal Turing machine: $\mathcal{T}_u(m,n)$ such that $\mathcal{T}_u(k,n) = \mathcal{T}_k(n)$ where $\mathcal{T}_k$ is the k^{th} Turing machine in some effective enumeration. One can legitimately think of m as the Gödel number of the sequence of grammar rules and lambda expressions of the formal language corresponding to $\mathcal{T}_k$. If n is the Gödel number of a well-formed question of this language, $\mathcal{T}_u(m,n) = h$ can then, as a Gödel number, be decoded as the answer to that question. In this formalism, the sublanguage paradigm is equivalent to saying that at any instant our thought processes can be characterized as a Turing machine, precisely Church's thesis. Moreover Church's thesis is its own converse; thus each of us encompasses the capabilities of a Universal Turing Machine. Yet if this is true, then we must be essentially more, a Universal Turing machine with a Demon $\mathcal{S}$; having observed n, our Demon selects that Turing machine that is most informing, and $\mathcal{T}_u(\mathcal{S}(n),n)$ becomes our immediate cognitive model.

The paradigm for human information processing that underlies this thesis can now be stated:

It is the constant re-evaluation and adjustment of the relevant view that characterizes human information processing. A succinct expression of such a view is as a formal language. When there are strong social and physical inertias in an area of broad concern, that part of our sublanguage concerned with that area stabilizes. For these stable areas, it is economically expedient to develop computer systems that can understand these sublanguages.

3. IMPLICATIONS OF THE PARADIGM

The Universal Turing machine analogy suggests that one could write a computer program that would operate in two modes. In mode one, the program would ingest grammar rules of a formal language (including their semantics), one at a time, building them into an appropriate grammar table. In the second mode, the program would accept any string in the terminal vocabulary of the language, process it, and output the appropriate response. These two programs together would constitute a Universal Language Processor that could handle any sublanguage. This germ of an idea constitutes one of the core theoretical concepts of the New World of Computing System.

Continuing along these lines, look again at the "one-at-a-time" way of adding grammar rules. Grammar rules, including both their syntactic part and their semantic part (which can be conceived equivalently as a Lambda expression or as a subroutine embodying the same abstract algorithm) are indeed separable, isolated modules. A language processor, whether implicit in a compiler or explicit as a separately identified process (the Universal Language Processor) has two parts: (1) a parser, which uses the syntactic parts of the grammar rules to build the parsing graph, and (2) the evaluator, (such as the "eval" procedure of LISP) that uses the parsing graph to compose the semantic procedures, completing the processing of the input string.

There is one area where this computerization of sublanguages is already highly developed and sophisticated, namely programming languages. The stability of the Von Neumann architecture has resulted in an evolutionary development of sublanguages that exploit this stability. In the last sentence of the previous paragraph, we spoke of the computer "understanding" a sublanguage. The meaning of "understanding" is epitomized by the way the computer understands a programming

language: it is able to carry out instructions stated in the programming language in a way that was intended by the programmer who wrote them.

Consider a middle level manager in a large engineering concern. He writes instructions to his computer, typing:

"Task G will be delayed 10 days."

The computer:

1. changes appropriately the database entry for the duration of task G;
2. recomputes a Pert chart for the entire project;
3. for those tasks that will experience a significant delay, generates e-mail to the affected managers, identifying the cause and potential consequences of the delay.

The sublanguage of engineering management is thus "understood" by the computer, just as the manager would expect a staff assistant to have responded in a pre-computer era.

One could conceive of language segments — each made up of a family of grammar rules — establishing relationships among a restricted set of related concepts. The material needed to discern the inner content and workings of these relationships can be visualized as being held in appropriate structures and processes on these structures. These language segments could then be fed into the language building mode of our program, a rule at a time. Picturesquely, a new language segment could be purchased at our nearby computer store and "plugged into" our computer. The result would be an extension of our sublanguage. All of our previous data and prior "programs" — as language segments — would be unchanged. In this way we add a new capability to our existing sublanguage, with virtually no disruption of our ongoing processing. For example, one could delete the grammar rules constituting a

two-dimensional graphics package and replace it by a package that could handle three-dimensional graphics. One could still say just as before: "Histogram the height and length of ...."; but now one could also say: "Histogram the height, length and depth of ...." and the three-dimensional package would take its inputs from a three-dimensional array and produce the three-dimensional graphics output.

Look at this last example a little more carefully. Multidimensional arrays, ever since FORTRAN, are a common data structure. Their implementation in a limited memory is not solved with finality, as the recent evolution to OS/2 and extended DOS 5 have made apparent. The grammar of array processing, that is, the way one phrases in "ordinary English" how one wants arrays processed, is quite standard (cryptic commands and icons may have been added to standard phrasing in a particular user's sublanguage). These would carry over naturally from the two-dimensional to three-dimensional graphics. It is the semantic processing procedures underlying this syntax that take rethinking. So it is precisely a new "array" language segment that retains and appropriately extends existing syntax that is needed. There are certain basic functions one needs to process arrays, as long as one abstracts out details of representation. One needs to create an array, delete it, insert and access a value of known position, and search for the position of a given value. Add a few binary operations and one has accounted for all the utilities needed. The rest can be left to the Universal Language Processor. This notion, illustrated in the above paragraph, i.e., that an abstract data structure may be implemented in a variety of ways and the details of such an implementation may usefully be hidden, encapsulated in "utility procedures", is well known, being a central notion of object oriented programming.

B. The New World of Computing System

The New World of Computing System is a multimedia distributed information management system developed at Caltech under the guidance of Drs. Frederick and Bozena Thompson. It is difficult to give a more descriptive one-line definition due to its extensive far-reaching nature. One's first impression is that it is a multilingual natural language interface to a semantic net database — or equivalently, an entity attribute database — where the entities may be from arbitrarily many object classes (in the sense of object oriented programming). However, it contains its own modules for list processing, paging and networking, as well as providing a new and powerful applications development environment. We now describe in somewhat more detail those aspects of this System that are particularly germane to research reported in this thesis.

1. THE LANGUAGE PROCESSOR

How are sublanguages implemented in the computer? One class of sublanguages is already implemented in computers, namely programming languages. How are they currently implemented? A compiler is written which embodies both the syntax of the language and the semantics. The compiler accepts a sentence of the language and returns a single, machine language program. When used in interactive mode, this program is then executed. That is, the abstract computer that understands the programming language consists of a hardware computer with its own machine language and the compiler that translates the programming language into machine language. To change the programming language, one rewrites the compiler. There are compelling reasons why this is a bad technology for implementing sublanguages, including programming languages. We present here a radically different technology.

Let the computer be a Universal Language Processor. (A hardware computer with a universal language processor replacing the compiler of a particular language, if you like.) It operates in two modes:

1. It accepts, one at a time, the rules of grammar and their associated semantic procedures that define the sublanguage, building them into its internal grammar table.
2. It accepts an input sentence, parses it according to the grammar, uses the resulting parsing graph to compose the associated semantic procedures, evaluates them, outputs the result, and cycles.

Thus it is a simple, straightforward implementation of compositional semantics. It differs from current practice in a radical way; namely, there is a single language processor that is completely separate from the declaration of any particular sublanguage. A sublanguage — any computer language — is declared as a collection of independent grammar rules, each with its own syntax rule and associated semantic procedure. Sublanguages are defined to the computer in terms of grammar rules, consisting of a syntactic aspect and an associated semantic procedure. (See Figure 1.)

A Typical Rule of Grammar
(as understood by the computer):

```
RULE
>Syntax: <noun_phrase> => <adjective> " "<noun_phrase>
>Semantics: POST    adjective_modification_procedure
```

Figure 1

Given the constituents of a meaningful phrase, for example: "government" and "contracts," the semantic procedure goes to the two associated data files and produces the "meaning" of the entire phrase: "government contracts". The rôle of syntax is to

show how words and phrases can be combined into meaningful statements. Once the syntactic structure of a sentence is seen, the associated semantic procedures can be composed appropriately. The rules of grammar, along with the corresponding semantic procedures, constitute the building blocks. Each of these rules is implemented as a separate unit. The syntax of a sentence provides the plan for combining these building blocks into the complex meaning of the entire sentence. Thus the individual semantic procedures can be efficiently composed in innumerable ways to produce the needed answers to immediate user concerns.

The notion of "phrase", as needed below, is simply that of a meaningful segment of the sentence. Thus in the sentence: "What is 3 + 4?", among the phrases are: "3", "3 + 4", "is 3 + 4", as well as the whole sentence. On the other hand, "3 +", and "4?" are not. "is 3" is indeed a phrase, but one that does not appear in the final analysis of the sentence. The formal representation of this notion as an abstract data structure, which includes a specification of its part of speech, constituent phrases, and the grammar rule involved, is central to the New World of Computing language processing. Instances of this data structure are ubiquitous aspects of language processing. We will refer to "phrases" below in this sense (it will be unnecessary to know further detail).

It is important to notice the implications of the independence of the grammar rules — syntax and associated semantic procedures. As said above, in building a sublanguage, rules are added one at a time. These same rule-adding utilities can obviously be used at any time to add an additional rule or, for example, a whole family of rules implementing a new object class. It is these same utilities that implement the user's ability to extend his own sublanguage by definitions.

A common consideration that arises when faced with an architecture of this vastness is the effect it has on efficiency. Let us deal with this immediately. In the current implementation of this architecture, against a moderate sized database concerning ships and shipping (the DARPA "blue" file of computational linguistics fame), and using a sizable grammar, the parsing time for the following sentence:

"What is the cargo type and destination of each ship whose port of departure was some Soviet port?"

is in the tenth of a second, the overall response time including database access is a few seconds. The key to this speed lies in the fact that in such very high level sublanguages, the object class data structures and processes are highly optimized, so that in processing a sentence one is composing a few highly optimized procedures.

2. THE PAGING MODULE

An "insider's" problem is to determine how the great number of highly complex procedures that may all be needed at some time or another can be retained in a form that makes them available for rapid response to a query. This problem is equally relevant concerning the many forms data resources may take — various database systems, text files, postscript and pixel files, etc. One way that has proved particularly effective is to use "pages" in peripheral memory, but pages that are organized on the basis of semantic content. In response to a particular query, only those pages that are required are brought into main memory — whether they hold data, procedure or other information resources. Pages holding all manner of material are brought into the same paging area. Obviously, procedure pages require a modicum of run time binding, but since the number of paging slots is large, there is very little trashing of pages between main and peripheral memories.

A page is a fixed-size block of mass storage. In the current implementation of the New World of Computing, these pages can contain indiscriminately database records, executable code, data about internal management, lexical and dictionary items, lists, digitized sound and pictures, strings, etc., in other words any type of manipulable data. This does not necessarily require that all the data that the System manipulates be loaded on pages. As we will see in Section III.A below, there exists an efficient "grain size" for any media, simplifying description, manipulation and transport of relevant database items; the choice of storing raw media data on pages or not will also be discussed in that section. The area of mass storage where these pages are kept is called a paging file. Copies of some of these pages will be in memory when they are in use. The area of memory where the pages are swapped in and out is called the paging area. When there are no unused page slots left in the paging area, a swapping algorithm selects which page has had the least use, and returns it to mass storage if it has been modified. For accessing data or object code stored on one of these pages, each page has a page address consisting of the page identifier and a byte offset on the page. The precise form of this address will be discussed below.

3. THE DATABASE AND OBJECT ORIENTED NATURE OF THE SYSTEM

The information available to the computer is organized into a network of "nodes" and "links". The "nouns" of a sublanguage point to certain of the nodes in this semantic net. The syntax rules also have a geometric interpretation in terms of the semantic net; they indicate how to move from one set of nodes to another. Thus the parser composes the path from the words in the initial expression of a question to the

nodes constituting the desired answer. The information about a node is kept on a database record on one or more pages of peripheral memory.

Organizing information in this way provides a highly efficient and flexible method for maintaining a rather shallow level of information organization (essentially equivalent to an entity-attribute database or relational database, plus inheritance). By linking such "database" records — equivalently the semantic net nodes — to more complex forms of representation (e.g., texts, pixel files, postscript files, engineering drawings) and by providing sophisticated semantic procedures that can exploit the additional complexities of these structures, the computer can give wide-ranging responses to highly complex technical questions. In the terminology of object oriented programming, these database records constitute the object representations for the single all-encompassing object class: "noun". Any hierarchy of subclasses of objects may be created, such as "image noun", "matrix noun", "covariance matrix noun", etc., with their associated processing procedures.

A new object class can be easily implemented (this is supported by the System) as a new subclass of the "noun" object class. When an instance of the new object class is created, first its record as an instance of "noun" is created, and then a link from this record to an instance of the data structure of the new class is added. As an example, suppose one were building a new sublanguage to be used by the structural engineers in an aerospace company. Suppose the company already had a major investment in files of stress data and, say, FORTRAN procedures that processed these files. The new object class, a sub-object class of "noun," would be created whose associated data structure was that of the stress data files. Syntax rules for noun phrases that engineers commonly use in referring to the stress data would be added,

together with their corresponding semantic procedures consisting largely of calls to the relevant FORTRAN routines. Such queries as:

"Plot the stress against wing tip loading for both Model A12 and Model A14 wing aileron designs"

would be immediately available.

4. THE COMMON TERMINAL VOCABULARY OF ALL SUBLANGUAGES

In today's highly visual world, sublanguages are seldom limited to written text. However how can this complete integration of media be implemented? Certainly the identification of the object class with its encapsulation of structure and process is a major step. Another step concerns the extended "alphabet" available to all sublanguages. All letters and characters of the usual alphabet as well as the entire extended ASCII character set; all graphic "events", such as clicks of the mouse and movements of the cursor; and all "interrupts" from internal and external sources (properly screened and identified) can be used in the input string that is fed to the language processor. All sublanguages have the same terminal vocabulary, namely this extended alphabet. Grammar rules can supply the recursive, flexible link between the input string and the internal object classes. For example, one can at any time introduce a new icon, placing under it any sentence or phrase of the sublanguage that then is evaluated in line whenever the icon is clicked during input of a query. Consider two illustrations of how this can work.

An airline mechanic, working on the radar nose cone of a Boeing 747 aircraft, turns to his computer for detailed technical support. He has already entered information identifying the particular aircraft he is working on, and has called for a

display of the nose cone area. The computer-generated photo image of the relevant area (plus an invisible back-plane drawing outlining all significant parts) provides a highly efficient medium for communication. For example, he may type "no hydraulic pressure" and click his mouse on the image of the valve where he just took a measurement. The computer may respond with the spoken word: "replace" and blink the sensor it identified in its diagnosis as the probable cause of the problem. In response to a sparsely stated but technically involved question, the mechanic receives an immediately useful response that reflects a high degree of built-in understanding.

Again a maintenance professional is using a professional, completely mobile telephone-computer which eliminates any need for the usual truck full of manuals. He types in:

"I am at 477 Oak Street.
Show me the electric panel wiring diagram."

The professional's efficiency is greatly increased, since the computer tailors its responses to the specific installation. Astute use of hypermedia links from one data display to another quickly provides pathways to the details the professional really needs. References that establish context (e.g., "I am at ..."), as well as pronouns and elliptic constructions (e.g., "What about the other connector?") play important rôles in effective dialogue. Note that pointing to and blinking significant areas in pictures and drawings constitute visual "pronouns" (e.g., "voltage 'there'?" or "tighten 'that'", "[show schematic icon] of 'that'").

5. THE CREATION AND BASING OF SUBLANGUAGES

The typical industrial manager will have many sublanguages, for example:

- Schedules and deliverables

- Budgets and fiscal control
- Personnel assignments and administration
- Correspondence

Underlying each of these, and a part of every sublanguage, are the general dialect of the manager's natural language, a complete graphics package, text editor, electronic mail, voice messaging, etc. Once he has chosen to use any one of his sublanguages, all of these services will be immediately available; the manager will not be aware of which service a phrase of his query may have invoked as he proceeds in his normal way:

"Send this draft budget to my section managers with the following message:"

`'...(voice)...'`

"Schedule a meeting with them sometime on Wednesday afternoon."

How are sublanguages created? Initially, there is one sublanguage: BASE. It contains a limited dialect of English which is adequate to handle expressions concerning typical relational or entity-attribute databases with inheritance. It also contains a graphics package, text editor, electronic mail, etc., as mentioned above. To create a new sublanguage, say "Finances," one "bases" it on BASE:

"base Finances on BASE"

Then, choosing this new sublanguage:

"enter Finances"

one has all the capabilities of the based upon sublanguage immediately available. One can then extend this new sublanguage in many ways (these will be discussed below).

Engineering manager E. D. Moore wishes to create a sublanguage to share with his three subordinate managers:

"base EngSec on BASE"

... add data, graphics, addresses, icons, ...

"authorize C. E. Jones, P. E. Smith and A. E. Johnson to enter EngSec"

Now any of the four of them can use, modify, and extend this common sublanguage "EngSec". Thus they jointly maintain a common, up to date view of their joint activities (e.g., preliminary designs, personal schedules). This is the significance of being able to enter.

There is a strong asymmetric relationship between a sublanguage and all of the sublanguages on which it is based, either directly or indirectly. Suppose one sublanguage "Accounting," is based on another "Personnel Accounting":

"base Accounting on Personnel Accounting"

Any changes in Personnel Accounting are immediately reflected in Accounting. However, Accounting can be changed in any way without affecting Personnel Accounting at all. This asymmetric relationship is characteristic of basing.

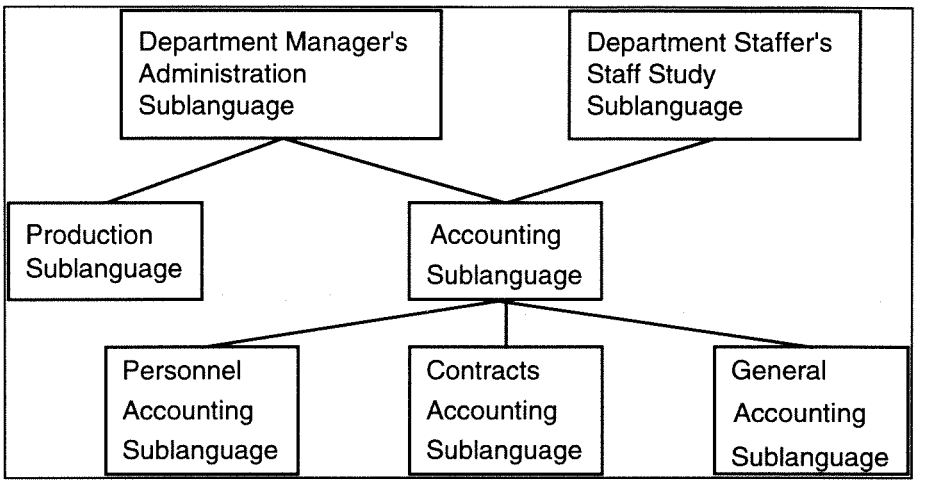

Figure 2

In Figure 2, showing the accounting sublanguages, the people in the Personnel Accounting Section are the only ones who are authorized to enter the Personnel Accounting sublanguage. Therefore they are the only ones who can change it. The procedure is similar for the Contracts Accounting and General Accounting sublanguages. Accounting is based on each of these three. No one is authorized to enter Accounting; therefore no one can make any changes to it. Of course, it is automatically always up to date with the latest data from Personnel Accounting, Contracts Accounting and General Accounting.

Appropriate managers are authorized to base on Accounting. One of the Department Manager's sublanguages is based on both Accounting and Production, and therefore always has available the latest accounting information. (See Figure 2.) The manager may well have had the application programmers add a number of grammar rules, graphic output formats, and icons so that overviews of the complete operation are always readily available. These added facilities would be available only in this particular sublanguage, but would always utilize the latest accounting and

production data. A member of the manager's staff, looking into the possible change in the pricing structure for company products, could also base a staff study sublanguage on Accounting, change many of the entries to values reflecting the new pricing structure, then examine the inferred results, and finally arrange appropriate graphics for a presentation, without, of course, affecting the Accounting sublanguage at all.

A detail of basing will be included here, since it will be referred to below. Suppose sublanguage A is based on sublanguage B and sublanguage B is based on sublanguage C; further suppose that in C we can talk about the class of cities, i.e., "city" is in the lexicon of C. In this lexical entry will be the page address of the database record for the class city. The lexicons of both B and A will, of course, also include the word "city." A's lexical entry for "city" will include the same page address that is in C's lexical entry; circumventing an indirection through B. (See Figure 3.) We will see that this has consequences beyond the obvious efficiency.

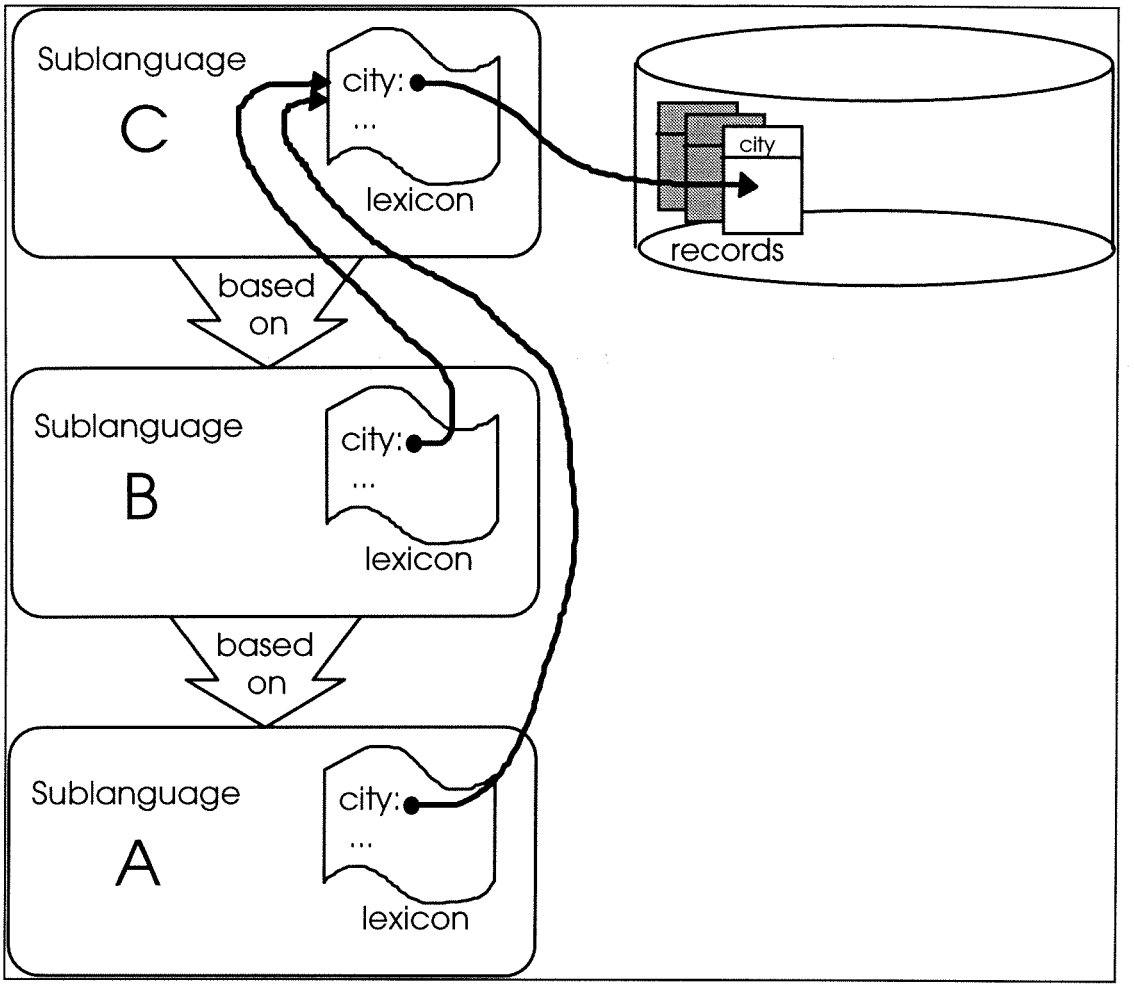

Figure 3

6. MANAGEMENT OF SUBLANGUAGES AND THE COMMAND SUBLANGUAGE

People are expected to have a number of sublanguages available on their telephone-computer. How do they manage and use them? In the long run, the telephone-computer, like the telephone today, will always be on. They will either be in one of their sublanguages or in the COMMAND sublanguage. In the short run, what happens when they invoke the New World of Computing application from say a

window environment? After the system logo appears, they will enter the following dialogue:

```
>Please identify yourself: John Jones
>Password:
You are in COMMAND. Proceed.
>
```

The COMMAND sublanguage is a limited, not very English-like sublanguage in which they can do a variety of sublanguage management things. They can ask for a directory of their sublanguages; and can enter any one of them:

```
>enter Correspondence
You are in Correspondence. Proceed.
>
```

They can also base a new sublanguage on an existing one. For any existing sublanguage of which they are the creator, they can authorize anyone else (who is recognized by the System) to enter and/or base on it. They can set the parameters of their profile, etc., and of course they can "exit" the System.

III. Implications of the Paradigm for Communications and Computers

(The remainder of this thesis reports on the author's research contributions.)

What we are talking about in this thesis, and in general in the New World of Computing project, is defining standards and procedures that will make access to information clean, fast and inexpensive. Even when the hardware technology of the telephone-computer is in place, the main deterrent to loading it will remain as it is today — the excessive cost of software development. However the cost of software development is a symptom; the problems that underlie the delay in the arrival of the telephone-computer are to be found elsewhere. A central problem is that of addressibility.

A. The Problem of Addressibility

With increasing frequency, the public press is discussing the telephone-computer — how it will provide access to "all the world's information" and bring into being the "electronic market-place". It is commonly conceived that one will be able to summon on one's monitor articles, encyclopedia entries, news material on any topic that might be of interest, and advertisements for innumerable products from endless sources. More disturbing, it is assumed, without further thought, that all of this

undigested — and in this form, indigestible — plethora of so-called information constitutes a solution to the information explosion problem. There is more and more evidence that such access, when available, is seldom used and has a very low cost effectiveness ratio. We are seeking capabilities in a far different direction.

In considering the question of access to all the world's information, the crux of the problem is the two-level address structure that is almost universally the standard today. Each computer has its own virtual memory (this is not strictly true in the case of distributed databases; but then the small group of interrelated computers has its own virtual memory). Beyond this virtual address space, the computer can address files in an essentially unlimited file space. In order to execute a program residing in a file, the file must first be brought into virtual memory, and then given control. Since the cross-references within the program have only been linked together within the boundaries of the program, interaction between programs becomes extremely expensive, consisting of passing information between programs through auxiliary file transfers.

The same problem is involved in the use of data. In order that data processing be sufficiently efficient, optimal data representations are developed, and the programs that access this data must know about the intrinsics of these data structures. So one may have a general purpose database system and special application databases. Both a specific database and the database processing system must be brought into virtual memory before they can be used, and then efficient processing is confined to only those parts of virtual memory. In commercial products, the communication between processes must be confined to data structures of such generality — most often ASCII files — that computations involving more than one process result in unacceptable

response times; while in one of a kind, enterprise-wide systems, the cross-linkages become so extensive that program maintenance costs soar.

When the computer fully understands the sublanguage of a highly trained professional, it is the computer itself that will need to be able to digest the articles, encyclopedia entries and tabular data that are relevant to the professional's cryptic instructions. Since the computer must access all the sources of information that it will use in responding to the user's input, it must have both an intrinsic understanding of the infrastructures of these sources and the capability to address into these structures. The two-level address structure that is used today enforces such high inefficiency-partitions between information sources as to make practically impossible any reasonable notion of access to all the world's information. This, then, is the problem of addressibility.

This point is of sufficient importance to this thesis that we pause to clarify it with several examples. There exists a specialized magazine, *Bankers News*, that serves the banking industry. Like many trade publications, it possesses a direct on-line electronic access from several commercial data services. Three successive issues of this journal contain separate tabular listings of data about Savings & Loan companies. Through simple queries one can display these issues on one's monitor; thus these issues can also be available in ASCII file form. Suppose the user wished to ask a question whose answer required the computer to access, among other sources, these various tables and to carry out calculations on these entries. There are several methods within the current state of the art that could be used. For example, the user visually scans the data file, finding what he wants to extract, dumps the screen to a file, and edits it or cuts and pastes from the screen to the relevant application. This is slow and distracting, and the reliability will probably significantly decrease when it has to be

done repeatedly. Alternatively, a program could download the whole file into the user's virtual space where it could be appropriately massaged, and relevant information extracted. Here the response time would suffer badly, not to mention the amount of virtual space that would be eaten up by a query of any complexity.

Two other alternatives impose the chore of extracting the appropriate information on the information server's computer. Then the user's computer picks it off the section of the screen where it is known to appear, or it uses a list of indices that the information service must keep up to date . These two solutions are efficient enough to have had some success in highly specialized, narrow areas serving a broad clientele (e.g., New York Stock Exchange ticker), or in applications where high costs can be afforded (e.g., military or space). As stand-alone applications, they have enjoyed a growing popularity, but whenever the data had to be used in an integrated environment, they have put an undue burden on either the customer or the provider. Their software costs quickly become prohibitive, because the ever changing aspect of information requires more and more specialized access subroutines in the first case, and index upkeep time in the second.

In all of these methods, someone must attend to the interface between each specific source material and the application. All suffer from at least one of the following deficiencies: necessity of user intervention, considerable waste of storage or transmission resources, need for individually tailored access subroutines, and software upkeep cost. An equivalent to the functionality of addressing, to be proposed in this chapter, can be implemented without our described infrastructure (and already is in some specific cases). In the case of textual material, it includes indexing into ASCII databases using keyword and header information. However, this in itself is not sufficient. The cost of implementing, updating and maintaining these databases, the

cost to the user of a specialist to stay in touch with these constant changes, and their access inefficiencies are common experiences that illustrate the need for a new addressing scheme of the nature we propose — a scheme that would allow very specific and specialized databases to be easily integrated into the world's virtual memory and thus made available to a wide audience at an affordable cost.

B. What Needs to be Addressed

What kind of things need to be addressed? There is no fundamental need to access the smallest atom of data the computer can handle from the sublanguage's perspective. Neither is there a fundamental requirement that each cell be identical in size. What does a sublanguage need to address? Clearly one group is made up of the atomic elements constituting the objects in its discourse. There also exist links between these objects, as well as links tying these objects to data types, such as images, texts and messages; these links are addresses. However, in a sublanguage concerned with fine art, talking about individual pixels forming the portrait of the *Mona Lisa* has no relevance. In another perspective, when the subject of discourse is the dashboard of a car, as in a driver's education application, a large portion of the picture is also irrelevant. What is relevant are the objects of discourse you can point to, manipulate, and ask questions about, such as the steering wheel, pedals, turn signals, indicators, for example. The manhole cover you can spot through the windshield will not be part of the sublanguage. Neither will the sky, or the plush rosewood trimming. The need to address a whole picture is an operating system requirement (what file is it stored in?); the need to address the individual pixel is a hardware graphical subsystem requirement. The sublanguage requirements are to subsets of pictures, groups of relevant pixels. The way this is implemented in the New

World of Computing System is by associating each picture with an invisible set of polygons that delineate the various relevant areas. Each polygon is linked to its associated object in the database, thus can be linked to —"addressed from"— any other object in the database, and the use of a whole new set of vocabulary becomes instantaneously obvious (blink, show, display, highlight, plot), as well as a whole new set of input primitives because clicking on any of those polygons refers to the underlying object in a manner that is equivalent to giving its name. For example, saying:

"Which is the cheapest of these?"

and clicking on a handful of items on a page of a catalog, has the same result as typing:

"Which of the following VCRs is the cheapest: Model 1241, Model 1243, Model 1841 or Model 1843?"

Similar scoping of relevant data objects exists with other data types, as in a text, where addressing down to the character is unnecessary in the great majority of cases. However by providing a mechanism for instances of words or groups of words in the text to be linked to the database, a hypertext style application becomes trivial to implement. Imagine a puzzled mechanic working on an unknown car engine. He turns to his peripatetic computer and asks for a display of the engine for that particular model. He then types:

"How do you replace that?"

and clicks on a particular area of the diagram. The computer retrieves a specific paragraph from the manual that is kept in the documentation repository of the car's manufacturer. The paragraph is displayed on the screen. It starts like this:

Disconnect the nozzle of the turbo's loopback oxygen sensor.

. . .

A little befuddled, our mechanic just needs to click his "help" icon, and point to the words he didn't understand and the corresponding item on the diagram will flash.

In the case of drawings, it is "subdrawings" that are the addressable elements. If we limit our attention solely to the graphic aspect, "subdrawings" are like subdrawings in other common graphics packages. A subdrawing is composed of a drawing and a transformation matrix that defines the position of the associated drawing in the higher drawing. However, the concept of subdrawing here stems directly from the idea of items having conceptual relevance in the sublanguage's area of discourse. Thus in addition to its graphic aspects, it has an associated object in the semantic net database. As such, it also can (and usually will) have relational tie-ins to the rest of the sublanguage's elements since it is, in and of itself, a sublanguage object. As you would expect, this is fully recursive: the drawing associated with a subdrawing can itself have subdrawings. An example of a drawing would be a diagram of an amplifier. The subdrawings are the components and the connecting wires. Queries about the subdrawings are in the nature of:

"What is the amplification factor of this transistor?"

"What is the frequency of this crystal?"

"What is the impedance between here (clicking on the subdrawing of a pin) *and here* (clicking on an input of an op-amp)*?"*

This transistor, this crystal, the output-stage op-amp, its pin 4 and ground are all subdrawings. The transistor subdrawing has an associated transformation matrix that

reduces it to the diagram's scale, rotates it 90 degrees counterclockwise, and translates it to the appropriate spot. Its associated drawing is an instance of the N2661J transistor drawing located in the Texas Instrument transistor database in Dallas, Texas. That drawing is composed of four subdrawings: a vertical thick line, an emitter subdrawing, a collector subdrawing and a base subdrawing. By basing on this Dallas database, the diagram designer has inherited all of the characteristic vocabulary pertaining to the transistor sublanguage, so that the amplification factor of this transistor is automatically part of the designer's sublanguage.

So far we have looked at the granularity of identifiable objects in the few media that one commonly encounters in the computer world. Similarly, the question of the appropriate granularity comes up upon the addition of every new object class. The design decisions then determine to a large extent what will be conceptually "visible" to the end user of the sublanguage under development. So for every medium and for a given sublanguage in which that medium is an object class, one can usually identify a relevant granularity that conveys the optimal amount of information from the viewpoint of that sublanguage. Since the result is a sublanguage of practical significance to people, the total number of objects available in a sublanguage as well as the amount of data storage associated with a given object will be under reasonable control, i.e., without choking the communication channel. In fact, it will often be the case that computer load and response time will play a major rôle in determining the object granularity, and thus what the user is able to "see". (That is an aspect that needs to be addressed.) Establishing the granularity for a medium isn't always an easy affair, as is the case in an animated or video sequence. Now, current tools only allow the description of a set of successive frames. This is not optimal because it is usually the case that a number of relevant items persist from frame to frame while many other

items on the same frames are totally irrelevant. Being able to keep track of relevant items through successive frames is a difficult task to automate, and an extremely tedious one to do by hand.

It is noteworthy that many, if not the majority, of objects referred to by sublanguages do not have "names" in the lexicon. Consider the marriage of Edward D. Moore and Patricia Jones Moore. Friends of the Moores often refer to their marriage; for example, "her daughter by her second marriage," but it does not have a name. This technique is especially applicable to objects associated with graphic media. For example, "Texas Instrument's N2661J transistor" may be the name of a class of all subdrawing objects as they exist in a great variety of circuit drawings. None of these subdrawing objects need have a name. They can be individually identified either by a definite description or by clicking on them in the display of the circuit drawing. In the later case, one can identify the particular transistor either by:

"the N2661J transistor used in the output-stage of the amplifier"

or by pointing at the transistor and clicking the mouse while viewing the circuit diagram. In either case they will inherit the properties, such as amplification factor, of the N2661J transistor class. Although this does not reduce the number of objects in the sublanguage, it does keep the lexicon uncluttered with useless names that would never be referenced. This in turn speeds up lexical lookup and thus the response time.

As for auditory media, even if the technology to generate them is becoming more and more available, their storage costs are still too high to generalize on a large scale, and the technology to use them as input is available only for limited situations. So even if the granularity is fairly obvious, creating the linkages can sometimes stumble on technological barriers. In educational applications for example, (an area

not known for the depth of its budget,) a typical application could be a spelling bee program. The application would take a word at random from a dictionary and enunciate it to a candidate, expecting a typed answer. Most grade schools cannot currently afford the hardware necessary to store such a vast amount of digitized speech. Text-to-speech can alleviate this problem, but there are other more interactive applications where this remedy will not apply.

C. The Requirements for Global Addressibility

Let us go back to the example of the car mechanic. Notice when he made a request for information, the most up to date data came directly from the manufacturer's database. The program he was using knew how to access that data, where it was located and how to retrieve it. Obviously that type of program should be available to any mechanic anywhere in the world, without having to provide explicitly at every location the specifics of the network path to each item that may be desired. We will now look into the requirements necessary to make an application oblivious of the specifics of the network. The principal requirement is an addressing scheme that uniquely identifies each data repository site, independently of the network's size, configuration, evolution or location. If two networks are merged, each station in the network, or at best one server in each network, has to become knowledgeable about the addresses the other network contains. Sometimes this process even requires the station or the network to come down. If the two networks had common addresses, the merging would become an even more complex process requiring modification of the network databases. The scope of the addressing scheme must thus be worldwide. This implies that each station has at least one address, and that each address is unique.

It turns out that such a network already exists. We refer to it as the telephone network. It spans the whole world and each station has a unique number. By mapping the telephone number into our address, we are able to point to a data item located on any connected computer effortlessly. The costs of tying into the network are minimal. A station can be mobile, or can be relocated easily using the current mechanisms of cellular phone and call forwarding. The only requirement is the ability to transfer packets of digital information. This is already available, but higher bandwidths and faster call setup times are required for a smooth operation. The bandwidth already exists in the phone companies' fiber optic trunks. The current bottleneck is in the twisted-wire pair between the station and the local exchange. Until the fiber optic pipeline reaches the individual home or office, we will have to rely on high speed modems to the local exchange.

The call-setup latency poses more of a problem. In its current state, the phone network has a setup time of three to four seconds, and can only multiplex two calls at a time. This situation encourages few connections, and large amounts of data transfer per connection (i.e., long connection times). It virtually enforces monolithic applications where most of the information gathering is done on large servers that have the connection capabilities to do so in an efficient way and where the station-to-server link is used essentially for display purposes. Since the connection price is so high, even the servers have a tendency to limit their sources to an "optimal" set that is not always in accordance with the customer's preferred choice. The mechanism is the same as the one that prevents us from calling all the stores when we are bargain shopping: it takes too long to gather all the information, so we sample a few well-chosen sites.

The ideal solution is to bring packet-switching technology to the station, since this would give the computer a zero call-setup time and infinite multiplex capability, physically allowing the answer to a question to be derived from several data sources. An example would be to ask for the fastest way to go from one city to another at a specific date and time. It could turn out that the train gets there sooner because it leaves more often than the plane. This could be discovered in a single query that would go out and interrogate both schedules. We should keep in mind that all these issues (using the phone as a data network, high bandwidth and quick call-setup time by packet-switched network) are already technologically addressed in the Integrated Services Digital Network (ISDN). The main problem left is massive acceptance of the standard worldwide. Another important requirement is for a uniform information packet at the application layer over the whole network. Now this conflicts with the widespread belief that at such a high level the programmers should be insulated from such mundane considerations. Reality shows us that even with our exponential growth in computational speed, size of memory and communication bandwidth, the size and complexity of information grows in parallel. So that our real-time latency on a per-dollar basis is no better, if not worse than it was ten years ago, even if the quantity and quality of the information we are handling now are much higher.

With this in mind, a standard sized packet is herein specified to optimize data transfer and simplify networking sub-routines. The actual number was derived from packing considerations on the grain sizes of the various data structures used by the system. However at least one level of the application must be knowledgeable of the packet size, in order to pack and unpack data efficiently. Another advantage of the uniformity is that it enforces standardization over heterogeneous hardware platforms. Moreover, if a standard storage format is used on the packet, an automatic cross-

platform compatibility ensues at minimal cost. This would allow computers not only to share data independently of their architecture, but also to understand similar structures and have similar displays over similar media. What we have done is reinvent the paging concept on a world-network-wide scale.

In current database technologies, the database consists largely of links from one node of the database to another. The "links" in our semantic net database consist of page addresses. Thus for a network of more than one telephone-computer, the links within a database may refer to pages that reside anywhere in the net, anywhere in the world. A database in such a network is "distributed" in an intrinsic way; the basing procedure implements this.

D. Page Address and the Size of Virtual Memory

If there is to be a single virtual address space for all of the world's telephone-computers, how large must it be? In this single virtual memory, addressing should be down to the individual byte, but certainly not down to the individual bit. This consideration implies that the length of an address pointer be an integral number of bytes, since such addresses must be on a byte boundary. The page will be the principal unit to be sent across the telephone network. Each page will have its own "home" — the telephone-computer that owns the page and in whose page file it normally resides.

Thus a page address has three parts:

- the byte offset on a page;
- the page number identifying the page among all the pages residing at a given station;
- the telephone number uniquely identifying the station.

The first two constitute the intra-station part of the pointer, the latter the inter-station part. (See Figure 7.)

How large should each of these numbers be?

The page offset needs to be able to address any part of the page down to the byte level; thus the number of bits it requires is equal to log2(page_size). Because of the various uses of pages, each use having an optimal page size governed by storage versus performance issues and with strong indications of being sublanguage dependent, determining the optimal page size overall is a multivariate problem. It would be desirable to have statistics from a field test on a real life application. However that has yet to be done.

> "Neither extrapolation techniques for current trends nor normative approaches work well in this time frame. As a result, only large-scale field trials can provide data with which to gain insight into such areas as work at home; family applications; applications to the disadvantaged; democratic processes; social services; transportation-communication tradeoff; mass media impacts; new employment options; social engineering." [HILTZ88]

The size should be an integer multiple of the packet size of the telecommunication standard that is now 512 K. Our experience with data indicates that the size should be between 512 and 4 K; currently the New World of Computing uses a 2 K page size. We will assume this page size here; thus the byte offset is 2^{11} bits.

What is the total number of pages necessary on any given station? Here are several considerations that bear on the determination of this number.

- The average human brain uses less than 20,000 vocabulary items; a highly educated brain 35,000 to 40,000. These serve him for the transitive closure of all his sublanguages. Rough analysis of our current experience with the System indicates that each vocabulary item requires on average about five pages of supporting structures (i.e., lexical entry, dictionary entry, semantic procedures, database records, associated graphic structures, and so on). This suggests that a New World of Computing station with over 100,000 pages would possess more than enough storage even for the most well-versed of scholars.
- Even when large user applications like spreadsheets and word processors become integrated with the paging aspects of the System, it is doubtful that they would require orders of magnitude more space than they do nowadays, especially since paged code provides easy elimination of code duplication that is pervasive in software originating from multi-disciplinary companies. An upper bound of a million pages would seem adequate.
- Unfortunately, there appears to be a tendency on the part of people using bodies of data not to rely on information that doesn't possess a quality of completeness. Given a choice of phone books, we will usually choose the thickest one, even though we will never have need for all the names that are entered into it. In the same sense, we tend to populate our databases in ways that would give human users a severe case of information overload, forcing dependence on the search capabilities of today's information technology. There are many reasons why these very large databases are bad, one of them being the Herculean effort required to maintain a reasonable level of data integrity. A much more cost effective design is to preserve the sublanguage

structure that this type of data inherently possesses — or else it would not be manageable by humans — and choose as the person responsible for each of these sublanguages the one with the greatest involvement. This type of database modeling is ideally supported by the basing concept. Since these large databases do exist, the need to support them and to be able to assimilate them into the New World of Computing System needs to be addressed. Examples of very large databases include credit card databases that can hold several million items. Textual reference databases like those available through DIALOG, by successive mergers, have coalesced to even greater sizes. To assimilate similar databases into a New World of Computing sublanguage would require millions of pages.

- An upper bound will certainly be the amount of physical mass storage accessible by a single computer in the near future. With banks of gigabyte drives and optical disks the concept of a terabyte storage system is conceivable. With new mass storage technologies appearing every year, it becomes a definite possibility.

These considerations suggest the bound on the number of pages per telephone-computer be somewhere in the range 2^{20} to 2^{40}. Thus adding page offset, the number of bytes per telephone-computer is in the range 2^{31} to 2^{51}.

What bound can be assumed for the number of telephone-computers world-wide? How many will the present format of the international telephone number accommodate? An international phone number is decoded by an algorithm roughly equivalent to a prefix grammar. It is not defined precisely as a prefix grammar because there are some ambiguities that are resolved with string length or time delay

resolvers — i.e., input is finalized upon reception of a special key, or a lengthy enough pause — but these exceptions are rare. For the purpose of this analysis, we will assume that an international number is constituted of a maximum of twelve digits that are usually partitioned into country code prefix, area code prefix, service area code prefix and actual phone. For example, a long distance call originating from the United States requires a one digit long distance code, three digits for the area code, three for the service area code and four for the actual phone. This number is designated in the telephone lingo as a 1-NPX-NXX-XXXX number or a 1-3-3-4 pattern. If the destination of the call is Germany, then the number has a 2-3-3-4 pattern. For Japan, the pattern is is 2-2-4-4. Twelve digits amounts to a theoretical upper bound of one trillion for the number of phones in the world. The practical limit is substantially lower because of the idiosyncrasies of mapping the real world into a compact model. The possibility of expanding on the right by adding a new digit always exists but is limited by the enormous investment into the current structures, the inertia faced when trying to modify human attitudes and perceptions, and the difficulties of orchestrating a simultaneous changeover with a minimum of disruptions. For these reasons, such a change is addressed uniquely as a last resort and has been attempted only at the level of a country, never yet on an international scale. To map all the potential phone numbers into a single number would take a little under forty bits.

We mentioned earlier that all American phone numbers follow the NPX-NXX-XXXX rule. The letter N stands for a non-zero digit, the letter P stands for a zero or a one, while the letter X can be any digit. This rule encompasses a theoretical maximum of $2 \times 9^2 \times 10^7 = 1620$ million phone numbers. At present there are approximately 558 million phones in the world, i.e., about 2^{29}. This indicates how sparse the current

phone number tree is. There are several reasons for this. One is that the syntax for decoding a phone number being equivalent to a prefix grammar, whole limbs of the parsing tree are ripped out by special purpose prefixes. For example, all the prefixes that don't abide by the NPX rule, and even for some that do like the ten thousand potential numbers with a service area code of 911 that will never exist. Another reason is tied to the blocking factor. Each code hierarchically subdivides the world population into blocks but these blocks often conflict with geographical or political areas that cannot be conveniently split. One area code that can cover 10 million numbers could very well account for the population of North and South Dakota. In the same vein, countries like Liechtenstein with a population of 28,000 will never have the use of the 10 billion numbers their country code (41) represents. Yet another reason can be frequently encountered in the United States where alternate long distance companies need some distinguishing technique to service the same phone number for accounting purposes. The most general method is to use a special dialling pattern, followed by the customer's identification number and finally ending with the requested phone number. For these reasons, 2^{40} is somewhat too large.

Ten years ago, there were 400 million phones in the world, now there are approximately 558 million. This represents a 40% increase, mainly occurring in the industrial countries that average four phones for every five persons. This explains why 87% of the phones in the world are used by only 15% of the world's population. (See Figures 5 & 6.)

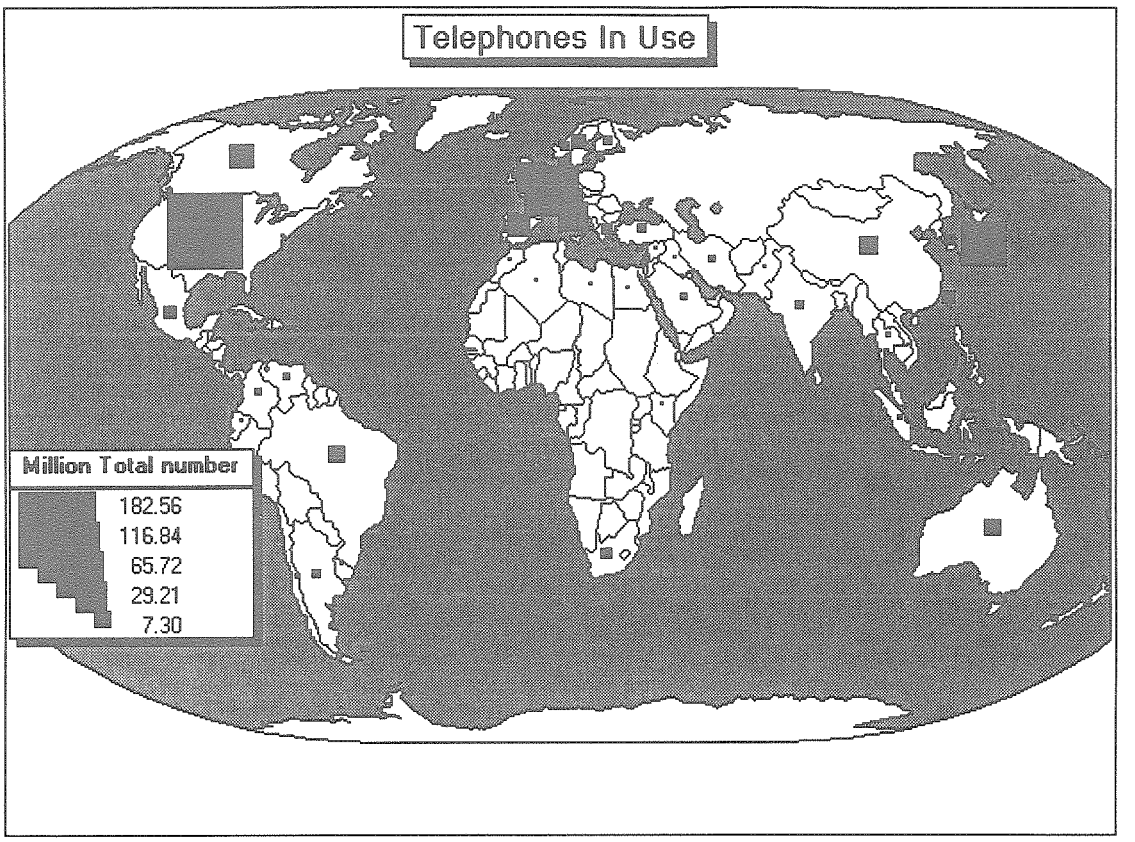

Figure 4

"Statistics indicate that while the number of letters and of book and newspaper publications has not changed appreciably during recent years, domestic and international telephone traffic is increasing at an annual rate of 15% and 25%, respectively." [INOSE]

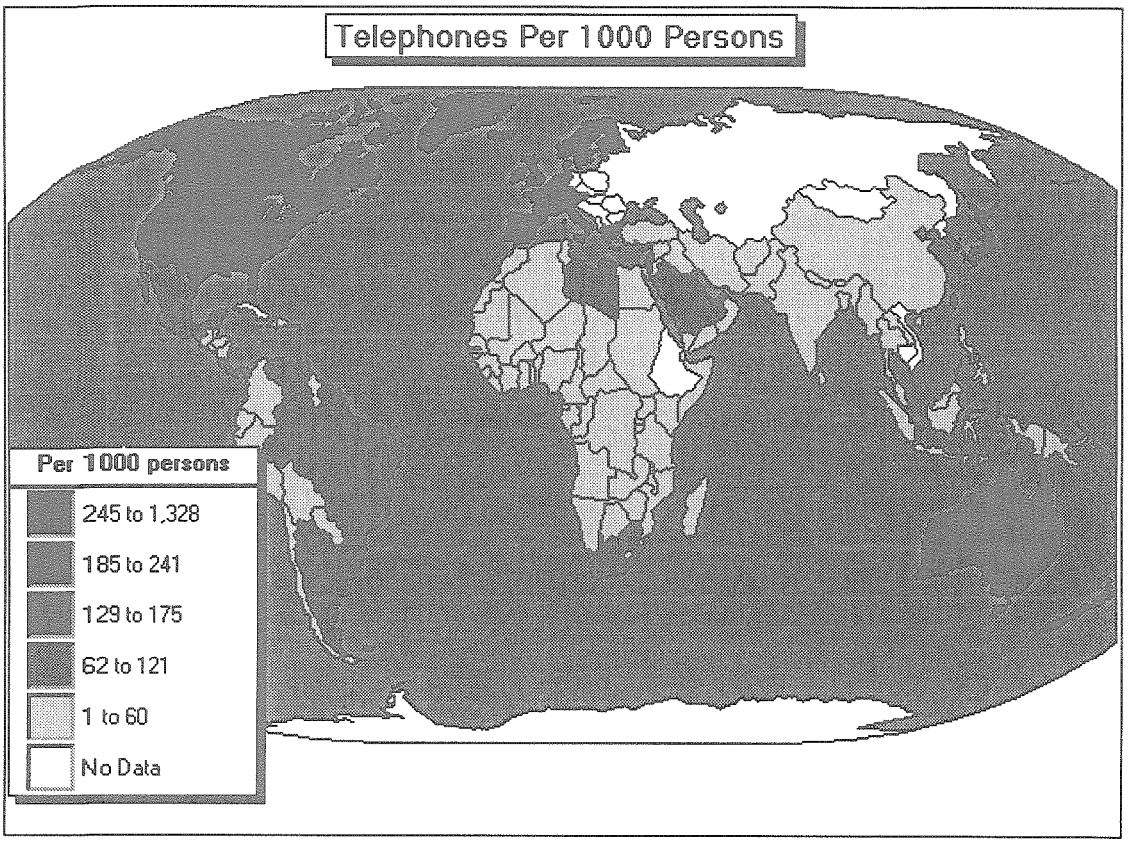

Figure 5

> "According to the statistics, an American makes some 800 telephone calls and sends some 400 pieces of mail per year. A Japanese makes some 400 telephone calls and sends some 100 pieces of mail per year." [INOSE]

How these numbers will evolve in the future is anybody's guess, but a reasonable estimate would forecast a curbing of the demand as it approaches the two phone-per-person mark. That would be close to market saturation, especially if one takes into account the fact that the New World of Computing System will eliminate the need for duplicate requirements per household (like facsimile devices). This estimation makes allowance for the fact that 33% of the world's population is under

fifteen years of age. The lower needs in telephony of that segment of the population (with the well-known exception of American teenagers) will offset the requirements for duplicate phones at the workplace, in public areas and for providing service. It also assumes an enhancement of the cellular phone trend, which reduces the need for duplicate phones, since a portable phone can always be present at one's side, within a cellular service area. With the advent of projects in the spirit of the Motorola Iridium project that is in the process of establishing a belt of 77 geo-stationary communication satellites, worldwide cellular service should be commonly available.

Taking the above elements into consideration, we have developed three near term models of possible growth patterns of the phone infrastructure to estimate what kind of reprieve various numbers of bits allocated for telephone numbers could buy us (see Figure 7.) The overload criterion that was selected was the 1:4 compaction ratio for the packing algorithm used to store phone numbers into the address of a New World of Computing System page. Historically, this seems to be a point where phone number allocation becomes cumbersome. For example, in France it was close to a 1:3 ratio when it finally extended its old 8-digit system. The three models tend to represent the three possible trends of the world economy: average, pessimistic and optimistic. We will look at one billion telephone-computers — 2^{32} — as a point of comparison (with a packing factor of 1:4).

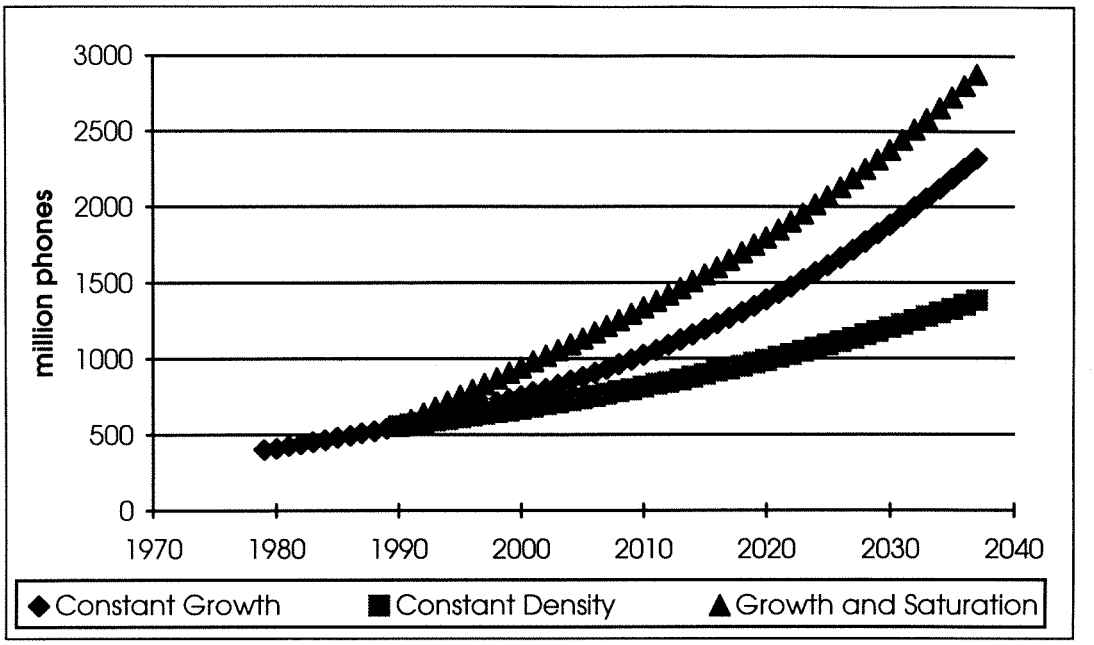

Figure 6

The first model (Constant Growth) is a mere projection of the past ten years' growth into the future. It assumes that market trends of the past decade are still dominant and will continue to dominate for the next few decades. It forecasts a crossover of the one billion mark in the year 2010, or fifteen years from now.

The second model (Constant Density) assumes a stagnant economy that will keep the demand for new phones low, and will thus leave the number of phones per capita stable. The only growth comes from population expansion. This unlikely situation is a good lower bound on the total number of phones. It predicts problems with the one billion level by 2021.

The third and more elaborate model (Growth and Saturation) estimates that demand for phones will grow in the near future but will peak out when the phone density reaches a certain level. It distinguishes between the industrial countries that have a rapidly growing market, and the developing countries whose initial growth is

slower but that are much farther away from saturation. This rapidly expanding model predicts trouble on the horizon of this millennium, i.e., 2002.

The considerations we have recorded in the last two paragraphs give us a feel for the situation. However from a practical viewpoint, taking into account the requirement that a page address must always fall on a byte boundary, there is very little latitude; the address should be either 8 bytes or 9 bytes long. (Now recall that we have estimated that bounds on the size of each telephone's virtual memory should be between 2^{31} to 2^{41}, and that the current telephone number format will need to be changed when the number of telephones reaches approximately 2^{40}.) The choices are given in the following table:

		Number of telephones	**Size of each telephone's virtual memory in bytes**	**Saturation deadline (third model)**
8 byte address		512 million (2^{31} / 4) (current number)	2^{33}	now
2^{64}	**A**	2 billion (2^{33} / 4)	2^{31} (minimum)	2024
9 byte address		512 million (2^{31} / 4) (current number)	2^{41} (maximum)	now
2^{72}	**B**	16 billion (2^{36} / 4)	2^{36}	at least another century
		2^{40}(current format)	2^{32}	many centuries

We will select rows A and B as the two best alternatives to be considered.

Row A: These bounds are very tight both for the number of telephone-computers and the number of bytes of virtual memory at each telephone-computer. Although it is hard to

imagine that there will be a four-fold increase in the number of telephones and at the same time a complete transition from telephones to telephone-computers in just thirty years, it is equally hard to imagine the societal impact that this new technology will bring. The bound will most likely be reached in only a few years beyond 2024, and a conversion of all addresses in all files on all telephone-computers, world wide, would be a momentous undertaking. As for the size of virtual memory at each telephone-computer, few, out of the billions of telephone-computers, would use anywhere close to two billion bytes of memory, but it is not difficult to imagine that a sizable number would. It should be kept in mind that we are considering an inelastic bound.

Row B: The bounds here are reasonable. Certainly there will be many applications of computer technology where computers will indeed require much larger memories than 64 billion bytes, but most of the contents of these memories will not be on New World of Computing pages. As far as the number of telephone-computers is concerned, we are beyond any imaginable circumstances of future change, both societal and technological.

The one question remaining to be addressed in consideration of rows A and B are the various costs in response time and storage for adding that extra byte. An

increase in field size from 8 bytes to 9 bytes is an increase of 12½%. Therefore, various aspects of the system like the indexing of data records or the list processing area, will be directly affected in a proportional way. In other respects, due to the highly blocked structure of data storage in the New World of Computing, the overall impact will be less, the increase in size being absorbed by empty space at the bottom of the blocks. The greatest hindrance, however, will be generated by alignment problems on platforms that are aligned on sixteen or thirty-two bits. Thirty-two bit optimizing compilers can generate extremely efficient machine-code for handling (moving, comparing, testing) 8 byte structures. On the other hand, a 9 byte structure will require at least one extra machine instruction per handling instruction, hence a minimum of a 50% increase in handling time. Profiling the New World of Computing code has shown that it spends between 9% and 15% of its time doing this type of structure handling. We can expect an average loss of around 6% in response time.

Storing these 9 byte structures will also generate a performance versus capacity trade-off. Packing these structures on pages in a compact manner will prohibit the machine from directly using them, since this will guarantee that half of them are non-aligned on a sixteen bit aligned platform, and three-quarters on a thirty-two bit aligned platform. This will force these structures to be copied every time they are used, introducing a significant amount of extraneous code, again degrading performance. On the other hand, storing the structures so that they are aligned on the page would require an increase in storage capacity of 11% to 33% depending on alignment of the platform. This would be a prohibitive cost. The degradation of performance of the former case will be compensated by improvements in both MIPS ratings of PC and workstation computers, and changes in transmission times. Going

to ISDN, then to BISDN will simply dominate out the effect of going from 8 byte to 9 byte fields.

On the other hand, a changeover from 8 byte fields to 9 byte fields any time in the future, especially after a great deal of the world's information resources are put on New World of Computing pages would be enormously expensive. Even if the product development time is slightly greater because of these alignment considerations, the choice is overwhelmingly on the side of a 9 byte field.

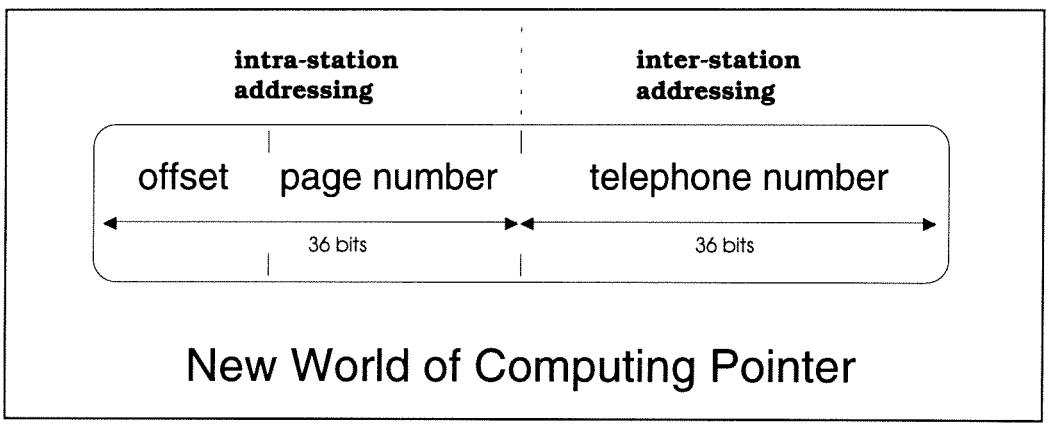

Figure 7

E. Networks in the Telephone-Computer Era

Basing one sublanguage on another establishes addressibility among the data and process pages that constitute the physical manifestation of these sublanguages; basing results in the sharing of a common address space (down to the byte level) across the entire hierarchy of associated sublanguages and whose key element is the addressable page. In processing a query or command, pages whose "home" is on one station are commingled with pages from another. Each individual item of information in a network of sublanguages is to be found at some offset on one of these pages. The

stations whose peripheral memories are the depositories for these pages are uniquely addressable by their telephone numbers. Therefore, each item has a unique address by which it can be identified: its byte offset on the page, its page number and its station telephone number. It is these hierarchies of sublanguages with their associated common address space that constitute networks, and not computers. In a single computer, a person may have many sublanguages, each in its own network.

When one installs a new telephone and is assigned a new telephone number, one is thereby automatically assigned one's own slice of "virtual memory," one's own corner of the world's address space. This gigantic virtual address space, spanning the whole world, eliminates the need for the wasteful redundancies of today's configurations. Compare the total memory configuration of any two computers of the same type. An amazing percentage of their mass storage space is identical: same operating system, same firmware, many identical general software packages. If generalized over all computers, the amount of unique memory becomes relatively small, and can fit without strain into the two gigabyte virtual address space of each/all computer(s).

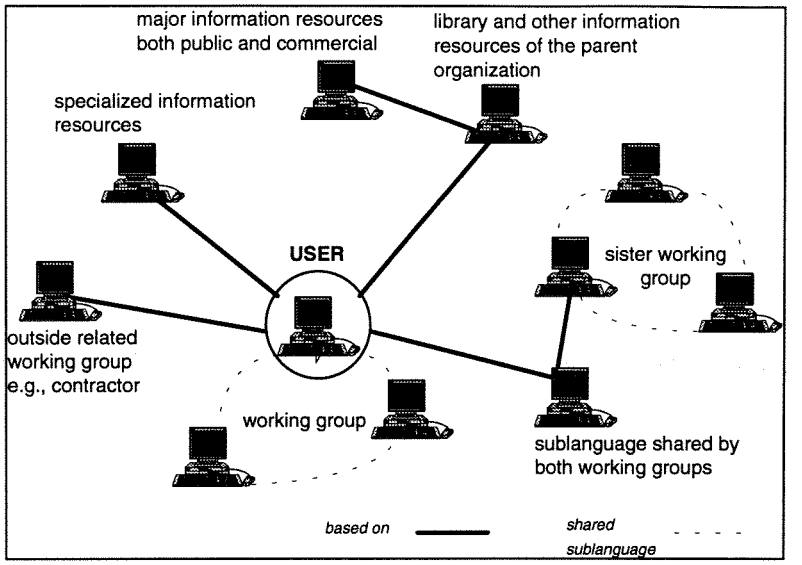

Figure 8

Figure 8 shows a schematic of a typical sublanguage network. The inclusion in these networks of large volume servers of archival information will be typical. We have seen this need in maintenance situations. Companies like Mead Data Central in Dayton, IMSL, a Houston-based supplier of scientific subroutines, Springer-Verlag's *Bielsteinshandbuch der Organischen Chemie*, cookbooks and garden catalogs, the New York Stock Exchange stock closings and the show records of the American Kennel Club will all be available, page by page as needed by thousands of users. A department store, serving several hundred thousand charge customers, will do so through a high transaction server that makes available all manner of sales material, processes incoming queries and orders, and connects customers to knowledgeable sales personnel.

Here is another maintenance professional working for a local service company. The professional's mobile station is in at least two networks: the first, previously illustrated, is with the home maintenance station whose server has all the records for the field locations served and all the maintenance information. The second is the

dispatcher network. The central dispatcher can see on the map displayed before him the location of all maintenance trucks as they move about the city. When the dispatcher receives a call requesting maintenance service, he can type in the caller's address and immediately see its location on the map, spot the nearest maintenance unit, click it with his mouse, and talk with the maintenance person directly to coordinate the new service call.

We have identified what needs to be addressed. We have specified a universal address space and seen how a 72 bit address structure is more than adequate to provide a single worldwide virtual memory. The question remains: how is the addressibility problem for all of the world's information to be solved? We identify the notion of the Archival Station. You call up a station that supplies information you wish to be accessible to one of your sublanguages. A form appears on your monitor and you are asked to fill it out. After doing so, knowing your charge number from your personal data page, it sends you an identification request that you can satisfy by either giving a password or sticking a microchip card in the slot if your telephone-computer is provided with one. You are then free to base your sublanguage on any material of interest to you and available from this source. Your initiating call to the Archival Station provides it with all it needs for billing, notification, etc. The act of basing automatically identifies authorization information necessary for security. The Archival Station only infrequently initiates a call to you. Your computer calls it, requesting a page. Since the request is sent as one of your own pages, it carries not only the requested page address but the return address as well. Thus it carries all the information the Archival Station needs to identify you and your account and provide you with the information you need. In this manner, the Archival Station information resource sublanguage can be "in" as many "networks" as there are clients who wish to

have its resources available. Since clients' sublanguages are based upon this resource, it itself is protected from change.

Billing services for the use of these pages are handled by the telephone company in the manner of "900" numbers today. Suppose a software house in Chicago is marketing a CAD/CAM package and that this package calls a differential equation solving program marketed by a Houston firm. Someone in Los Angeles contracts for the CAD/CAM package. Los Angeles bases on Chicago which in turn is based on Houston. It is only when an engineer in Los Angeles uses the design package, and pages carrying the relevant code go across the telephone lines, that any charges accrue. Page requests for CAD/CAM procedures will be filled from Chicago; when the Chicago procedure requests a Houston page, that request materializes as a direct page request from Los Angeles to Houston; thus the billing, handled completely by the telephone companies involved, correctly compensates each of the two software firms involved.

All telephone-computer users in each of their sublanguages have complete freedom to choose and base on whatever information resources they desire, paying for access to only those pages required in the course of their processing. Furthermore, this information does not come as isolated independent services (as for example, in the French Minitel System where each of the 20,112 services has its own impregnable virtual space). Any number of such resources may be integrated in response to a single user query in a single sublanguage. This may be a sublanguage a telephone-computer user has developed in conjunction with one of his particular interests, having personally selected the several information resources it has been based upon. The process of adding such resources and of extending and modifying the

sublanguage and its data in many ways then becomes just part of normal day-to-day activities.

IV. Some Details on Implementation

A. Pages and Headers

The paging structure of the New World of Computing is a fundamental element of the way distributed information is handled, thus is central to the design of the communicational aspects of the system. A page is a fixed size block of mass storage. Pages can contain indiscriminately database records, executable code, data about internal management, lexical and dictionary items, lists, digitized sound and pictures, strings, etc. — in other words any type of manipulable data. This does not necessarily require that all the data manipulated by the New World of Computing always be on pages. As we have seen, there exists an efficient "grain size" for any media that simplifies description, manipulation and transport of relevant data. The choice of storing raw media data on pages is a response time versus cost trade-off. Some media (e.g., pictures) require a large amount of space; it would be foolish to strain the New World of Computing's paging resources with all the digitized photographs, etc., a single user might address at one point in time. ASCII text, pixel files, pre-existing files such as special word processor files, etc., are stored directly in their original form on disk. Some may be stored on specialized hardware, e.g., videotape, optical disks. Access using specialized hardware is not constrained to storage limitations; some

data's usefulness only stems from the fact that they came from an instrument, e.g., "the gas meter reading from the house at 110 Park Street," or "the voltmeter reading of the control experiment" All of these, however, are packed as bit files on pages when networking requests are made and a subset of these pages may be kept in memory when they are in use.

As we mentioned earlier, the area of mass storage where these pages are kept is called the paging file; the area of main memory where the page images are swapped in and out is called the paging area. To access data or a procedure stored on a given page, a paging utility is called, with the given page's page address, to load an image of the page into the paging area. This utility returns a pointer to the page image, and to the byte offset on the page. When the telephone number in the page address coincides with the computer's own telephone number, and the page image is not already in the paging area, it is fetched from the computer's own paging area. Otherwise, the page must come over the network from another station's addressing space; the "telephone number" of this station — a part of the page address — uniquely identifies any given computer in the world. In an Ethernet configuration this would map to the Internet number, but in the telephone-computer era, it will map to what is nowadays the actual telephone number.

When a station requests a page whose "home" is on a distant telephone-computer, a request is sent to the appropriate network layer along with a header specifying the reason for the request. Any lookup or translation of the network address into the network protocol's standard is done at this time. If the New World of Computing System daemon at the requested address is not already open, the network layer opens up a connection. It forwards the header and the page to the daemon that will then reply with a status header indicating the success of the operation. If the

header indicates success, the requested page will accompany it. This process is the complete communication handshake protocol. The page serves as the atomic packet of communication between two New World of Computing stations.

The communication header as introduced above is a small strip of sideband information that determines the specific handling requested of the accompanying page. It can be shipped along a different channel if desired; this is the case with ISDN where the header is threaded through the D channel, and the pages are exchanged on one of the B channels. The requested handling is specified by a number called a message code. The header can also contain additional information of use to the communication layer, and thus should not be stored on the page itself.

Some message codes are always accompanied by a page, others are not. This latter group is mainly used to return a diagnostic, in general to inform of a problem. Here are the several groups of network System paging message codes:

- Basic paging requests identical to the mass media paging interface (asking for a page, marking it, requesting a new unused page, writing it back).
- Configuration requests like requests for a Sublanguage Control Page (SCP).
- Communication message codes: The protocol for establishing a shared session*: these are mainly sideband protocols and most of them are handled by the network software: putting the call through, checking on the availability of the caller, transferring caller identification information, urgency level, call acceptance, hangups..
- The protocol for maintaining a common interface during a shared session*: exchanging voice packets, maintaining the shared displays, windows and cursors.

---

* Shared sessions are described in detail later in the paragraph entitled PHONING.

- Data integrity and security message codes.
- Requesting system pages relating to user profiles such as mailbox pages and message-box pages, hardware device configuration requests (like the sound digitizer settings).
- Media transfer message codes: transferring large objects like files, pictures, digitized voice...
- Error message codes: illegal page requests, security violations, network errors, corrupt headers. Communication status: busy signal, station not up, no answer, requested party not available. System errors on the called side: nonexistent file requests, protection violations.
- Phrase evaluation — this will be discussed below.

B. Daemons

Every station has a daemon (or non-stop background process) running whose sole task is to serve page requests. In a UNIX environment the daemon is a detached background process, similar to processes like nfsd, inetd or pagedaemon. In the Microsoft Windows environment, the daemon is an independent application that interfaces with the New World of Computing process using the Dynamic Data Exchange (DDE) protocol. In a purely DOS environment, the daemon would be a Terminate-and-Stay-Resident (TSR) program and would thus be essentially interrupt driven. The following descriptions assume a UNIX environment because of its generality. Extensions to other environments are straightforward.

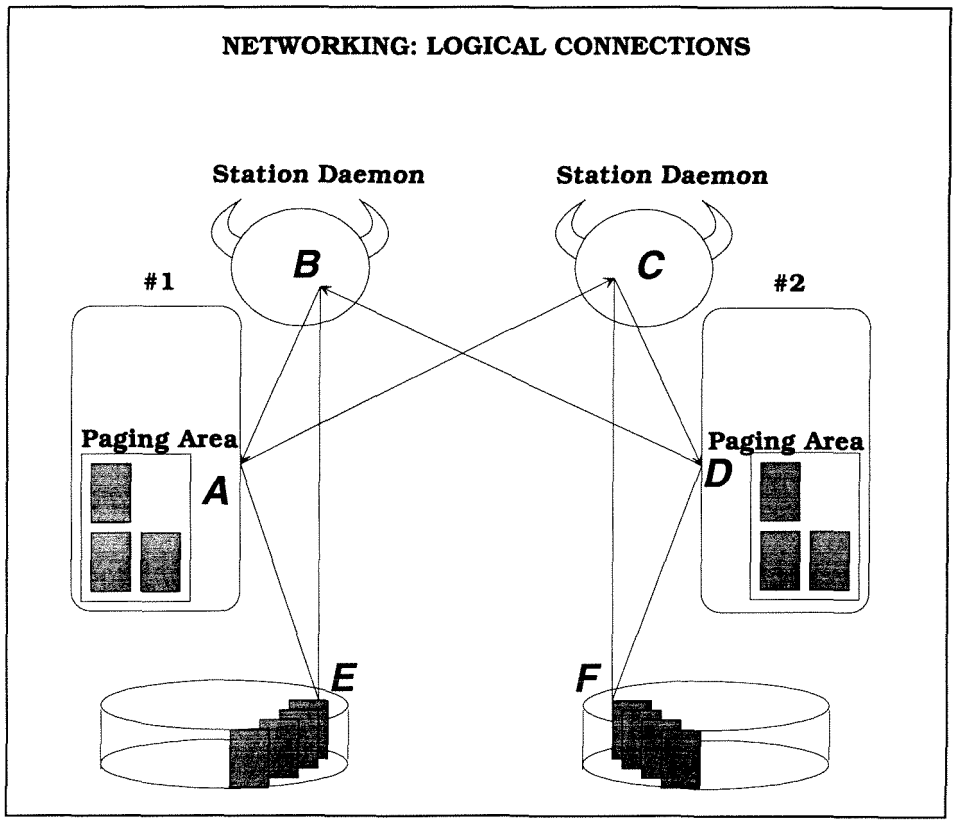

Figure 9

Every time the New World of Computing is called by the underlying operating system, a connection with the daemon is established. If establishing the connection fails, the New World of Computing will assume there is no daemon present, and will spawn a new one. Then the connection is attempted again. The daemon of any particular station is always listening on a well-known port address. Any New World of Computing process that wishes to get in contact with another station initiates a connection with its daemon, at that station's network address, on the well-known port. As soon as the daemon answers, the New World of Computing process hands over its own identification number, and they re-contact each other immediately on a port number that is derived from the identification number. They do so in order to leave the well-known port number available for further calls. This derived connection (labeled **AB** in Figure 9) stays open as long as that New World of Computing application is running. The mechanism for establishing a connection with the daemon is the same for a distant station as it is for the station the daemon is bound to. The only difference is that the daemon does not add this connection to its list of ports to listen to. The New World of Computing regards this connection solely as an input. This is where external page requests come from. The daemon then keeps cycling among its list of active connections, waiting for a header and page. When the daemon receives a page request, it executes the actions specified by the header. These can vary from delivering the requested page, to identification, to establishing a phone conversation. It usually forwards the request to the New World of Computing process if it is up, leaving the control of the paging area to one organism. For all intents and purposes, the paging services could be a totally independent process and both the New World of Computing daemon and the New World of Computing process itself would make requests to it. If the New World of Computing is down, the daemon knows how

to retrieve the pages directly from the paging file (through the connection labeled **BE** in Figure 9). It then sends back along a reverse path (**ABD** in Figure 9) a small header establishing the success of the request along with the desired page. When irrelevant, the accompanying page is omitted to prevent unnecessary network usage. Typical cases are error replies (e.g., the desired page is invalid) and side-band information (e.g., the phone line is busy). The daemon then sends the page back along an identical path along with a header that determines how successful the operation was.

C. Functionalities Implemented by these Methods

In this section, we will discuss how the System uses the basic page/header passing to implement rather low level functionalities. In the following section we summarize the way higher level — application oriented — functionalities are handled.

In order not to leave any mysteries, it will be useful to know about the bridge procedure. Here is a typical example of how it works. The user wishes to create a new class of entities, say the class of males. He types:

```
>class:male

>male: John Jones
```

The corresponding rules of grammar are:

```
RULE
<sentence> => "class:"
SYN new_word_proc


RULE
<sentence> => <noun> ":"
SYN new_word_proc
```

These rules are general rewrite-rule grammar rules where any symbol encased in angle brackets is a part of speech, and literal strings are quoted with double quotes. The constituents on the right of the "=>" symbol are rewritten as the left hand part. The rule is triggered at the time specified by the all uppercase identifier that is at the beginning of every third line (e.g., SYN for syntax time in this particular case).

Following that identifier is the name of the semantic procedure to be invoked when the rule is triggered. In our current case, *new_word_proc* calls the procedure "*bridge*". This is a New World of Computing System procedure that makes use of knowledge of deep internal structures of the language processor. (This is not the place to go into further details.) It goes to the point in the input string that has been parsed by the current rule — in each of the above examples, to the ":" character — and returns the rest of the input string. Its side effect is to fool the parser into thinking that the entire input string, not just the part that fits the rule, has parsed. Thus in the first instance, it returns "male"; in the second instance it returns "John Jones". *New_word_proc* then proceeds to add the appropriate entry to the lexicon and carries out the rest of the implied instructions.

In the great majority of cases when a page is requested from another station, the page address of the desired page is known to the requesting procedure. This is the case, for example, when a database record is being used and a link on that record is being followed; for example, the members of a class are being evaluated and one of the members is from a based upon sublanguage. We will refer to such requests as "direct requests."

1. SUBLANGUAGE AND USER MANAGEMENT

When do page requests that are not "direct requests" arise? A common occurrence of such a request is when the name of a sublanguage or of a user is known as a string but not as a page address. This occurs, for example, when a person first begins a New World of Computing session and responds to the prompt: "Please identify yourself:", or when a person in the COMMAND sublanguage wishes to enter a particular one of their sublanguages. Two utility procedures are available to handle such situations.

The procedure *id_is_ver*(char *subl) checks to see whether "subl" is the name of a sublanguage. If so, it returns the page address of the Sublanguage Control Page (SCP), otherwise NULL. If subl contains a single instance of the character "@" followed by a whole number, e.g., Correspondence @ 3566230, the system interprets it as the name of a sublanguage on a distant station with the number being its telephone number. It requests of this distant station its COMMAND sublanguage's SCP. From this it obtains the page address of that station's Sublanguage Table. This table contains a list of pairs, a string and a page address, where the string is a sublanguage name and the page address is that of this sublanguage's SCP. If there are a great many sublanguages at this station, it may be necessary to request continuation pages of the Sublanguage Table. Other than the first request for the distant station's COMMAND sublanguage SCP, these are all direct requests.

Similarly, the *user_id*(char *auth_user) procedure finds the page address of an authorized user — actually the user's profile page. The profile page address is found on the AuthUser Table on the COMMAND sublanguage's SCP.

It should be obvious how, using bridge, *id_is_ver* and *user_id* procedures, the following functions of the command language involving two telephone-computers are implemented:

- authorizing new users to use a given telephone-computer;

```
>new user: Tom Jackson @ 2025556789
```

- authorize persons at distant stations to enter and/or base on a local sublanguage;

```
>authorize Tom Jackson @ 2025556789 to base on
 Accounting
```

- to be recognized at a station when one is only authorized at a distant station;

```
>Please identify yourself:Tom Jackson @ 2025556789
```

- to enter a local sublanguage or a sublanguage on a distant station, once you have been recognized at the local station;

```
>enter Accounting
>enter Sales Statistics @ 2135556811
```

Both telephone numbers, sublanguage names and authorized user names can be assigned aliases. For example:

```
>definition: Western Region Sales : 2135556811
>definition: Bob : Robert Smith @ Western Region
 Sales
>definition: Western Statistics : Sales Statistics @
 Western Region Sales
     .........
>Please identify yourself: Bob
You are in COMMAND. Proceed
```

```
>enter Western Statistics
You are in Sales Statistics @ Western Region Sales.
Proceed
```

2. BASING

Suppose the user, while in the COMMAND sublanguage, types:

```
>base Accounting on Contracts Accounting
```

Then *id_is_ver*("Accounting") and *id_is_ver*("Contracts Accounting") are called. If the first returns a NULL, then the system procedure: *create_ver* is called. If the second returns a NULL, an appropriate diagnostic message is output to the user. When basing, the "based" sublanguage, e.g., "Accounting" in the example, must be local. Thus as far as station to station communication is concerned, we are interested only in the "based upon" sublanguage. *Id_is_ver* returns the page address of the "based upon" sublanguage's SCP. This sublanguage's Sublanguage Record page address is on this SCP, and from that record can be obtained the list of authorized users (their profile page addresses) who have been authorized to base on this sublanguage. Once authorization has been ascertained, basing can proceed using only direct requests. All of the material about this sublanguage is directly requestable from this SCP — its lexicon, grammar table, part of speech table, etc.

3. E-MAIL, VOICE MESSAGES AND FAX

In the New World of Computing, sending e-mail or a voice message consists of putting the page address of an existing text entity or message entity into the mail box or message box of the recipient. The actual text or message is not transferred at that time. If, upon "receipt" of such a text or message, the recipient wishes to view or hear it, they type:

```
>display my mail
```

or

```
>play my messages
```

or for a more complex example:

```
>display any mail from John
```

At that time, the text or digitized voice is copied onto pages and transmitted from station to station. Thus the only page requests involved with e-mail or messaging that are not direct requests are to obtain the page address of the mail box or message box of the recipient.

First, note that since it is the page address of an already existing entity that is sent, and entities exist only in sublanguages that are based, directly or indirectly on BASE, the sender cannot be in the COMMAND language. If the recipient is at a distant station, then the method of addressing that we have seen above can be used:

```
>send the draft of the budget to John Jones @
 2135556789
```

In this case, the procedure *user_id* is used to obtain the page address of the recipient's mail box or message box.

However, another method is more widely useful. Anyone authorized at some station can be made an individual entity of the current sublanguage. The local user just types:

```
>local name: John: John Jones @ 2135556789
```

This puts "John" in the current sublanguage lexicon, just like any other entity — individuals, text, messages, etc. The only difference is that the database record for "John" has the page address of John Jones' profile page in its header. So now e-mail can be sent to him directly:

```
>send the draft of the budget to John
```

His profile page address will be obtained from his resident record, and the page address of his mail box is on his profile page. A person does not have to be at a distant station to be an individual in the current sublanguage. Thus if Pete Jackson is also authorized on this current station, one can type:

```
>local name: Pete: Pete Jackson
```

then, for example:

```
>class:section manager
>Pete and John are section managers
>send the draft of the budget to section managers
>message:Birthday Greetings
```

. . . . voice . . .

```
>send Birthday Greetings to all individuals whose
 birthday is today
```

(Since we are in a sublanguage, "all" means "all individuals that have been declared as such to this sublanguage.")

One can expect that this along with site-to-site digital telecommunication like ISDN will render facsimile devices obsolete, since the structured data can be transmitted and printed if necessary without having to scan and basically destructuralize the information.

4. GRAPHICS AND LARGE FILE TRANSFER

As we discussed earlier, some data objects are so large that it is not worthwhile to try to store them on pages. For greater efficiency, these cumbersome objects are stored in files, and are referenced by their file name. When a foreign station needs to

access one of these files which it doesn't hold in its local virtual space, it must initiate a network file transfer, sending a header with the appropriate message code to the possessing station. The accompanying page frame holds the requested filename. After checking the validity of the requested file, the station then initiates a handshake with the requesting station. The possessing station breaks the file down into page size chunks; these chunks are sent back in a continuous stream along with the appropriate message code, and the amount of relevant data that is on the page. As far as the system processes that implement this functionality go, the data is simply bits. The last page is sent with an end of file message code signalling the end of transmission. The requesting station reconstitutes the file on its side, either on disk for texts or in video display memory for pixel sets. This protocol is general enough to be used to transfer any type of file.

5. PHONING

> "Computer conferencing, or computer-mediated discourse, as academics like to call it, is only 15 years old. The first was a single conferencing system, used by the Office of Emergency Preparedness during Nixon's price freeze economic program; there are now several such systems in existence." [MEEKS]

The shared session concept provides means for two or more users to look into the same sublanguage without leaving their station. The complete phoning protocol has been implemented as a finite state machine. The protocol is handled by the phone dispatcher (*pd*) function. In person to person mode, there are four possible states: IDLE, RING, WAIT, TALK with the initial state being IDLE. There are ten transitional events. The events are queued in six separate queues: Incoming Calls,

Outgoing Calls, Incoming Responses, Outgoing Responses, Incoming Info and Outgoing Info that *pd* scans periodically. The transition table is displayed in Figure 11 where italics indicate the state the machine transitions to after completing all the actions specified in the list above it.

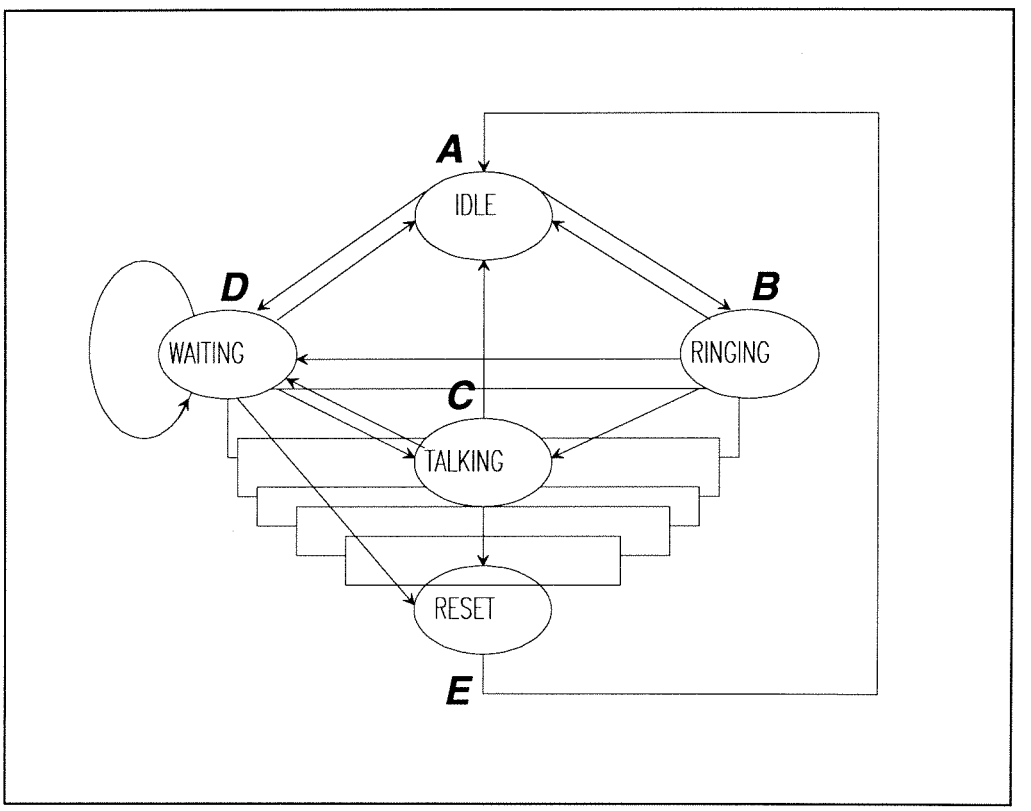

Figure 10

ASK PHONE PROTOCOL CHART

Queues	Events	Phone States			
		IDLE	**RING**	**WAIT**	**TALK**
Incoming Calls	**P_INI**	set party ACK/IGN icon audio bell *RINGING*	none	none	none
	P_FIN	none	close icon *IDLE*	none	close window report time IDLE
Outgoing Responses	**P_IGN**	none	send P_IGN close icon *IDLE*	none	none
Outgoing Responses	**P_ACK**	none	send P_ACK open window init recorder HANGUP icon *TALKING*	none	none
Incoming Responses	**P_IGN**	none	none	close icon *IDLE*	none
	P_ACK	none	none	open window init recorder HANGUP icon start timer *TALKING*	none
Outgoing Calls	**P_INI**	set party send P_INI call HANGUP icon *WAITING*	send P_IGN reset party send P_INI call HANGUP icon *WAITING*	send P_ABR reset party send P_INI call *WAITING*	send P_FIN reset party send P_INI call *WAITING*

	P_FIN	none	send P_ABR kill party close icon *IDLE*	send P_ABR kill party close icon *IDLE*	send P_FIN close window report time *IDLE*
Incoming Info		none	none	none	display line
Outgoing Info		none	none	none	send line

Figure 11

A typical person-to-person session is displayed in Figure 12. It is broken down into the three typical phases of a conversation: setting the call up, exchanging information and hanging up. When Mary, sitting at her station, decides to initiate a shared session with John, she types in: "*call John @ 5551234.*" She could have typed as well "*urgent call to John @ 555-1234*" or "*emergency call to John @ 5551234,*" and she would have upped her priority in doing so. She could also have used a "local name" for "John@5551234", i.e., "*phone John*". All these phrases parse to a syntax time procedure named *phone_pr*, which calls *bridge* to gather the callee's name and number. If she calls John frequently, she could place the phrase "phone John" under an icon so that pressing the icon would initiate the call. All of these result in calling a semantic time procedure named *phone_se*. When this procedure is invoked, and if a number is present, a request is sent to that number to see if the station is active, if John is a legitimate user at that station and to gather information about him that is stored on his profile page (e.g., his priority level). If he is a legitimate user and the call's priority level is higher than John's setting, a U_STAT request is sent that checks if he is currently logged in, if his line is busy, and if he is accepting calls. If any of these checks fail, Mary is offered an electronic mail alternative. Otherwise, the

actual call request is put on the phone dispatcher's outgoing call queue. When *pd* gets around to picking up that event, a network request is sent to John's station through the phone network, a HANGUP icon is displayed on Mary's screen allowing her to cancel the call at any time and the state changes from IDLE to WAITING. At John's station which is also in the IDLE state, the network interrupt handler receives the network request and pushes it on the phone dispatcher's incoming call queue. That event causes *pd* to signal John in a user customizable way (e.g., bell, blinking phone icon with caller name) that a call is pending. The state goes from IDLE to RINGING. To get out of that state requires active intervention from John. He can either indicate he wishes to ignore the call altogether or answer it either from the prompt or by clicking the phone icon. The former case puts a "busy" event in the dispatcher's queue, erases the phone icon and goes back to IDLE state. The latter puts a "phone" event in the queue, sets the station up for a shared session (displays the shared session window, synchronizes parameters of the two stations' voice digitizers, displays the HANGUP icon) and goes to TALKING state. The "busy" or "phone" event is sent by John's phone dispatcher through the network to the incoming response queue of Mary's station. Mary's phone dispatcher picks it up and either erases the HANGUP icon and goes to IDLE mode in the "busy" case or sets her station up for the shared session, starts a timer that records the length of the conversation and goes to the TALKING state. At this point the two stations are set up for a shared session.

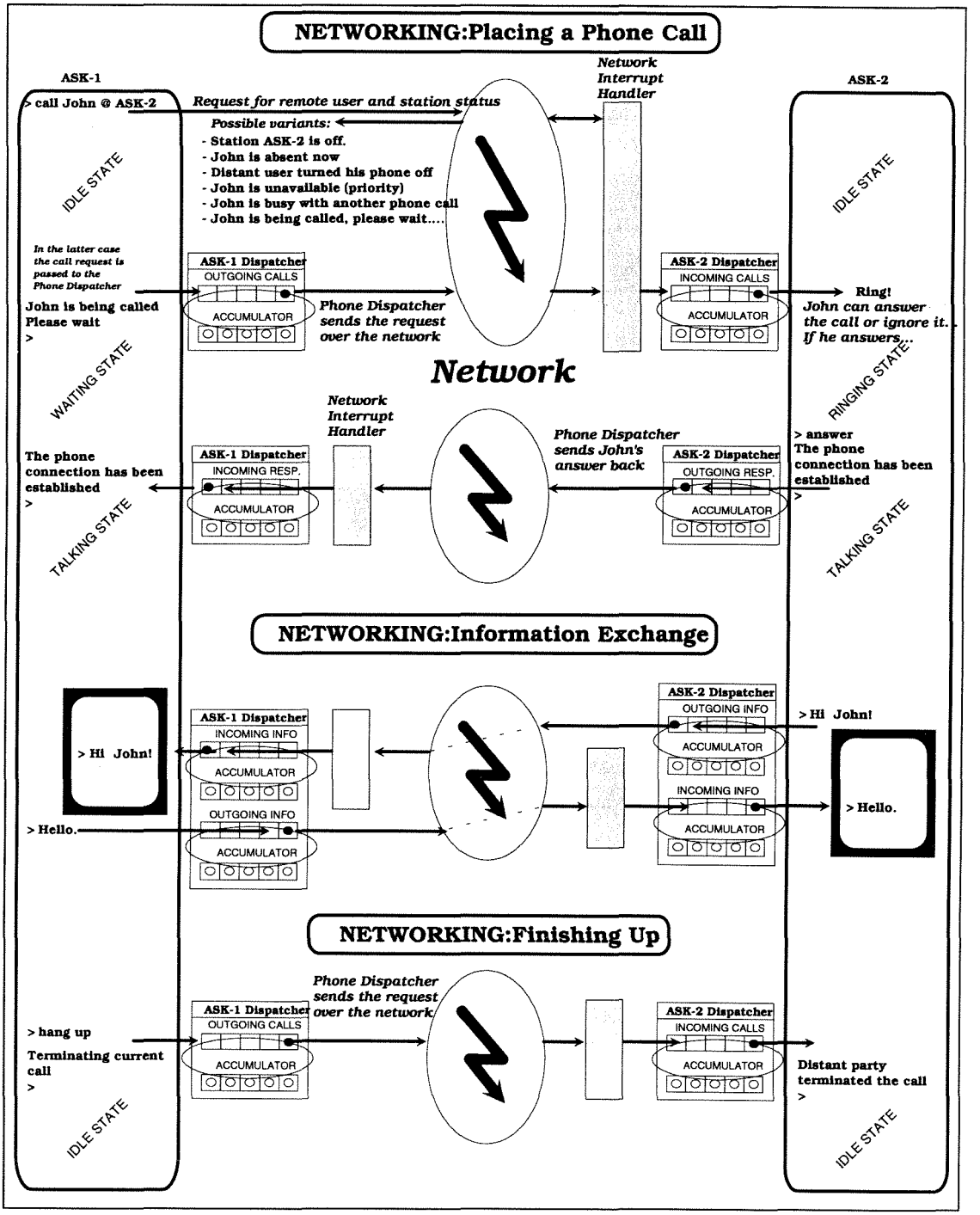

Figure 12

This is the information exchange phase. Voice is flowing back and forth and all input and output events are echoed on either station. As they occur, these events are

duplicated and one instantiation is put in the Outgoing Info queue. The dispatcher in the usual way sends them across the network and they land in the other station's Incoming Info queue. There the other dispatcher displays them appropriately. Because of its high bandwidth requirement, and the fact that transmission reliability is not a major issue, voice is handled in a different way. A timer driven interrupt routine continuously sends digitized speech pages over to the other station while stations are in the TALKING state. In any other state the routine acts as a no-op. This decreases the overhead of sending these pages as events. When the conversation has been carried to its end, anyone can end it by clicking the HANGUP icon or inputting "hang up". This generates a "hangup" event that is put in the outgoing call queue, the shared session window is closed and the length of the conversation is displayed. The state goes back to IDLE. When the other dispatcher receives that event it behaves in an identical manner.

What happens when several people want to establish a conference session? The intrinsic situation is identical except that a call can come in at any time from another party. This introduces four new possible states (see Figures 13 & 14):

1. TALK-RING when a request comes in while a conference is already occurring;
2. TALK-WAIT when a request for another party has been sent in the midst of an occurring conference;
3. TALK-RING-WAIT when both of the above events occur;
4. RING-WAIT when somebody calls while a request is being made for another person.

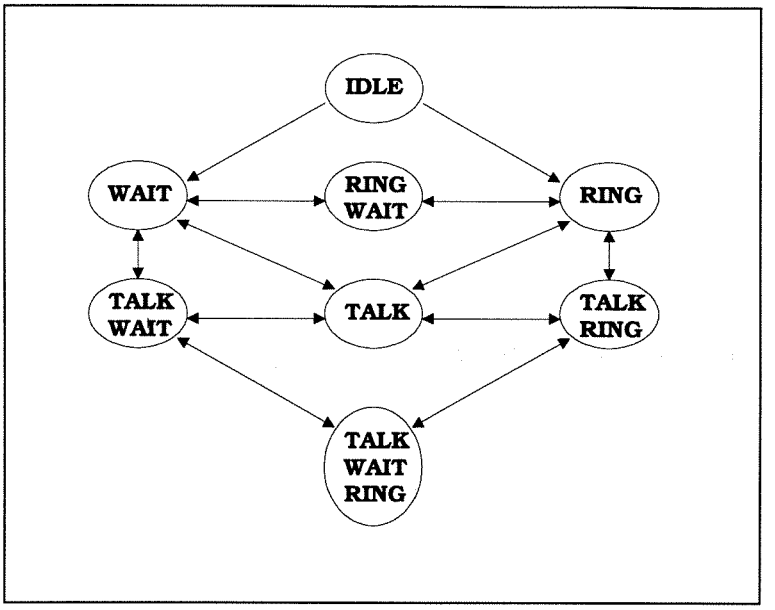

Figure 13

ASK PHONE PROTOCOL CHART

(conferences)

Queues	Events	Phone States							
		IDLE	**RING**	**WAIT**	**TALK**	**TALK-RING**	**TALK-WAIT**	**TALK-RING-WAIT**	**RING-WAIT**
Incoming Calls	**P_INI**	set ring party RING	set ring party	set ring party RING-WAIT	set ring party TALK-RING	set ring party	set ring party TALK-RING-WAIT	set ring party	set ring party
	P_FIN	none	"ignored by dist" kill ring party IDLE*	none	"bye from dist" kill party IDLE***	"bye from dist" kill party RING*** ——or—— "ignored by dist" kill ring party TALK*	"bye from dist" kill party WAIT***	"bye from dist" kill party RING_WAIT*** ——or—— "ignored by dist" kill ring party TALK-WAIT*	"ignored by dist" kill ring party WAIT*
Outgoing Responses	**P_IGN**	none	send P_IGN kill ring party "ok, ignore" IDLE*	none	none	send P_IGN kill ring party "ignore" TALK*	none	send P_IGN kill ring party "ignore" TALK-WAIT*	send P_IGN kill ring party "ignore" WAIT*
	P_ACK	none	activate ring party TALK* (TALK-RING)	none	none	activate ring party TALK*	none	activate ring party TALK-WAIT*	activate ring party TALK-WAIT* (TALK-RING-WAIT)
Incoming Responses	**P_IGN**	none	none	kill wait party "ignored by dist" IDLE**	none	none	kill wait party "ignored by dist" TALK**	kill wait party "ignored by dist" TALK-RING**	kill wait party "ignored by dist" RING**
	P_ACK	none	none	activate wait party TALK** (TALK-WAIT)	none	none	activate wait party TALK**	activate wait party TALK-RING**	activate wait party TALK-RING** (TALK-RING-WAIT)

Outgoing Calls	**P_INI**	set wait party	set wait party WAIT	set wait party	set wait party RING-WAIT	set wait party TALK-WAIT	set wait party TALK-RING-WAIT	set wait party	set wait party
	P_FIN	none	none	send P_FIN "ignore" kill wait party IDLE**	send P_FIN "bye to dist" kill party IDLE***	send P_FIN "bye to dist" kill party RING***	send P_FIN "bye to dist" kill party WAIT*** ——or—— send P_FIN kill waiting party "ignore" TALK**	send P_FIN "bye to dist" kill party RING-WAIT*** ——or—— send P_FIN kill waiting party "ignore" TALK-RING**	send P_FIN "ignore" kill wait party RING**
Incoming Info		none	none	none	display	display	display	display	none
Outgoing Info		none	none	none	send	send	send	send	none

* - transition only if there is no more RINGing virtual party

** - transition only if there is no more WAITing virtual party

*** - transition only if there is no more TALKing party

Figure 14

Each session is represented by a window that symbolizes the opening to the current conversation's sublanguage. Iconizing that window is the equivalent of putting the session on hold. Other participants can keep on sharing information, but the user will not see it and they will not see his current activity. This allows several sessions to be opened at a time, but only one can be active. Opening up a session window, or answering a new call automatically closes other session windows. As new

call requests arise, the user has at his disposition a varied set of possibilities that allow him to deal with them. Each call request is symbolized as an icon with the caller's name. The icon can be a digitized picture of the caller or of his company's logo. The user can grab that icon and put it in his window thus adding the caller to his current session. He can also put it in his exclusion list. This will cause any subsequent calls to be ignored. There also exists an inclusion list of people whose calls are accepted even if their priority level is too low. This list is used when all calls are ignored, thus enabling a selective filter. These are all, of course, graphic representations of activities that can be requested as a normal sentence input. (e.g., "Ignore all calls except those from my broker and my family.")

6. PHRASE EVALUATION AND DISTRIBUTED PROCESSING

In Chapter II, Section B. The New World of Computing System, THE LANGUAGE PROCESSOR, the notion of a New World of Computing "phrase" was introduced. Recall that a "phrase" is a data structure corresponding to some meaningful substring of the input sentence. Here, we go a little more into detail as to what a phrase looks like. A phrase has two structural forms: the "before evaluation" form and the "after evaluation" form. These can be most easily understood by an example. We will consider the example:

```
>Show the three month tabular display of each of
 John Jones' high tech stocks.
```

We will primarily be interested in the phrase associated with the string:

"three month tabular display of each of John Jones' high tech stocks"

However first consider the subphrase:

"each of John Jones' high tech stocks"

Assume that this has already been evaluated, and that the list of these stocks is now on the scratch page* "[STOCKS]". Then the "after evaluation" structure of this phrase is:

PHRASE-1: part of speech: <noun_phrase>
string: "each of John Jones' high tech stocks"
data: [STOCK]

Assume that the subphrase:

"three months"

has the "after evaluation" form:

PHRASE-2: part of speech: <time>
string: "three months"
data: three months (in terms of the internal unit of time)

The "before evaluation" form of the phrase corresponding to the substring:

"three month tabular display of each of John Jones' high tech stocks"

is:

PHRASE-0: part of speech: <noun_phrase>
string: "three month tabular display of each of John Jones' high tech stocks"
procedure: *tab_disp_proc*
constituents:
PHRASE-1
PHRASE-2

---

* A scratch page is a page whose lifetime is that of the sentence being processed.

At this point in sentence processing, the "semantic evaluation" is called to evaluate PHRASE-0. It calls *tab_disp_proc* with the two constituents as arguments. *Tab_disp_proc* has been supplied to this user by a software house that specializes in sublanguage packages for financial institutions, including access to their database of financial statistics. That is to say, the current sublanguage being used was based upon the sublanguage supplied by this software house. Thus the page address of *tab_disp_proc* was put into the grammar table of the current sublanguage at the time of basing, therefore is directly requested from the software house page files at this moment. We consider two cases:

1. *tab_disp_proc* does all of the calculations necessary to produce the desired graphic at the user's station; in doing so it will most likely use direct requests to the software house's station to get individual items of data from the database.
2. the rôle of *tab_disp_proc* is to send the inputs to the calculation back to the software house where the calculations will actually be made, returning to the local station the desired graphics. In this case, *tab_disp_proc* asks whether it is at its home station — the software house. If not, it puts the above PHRASE-0 on pages (there is a utility for this) and sends this page back to the software house with the message code: phrase evaluation. Upon receipt of this page and header, the software house evaluates the phrase, again calling *tab_disp_proc*. This time it is at its home station, so it completes the calculations, creating a graphic file for the graphic, say [PS FILE], and returns a page containing the "after evaluation" phrase:

   PHRASE: part of speech: <graphic>

   string: "three month tabular display of each of John Jones' high tech stocks"

   data: a page containing "[PS FILE]", i.e., the name of the file

This phrase will then be further processed by the semantic procedure of the RULE:

```
<sentence> => "show" <graphic>
```

which initiates a file transfer, as discussed above, and displays the graphic.

The second case is an example of the use of the "phrase evaluation" message code. This message code provides a powerful mechanism for handling all manners of "distributed computation."

D. Implementation Solutions to High Level Applications Functionalities

1. "INTELLIGENT" COMPUTER RESPONSE

In a networked set of sublanguages, each one can be seen as containing links (page references) to the others. These links are implicitly created either at the time the sublanguage is based on another one, or when the based upon sublanguage is modified. As we have seen, these page references can select any byte from the world's virtual pool of data. Because of the different topological configurations of these based sublanguages, various link patterns can be observed.

a) Intelligent computer intervention:

In a manufacturing environment, one can imagine a task database sublanguage containing all the information pertaining to the many manufacturing tasks of a large project. In this "task" sublanguage, there would be an entry for the manager in charge of each task. All task managers would have their own sublanguages based upon the task sublanguage. Let us suppose a supply of raw material is delayed. The notification from the supplier will cause a modification of the expected arrival date of that shipment in the database sublanguage. Now it is the case that any data record contains a special field of its header for the address of an action to be taken whenever that

record is modified. (An "action" can be thought of as any phrase or linked sequence of phrases.) In this example, a typical action would be to re-PERT the chart of the associated project. This will in turn trigger a chain reaction of data updates. Any one of these can be set up to send a relevant notification to the manager in charge of the associated task and to the project manager, at their own stations. So here we have an "intelligent" sublanguage that "knows" how to react to specific situations and help the users in their everyday tasks. Moreover, the paging and networking mechanisms have given this sublanguage the ability to conglomerate and reach its various end users without any evident effort on their part.

b) Marketing of the access to data and procedures:

What are the commercial implications of such paging and networking mechanisms? The least that can be said is that after revolutionizing the software development environment, they will bring along radical changes in the software distribution market. Since the New World of Computing networking can provide access to any page containing a procedure, no matter how distant, storing, upgrading, updating or obtaining software now become simple operations executed from the keyboard. "Extending" a sublanguage with new programmatic features, or with an appropriate database, is just a matter of providing the phone-network number of the provider. All the financial aspects can be treated right there and then, by choosing among various possibilities:

- single payment either to the software house itself or to a distribution outlet for a disk from which "extend" will add a new sublanguage;
- direct electronic payment from the buyer's bank account in a lump payment for the right to base on a specific sublanguage;

- charging on a per-page access, the billing being done by the telephone company; (Recall that if sublanguage AA is based on BB, and BB on CC, then any links in AA to CC pages is direct, not through intervening BB pages; therefore each software supplier, throughout the basing hierarchy under a sublanguage, is reimbursed for the use of its paged material on a per-page basis.)
- requiring a monthly payment for the right to be based on a specific sublanguage.
- The various modes of payment would be attuned to the different marketing aspects of the type of information being sold. If a sublanguage needed to access Dow Jones data on a regular basis, a per-page charge would probably be the most appropriate. On the other hand, if it wanted a financial analysis of a certain portfolio of stocks to be performed by a reputable financial company that did not want its analytical techniques to be widely distributed, even in object code format, a per-process payment would be the way to go. Regular information providers like trade journals or magazines could charge a periodical fee as is the practice today (though competition may drive them into a per-page tarification).

c) Offloading computational loads:

In view of the large variety of tasks that can be required of a computer, some of them are bound to require sophisticated computing platforms to achieve a reasonable throughput. The commercial aspect of this premise is that most of these tasks do not require the whole attention of the CPU for any extended period of time. Exceptions to this are applications that require lengthy periods of super-computer time, like chemical reaction modeling or weather simulation. Networked paging will allow people to buy a computer designed for their average use, instead of being forced to buy a machine that will be able to handle the upper bound of their computational needs when exceptional needs occur only occasionally. The way this concept

functions is by offering the opportunity to unload occasional arduous tasks to higher end machines or specialized hardware-like transaction processors by means of a "phrase evaluation" event described above. In fact, a single application could very well send "phrase evaluation" requests to a range of different computers, either to parallelize the process or just because the task is diversified. A car loan application could very well be required to access the customer's credit history record, and query the "blue book" database to retrieve an estimate of the car's worth. Note that the "blue book" database need not be an integral part of the bank's software environment. One could very well imagine its maintenance by an external company like Kelly's or J. D. Powers & Associates, who would then charge a small standard fee per query.

2. METHODS FOR DISTRIBUTED PROCESSING

What is the current state of the art in distributed processing? Since most processes are self-contained, the only interface available is done by file exchange. This means that an application that wishes to have some data processed by another application needs to save the data into a file. Then it needs to invoke the second application with the file name as a parameter along with a list of commands specifying the action to be taken (including the command to save the data into a file again). With the advent of applications that adhere to the Graphic User Interface (GUI) as opposed to the Command Line Interface (CLI) type that is more predisposed to batch style action, came a mechanism to transfer data easily from one application to another, namely cutting and pasting to a clipboard area. This has simplified the user's work, and generalized data transfer. Singularly, it has also made automation much more difficult. While the commands used to consist of a stream of characters that could be rerouted at will, in a GUI environment they consist mainly of mouse events

(moves, clicks and drags). A good design rule is always to have a keyboard equivalent for any such event, but that rule has been violated by almost every application on the market. Another attempt to avoid this hindrance was made by providing the user with an event recording mechanism. All events are stored in a file in their order of occurrence. They then can be played back at will. This initially created very brittle scripts, since all events were replayed no matter what the state of the desktop was. If icons had been moved, if windows did not appear in the same position, the script would wander off with random consequences.

A more recent approach has been to reference the target of the events by name in an object-oriented manner rather than by absolute screen position. While being a significant improvement, this technique is still inadequate in several aspects. If applications do not buy into the object-oriented paradigm by not publishing their object names, a little brittleness can still be expected, since event positions then become window relative. If the application has a "creative" interface, or has various options that it could disable, it will disrupt the script. Regardless of the brittleness issue, event recording lacks two fundamental aspects that would give it the flexibility and the generality of a programming language: state modification and test capability. Since there are no variables, there is no method for creating, for example, a different temporary file name every time an application is invoked. Since there is no branching technique, there is no method for taking special action if a file already exists. This makes recovery from simple errors impossible. The standard way of signalling errors in a GUI is popping an alert window which in general requires some sort of acknowledgment from the user. However a script cannot discern such a window, and even if it achieved that, it would not be able to take any conditional action. A common need that no event recorder has been able to achieve, is a terminate macro

that would correctly close all the current applications or windows that are lying on the desktop, save the open files if necessary, and then shut the system down. To palliate these shortcomings, very often an "authoring" language has been provided. These languages, though, are cryptic and cumbersome, hence discouraging people away from the simplicity and straightforwardness that attracted them to a GUI in the first place.

In the New World of Computing System, the "phrase evaluation" apparatus, introduced in paragraph IV.C.6. above, provides distributed computing capabilities that overcome the disadvantages of current methods and are straightforward to use. Notice how this concept radically differs from the current monolithic applications. The "phrase evaluation" apparatus allows another computer to be given temporary control of the semantic engine. It does so by sending a page of events that are fed to the various input queues of the system. There are two main "phrase evaluation" activities. The first one is used when some type of action needs to take place on a distant station. Typical instances are the update of cursor movements and the creation of new windows in a shared session. The transmitted page contains a list of cursor moves and clicks that are shadowed by the mimicking display. Outside of the success of the transmission, no acknowledgment is required. The use of this type of action is restricted to a "fire-and-forget" variety of activity. The second is more of an interrogation or evaluation that needs to be accomplished on a station that contains information. The information

- is too restricted or too specific to justify basing a whole context on it;
- can be retrieved by special hardware that exists only at that particular location;
- requires real time input from a source accessible only to another computer;
- requires cooperative computing involving interactive inputs from multiple users.

The query is copied onto a page and forwarded to the appropriate computer, which extracts it and puts it in the linguistic engine's input queue. It is then processed as any normal query; the result is put back on a page and shipped back to the requester. An example would be the case of the gas company reading meters at all their subscribers' homes from their central accounting computer.

A multi-user, interacting computer application can be implemented using this technique. An example is a multi-player game such as the game of bridge. One program would be written that recognizes itself as either "North", "South", "East", or "West", that knows how to handle the display of the playing surface, and that queries its player to ascertain his/her bid or the card to be played. A start-up program would initiate this program on each of the four participants' computers; then the common programs would use message passing to carry out the game. Many other forms of cooperative work could be implemented.

An important characteristic of the "phrase evaluation" process is that it does not require any reprogramming on the distant station. Any input activity that a user could attempt on that station can be reproduced programmatically from the originating station without even having to access the distant station beforehand to settle on a communication or handshake protocol, or having to write a special driver for the new input. "Phrase evaluation" is thus an ideal Application Programming Interface (API) for the New World of Computing, which allows any outside application that is connected to the network or knows how to dial a phone number, to access, modify or query the New World of Computing or any application managed by the New World of Computing, in a very exhaustive manner. This application could have control of every aspect of the New World of Computing interface including the capability of clicking

icons, selecting menu entries or opening windows. It can send digitized voice to a voice recognition based interface, or scan images into a pattern recognition interface.

3. HIGH LEVEL PHONING FUNCTIONALITIES

A shared session, whose implementation was discussed above, goes far beyond both "televised face" conferencing and file sharing. The number of participants in such an exchange is not limited to two. Teleconferencing a large number of people is possible by making available a sublanguage where the speakers can share their views. Their common sublanguage contains all the elements that are relevant to the topic of the conversation, and these are automatically displayed to each participant using the associated media. All input events are distributed, so that any participant can influence the shared display. When invoking a shared session, a new window is created for each participant. Any input — displaying of pictures, invoking an application package — is echoed on all the other participants' screens. Pointing is also shared, with each mouse cursor being echoed with an identifying feature. Participants can bring up a drawing of a project, point at various items, ask questions and all along their conversation is being digitized, packed on pages and sent across to the other participants as in a real life phone conversation.

Consider, for example, our maintenance professional working on the nose cone radar of a Boeing 747 aircraft, his computer by his side. What are the networking aspects of this maintenance situation? The computer is networked with the Boeing maintenance shops in Seattle, Washington. That is, a sublanguage in this computer is based on the Boeing Maintenance Sublanguage in the Seattle shops. None of the maintenance material is in the computer being used by the maintenance person. First, the identification of the aircraft being serviced is established. In response to a call for

a full color annotated image of the radar nose cone area, the pixel data sets come, via ISDN, from a single source — the multimedia server in Seattle. Although some processing is being done locally, all maintenance data and diagnostic analysis is being done in Seattle. The bane of having out-of-date maintenance manuals will be a thing of the past. If the maintenance professional is still puzzled, clicking the mouse on a special icon will establish an immediate conversation with a maintenance specialist at Boeing, Seattle. Both monitors will display the same material, both people can use their mouse to point, and both can have a voice discussion of the problem at hand. Moreover if a third party needs to be brought in, no special preparation or interruption is required.

E. A Detailed In-Depth Look at an Example

The purpose of this chapter is to give a detailed run-through of a simple but complete session with the New World of Computing that exercises the communicative aspects of the tool. The setup is as follows. There are two computers. The first one models a small library's database (so small it contains only one reference, the Ph.D. thesis of Kwang-I Yu, Communicative Databases). The second computer contains my personal sublanguage dealing with reference work. I have based it on the Library sublanguage to inherit its contents, and then added this thesis, Computer Mediated Communication, as a personal reference, thus creating an environment customized to my needs. For each computer station in turn, I will describe the preamble statements that tailor the sublanguage to its specific use. I will then submit an identical query to both computers, "*Who is the author of each thesis?*," and describe the whole process, illustrating the differences between the first case where the sublanguage is self-contained, and the second one where it spans the data

space of both stations. However, before doing that, I will need to go into the details of the various data structures that comprise a sublanguage.

1. THE LOCAL CASE

a) Library station: setting up the data

In the following transcripts, the user's input is in italics. Preceding this input is the prompt consisting of the name of the sublanguage followed by an angle bracket. The rest is output from the computer.

The initial activity is to request a directory from the COMMAND, or administrative, sublanguage to ensure that the system is virgin, and we see that it is, since the only sublanguage available is the BASE, or initial, sublanguage which we cannot enter.

```
COMMAND> directory
Context Creator enter base
_______ _______ _____ ____
BASE    MASTER   no    yes
```

We create our Library sublanguage by basing it on BASE.

```
COMMAND> base Library on BASE
     transfering dictionary for Library
     transfering lexicon entries for Library
     transfering dictionary for Meta-Library
     transfering lexicon entries for Meta-Library
 Context Library has been created based on BASE.
```

If we do a directory now, we see that the Library sublanguage exists and that we can enter it.

```
COMMAND> directory
Context Creator enter base
_______ _______ _____ ____
Library John     yes   yes
BASE    MASTER   no    yes
```

We must provide basing privileges to the person on the other computer if we want him to be able to use this sublanguage. Note how the other machine is uniquely identified by its phone number.

```
COMMAND> authorize Remy@(818)395-6232 to base on Library
 Remy has been authorized to base on Library.
```

We are still the only person enabled to modify this sublanguage. We will do so by accessing it.

```
COMMAND> enter Library
 You are in Library.  Proceed.
```

We can now start populating the COAR semantic net database with the elements necessary for our application.

```
Library> individual/individual attribute:author
 The new attribute author has been added.
Library> class:thesis
 The new class thesis has been added.
Library> individuals:Kwang-I Yu, Communicative Databases
The following new individuals have been added:
    Communicative Databases
    Kwang-I Yu
```

We now create the links in the semantic net database with the following assertions.

```
Library> Communicative Databases is a thesis.
The individual entity Communicative Databases has been added
to the class thesis.
Library> Author of Communicative Databases is Kwang-I Yu.
 Kwang-I Yu was added as author of Communicative Databases.
```

We are now ready to ask our question.

```
Library> Who is the author of each thesis?
thesis                    author
Communicative Databases   Kwang-I Yu
```

We are done.

```
Library> exit
 You have returned to COMMAND.
```

b) Examining the internal data structures

Let's take a probing look into the internals of what we have created.

```
COMMAND> enter Library
 You are in Library.  Proceed.
Library> metalanguage
 You are now in META-Library.  Proceed.


META-Library> dump record for Kwang-I Yu

Page number:     <1,4406,0,(818)395-6231>
   Parent page : <1,4406,0,(818)395-6231>
   Record type :      1
   Header size :      9
   Fields/entry:      1
   Num entries :     11
Not Timed Record
Not Indexed
Permanent
Individuals only
Header_bool : [ 0 5 ]
Data_page: <0,0,0,0>
Name in English : "Kwang-I Yu"
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:20101 <1,4402,0,(818)395-6231>
 Record for Kwang-I Yu dumped.
```

Let's explain briefly what all these different fields mean.

- The **page number** is our universal pointer that uniquely identifies this record throughout our world address space. This particular one shows us that the record starts at the top of the page (offset is 1), the page is the 4406th one in the paging file number 0 on the station identified by the (818)395-6231 phone number.
- The **parent page** would be relevant only if we had enough data for this record to be indexed. This clearly is not the case.
- The record type is that of an individual object.

- The **header size** is a bit of a misnomer since it indicates the offset in fields where the actual data starts.
- **Fields per entry** tells us that each entry contains one page pointer of data. Each entry is in fact two fields long, the first field being used as a tag.
- **Number of entries** is also a misnomer since it holds the offset to the first available data field where additional information can be added.
- **Not Timed Record** indicates that there is no date information associated with the data in this record. For example, if we had specified when Kwang-I wrote his thesis, then that entry would hold an additional field containing that date, and the whole record would be flagged as timed.
- **Not Indexed** tells us what we already knew from the fact that we have so minimal an amount of information.
- **Permanent** tells us that this record is persistent, even if we turn off the computer.
- **Individuals only** is another flag indicating that all the data fields contained in this record are simple objects. This tells us that there are no classes that need to be expanded which is a useful shortcut as compared to checking all the data fields, especially if the record is indexed.
- The **header bool** is a collection of sixty-four flags, the preceding four cases being four examples. A large set of these are accessible to the application programmer.
- The **data page** field is a free-for-all that has a wide variety of uses depending on the record type. In the case of an individual record, it can hold the page pointer of an action page or a paged procedure that will be invoked

whenever the record is accessed or modified. It would be a convenient way of gathering database usage statistics as a typical case.

- The three **name fields** contain a permanent string pointer that points to the name for this record in the three languages that can be used with this sublanguage. The fact that the two other names are blank indicates that they have not been translated yet.

Finally we get to the record list where the actual data begins. In this case we have only one entry. It is located at offset 9 on this page. The flag describes what this entry is. It is in centennial base. The initial two specifies that this record (Kwang-I Yu) is a value for the attribute pointed to by <1,4402,0,(818)395-6231> which is the record for author. The two following ones describe this attribute as an individual/individual attribute.

```
META-Library> dump record for Communicative Databases

Page number:     <1,4405,0,(818)395-6231>
   Parent page : <1,4405,0,(818)395-6231>
   Record type :      1
   Header size :      9
   Fields/entry:      1
   Num entries :     13
Not Timed Record
Not Indexed
Permanent
Individuals only
Header_bool : [ 0 5 ]
Data_page: <0,0,0,0>
Name in English : "Communicative Databases"
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:   20 <1,4403,0,(818)395-6231>
   Offset: 11 flag:  101 <1,4402,0,(818)395-6231>
 Record for Communicative Databases dumped.
```

This record is very similar to the preceding one since it is an individual object record also. We have an additional entry with a flag of twenty indicating that Communicative Databases is a member of the class thesis pointed to by <1,4403,0,(818)395-6231>. The zero-one-one flag indicates that this record is the argument for the same attribute as above.

```
META-Library> dump record for thesis

Page number:     <1,4403,0,(818)395-6231>
   Parent page : <1,4403,0,(818)395-6231>
   Record type :    21
   Header size :     9
   Fields/entry:     1
   Num entries :    11
Not Timed Record
Not Indexed
Permanent
Individuals only
Header_bool : [ 0 5 ]
Act_page: <0,0,0,0>
Name in English : "thesis"
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:    1 <1,4405,0,(818)395-6231>
 Record for thesis dumped.
```

The record for thesis has a type of "class of individuals", and contains one entry. Its flag of one characterizes whatever is pointed to by <1,4405,0,(818)395-6231> (Communicative Databases) as a member of this class.

```
META-Library> dump record for author

Page number:     <1,4402,0,(818)395-6231>
   Parent page : <1,4402,0,(818)395-6231>
   Record type :   101
   Header size :     9
   Fields/entry:     2
   Num entries :    12
Not Timed Record
Not Indexed
Permanent
Individuals only
Header_bool : [ 0 5 ]
Act_page: <0,0,0,0>
Name in English : "author"
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:  100 <1,4405,0,(818)395-6231>
<1,4406,0,(818)395-6231>
 Record for author dumped.
```

Finally, we get to the record for the individual/individual attribute author as indicated by its type. Attribute and relations have two data fields per entry, the argument and the value. Think of these as the links in the semantic net. The flag in this case indicates that the argument is an object as opposed to a class or a sub-attribute.

c) Processing the query

Now that this picture is in place, let's look at what happens when we input our query. The string "*Who is the author of each thesis?*" is first handed over to the preprocessor who will among many other things build the skeleton of the parsing chart and break the string up into identifiers, looking up each one in the lexicon using the ultra-fast double hashing table algorithm. Here is an excerpt of the relevant items contained in the lexicon:

```
Lexicon entries for Context Library
     lex_master_page: <1,4076,0,(818)395-6231>

Bucket #10 page: <2,4258,0,(818)395-6231>

thesis
    pos: noun  type: standard  feat: [ 7 26 ]
    id for psem: 1025  flag: 0
    payload: <1,4403,0,(818)395-6231>
    amb link: <0,0,0,0>  def msg: thesis

     Bucket #12 page: <2,4260,0,(818)395-6231>

(818)395-6232
    pos: station_name  type: initial  feat: [ ]
    id for psem: 1025  flag: 2  payload: <0,0,0,0>
    amb link: <0,0,0,0>  no def msg

     Bucket #15 page: <2,4263,0,(818)395-6231>

the
    pos: definite_article  type: initial  feat: [ ]
    id for psem: 1025  flag: 0  payload: <0,0,0,0>
    amb link: <0,0,0,0>  def msg: RULE "the"

     Bucket #24 page: <2,4272,0,(818)395-6231>

author
    pos: noun  type: standard  feat: [ 7 28 ]
    id for psem: 1025  flag: 0
    payload: <1,4402,0,(818)395-6231>
    amb link: <0,0,0,0>  def msg: author

     Bucket #25 page: <2,4273,0,(818)395-6231>

who
    pos: relative_pronoun  type: initial  feat: [ 3 ]
    id for psem: 1025  flag: 0  payload: <0,0,0,0>
    amb link: <0,0,0,0>  def msg: RULE the relative pronoun
"who"

     Bucket #28 page: <2,4276,0,(818)395-6231>

Communicative Databases
    pos: noun  type: standard  feat: [ 7 25 ]
    id for psem: 1025  flag: 0
    payload: <1,4405,0,(818)395-6231>
    amb link: <0,0,0,0>  def msg: Communicative Databases
```

```
     Bucket #29 page: <2,4277,0,(818)395-6231>

of
    pos: preposition  type: initial  feat: [ 0 ]
    id for psem: 1025  flag: 1  payload: <0,0,0,0>
    amb link: <0,0,0,0>  def msg: RULE "of"

     Bucket #35 page: <2,4283,0,(818)395-6231>

(818)395-6231
    pos: station_name  type: initial  feat: [ ]
    id for psem: 1025  flag: 1  payload: <0,0,0,0>
    amb link: <0,0,0,0>  no def msg

     Bucket #45 page: <2,4293,0,(818)395-6231>

Kwang-I Yu
    pos: noun  type: standard  feat: [ 7 25 ]
    id for psem: 1025  flag: 0
    payload: <1,4406,0,(818)395-6231>
    amb link: <0,0,0,0>  def msg: Kwang-I Yu

     Bucket #50 page: <2,4298,0,(818)395-6231>

each
    pos: quantifier  type: initial  feat: [ 1 2 5 ]
    id for psem: 1025  flag: 31  payload: <0,0,0,0>
    amb link: <0,0,0,0>  def msg: RULE the quantifier "each"

     Bucket #56 page: <2,4304,0,(818)395-6231>

is
    pos: copula  type: initial  feat: [ 0 1 2 ]
    id for psem: 1025  flag: 1  payload: <0,0,0,0>
    amb link: <0,0,0,0>  def msg: RULE "is"
```

Each lexicon entry contains:

- A part of speech (pos) which is what that identifier will parse to.
- A type which can be:
  - ♦ initial: It is part of the bootstrap language.
  - ♦ standard: It has been added by the user, like our Kwang-I Yu case.
  - ♦ referent: An example of which we will encounter in the second case.
  - ♦ agent: It serves as an access point to a based-on lexical entry.

- The features: a set of flags that provide considerable flexibility to the parsing process but are not of much relevance to this topic.
- The psem id, flag and payload which constitute the semantic part of the item being built (more on this later, but notice how for the standard entries, the payload actually points to the record for that item).
- An ambiguous link which points to another lexical entry in the case where two items with the same name but different semantic meanings exist. In our case, they are all null pointers, but we will see an additional usage for this later.
- A definition message which is used as disambiguating feedback to the user if an ambiguity does exist.

Once the preprocessor is done, the skeleton chart is passed to the parser who attempts to apply the grammar rules contained in this sublanguage's dictionary. By doing so, it populates the parse chart until it finds an arc with part of speech sentence which covers the whole input. In our case, the input being a well-formed sentence, it succeeds and the resulting parsing graph is illustrated in Figure 15. Each gray box in Figure 15 symbolizes a grammar rule. It is labeled with the resulting part of speech, and contains in bold the type of rule and in italics the semantic procedure associated with that rule.

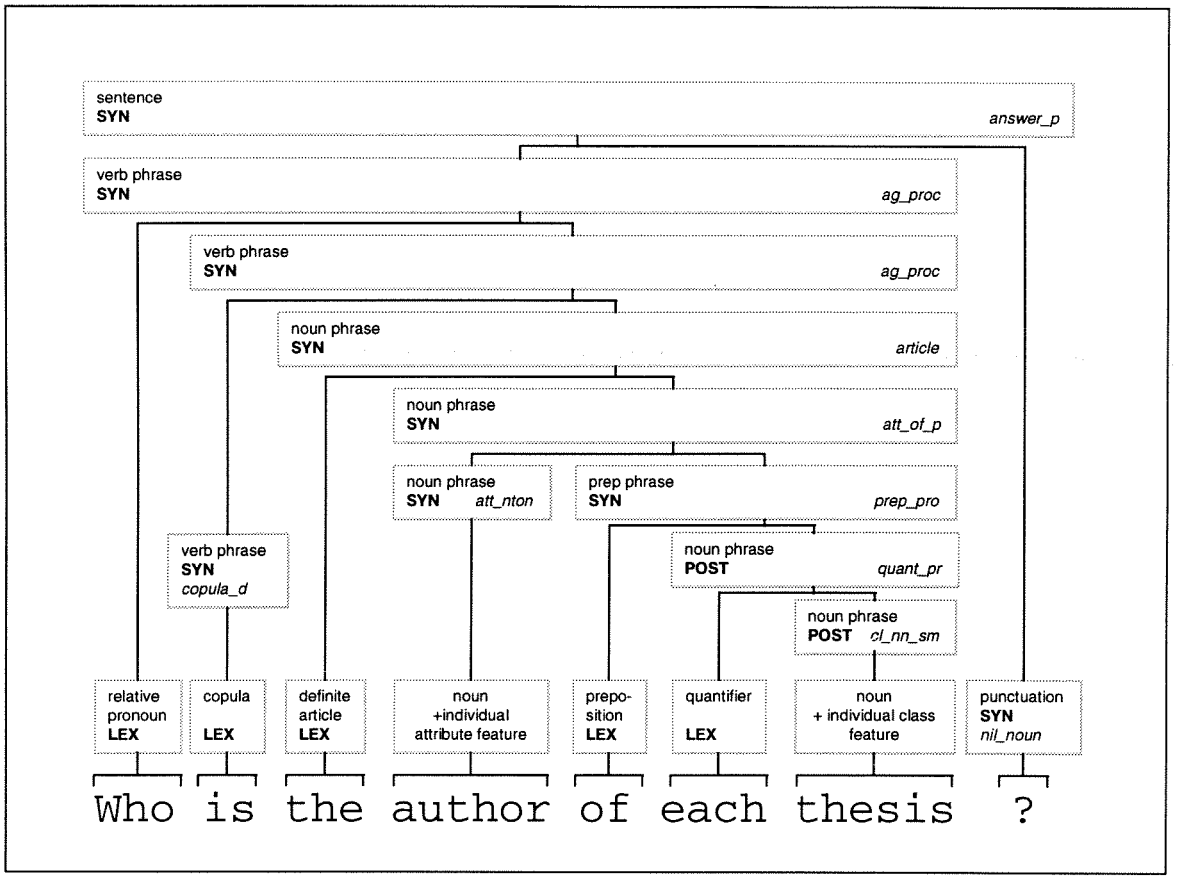

Figure 15

Here are the relevant grammar rules excerpted from the dictionary in their native format described earlier in Figure 1.

```
Dictionary for Context: Library

RULE the relative pronoun  "who"
<relative_pronoun:+who> => "who"
LEX 0

RULE "is"
<copula:+prim+fsi+is> => "is"
LEX 1

RULE "the"
<definite_article> => "the"
LEX 0

RULE "of"
<preposition:+of> => "of"
LEX 1
```

```
RULE the quantifier "each"
<quantifier:+sif+otf+each> => "each"
LEX 31

RULE "?"
<pct:+question_mark> => "?"
SYN nil_noun

RULE quantification of a singular noun
<noun_phrase:2+qnf-cjf> =>
       <quantifier:+sif-num> " "
       <noun_phrase:-plf-attribute-psf-qnf-dtf-pnf-cjf>
* e.g., "some ship"
POST quant_pr

RULE prepositional phrase
<prep_phrase> => <preposition> " " <noun_phrase:-typefeat5>
SYN prep_pro

RULE class noun to noun phrase
<noun_phrase:1> => <noun:+class-meta>
POST  cl_nn_sm

RULE attribute noun to noun phrase
<noun_phrase:1> => <noun:+attribute>
SYN  att_nton

RULE putting a definite article on a noun phrase
<noun_phrase:2+dtf+the> => <definite_article> " "
<noun_phrase:-psf>
* e.g., "the ship", "the (sister of John)"
*        *"(the sister) of John",
*        *"the (boy and [the] girl)"
SYN article

RULE genitive mod. ("of" phrase)
<noun_phrase:1+pmf-attribute+class-typefeat5-cjf-and-or>
     =>  <noun_phrase:+attribute-pof-dtf-qnf-psf-rcf-cjf>
     " " <prep_phrase:+of-pmf>
SYN att_of_p

RULE copula to verb_phrase
<verb_phrase:+copula> => <copula>
SYN copula_d

RULE agent noun in the question inversion position
<verb_phrase:1+fsj+fqt+fph+fag+fce> =>
      <verb_phrase:+copula+is-fin-fsj-foj-fag-fda-fce>
      " " <noun_phrase:-pof>
* e.g., "is John [an employee?]"
SYN ag_proc
```

```
RULE adding a RP as agent of a verb
<verb_phrase:2+fph+fsj+fin+fag+fce+fqt> =>
        <relative_pronoun:-that-whose-which-whom> " "
        <verb_phrase:-copula-fin-fpa-fsj-fag-fti>
* e.g., "(Who/What) (tells the commander)",
*       "(Who) (is the commander)",
*       "(Who) (has the commander's report)"
SYN  ag_proc

RULE sentence = transitive verb "?"
<left_delimiter> <sentence:+question> =>
    <left_delimiter> <verb_phrase:+fsj+foj-fpp-fvi>
    <pct:+question_mark>
SYN answer_p
```

Once the parse chart is built comes the time for syntax analysis. This is the first pass of the semantic processor on the chart. During this pass the chart will be traversed with all the procedures attached to a rule of type SYN being invoked. These procedures have purely transformational goals. We will quickly gloss over the lengthy process. Since we are mainly preoccupied with the semantics we will cheat by looking directly at the result which appears in Figure 16.

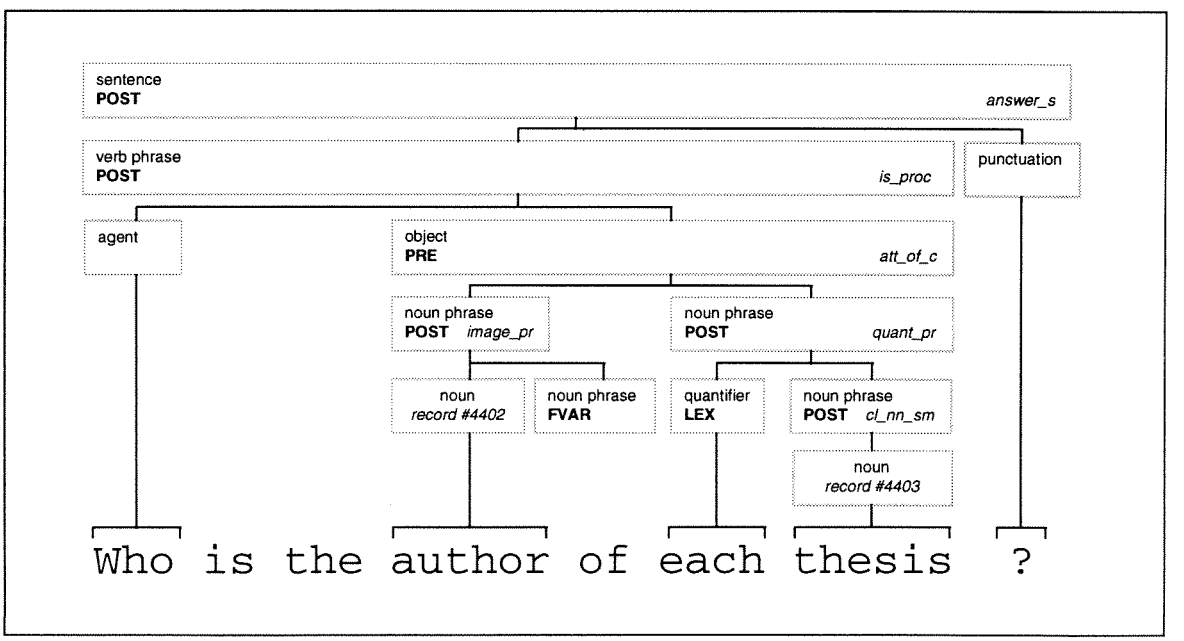

Figure 16

At this point, we have a chart expunged from its syntactic frivolities, and representing a road map to the true semantic meaning of the sentence. We are now going to follow that road map step by step. To do so, we have to familiarize ourselves with phrases in the bread and butter representation format of the semantic processor: I am talking about New World of Computing lists.

The New World of Computing list processor is a tagged ternary list engine. In fact the parse chart itself is a list since this list architecture is particularly well suited to sublanguage processing.

A list is constituted of a suite of linked list elements. A list element "Ln" is composed of five items, the id, the flag and three fields. It is displayed as:

```
L  n    ( id, flag, Field1, Field2, Field3 )
```

Each field can contain a wide variety of types, among others are: page pointers, string pointers, pairs of integers, reals, a set of flags called bools, dictionary pointers, generic pointers and of course list pointers to link to another list element.. The id serves as the tag, defining the types of fields and categorizing the element as a whole. The flag is a useful storage area for a 32 bit integer and serves numerous context-dependent purposes. List pointers are of a type much more common to traditional programmers. They only point inside the machine's virtual address space. This means that to reference a list across the New World of Computing's world-wide space, the list has to be flattened out and copied to a page. Utilities are provided to take care of this case.

With this architecture in mind, let's see what a phrase looks like in list format.

```
phrase   ( pos, 0, psyn, psem, link )
psyn     ( SYN, 0, [ feat ], name, arc )
psem     ( id, flag, ll, cons, payload )
```

Phrase glossary:

- **pos** : part of speech
- **psyn** : list pointer to the syntactic part of the phrase
- **psem** : list pointer to the semantic part of the phrase
- **link** : list pointer to the following phrase in the chart
- **SYN** : characterizes the list element as a psyn
- **feat** : a bool containing the features for the part of speech (e.g. noun:+class)
- **name** : a string pointer to the literal part of the phrase
- **arc** : a list pointer to the initial arc for this phrase in the parse chart, (usually not displayed)
- **id** : holds the rule type (e.g., LEX, SYN, PRE, POST)
- **flag** : integer; in the case of a LEX, it holds the flag from the lexicon entry.
- **ll** : list pointer to the label list which will describe the output format.
- **cons** : list pointer to the constituents of the phrase.
- **payload** : page pointer to the semantic routine associated with this rule.

It is the goal of each semantic routine to replace the psem with actual data (meaning.) In this case, the id is changed to a type descriptor (e.g., llp for a record; llp meaning the first and second fields are list pointers and the third is a page pointer.)

The following is the parse chart from Figure 16 represented as a list. The phrases are automatically separated with dashed lines.

```
L  0    ( sentence, 0, L  1, L  2, nil )
L  1    ( SYN, 0, [ 1 ], S  1, nil )
S  1    who is author of each thesis?
L  2    ( POST, 0, nil, L  3, <1,3239,0,(818)395-6231> )
         -----------
L  3    ( verb_phrase, 0, L  4, L  5, L  6 )
L  4    ( SYN, 0, [ 11 12 15 17 24 25 26 28 32 ], S  2, nil )
S  2    who is author of each thesis
L  5    ( POST, 0, nil, L  7, <1,3812,0,(818)395-6231> )
         -----------
L  7    ( agent, 0, L  8, nil, L  9 )
L  8    ( SYN, 0, [ 3 ], S  3, nil )
S  3    who
         -----------
L  9    ( object, 0, L 10, L 11, nil )
L 10    ( SYN, 0, [ 7 26 42 47 55 ], S  4, nil )
S  4    author of each thesis
L 11    ( PRE, 0, nil, L 12, <1,2921,0,(818)395-6231> )
         -----------
L 12    ( noun_phrase, 0, L 13, L 14, L 15 )
L 13    ( SYN, 0, [ 7 28 ], S  5, L--- )
S  5    author
L 14    ( POST, 0, L 16, L 17, <1,2817,0,(818)395-6231> )
L 16    ( FVAR, noun_phrase, nil, nil, <0,0,0,0> )
         -----------
L 17    ( noun, 0, L 18, L 19, L 20 )
L 18    ( SYN, 0, [ 7 28 ], S  6, nil )
S  6    author
L 19    ( llp, 0, L 21, nil, <1,4402,0,(818)395-6231> )
L 21    ( lil, 1, L 22, 1:0, L 23 )
L 22    ( lil, 1, nil, 1:0, nil )
L 23    ( lls, 0, nil, nil, S  7 )
S  7    author
         -----------
L 20    ( noun_phrase, 0, nil, L 16, nil )
         -----------
L 15    ( noun_phrase, 0, L 24, L 25, nil )
L 24    ( SYN, 0, [ 7 26 40 ], S  8, L--- )
S  8    each thesis
L 25    ( POST, 0, nil, L 26, <1,2854,0,(818)395-6231> )
         -----------
L 26    ( quantifier, 0, L 27, L 28, L 29 )
L 27    ( SYN, 0, [ 1 2 5 ], S  9, nil )
S  9    each
L 28    ( llp, 31, nil, L 30, <0,0,0,0> )
L 30    ( LLEX, 50, nil, nil, <23,4298,0,(818)395-6231> )
         -----------
L 29    ( noun_phrase, 0, L 31, L 32, nil )
L 31    ( SYN, 0, [ 7 26 ], S 10, L--- )
S 10    thesis
L 32    ( POST, 0, nil, L 33, <1,2904,0,(818)395-6231> )
```

```
          -----------
L 33      ( noun, 0, L 34, L 35, nil )
L 34      ( SYN, 0, [ 7 26 ], S 11, nil )
S 11      thesis
L 35      ( llp, 0, nil, L 36, <1,4403,0,(818)395-6231> )
L 36      ( LLEX, 10, nil, nil, <44,4258,0,(818)395-6231> )
          -----------
L  6      ( pct, 0, L 37, nil, nil )
L 37      ( SYN, 0, [ 0 ], S 12, nil )
S 12      ?
```

The semantic processor starts by walking down this chart, recursing on each constituent until it hits the object part of speech. Because its associated rule is of type PRE, it stops and invokes the attached semantic procedure. "*att_of_c*" is invoked whenever an attribute like author is applied to a class like thesis. As input it is fed the object phrase "*author of each thesis*".

The rule is a PRE because it needs to coerce the flow of the recursion of the semantic processor on its second constituent, the noun phrase "*each thesis*" before it lets the attribute procedures loose on it. If left to the regular operation, the semantic processor would have walked down the left branch first.

Hence, "*att_of_c*"'s first step is to evaluate the noun phrase "*each thesis*". This restarts the semantic processor on that particular branch of the chart, which keeps recursing until it hits the leaf phrase which is the noun "*thesis*".

We saw that the lexicon entry for thesis contains as payload the associated record pointer <1,4403,0,(818)356-6231>. This in turn happens to point to a page in the local paging file. Having hit data paydirt, the semantic processor then starts "rolling back up the rug" on the recursion.

The last procedure recursed upon was "*cl_nn_sm*". It is now invoked and since the noun_phrase it handles is simple, its sole task is to instantiate the psem of the phrase it is going to return with the record for "*thesis*". It also adorns it with a trivial label list that is of no concern to us for the purpose of this demonstration.

The result is:

```
(  5) SEM output:
          -----------
L  0     ( noun_phrase, 0, L  1, L  2, nil )
L  1     ( SYN, 0, [ 7 26 ], S  1, L--- )
S  1     thesis
L  2     ( llp, 0, L  3, nil, <1,4403,0,(818)395-6231> )
L  3     ( lil, 1, nil, 1:0, L  4 )
L  4     ( lls, 0, nil, nil, S  2 )
S  2     thesis
```

The number in parentheses indicates the depth level of the recursion, and if you check figure 16 you will see that "*cl_nn_sm*" is in fact five rules down from the top sentence rule.

The next procedure in our recursion stack is "*quant_pr*". It takes the psem of the noun phrase that was just returned by "*cl_nn_sm*" and promotes it up as the psem of the phrase it will return. It then quantifies it with the quantifier "*each*" contained in its first constituent, by tacking it into the resulting noun phrase's label list as its decimal value: 31.

```
(  4) SEM POST call procedure: <1,2854,0,(818)395-6231>
(  4) SEM output:
          -----------
L  0     ( noun_phrase, 0, L  1, L  2, nil )
L  1     ( SYN, 0, [ 7 26 40 ], S  1, L--- )
S  1     each thesis
L  2     ( llp, 0, L  3, nil, <1,4403,0,(818)395-6231> )
L  3     ( lil, 1, nil, 31:0, L  4 )
L  4     ( lls, 0, nil, nil, S  2 )
S  2     thesis
```

This ends the evaluation of that particular branch. The resulting phrase looks as follows:

```
ARG return from: evaluate, output:
L  0    ( lll, 0, nil, nil, L  1 )
L  1    ( noun_phrase, 0, L  2, L  3, nil )
L  2    ( SYN, 0, [ 7 26 40 ], S  1, L  4 )
S  1    each thesis
L  3    ( llp, 0, L  5, nil, <1,4403,0,(818)395-6231> )
L  5    ( lil, 1, nil, 31:0, L  6 )
L  6    ( lls, 0, nil, nil, S  2 )
S  2    thesis
L  4    ( quantifier, 0, L  7, L  8, L  9 )
L  7    ( SYN, 0, [ 1 2 5 ], S  3, nil )
S  3    each
L  8    ( llp, 31, nil, L 10, <0,0,0,0> )
L 10    ( LLEX, 50, nil, nil, <23,4298,0,(818)395-6231> )
L  9    ( " ", 0, nil, nil, L 11 )
L 11    ( noun_phrase, 0, L 12, L 13, nil )
L 12    ( SYN, 0, [ 7 26 ], S  4, L 14 )
S  4    thesis
L 13    ( POST, 0, nil, L 15, <1,2904,0,(818)395-6231> )
L 15    ( noun, 0, L 16, L 17, nil )
L 16    ( SYN, 0, [ 7 26 ], S  5, nil )
S  5    thesis
L 17    ( llp, 0, nil, L 18, <1,4403,0,(818)395-6231> )
L 18    ( LLEX, 10, nil, nil, <44,4258,0,(818)395-6231> )
L 14    ( noun, 0, L 16, L 17, nil )
```

"*att_of_c*"'s next task is to take the free variable (tagged with FVAR) that is "hidden" in its first constituent, the attribute "*author*", bind it, and instantiate it with the "*thesis*" noun phrase we just finished evaluating. It will now be tagged as a BVAR, or bound variable.

With this free variable out of the way, "*att_of_c*" can now combine its two constituents and attempt to evaluate the combination as a noun phrase. All these iterative evaluations may seem quite tedious, but one must remember that the semantic processor works directly on the parse chart, and thus any segment that is evaluated will be replaced by its semantic value. The semantic processor then only glosses over these, avoiding any duplicate or extraneous work.

The free variable being instantiated with the "*thesis*" noun phrase, "*image_pr*" now has all the arguments necessary to start doing the actual real data processing work required by this query. After doing some initial testing to see that the data is consistent, (e.g., the attribute and the class have compatible types and neither are empty,) the image of the "*thesis*" class under the "*author*" attribute is taken. In this simplistic case, this means that each member of the class is taken in turn and the attribute record is searched to see if it is an argument. For every argument found, the corresponding value is copied into an output record. Since the class is quantified with "*each*", the member argument is also copied into the output record as an output label.

The resulting scratch output record <1,4411,0,(818)395-6231> becomes the psem of the phrase.

This is what "*att_of_c*" returns:

```
(  3) SEM output:
          -----------
L  0     ( object, 0, L  1, L  2, nil )
L  1     ( SYN, 0, [ 7 26 42 47 55 ], S  1, nil )
S  1     author of each thesis
L  2     ( llp, 0, L  3, nil, <1,4411,0,(818)395-6231> )
L  3     ( lil, 1, L  4, 1:0, L  5 )
L  4     ( lil, 1, nil, 31:0, L  6 )
L  6     ( lls, 0, nil, nil, S  2 )
S  2     thesis
L  5     ( lls, 0, nil, nil, S  3 )
S  3     author
```

At this point all the actual semantic processing is done. The semantic routine "*is_proc*" that handles the "*Who is the author of each thesis*" verb phrase has very little to do. It must first realize that this is a simple case, which is not as obvious as it first may seem. There are so many occurrences of verb phrases of the form "<subject> is <object>" that the discrimination process is quite exhaustive. It must check for things like adverbs and also verify that the query is not negated, i.e., "<subject> is not <object>". Since both positive and negative verb phrases have very similar semantics,

it would be silly not to use the same procedure. Once it finds out that the agent is a mere "*who*", it pops it off, and the underlying noun phrase psem becomes the verb phrase psem.

```
(  2) SEM POST return from recursion on constituents:
(  2) SEM POST call procedure: <1,3812,0,(818)395-6231>
(  2) SEM output: (for is_proc)
           -----------
L  0     ( verb_phrase, 0, L  1, L  2, nil )
L  1     ( SYN, 0, [ 11 12 15 17 24 25 26 28 32 ], S  1, nil
)
S  1     who is author of each thesis
L  2     ( llp, 0, L  3, nil, <1,4411,0,(818)395-6231> )
L  3     ( lil, 1, L  4, 1:0, L  5 )
L  4     ( lil, 1, nil, 31:0, L  6 )
L  6     ( lls, 0, nil, nil, S  2 )
S  2     thesis
L  5     ( lls, 0, nil, nil, S  3 )
S  3     author
```

At this point, we have come all the way back to the surface, i.e. the sentence. "*answer_s*" is now in charge of turning the resulting semantic data into a more humanly palatable form. Since our output record is a simple labeled class, it quickly ends up in the hands of the "*out_rec*" subroutine. No fields need to be crunched out, and all the data has been successfully processed, so "*out_rec*"'s job consists mainly in formatting the record for output. It takes each entry in the output record in turn, and in our case, since all our fields are records (as opposed to number values for example), it fetches the permanent string (also a page pointer) which holds the name of the record in the current language. It tabulates these strings so as to have them all align in columns. Since this data is not time dependent, "*out_rec*" is done.

```
(  1) SEM POST return from recursion on constituents:
(  1) SEM POST call procedure: <1,3239,0,(818)395-6231>
(  1) SEM output: (for answer_s)
          -----------
L  0     ( sentence, 0, L  1, L  2, nil )
L  1     ( SYN, 0, [ 1 ], S  1, nil )
S  1     who is author of each thesis?
L  2     ( OUT, 0, L  3, nil, nil )
L  3     ( lls, 0, nil, L  4, S  2 )
S  2     thesis                    author
L  4     ( lls, 0, nil, nil, S  3 )
S  3     Communicative Databases  Kwang-I Yu
```

The list of formatted output is handed over to the outputer which produces the following:

```
~~~~~~~ OUTPUTER ANALYSIS ~~~~~~~~~~~~~~~~~
thesis                    author
Communicative Databases   Kwang-I Yu
```

2. THE NETWORKED CASE

Let us now switch our focus on the other computer. This one should be imagined as my home workstation where personal information is kept.

a) Personal station: setting up the data

We start by creating a new working sublanguage by basing it on the one we just left on the other computer. The only identification required to specify it, is its phone number.

```
COMMAND> base Reading on Library@(818)395-6231
     transfering dictionary for Reading
     transfering lexicon entries for Reading
     transfering dictionary for Meta-Reading
     transfering lexicon entries for Meta-Reading
 Context Reading has been created based on Library@(818)395-
6231.
```

We then enter that environment and start adding our own data on top. Note that we do not need to redefine thesis or author since we inherited them from our Library sublanguage.

```
COMMAND> enter Reading
 You are in Reading.  Proceed.
Reading> individual:Remy Sanouillet
 The new individual Remy Sanouillet has been added.
Reading> individual:Computer Mediated Communication
 The new individual Computer Mediated Communication has been
added.
Reading> Computer Mediated Communication is a thesis.
The individual entity Computer Mediated Communication has
been added to the class thesis.
Reading> Author of Computer Mediated Communication is Remy
Sanouillet.
 Remy Sanouillet was added as author of Computer Mediated
Communication.
```

We now submit the identical query and notice that our data has been seamlessly integrated into the existing web of information.

```
Reading> Who is the author of each thesis?
thesis                              author
Computer Mediated Communication     Remy Sanouillet
Communicative Databases             Kwang-I Yu
```

We will now look in detail at how this worked, and how the data was transported across.

b) Examining the internal data structures

First let's examine the data records and see how they differ from the self-contained case.

```
META-Reading> dump record for Kwang-I Yu

Page number:     <1,4406,0,(818)395-6231>
   Parent page : <1,4406,0,(818)395-6231>
   Record type :      1
   Header size :      9
   Fields/entry:      1
   Num entries :     11
Not Timed Record
Not Indexed
Permanent
Individuals only
Header_bool : [ 0 5 ]
Data_page: <0,0,0,0>
Name in English : "Kwang-I Yu"
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:20101 <1,4402,0,(818)395-6231>
 Record for Kwang-I Yu dumped.
```

The record for Kwang-I Yu is identical. Not only is it identical, it is the actual record from the Library as can be seen from the phone number field of its pointer. The same would be true of the record for Communicative Databases. All the records in the Library sublanguage are unaffected by our activity here.

```
META-Reading> dump record for Remy Sanouillet

Page number:     <1,4403,0,(818)395-6232>
   Parent page : <1,4403,0,(818)395-6232>
   Record type :      1
   Header size :      9
   Fields/entry:      1
   Num entries :     11
Not Timed Record
Not Indexed
Permanent
Individuals only
Header_bool : [ 0 5 ]
Data_page: <0,0,0,0>
Name in English : "Remy Sanouillet"
Name in French  : ""
Name in Russian : ""
```

```
Record list:

   Offset:  9 flag:20101 <1,4406,0,(818)395-6232>
 Record for Remy Sanouillet dumped.
```

Now, let's look at a local record. Outside the fact that it is pointing to this computer, its structure is similar to that of Kwang-I Yu's record. It is a value of the individual/individual attribute held in <1,4406,0,(818)395-6232>. Note that this is a local record, not the record for the attribute author held in Library.

```
META-Reading> dump record for Computer Mediated
Communication

Page number:     <1,4404,0,(818)395-6232>
   Parent page : <1,4404,0,(818)395-6232>
   Record type :      1
   Header size :      9
   Fields/entry:      1
   Num entries :     13
Not Timed Record
Not Indexed
Permanent
Individuals only
Header_bool : [ 0 5 ]
Data_page: <0,0,0,0>
Name in English : "Computer Mediated Communication"
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:   20 <1,4405,0,(818)395-6232>
   Offset: 11 flag:  101 <1,4406,0,(818)395-6232>
 Record for Computer Mediated Communication dumped.
```

The same holds true for the Computer Mediated Communication's record which is a member of a local class, <1,4405,0,(818)395-6232>, and an argument of the attribute <1,4406,0,(818)395-6232>. How can this be since we did not create any local classes or attributes?

```
META-Reading> dump record for thesis

Page number:     <1,4405,0,(818)395-6232>
   Parent page : <1,4405,0,(818)395-6232>
   Record type :     21
   Header size :      9
   Fields/entry:      1
   Num entries :     13
Not Timed Record
Not Indexed
Permanent
Referent
Header_bool : [ 5 6 ]
Act_page: <0,0,0,0>
Name in English : "thesis"
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:     0 <51,4259,0,(818)395-6232>
   Offset: 11 flag:     1 <1,4404,0,(818)395-6232>
 Record for thesis dumped.
```

The answer lies in the basing process. Our assertion "*Computer Mediated Communication is a thesis.*" would have required the addition of an entry in the Library's record for thesis. The librarian would probably not eye such an activity with great enthusiasm. To avoid this, the system created an ambiguous "*thesis*" entry in the lexicon, changed the old entry type from agent to referent, and created the above record as payload for the new entry. We can see that its bool flags have changed from individuals_only to referent, and that it contains a member entry for our Computer Mediated Communication record. But more importantly, it contains a referent entry (flag of zero) that points to the ambiguous lexical entry. We will see the mechanisms of dereferencing this a little further on.

```
META-Reading> dump record for author

Page number:     <1,4406,0,(818)395-6232>
   Parent page : <1,4406,0,(818)395-6232>
   Record type :   101
   Header size :     9
   Fields/entry:     2
   Num entries :    15
Not Timed Record
Not Indexed
Permanent
Referent
Header_bool : [ 5 6 ]
Act_page: <0,0,0,0>
Name in English : "author"
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:    0 <86,4273,0,(818)395-6232>
<86,4273,0,(818)395-6232>
   Offset: 12 flag:  100 <1,4404,0,(818)395-6232>
<1,4403,0,(818)395-6232>
 Record for author dumped.
```

Similarly, the record for author was added as a referent when we said: "*Author of Computer Mediated Communication is Remy Sanouillet.*" Besides the object entry (flag of one hundred) that the assertion added, we have a similar referent entry pointing to an ambiguous lexical entry.

Because the Reading sublanguage is based upon Library, and because the only modifications we've made are semantic in nature (i.e., we haven't added new grammar rules, or modified the procedures), the resulting parse chart for the query will be structurally identical. All the data will be virtually identical since copied through the basing process. All page references are local except for "standard" data (i.e., data from the Library sublanguage that has been added by the user). Except for the following exceptions, the lexicon is a clone of the Library lexicon.

```
Reading> metalanguage
 You are now in META-Reading.  Proceed.
META-Reading> dump lexicon
Lexicon entries for Context Reading
     lex_master_page: <1,4076,0,(818)395-6232>

     Bucket #10 page: <2,4259,0,(818)395-6232>

thesis
    pos: noun  type: standard  feat: [ 7 26 ]
    id for psem: 1025  flag: 0
    payload: <1,4405,0,(818)395-6232>
    amb link: <51,4259,0,(818)395-6232>  def msg: thesis
initially created in Context Library.
  (amb)
    pos: noun  type: referent  feat: [ 7 26 ]
    based on lexical pointer: <44,4258,0,(818)395-6231>
    amb link: <0,0,0,0> agent: <1,4054,0,(818)395-6231>

     Bucket #12 page: <2,4261,0,(818)395-6232>

(818)395-6232
    pos: station_name  type: initial  feat: [ ]
    id for psem: 1025  flag: 2  payload: <0,0,0,0>
    amb link: <0,0,0,0>  no def msg

     Bucket #19 page: <2,4268,0,(818)395-6232>

Computer Mediated Communication
    pos: noun  type: standard  feat: [ 7 25 ]
    id for psem: 1025  flag: 0
    payload: <1,4404,0,(818)395-6232>
    amb link: <0,0,0,0>  def msg: Computer Mediated
Communication

     Bucket #24 page: <2,4273,0,(818)395-6232>

author
    pos: noun  type: standard  feat: [ 7 28 ]
    id for psem: 1025  flag: 0
    payload: <1,4406,0,(818)395-6232>
    amb link: <86,4273,0,(818)395-6232>  def msg: author
initially created in Context Library.
  (amb)
    pos: noun  type: referent  feat: [ 7 28 ]
    based on lexical pointer: <79,4272,0,(818)395-6231>
    amb link: <0,0,0,0> agent: <1,4054,0,(818)395-6231>
```

```
     Bucket #28 page: <2,4277,0,(818)395-6232>

Communicative Databases
    pos: noun  type: agent  feat: [ 7 25 ]
    based on lexical pointer: <72,4276,0,(818)395-6231>
    amb link: <0,0,0,0> agent: <1,4054,0,(818)395-6231>

     Bucket #29 page: <2,4278,0,(818)395-6232>

Remy Sanouillet
    pos: noun  type: standard  feat: [ 7 25 ]
    id for psem: 1025  flag: 0
    payload: <1,4403,0,(818)395-6232>
    amb link: <0,0,0,0>  def msg: Remy Sanouillet

     Bucket #35 page: <2,4284,0,(818)395-6232>

(818)395-6231
    pos: station_name  type: initial  feat: [ ]
    id for psem: 1025  flag: 1  payload: <0,0,0,0>
    amb link: <0,0,0,0>  no def msg

     Bucket #45 page: <2,4294,0,(818)395-6232>

Kwang-I Yu
    pos: noun  type: agent  feat: [ 7 25 ]
    based on lexical pointer:<135,4293,0,(818)395-6231>
    amb link: <0,0,0,0> agent: <1,4054,0,(818)395-6231>
```

c) Processing the query

The semantic process follows the same venue as in the previous case. The crucial differences occur in "*cl_nn_sm*" and in "*att_nton*", the two routines that transform the underlying nouns into noun phrases. Upon carrying over the psem record for the noun "*thesis*", "*cl_nn_sm*" detects that it is a referent record. It contains a record from the other sublanguage as a subclass and therefore must expand it into a scratch record which will serve in lieu of the actual permanent record, but will be discarded when the query finishes.

To perform the expansion, it takes each entry in the record in turn, copies it into the temporary record if it is not flagged as referent. If it is flagged as a referent as is the case for the first entry in the record for "*thesis*", then that entry points to the

ambiguous referent entry we noticed earlier in the lexicon. "*cl_nn_sm*" loads that lexicon bucket, checks the entry, and finds, in that entry, the lexicon pointer that "*thesis*" is based upon.

It then attempts to load that foreign bucket: <44,4258,0,(818)395-6231> by calling the "*page*" routine it has been using all along for local pages. "*page*" is just an alias for a "*generic_page*" request for an unlocked, unmarked page.

The first step consists in checking the local paging area via "*locate_page*" to see if we happen to have that page already in cache. Supposing the page hasn't been recently requested, "*locate_page*" will fail, and "*generic_page*" will grab an available page slot in the cache and invoke "*bring_in_page*" specifying that this is a standard request, as opposed to pages which require special handling (e.g. the Version Control Page). The rôle of "*bring_in_page*" is to load the desired page into the designed slot and perform typical administrative tasks.

To load the page it in turn invokes "*pg_read*" passing along the standard request message code. "*pg_read*" is the one that actually distinguishes between a request for a local page and one for a distant page. In our case (818)395-6231 being different from our phone number, this is a distant page so it tags the empty page slot with the page pointer it desires and calls "*foreign_page*" which does a "*send_page*" and then waits for a reply using a "*get_page*" – the basic data exchange.

In the mean time, "*send_page*" checks to see if the appropriate connection is already established, otherwise it will dial up the foreign computer and initiate the link using "*contact_station*". If the connection is established, it will pass the request over to the distant computer using "*send_foreign_page*" which is the device handler at the bottom of the protocol stack that writes to the communication device.

On the other side of the communication channel sits the New World of Computing daemon. It is awakened from its sleep by the arrival of this request. It identifies the station from which this request is coming (i.e., its return address) and does a "*get_foreign_page*" to retrieve the request.

It recognizes the message code as a standard page request, and if the New World of Computing process were not running, it would know how to retrieve the page directly from the paging file. Assuming that in our case the Library station is always up, it interrupts the program and forwards the request by using a "*send_foreign_page*" on the internal communication channel. It then waits for a reply using a "*get_foreign_page*".

The interrupt handler for the internal communication channel of the New World of Computing process is called "*err_urg*". Its first step is to retrieve the request from the channel using a "*get_foreign_page*". Then decoding this as a standard request, it does exactly the same "*page*" as the one that started this whole process, but since now the request is local, the page is copied from disk (or cache) into a buffer and sent back the way the request came using a "*send_foreign_page*".

Provided no communication problems occurred on the way, the page buffer returns along the reverse path into the desired page slot, and barring the delay, the whole process is indistinguishable from a local page request.

We now have the foreign lexicon entry for "*thesis*", and we recurse on its payload. Since this record is not a referent (it wasn't based on anything else), its single entry for Communicative Databases is added to the temporary record. So our temporary "*thesis*" record should now look like this:

```
Page number:     <1,4475,0,(818)395-6232>
   Parent page : <1,4475,0,(818)395-6232>
   Record type :    21
   Header size :     9
   Fields/entry:     1
   Num entries :    13
Not Timed Record
Not Indexed
Temporary
Individuals only
Header_bool : [ 0 4 ]
Act_page: <0,0,0,0>
Name in English : ""
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:     1 <1,4404,0,(818)395-6232>
   Offset: 11 flag:     1 <1,4405,0,(818)395-6231>
```

"*att_nton*" follows a parallel process resulting in the following temporary record for "*author*":

```
Page number:     <1,4476,0,(818)395-6232>
   Parent page : <1,4476,0,(818)395-6232>
   Record type :   101
   Header size :     9
   Fields/entry:     2
   Num entries :    15
Not Timed Record
Not Indexed
Temporary
Individuals only
Header_bool : [ 0 4 ]
Act_page: <0,0,0,0>
Name in English : ""
Name in French  : ""
Name in Russian : ""

Record list:

   Offset:  9 flag:  100 <1,4404,0,(818)395-6232>
<1,4403,0,(818)395-6232>
   Offset: 12 flag:  100 <1,4405,0,(818)395-6231>
<1,4406,0,(818)395-6231>
```

When "*image_pr*" takes the image of the first record through the second attribute record, it ends up getting the record for Kwang-I Yu in a totally transparent manner. Similarly when the outputer requests the permanent strings that are the labels for these records, it will have no knowledge that these strings are located on a page that had to travel across a network. As far as it is concerned, the record name pointer just pointed to an area of relevant data, and in every detail that is how this whole metaphor works.

3. CONSEQUENCE

Without my work, the New World of Computing would be limited to cases like the first one. The pervasive introduction of the universal pointer has enabled these systems to preserve information closest to the person best qualified to maintain it (e.g., the librarian in our case), while still allowing individuals the flexibility to customize their information space without fear of either the base data becoming obsolete, or of corrupting the shared information.

V. In Conclusion

The author has implemented the capabilities discussed above. They are functioning within the encompassing New World of Computing System on Sun SPARC Stations, using Ethernet communications and OpenWindows. The NCR Corporation is currently participating with me in porting this System to the MSDOS/Windows environment on NCR Personal Computers, as well as in its product development. The forecast is highly optimistic for the era of the telephone-computer.

VI. Bibliography

[ANDERSON75] Anderson, B. F. 1975. Cognitive Psychology: the Study of Knowing, Learning and Thinking. New York: Academic Press.

[ANDERSON76] Anderson, J. R. 1976. Language, Memory and Thought. Hillsdale, N.J.: Lawrence Erlbaum Associates.

[ARBIB] Arbib, M. A. 1972. The Metaphorical Brain: An Introduction to Cybernetics as Artifical Intelligence and Brain Theory. New York: Wiley Intersciences.

[BARBIZET] Barbizet, J. 1969. Pathologie de la mémoire. Paris: Presses Universitaires de France.

[BARTEE86] Bartee, T. C.
1986. Digital Communications. New York: Sams.

[BARTEE87] Bartee, T. C.
1987. Data Communications, Networks and Systems. New York: Sams.

[BARTLETT] Bartlett, F. C. 1958. Thinking: An Experimental and Social Study. New York: Basic Books.

[BENNET84] Bennet, J. L. 1984. Managing to Meet Usability Requirements. Visual Display Terminals: Usability Issues and Health Concerns, editors J. L. Bennet, D. Case, J. Sandelin, and M. Smith. Englewood Cliffs, N.J.: Prentice-Hall, Inc.

[BENNET86] Bennet, J. L. 1986. Tools For Building Advanced User Interfaces. IBM Systems Journal, 25, (3/4): 354-368.

[BERNE] Berne, H. 1975. Des jeux et des hommes: psychologie des relations humaines. Paris: Stock.

[BEVER] Bever, T. G. 1970. The Cognitive Basis for Linguistic Structures. Cognition and the Development of Language, editor, J. R. Hayes. New York: John Wiley & Sons.

[BLOCK] Block, E. G. 1989. ISDN: The Telcos Are Ready, But Are the Users? Telecommunications, May 1989, 31.

[BOBROW75] Bobrow, D. G., and D. A. Norman. 1975. Some Principles of Memory Schemata. Representation and Understanding Studies in Cognitive Science, editors D. G. Bobrow, and A. M. Collins. New York: Academic Press.

[BOBROW77] Bobrow, D. G., and T. Winograd. 1977. An Overview of KRL, a Knowledge Representation Language. Journal of Cognitive Science.

[BROOKS] Brooks, F. P. Jr. 1975. The Mythical Man-Month. Reading, Mass.: Addison-Wesley Publishing Co.

[BRUCE] Bruce, B. 1975. Case Systems for Natural Language. Artificial Intelligence.

[BUSINESS] Artificial Intelligence, The Second Computer Age Begins. 1982. Business Week. March 8, 1982: 66-72.

[CHOMSKY67] Chomsky, N. 1967. The Formal Nature of Language. Biological Foundations of Language, editor, E. H. Lenneberg. New York: John Wiley & Sons.

[CHOMSKY73] Chomsky, N., and M. Halle. 1973. Principes de phonologie générative. Paris: Seuil.

[CHOMSKY75] Chomsky, N. 1975. Reflections on Language. New York: Pantheon Books.

[CICOUREL] Cicourel, A. V. 1974. Cognitive Sociology: Language and Meaning in Social Interaction. New York: The Free Press.

[COFER] Cofer, C. N., editor. 1976. The Structure of Human Memory. San Francisco: W. H. Freeman.

[DREYFUS79] Dreyfus, H. L. 1972, 1979. What Computers Can't Do: A Critique of Artificial Intelligence Reason. New York: Harper & Row.

[DREYFUS85] Dreyfus, H. L., and S. E. Dreyfus. 1985. Mind Over Machine. New York: Macmillan/The Free Press

[ENDERTON] Enderton, H. B. 1972. A Mathematical Introduction to Logic. New York: Academic Press.

[EVANS] Evans, C. 1979. The Micro Millenium. New York: Viking.

[FEIGENBAUM63] Feigenbaum, E., and J. Feldman. 1963. Computers and Thought. New York: McGraw-Hill.

[FEIGENBAUM83] Feigenbaum, E., and P. McCorduch. 1983. The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World. Reading, Mass.: Addison-Wesley.

[FILLMORE68] Fillmore, C. J. 1968. The Case for Case. Universals in Linguistic Theory, editors E. Bach and R. G. Harms. New York: Holt.

[FILLMORE69] Fillmore, C. J. 1969. Toward a Modern Theory of Case. Modern Studies in English, editors D. A. Reibel and S. A. Schane. New York: Prentice-Hall.

[FILLMORE75] Fillmore, C. J. 1975. An Alternative to Checklist Theories of Meaning. Proceedings of the First Annual Meeting of the Berkeley Linguistics Society. University of California, Berkeley.

[FLORES] Flores, F. C. 1982. Management and Communication in the Office of the Future. Report. San Francisco: Hermenet Inc.

[FODOR74] Fodor, J., T. Bever, and M. Garret. 1974. The Psychology of Language. New York: McGraw-Hill.

[FODOR81] Fodor, J. 1981. Methodological Solipism Considered as a Research Strategy in Cognitive Psychology. The Behavioral and Brain Sciences. Haugeland.

[FROMKIN] Fromkin, V. A., and R. Rodman. 1974. An Introduction to Language. New York: Holt, Rinehart, and Winston.

[GREEN] Green, P E, R. J. Chapuis, J. D. Fisher, P.S. Frosch, and C. E. Wood. 1987. A Perspective on Advanced Peer-to-Peer Networking. IBM Systems Journal 26 (4): 414-428.

[GROSZ] Grosz, B. 1980. Utterance and Objective: Issues in Natural Language Communication. AI Magazine (Spring).

[GULLO] Gullo, K. 1989. Untangling ISDN. Information Week 247 (November 27).

[HARRAH] Harrah, D. 1963. Communication: A Logical Model. Cambridge, Mass.: M.I.T. Press.

[HEIDEGGER68] Heidegger, M. 1968. Was heisst Denken? (What is Called Thinking?). New York: Harper & Row.

[HEIDEGGER77] Heidegger, M. 1977. The Question Concerning Technology. New York: Harper & Row

[HILTZ85] Hiltz, S. R. and M. Turroff. 1985. Structuring Computer Mediated Communication Systems to Avoid System Overload, Commun. ACM 28 (July 7): 680-689.

[HILTZ88] Hiltz, S. R. and M. Turoff. 1988. The Network Nation - Human Communication via Computer. Reading, Mass.: Addison-Wesley Publishing Co.

[HOSTATER] Hostater, D. 1979. Godel, Escher, Bach: An Eternal Golden Braid. New York: Basic Books.

[INFORMATION] Information Computer Communications Policy, New Telecommunications Services - Videotex Development Strategies. 1988. Organisation for Economic Co-operation and Development.

[INOSE] Inose, H. 1979. Digital Integrated Communications Systems. Tokyo: University of Tokyo Press.

[JOHNSON] Johnson-Laird, P. N. 1974. Experimental Psycholinguistics, Annual Review of Psychology.

[KASSON] Kasson, J. M. 1986. An Advanced Voice/Data Telephone Switching System. IBM Systems Journal 25 (3/4): 380-398.

[KATZ] Katz, J. J., and J. A. Fodor. 1964. The Structure of Semantic Theory. The Structure of Language. New York: Prentice-Hall.

[KINTSCH] Kintsch, W. 1974. The Representation of Meaning in Memory. Hillsdale, N.J.: Lawrence Erlbaum Associates.

[KORZENIOWSKI] Korzeniowski. 1988. Users Chart Ups, Downs of ISDN Technology, Communications Week (October 24): 9.

[KUHN] Kuhn, T. 1962. The Structure of Scientific Revolutions. Chicago: University of Chicago Press.

[LAKOFF] Lakoff G., and M. Johnson. 1980. Metaphors We Live By. Chicago: University of Chicago Press.

[LAURIERE] Lauriere, J.-L. 1990. Problem Solving and Artificial Intelligence. New York: Prentice-Hall.

[LEWIS] Lewis, H. R., and C. H. Papadimitriou. 1981. Elements of the Theory of Computation. New York: Prentice-Hall.

[LINDSAY] Lindsay, P. H., and D. A. Norman. 1977. Human Information Processing, an Introduction to Psychology. New York: Academic Press.

[MARTIN] Martin, J. 1981. Telematic Society - A Challenge for Tomorrow. New York: Prentice-Hall.

[MASSARO75-1] Massaro, D. W. 1975. Experimental Psychology and Information Processing. New York: Rand McNally.

[MASSARO75-2] Massaro, D. W., editor. 1975. Understanding Language: an Information-Processing Analysis of Speech Perception, Reading and Psycholinguistics. New York: Academic Press.

[MEEKS] Meeks, B. N. 1986. High-Tech Conferencing for Humans. Microtimes (February): 46-48.

[MINSKY] Minsky, M. 1979. The Society Theory of Thinking. Artificial Intelligence: An M.I.T. Perspective, editors, Winston and Brown. Cambridge, Mass.: M.I.T. Press.

[NEWELL] Newell, A., and H. A. Simon, 1972. Human Problem Solving. New York: Prentice-Hall.

[NORMAN70] Norman, D.A., editor. 1970. Models of Human Memory. New York: Academic Press.

[NORMAN76-1] Norman, D. A. 1976. Memory and Attention: an Introduction to Human Information Processing. New York: John Wiley & Sons.

[NORMAN76-2] Norman, D.A., D. R. Gentner, and A. L. Stevens. 1976. Comments on Learning: Schemata and Memory Representation. Cognition and Instruction, editor D. Klahr. Hillsdale, N.J.: Lawrence Erlbaum Associates.

[NORMAN81] Norman, D. A., editor. 1981. Perspectives on Cognitive Science. New York: Ablex Publishing Corp.

[PENROSE] Penrose, R. 1989. The Emperor's New Mind. London: Oxford University Press.

[PIAGET64] Piaget, J. 1964. Six études de psychologie. Paris: Gonthier.

[PIAGET68] Piaget, J. 1968. La Naissance de l'intelligence chez l'enfant. Neuchâtel: Delachaux-Niestlé.

[ROSCH73] Rosch, E. H. 1973. On the Internal Structure of Perceptual and Semantic Categories. Cognitive Development and the Acquisition of Language, editor, T. Moore. New York: Academic Press.

[ROSCH75] Rosch, E. H. 1975. Cognitive Representations of Semantic Categories. Journal of Experimental Psychology: General.

[RUSSEL] Russel, B. 1920. Introduction to Mathematical Philosophy. Boston: Allen and Unwin.

[RUTKOWSKI] Rutkowski, A. M. 1986. Broadband Integrated Services Digital Networks. Telecommunications (December): 68.

[SARIN] Sarin, S., and I. Greif. 1985. Computer-Based Real-Time Conferences. M.I.T. Laboratory for Computer Science, M.I.T./LCS/TM 282 (July).

[SCHANK72] Schank, R. C. 1972. Conceptual Dependency: a Theory of Natural Language Understanding. Cognitive Psychology.

[SCHANK73] Schank, R. C. c1973. Computer Models of Thought and Language, editors Schank, R. C., and K. M. Colby. San Francisco: W. H. Freeman.

[SCHANK75] Schank, R. C. 1975 Conceptual Information Processing. New York: American Elsevier.

[SCHANK77] Schank, R. C. and R. P. Alberson. 1977. Scripts, Plans, Goals, and Understanding : an Inquiry into Human Knowledge Structures. Hillsdale, N.J.: Lawrence Erlbaum Associates. Distributor, Halsted Press Division, John Wiley & Sons.

[SCHANK81] Schank, R. C. and C. Riesbeck. 1981. Inside Computer Understanding, Hillsdale, N.J.: Lawrence Erlbaum Associates.

[SCHANK82] Schank, R. C. 1982. Dynamic Memory : a Theory of Reminding and Learning in Computers and People. London: Cambridge University Press.

[SCHWARTZ] Schwartz, M. 1987. Telecommunications Networks. Reading, Mass.: Addison-Wesley Publishing Co.

[SEARLE69] Searle, J. R. 1969. Speech Acts: an Essay in the Philosophy of Language. London: Cambridge University Press.

[SEARLE71] Searle, J. R., editor. c1971. The Philosophy of Language. London: Oxford University Press.

[SEARLE79] Searle, J. R. 1979. Expression and Meaning: Studies in the Theory of Speech Acts. London: Cambridge University Press.

[SEARLE83] Searle, J. R. 1983. Intentionality: an Essay in the Philosophy of Mind. London: Cambridge University Press.

[SEARLE84] Searle, J. R. 1984. Minds, Brains, and Science. Cambridge, Mass.: Harvard University Press.

[SHIFFRIN] Shiffrin, R. M. 1976. Capacity Limitations in Information Processing, Attention and Memory. Handbook of Learning and Cognitive Processes, Vol. 4: Memory Processes, editor, W. K. Estes. Hillsdale, N.J.: Lawrence Erlbaum Associates.

[SIMON60] Simon, H. A. 1960. The New Science of Management Decision. New York: Harper.

[SIMON65] Simon, H. A. 1965. The Shape of Automation for Men and Management. New York: Harper & Row.

[SIMON72] Simon, H. A., and L. Siklossy. 1972. Representation and Meaning; Experiments with Information Processing Systems. New York: Prentice-Hall.

[SIMON76] Simon, H. A. 1976. Administrative Behavior. New York: The Free Press.

[SIMON77] Simon, H. A. 1977. Models of Discovery: and Other Topics in the Methods of Science. Boston: D. Reidel Publishing Co.

[SIMON79] Simon, H. A. 1979. Models of Thought. New Haven: Yale University Press.

[SIMON81] Simon, H. A. 1981. The Sciences of the Artificial. Cambridge, Mass.: M.I.T. Press.

[SIMON82] Simon, H. A. 1982. Models of Bounded Rationality. Cambridge, Mass.: M.I.T. Press.

[STALLINGS89-1] Stallings, W. 1989. Handbook of Computer Communications Standards. Vol. 1. New York: Stallings/Macmillan.

[STALLINGS89-2] Stallings, W. 1989. ISDN An Introduction. New York: Macmillan.

[STALLINGS85] Stallings, W. 1988-1985. Data and Computer Communications. New York: Macmillan,

[STAMPER] Stamper, D. A. 1989-1986. Business and Data Communications. Benjamin Cummings.

[STOCKTON] Stockton, W. 1980. Creating Computers to Think like Humans. Time Magazine, December 7.

[TARSKI46] Tarski, A. c1946. Introduction to Logic and to the Methodology of Deductive Sciences, translator, O. Helmer. London: Oxford University Press.

[TARSKI56] Tarski, A. 1956. Logic, Semantics, Metamathematics: Papers from 1923 to 1938, translator, J. H. Woodger. Oxford: Clarendon Press.

[THOMPSON61] Thompson, F. B. 1961. Design Fundamentals of Military Information Systems.

[THOMPSON75] Thompson B. H., and F. B. Thompson. 1975. Practical Natural Language Processing. Advances in Computers. New York: Academic Press.

[THOMPSON87] Thompson B. H., and F. B. Thompson. 1987. Operating System Considerations in The New World of Computing. Pasadena: California Institute of Technology.

[THOMPSON91] Thompson B. H., and F. B. Thompson. 1991. The New World of Computing: A Solution to the Problems of the Telephone-Computer Era. Pasadena: California Institute of Technology.

[TURKLE] Turkle, S. c1984. The Second Self: Computers and the Human Spirit. New York: Simon and Schuster.

[WEIZENBAUM66] Weizenbaum, J. 1966. ELIZA. Communications of the ACM 9 (1).

[WEIZENBAUM76] Weizenbaum, J. 1976. Computer Power and Human Reason: from Judgment to Calculation. San Francisco: W. H. Freeman.

[WINKLER] Winkler, S. 1972. Proceedings of the First International Conference on Computer Communication - Computer Communication - Impacts and Implications. ICCC.

[WINOGRAD72] Winograd, T. 1972. Understanding Natural Language. New York: Academic Press.

[WINOGRAD74] Winograd, T. 1974. When Will Computers Understand People? Psychology Today 7 (12) May 1974.

[WINOGRAD76] Winograd, T. 1976. Towards a Procedural Understanding of Semantics. Revue Internationale de Philosophie (3):117-118.

[WINOGRAD79] Winograd, T. 1979. Beyond Programming Languages. Communications of the ACM 22 (7).

[WINOGRAD82] Winograd, T. 1982. What Does It Mean to Understand Language. Perspectives on Cognitive Science, editor, D. A. Norman. Norwood, N.J.: Ablex Publishing Corp.

[WINOGRAD83] Winograd, T. 1983-. Language as a Cognitive Process. Reading, Mass.: Addison-Wesley Publishing Co.

[WINOGRAD85] Winograd, T. 1985. Moving the Semantic Fulcrum. Linguistics and Philosophy 8 (1).

[WINOGRAD86] Winograd, T. c1986. Understanding Computers and Cognition: a New Foundation for Design. Norwood, N.J.: Ablex Publishing Corp.

[WINOGRAD87-1] Winograd, T., and F. Flores. 1987-1986. Understanding Computers and Cognition. Norwood, N.J.: Ablex Publishing Corp.

[WINOGRAD87-2] Winograd, T. 1987. A Language/Action Perspective on the Design of Cooperative Work. Human-Computer Interaction 3.

[WINSTON73] Winston, P. H. 1973. Learning to Identify Toy Block Structures. Contemporary Issues in Cognitive Psychology: the Loyola Symposium, editor, R. L. Solo. Winston. Distributor, Halsted Press Division, John Wiley and Sons.

[WINSTON75] Winston, P. H., editor, B. Horn et al. c1975. The Psychology of Computer Vision. New York: McGraw-Hill.

[WINSTON77] Winston, P. H. 1977. Artificial Intelligence. Reading, Mass.: Addison-Wesley Publishing Co.

[WINSTON79] Winston, P. H., and R. H. Brown, editors. 1982, c1979. Artificial Intelligence, an M.I.T.perspective. Cambridge, Mass.: M.I.T.Press.

[WINSTON84] Winston, P. H., and K. A. Prendergast, editors. c1984. The AI Business: the Commercial Uses of Artificial Intelligence. Cambridge, Mass.: M.I.T. Press.

[WORLDPOP] 1990. World Population Data Sheet. Washington, D.C.: Population Reference Bureau, Inc.

[WORLDFACT] 1990. The World Factbook. Washington, D.C.: Central Intelligence Agency.

[YU] Yu, K.-I. 1981. Communicative Databases. Pasadena: California Institute of Technology.

•