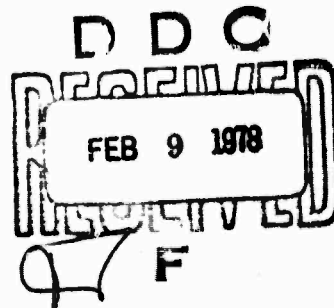


Bolt Beranek and Newman Inc.



12
B.S.

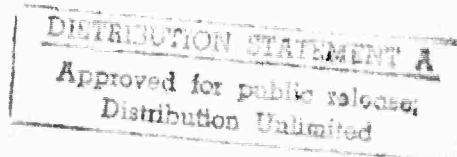
Report No. 3742



Research in Natural Language Understanding

Quarterly Technical Progress Report No. 1, 1 September 1977 to 30 November 1977

January 1978



Prepared for:
Advanced Research Projects Agency

AD A 049781

FILE COPY

**BEST
AVAILABLE COPY**

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER BBN ~~Report No.~~ 3742	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle) RESEARCH IN NATURAL LANGUAGE UNDERSTANDING Quarterly Technical Progress Report 1 Sep ~~1977~~ 30 November 1977		5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical Progress Report	
6. AUTHOR(s) W. A. Woods and R. J. Brachman		7. CONTRACT OR GRANT NUMBER(s) N00014-77-C-0378	
8. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02138		9. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 7D30	
10. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy Arlington, VA 22217		11. REPORT DATE December 1977	
12. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (1285p.)		13. NUMBER OF PAGES 80	
		14. SECURITY CLASS. (of this report) Unclassified	
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.			
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)			
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Knowledge representation, natural language understanding, parsing, semantics, situation recognition, structured inheritance networks, taxonomic structures.			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report is the first quarterly progress report of the ARPA-sponsored Natural Language Understanding project at BBN. The goals of this project are to develop techniques required for fluent and effective communication between a decision maker and an intelligent computerized display system in the context of complex decision tasks such as military command and control. This problem is approached as a natural language understanding problem, since most of the techniques required			

FEB 9 1978

F

cont'd.

060100

AB

20. Abstract (cont'd.)

→ would still be necessary for an artificial language designed specifically for the task. Characteristics that are considered important for such communication are the ability for the user to omit detail that can be inferred by the system and to express requests in a form that "comes naturally" without extensive forethought or problem solving. These characteristics lead to the necessity for a language structure that mirrors the user's conceptual model of the task and the equivalents of anaphoric reference, ellipsis, and context-dependent interpretation of requests. These in turn lead to requirements for handling large data bases of general world knowledge to support the necessary inferences. The project is seeking to develop techniques for representing and using real world knowledge in this context, and for combining it efficiently with syntactic and semantic knowledge. This report gives an overview of the approach to these problems and an initial specification of the knowledge representation system being used.

A

ACCESSION for	
NTIS	White Section ☒
DDC	Buff Section ☐
UNANNOUNCED	
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist.	SPECIAL
A	

RESEARCH IN NATURAL LANGUAGE UNDERSTANDING

Quarterly Technical Progress Report No. 1

1 September 1977 to 30 November 1977

ARPA Order No. 3414

Contract No. N00014-77-C-0378

Program Code No. 8D30

Contract Expiration Date:
31 August 1978

Name of Contractor:
Bolt Beranek and Newman Inc.

Short Title of Work:
Natural Language Understanding

Effective Date of Contract:
1 September 1977

Principal Investigator:
Dr. William A. Woods
(617) 491-1850 x361

Amount of Contract:
\$301,377

Scientific Officer:
Gordon D. Goldstein

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 3414

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-77-C-0378.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U.S. Government.

TABLE OF CONTENTS

Introduction	1
1. Knowledge-Based Natural Language Understanding	4
1.1 The Role of a Knowledge Network for an Intelligent Machine	4
1.2 Parsing Situations	8
1.3 The Process of Situation Recognition	9
1.3.1 Factored Knowledge Structures	12
1.3.2 Markable Classification Structures	18
1.4 The Structure of Concepts	21
1.5 The Need for Inheritance Structures	23
1.6 The Taxonomic Lattice	26
1.7 An Example	29
1.8 Conclusions	34
2. Structured Inheritance Networks	36
2.1 A Note on Notation	38
2.2 Node Types	43
2.2.1 Concept and Concept-node Types	43
2.2.2 Role- and SD-nodes	45
2.3 Eplink Types	45
2.3.1 Intra-concept-links	48
2.3.2 Intra-role-links	50
2.3.3 Intra-sd-links	54
2.3.4 Abstraction Hierarchies and Inter-concept-links	56
2.3.5 Inter-role-links	61
2.3.6 Inter-sd-links	63
2.4 SI-Net-1 Primitive Concepts	65
2.5 Meta-description and Procedural Attachment	68
2.6 The Individuation Process	73
2.7 Further Refinements	77
3. References	79

Introduction

The ARPA natural language understanding project at BBN is an effort to discover and develop techniques for dealing with large bodies of information in a command and control decision-making situation. Natural language understanding in the context of a knowledge-based intelligent display system constitutes a testbed for the research. Although this is an important application in its own right, the problems addressed in the research are general ones that cut across many other areas of application of rule-based and knowledge-based systems. The research project is primarily a design study to find and formulate fundamental techniques for efficiently solving a set of problems.

The specific problems that we are investigating are those that we consider the most critical for a fluent communication interface to an information display system for command and control decision-making. Although we approach this problem from the point of view of natural language understanding, most of the techniques required would still be necessary for fluent and natural communication even for an artificial language designed specifically for the purpose. The characteristics that are most important are the ability for the user to omit detail that can be inferred by the system, and the ability to express the elements of the user's requests in the order in which they occur to him, without extensive forethought and problem-solving by the user in developing the specification of what he wants done. Also important is the

capability of the system to prompt the user for detail that was omitted in his request when it is necessary to complete the specification of the request and is not inferable by the system.

To the largest extent possible, the user should be freed from the burden of remembering awkward and arbitrary syntactic conventions, from such anticipatory obligations as having to initially name any object that he may subsequently want to refer to, and from having to specify more detail in his requests than is necessary and natural. These objectives lead naturally to requirements for the system to handle the equivalent of anaphoric reference, ellipsis, default assumptions, vagueness, context dependent interpretations, and a fair degree of intelligent inferential processing in determining the intent of a user's request. This latter, in turn, results in a need to handle efficiently large data bases of general world knowledge to support such inferences. Moreover, the nature of the information processing required for such inferences, together with the current revolution in the cost of processing elements in integrated circuits, lead to the desirability and likely necessity of parallel processing algorithms and architectures for handling such processing in real time.

Consequently, the current research project involves an integrated attack on a number of related problems, each of which is a significant research problem in its own right. These include syntax and semantics of natural English, the interaction of

syntactic and semantic information during parsing, the use of anaphoric reference and ellipsis, knowledge representation conventions, knowledge base retrieval and inference techniques, and parallel processing algorithms for parsing and situation recognition. Although all of these areas are being pursued to some extent in parallel, various ones will be prominent at different points in the project. During the first quarter, primary emphasis has been given to the basic epistemological primitives of the knowledge representation system.

This first progress report will concentrate on the overall outline of what the project is trying to accomplish, in order to set the context for subsequent reports. The report consists of two sections, the first an overall introduction to the project as a whole, and the second a preliminary specification of the representational primitives of the knowledge representation structure.

1. Knowledge-Based Natural Language Understanding

W. A. Woods

1.1 The Role of a Knowledge Network for an Intelligent Machine

The kinds of intelligent information systems that we would like to be able to construct to aid a decision-maker are very much like intelligent organisms in their own right. Imagine for a moment an intelligent organism trying to get along in the world (find enough food, stay out of trouble, satisfy basic needs, etc.). The most valuable service played by an internal knowledge base for such an organism is to repeatedly answer questions like "what's going on out there?", "can it harm me?", "how can I avoid it/ placate it?", "Is it good to eat?", "Is there any special thing I should do about it?", etc. To support this kind of activity, a substantial part of the knowledge base must be organized as a recognition device for classifying and identifying situations in the world. The major purpose of this situation recognition is to locate internal procedures which are applicable (appropriate, permitted, mandatory, etc.) to the current situation.

In constructing an intelligent computer assistant for a decision maker, the roles of knowledge are very similar. The basic goals of food getting and danger avoidance are replaced by goals of doing what the user wants and avoiding things that the machine has been instructed to avoid. However, the fundamental problem of analyzing a situation to determine whether it is one for which there

are procedures to be executed, or one which was to be avoided, (or one which might lead to one that is to be avoided), etc. is basically the same. For example, one might want to instruct such a system to remind the user in advance of any upcoming scheduled meetings, to inform him if he tries to assign a resource that has already been committed, to always print out messages in reverse chronological order (when requested), to assume that "The First" refers to The First Infantry Division when the speaker can be inferred to be in the appropriate context, etc.

The principal role of the knowledge network for such a system is essentially to serve as a "coat rack" upon which to hang various pieces of advice for the system to execute. Thus the notion of procedural attachment becomes not just an efficiency technique, but the main purpose for the existence of the network. This does not necessarily imply, however, that the procedures involved consist of low-level machine code. They may instead, and probably usually will, be high level specifications of things to be done or goals to be achieved. The principal structure that organizes all of these procedures is a conceptual taxonomy of situations about which the machine knows something.

To support the above uses of knowledge, an important characteristic required of an efficient knowledge representation seems to be a mechanism of inheritance that will permit information to be stored in its most general form and yet still be triggered by any more specific situation or instance to which it applies.

Moreover, the nodes in the network (or at least a major class of nodes) should be interpretable as situation descriptions. One of the most fundamental kinds of information to be stored in the knowledge base will be rules of the form "if <situation description> is satisfied then do <action description>", or "if <situation description> then expect <situation description>". Situation descriptions are in general characterizations of classes of situations that the machine could be in. They are not complete descriptions of world states, but only partial descriptions that apply to classes of world states. (The machine should never be assumed or required to have a complete description of a world state if it is to deal with the real world.) A situation in this partial sense is defined by the results of certain measurements, computations, or recognition procedures applied to the system's input. Examples of situations might be "You have a goal to achieve which is an example of situation Y", "You are perceiving an object of class Z", "The user has asked you to perform a task of type W", etc.

More specific situations might be: "trying to schedule a meeting for three people, two of which have busy schedules", "about to print a message from a user to himself", "about to refer to a date in a recent previous year in a context where precision but conciseness is required".

The major references to this conceptual taxonomy by the intelligent machine will be attempts to identify and activate those

situation descriptions that apply to its current situation or some hypothesized situation in order to consider any advice that may be stored there. Note that "considering advice of type X" is itself an example of a situation, so that this process can easily become recursive and potentially unmanageable without appropriate care.

Conceptually, one might think of the process of activating all of the descriptions that are satisfied by the current situation as one of taking a description of the current situation and matching it against descriptions stored in the system. However, there are in general many different ways in which the current situation might be described, and it is not clear how one should construct such a description.

Moreover, until it is so recognized, a situation consists of a collection of unrelated events and conditions. The process of recognizing the elements currently being perceived as an instance of a situation about which some information is known consists of discovering that those elements can be interpreted as filling roles in a situation description known to the system. In fact, the process of creating a description of the current situation is very much like the process of parsing a sentence, and inherently uses the knowledge structure of the system like a parser uses a grammar in order to construct the appropriate description. Consequently, by the time a description of the situation has been constructed, it has already been effectively matched against the descriptions in the knowledge base.

1.2 Parsing Situations

As suggested above, the process of recognizing that a current situation is an instance of an internal situation description is similar to the process of parsing a sentence, although considerably more difficult due to a more open ended set of possible relationships among the "constituents" of a situation. That is, where the principal relationship between constituents in sentences is merely adjacency in the input string, the relationships among constituents of a situation may be arbitrary (e.g. events preceding one another in time, people, places, or physical objects in various spatial relationships with each other, objects in physical or legal possession of people, people in relationships of authority to other people, etc.) However, the basic characteristic of parsers, that the objects recognized are characterized as structured objects assembled out of recognizable parts according to known rules of assembly, is shared by this task of situation recognition.

Note that it is not sufficient merely to characterize a situation as a member of one of a finite number of known classes. That is, where it is not sufficient for a parser to simply say that its input is an example of a declarative sentence (one wants to be able to ask what the subject is, what the verb is, whether the sentence has past, present or future tense, etc.), in a similar way it is insufficient to merely say that an input situation is an example of someone doing something. One must generate a detailed description of who is doing what to whom, etc.

It is also not sufficient to characterize a situation as a single instance of an existing concept with values filled in for empty slots. In general, a situation description must be a composite structured object, various subparts of which will be instances of other concepts assembled together in ways that are formally permitted, in much the same way that the description of a sentence is put together from instances of noun phrases, clauses, and prepositional phrases. The specific instance built up must keep track of which constituents of the specific situation fill which roles of the concepts being recognized. Moreover, it cannot do so by simply filling in the slots of those general concepts, since a general concept may have multiple instantiations in many situations. Rather, new structures representing instances of those concepts must be constructed and pairings of constituent roles from the concept and role fillers from the current situation must be associated with each new instance.

1.3 The Process of Situation Recognition

The process of situation recognition consists of detecting that a set of participants of certain kinds stand in some specified relationship to each other. In general, when some set of participants is present at the sensory interface of the system (immediate input plus past memory), the task of determining whether there is some situation description in memory that will account for the relationships of those inputs is not trivial. If the total

number of situation descriptions in the system is sufficiently small, all of them can be individually tested against the input to see if any are satisfied. If the number of such descriptions is sufficiently large, however, this is not feasible.

Alternatively, if there is some particular participant that by virtue of its type strongly suggests what situation descriptions it might participate in, then an index from this participant might select a more manageable set of situation descriptions to test. Even in this case, however, the number of situations in which the constituent could participate may still be too large to test efficiently. In the most difficult situation, no single participant in the input is sufficiently suggestive by itself to constrain the set of possible patterns to a reasonable number. However, it may still be that the coincidence of several constituents and relationships may suffice, providing that the coincidence can be detected. It is this problem of coincidence detection that I believe to be crucial to solving the general situation recognition problem in complex applications with large data bases.

As an example, consider the following fragment of a protocol of a commander giving commands to an intelligent display system:

Cdr: Show me a display of the eastern Mediterranean.
[computer produces display]
Cdr: Focus in more on Israel and Jordan.
[computer does so]

Cdr: Not that much; I want to be able to see Port Said and the Island of Cyprus.

In the first clause of the third command of this discourse, (i.e. "not that much"), there is no single word that is strongly suggestive of the interpretation of the sentence. Moreover, there is nothing explicit to suggest the relationship of this clause to the one that follows the semicolon. The latter, if interpreted in isolation, would merely be a request for a display, or perhaps a succession of two displays, while in the context given, it is a request to modify a previous display.

If the system were given an explicit clue such as "change that display so that I can see Port Said...", then the problem of interpreting the meaning of this sentence would be greatly simplified. However, imposing constraints on a user that force him to be explicit in this way can result in a system that is unusable because of the inability of a user to discover how to express what he wants done. For a flexible and habitable system, one needs to be able to handle forms of expression, such as the one originally given, in which certain details of interpretation are not explicitly indicated but must be inferred.

There are two methods that I believe may be sufficient, either individually or in combination, to handle such cases. One is the use of factored knowledge structures that merge common arts of alternative hypotheses. The other involves the use of a markable

classification structure in which the individual recognition predicates triggered by the ongoing sequence of commands and displays will leave traces of their having fired, so that coincidences of such traces can be efficiently detected. I have been investigating a structure which I call a "taxonomic lattice", that combines some features of both methods.

1.3.1 Factored Knowledge Structures

Given a knowledge-based system with large numbers of situation-action rules, where it is infeasible to find the rules that match a given situation by systematically considering each rule, one needs to have some way of reducing the computational load. As mentioned before, one approach is to index the rules according to some salient feature that will be easily detectable in the input situation and can then be used to find a much more limited set of rules to apply. This has been done in many systems, including the LUNAR system for natural language question answering [Woods, 1973, 1977]. In that system, rules for interpreting the meanings of sentences were indexed according to the verb of the sentence and rules for interpreting noun phrases were indexed by the head noun. This approach certainly reduces the number of rules that need to be considered as compared with an exhaustive consideration of each rule. However, as we pointed out earlier, it has several limitations still. The first is that there may be some values of the index key for which there are still a large number of rules to

consider. In the case of the LUNAR system, for example, the verb "be" had a large number of rules to account for different senses of the word. Another is that there can be certain constructions for which there is no single easily detected feature that is strongly constraining as to possible meaning. In this case, there is no useful index key that can be used to select a sufficiently constrained set of rules to try. For simple question answering systems, one can design the language to exclude such cases, but for general fluency of expression, this is not possible.

Another limitation of this indexing approach as the range of language becomes more fluent is that in certain elliptical sentences, the constraining key may be ellipsed, and although one can have the rules indexed by other keys as well, the remaining ones may not sufficiently constrain the set of rules that need to be considered. Finally, even when the set of rules has been constrained to a relatively small set, there is frequently a good deal of sharing of common tests among different rules, and considering each rule independently results in repeating these tests separately for each rule.

One approach to solving all of the above problems is to use what I have been calling a "factored knowledge structure" for the recognition process. In such a structure, the common parts of different rules are merged so that the process of testing them is done only once. With such structures, one can effectively test all of the rules in a very large set, and do so efficiently, but never

consider any single rule individually. At each point in a factored knowledge structure, a test is made and some information gained about the input. The result of this test determines the next test to be made. As each test is made and additional information accumulated, the set of possible rules that could be satisfied by the input, given the values of the tests so far made, is gradually narrowed until eventually only rules that actually match the input remain. Until the end of this decision structure is reached, however, none of these rules is actually considered explicitly. This principle of factoring together common parts of different patterns to facilitate shared processing is the basic technique that makes ATN grammars [Woods, 1970] more efficient in some sense than ordinary phrase structure grammars. It is also the principle involved in simple decision trees, although the applications that we have in mind will require much more complex structures.

Although the concept has not previously been named and focused on as a generalized technique, this basic idea of sharing common parts of some structure as an efficiency technique has been used in a number of systems. It is the basic technique used by the lexical retrieval component of the BBN speech understanding system [Woods et al., 1976; Wolf and Woods, 1977], and it accounts for the efficiency of the finite state grammar approach of the CMU Harpy system [Lowerre, 1976]. A recent innovative use of this principle appears in Rieger's "trigger trees" for organizing spontaneous computations [C. Rieger, 1977].

A simple example of the kind of merged structure that we are trying to achieve can be illustrated with the problem of accessing an English word from a dictionary. If we want to store information associated with a word in a dictionary, we could represent the dictionary as a list of records, each of which contained the information associated with a particular word, including the spelling of the word itself. If we want to find the information associated with a given word, we could compare that word with each record in the list to see if there was a record for that word (and if so to retrieve it). However, this would be relatively inefficient since on the average half of the dictionary would have to be searched if the word was in the dictionary and the entire dictionary would have to be searched if it was not. (Ordering the dictionary in alphabetical order and using binary search strategies can make this process more efficient, but such techniques cannot be used to search a space where there is no such ordering.) If, however, we construct a tree structure which has an initial branch for each possible letter that can start a word, and each such branch is followed by branches for each of the letters which can follow it, and so on, then we can look up any word in this tree with no more steps than the number of letters in the word, independent of the size of the vocabulary. Such a structure is illustrated schematically below:

-A -I -R
-N -T -I
-R -M -Y
-B -A -L -L -I -S -T -I -C
-R -R -A -C -K
-G -E
-C -A -N -N -O -N
-O -R -P -O -R -A -L

(where the tree is represented with its root at the left, and alternative branches from a node are aligned vertically and starting on the same line as the node from which they branch.) In this structure, the common initial portions of all words are merged into a single path through the tree, and the process of looking for a match of a given word in this tree is effectively working simultaneously on whole classes of words that start with the same sequence of letters.

Words have a very simple structure. The structures that underlie both the commands to a command and control system and the facts and rules that need to be combined with those commands to carry out the intended effects are much more complex. For these, the simple accessing process outlined above is no longer adequate. However, the principle of sharing common parts of many rules still remains. If the system is sufficiently advanced to have many thousands of rules to be matched against the current situation, then one cannot afford to test each rule in succession to determine which

ones are satisfied. Instead one needs some analog of the dictionary tree structure that we described above, in which common parts of different rules are merged, enabling efficient retrieval processes to operate on this structure to find the rules that are satisfied.

The problem of accessing rules is more difficult than the dictionary accessing problem for a number of reasons. One problem is that rules don't match the input letter-for-letter like the words do -- rules have variables in them with various restrictions on what they can match. For example a rule might say that whenever an access is made to a classified file, then a record of the person making the request should be made. The description, "an access to a classified file" needs to be matched against the user's request (or some subpart of it) and in that match, the description "a classified file" will be matched against some specific file name. In this kind of situation, there is no natural ordering of the rules, analogous to the alphabetical ordering of words, that will help in finding the rules that are satisfied by the given situation. Nor is a structure as simple as the dictionary tree above adequate for this case.

Another reason that matching rules is more difficult than matching words is that a given situation may be matched by several rules simultaneously with differing degrees of generality. For example, there may be a rule that says "whenever access is made to a top secret file (more specific than classified), then check the need-to-know status of the user for that information and block access if not satisfied". In the case of a request to a top secret

file, both of the above rules must be found, while in the case of an ordinary classified file, only the first should. The actual input, however, will not explicitly mention either "top-secret" or "classified", but will merely be some file name that has many attributes and properties, among which the attribute "classified" is not particularly salient.

1.3.2 Markable Classification Structures

Another technique that holds promise for efficiently finding rules that are not indexed by a salient key is the use of a markable classification structure in which coincidences of relatively non-salient events can be detected. The keystone of this approach is a technique that Quillian proposed for modeling certain aspects of human associative memory [Quillian, 1966, 1968]. Quillian's technique of "semantic intersection" consisted of propagating traces of "activation" through a semantic network structure so that connection paths relating arbitrary concepts could be detected. For example, his system was able to connect concepts such as "plant" and "nourishment" by discovering the "chain" equivalent to "plants draw nourishment from the soil". If the appropriate information were in the network, this technique would also find chains of indirect connections such as "Plants can be food for people" and "People draw nourishment from food." The method was capable of finding paths of arbitrary length.

The problem of finding connections between concepts in a knowledge network is like the problem of finding a path through a maze from a source node to some goal node. At the lowest level, it requires a trial and error search in a space that can be large and potentially combinatoric. That is, if one element of the input could be connected to k different concepts, each of which would in turn be connected to k others, and so on, until finally a concept that connected to the goal was discovered, then the space in which one would have to search to find a path of length n would contain k^n paths. However, if one started from both ends (assuming a branching factor of k also in the reverse direction), one could find all the paths of length $n/2$ from either end in only $2 \cdot k^{n/2}$.

If one then had an efficient way to determine whether any of the paths from the source node connected with any of the paths from the goal node, such a search from both ends would have a considerable savings. This can be done quite efficiently if the algorithm is capable of putting marks in the structure of the maze itself (or some structure isomorphic to it), so that it can tell when reaching a given node whether a path from the source or the goal has already reached that node. However, without such ability to mark the nodes of the maze, the process of testing whether a given path from the source can hook up with a path from the goal would involve a search through all the paths from the goal individually, and a search down each such path to see if the node at the end of the source path occurred anywhere on that path. If this

were necessary, then all of the advantage of searching from both ends would be lost. The use of the graph structure itself to hold marks is thus critical to gaining advantage from this algorithm. Essentially, the nodes of the graph serve as rendezvous points where paths that are compatible can meet each other. The coincidence of a path from the source meeting a path from the goal at some node guarantees the discovery of a complete path without any path requiring more than a simple test at the corresponding node in the graph as each link is added to the path.

The Quillian semantic intersection technique, in its simplest form, is essentially identical to this bi-directional maze searching algorithm, where the maze being searched is the set of connections between nodes in a semantic network, and the source and goal nodes are the two concepts between which a connection is desired. We are interested in a generalization of the technique in which the source and goal nodes are replaced by a potentially large number of concept nodes, some of which are stimulated by immediate input, and some of which are remembering recent activation in the past. Moreover, we are looking not just for simple paths between two such nodes, but for the confluence of marks from multiple sources in predetermined patterns. Moreover, unlike Quillian, who considered all connections identically in searching for paths, we will consider marker passing strategies in which marks can be passed selectively along certain links. Recently, Fahlman [1977] has presented some interesting formal machine specifications of Quillian-type spreading activation processes which have this characteristic.

1.4 The Structure of Concepts

In building up internal descriptions of situations, one needs to make use of concepts of objects, substances, times, places, events, conditions, predicates, functions, individuals, etc. Each such internal concept will itself have a structure and can be represented as a configuration of attributes or parts, satisfying certain restrictions and standing in specified relationships to each other. Brachman [1977] has developed a set of epistemologically explicit conventions for representing such concepts in a "Structured Inheritance Network", in which interrelationships of various parts of concepts to each other and to more general and more specific concepts are explicitly represented. The essential characteristic of these networks is their ability to represent descriptions of structured objects of various degrees of generality with explicit representation of the inheritance relationships between corresponding constituents of those structures. A concept node in Brachman's formulation consists of a set of dattrs (a generalization of the notions of attribute, part, constituent, feature, etc.) and a set of structural relationships among them. Some of these dattrs are represented directly at a given node, and others are inherited indirectly from other nodes in the network to which they are related. More details of such representations are given in Chapter 2 of this report.

Let us assume that each concept that the system understands is represented as a node in one of these structured inheritance

networks. The network, as a whole, then serves as a conceptual taxonomy of all possible "entities" that the system can perceive or understand. Each node in this taxonomy can be thought of as a micro schema for the recognition of instances of that concept. Each has a set of dattr's with individual restrictions and a set of structural conditions that relate the dattr's to one another. These restrictions and structural conditions may themselves be defined in terms of other concepts defined by other micro schemata, and so on until a level of primitively defined, directly perceivable concepts is reached.

Each concept in the taxonomy can be thought of as having a level of abstractness defined as the maximum depth of nesting of its constituent structure. Instances of primitively defined concepts have level 0, constellations of those concepts have level 1, a concept having level 1 and lower concepts as dattr's has level 2, and so on. If a taxonomy contained only level 0 and level 1 concepts, then the situation recognition problem would be greatly simplified, since one never needs to recognize portions of the input as entities that participate as constituents of larger entities. The general problem, however, requires us to do exactly that. More seriously, the general case requires us to recognize a concept some of whose dattr's may have restrictions defined in terms of the concept itself. This is true, for example, for the concept of noun phrase in a taxonomy of syntactic constructions. Such recursively defined concepts have no maximum level of abstractness, although any

given instance will only involve a finite number of levels of recursion. This potential for recursive definition must be kept in mind when formulating algorithms for situation recognition.

1.5 The Need for Inheritance Structures

If the concepts to be recognized as situation descriptions were all sufficiently simple as to consist of single bundles of directly perceivable input constituents (i.e. level 1 concepts in the terms of the preceding section), and if only a single such description were to be matched against any given input, then the situation recognition process would be relatively straightforward. However, in general, several situation descriptions will be simultaneously satisfied by an input situation, no one of which will account for all of the input nor supplant the relevance of the others. For example, adding a ship to a display is simultaneously an example of changing a display and of displaying a ship. Advice for both activities must be considered. Moreover, a single description may have several different instantiations in the current situation. For example, there may be several instances of displaying a ship in a single display.

To further complicate matters, situation descriptions may subsume other descriptions at lower levels of detail, and advice from both may be relevant and may either supplement or contradict each other. For example, displaying an aircraft carrier is a special case of displaying a ship, and there may be specific advice

associated with displaying carriers as well as more general advice for displaying any ship. Neither description can replace the other. Moreover, conventions are required to determine which advice takes precedence over the other if conflicts arise.

Finally, situation descriptions can become arbitrarily complex by the addition of various qualifiers, by the conjunction and disjunction of descriptions, etc. For example, one might want to store advice associated with the situation [wanting to display a large ship at a location on the screen that is within one unit distance from either the top, bottom, or side of the screen when the scale of the display is greater than 1:1000].

The organization of large numbers of such situation descriptions of varying degrees of generality so that all descriptions more general or more specific than a given one can efficiently be found is one of the requirements for a system of the sort we envisage. In order to build and maintain such a structure, it is important to store each rule at the appropriate level of generality, relying on a mechanism whereby more specific situations automatically inherit information from more general ones. That is, when one wants to create a situation description that is more specific than a given one in some dimension, one does not want to have to copy all of the attributes of the general situation, but only those that are changed. Aside from conserving memory storage, avoiding such copying also facilitates updating and maintaining the consistency of the data base by avoiding the creation of duplicate

copies of information that then may need to be independently modified and could accidentally be modified inconsistently.

For example, one may want to store advice about displaying geographical features, about displaying such features that cover an area, about displaying bodies of water, about displaying lakes, etc. Thus, information about finding the area covered by a feature would be stored at the level of dealing with such area-covering features, information about displaying water in a certain color would be stored at the level of displaying bodies of water, and information about having inlets and outlets would be stored at the level of lakes. In any specific situation that the system finds itself, many such concepts at different levels of generality will be satisfied, and the advice associated with all of them becomes applicable. That is, any more specific concept, including that of the current situation, inherits a great deal of information that is explicitly stored at higher levels in the taxonomy.

In the case of the situation descriptions that we are dealing with, even the specification of what dattrs a given concept possesses is stored at the most general level and inherited by more specific concepts. Thus, for example, the descriptions of attribute dattrs for color and weight are stored for a general concept of physical object. These dattrs are then inherited by any more specific concepts of physical objects, such as planes, ships, desks, and pencils.

1.6 The Taxonomic Lattice

As mentioned above, I believe that a general solution to the situation recognition problem can be obtained by the use of a classification structure in which traces of individual elements of complex concepts can intersect to facilitate the discovery of coincidences and connections that may not be strongly inferable from constraining expectations. The structure that I propose to use is a version of Brachman's structured inheritance networks, in which descriptions of all potentially relevant situations are stored with explicit indications of general subsumption of one situation by another, and explicit indications of the inheritance of datums and of advice by one concept from another. This structure, which I have called a taxonomic lattice, is characterized by a multitude of situation descriptions at different levels of generality.

We say that a situation description S1 subsumes a description S2 if any situation satisfying S2 will also satisfy S1. In this case, S1 is a more general description than S2, and is placed higher in the taxonomy. For example, [displaying a portion of country] is a more specific situation than [displaying a geographical area], which is in turn more specific than [displaying a displayable entity]. All of these are subsumed by a general concept [purposive activity], which in turn is more specific than [activity]. Moreover, a given description can subsume many incomparable descriptions and can itself be subsumed by many incomparable descriptions. For example, an instance of [displaying a

geographical area] is also an instance of [accessing a geographical area], [displaying information], and [using the display], and may possibly also be an instance of [responding to a user command].

The space of possible situation descriptions forms a lattice under the relation of subsumption. At the top of the lattice is a single, most general situation we will call T, which is always satisfied and can be thought of as the disjunction of all possible situations. Anything that is universally true can be stored here. Conversely, at the bottom of the lattice is a situation that is never satisfied, which we call NIL. It can be thought of as the conjunction of all possible (including inconsistent) situations. Assertions of negative existence can be stored here.

At the "middle" level of the lattice are a set of primitive perceptible predicates -- descriptions whose truth in the world are directly measurable by the "sense organs" of the system. All classes above this level are constructed by some form of generalization operation, and all classes below are formed by some form of specialization. At some point sufficiently low in the lattice, one can begin to form inconsistent descriptions by the conjunction of incompatible concepts, the imposition of impossible restrictions, etc. There is nothing to prevent such concepts from being formed; indeed, it is necessary in order for the organism to contemplate, store, and remember their inconsistency.

There are a number of specific relationships that can cause one situation description to subsume another. A given situation description can be made more general by relaxing a condition on a dattr, by eliminating the requirement for a dattr, by relaxing the constraints of its structural description, or by explicitly disjoining it (or'ing it) with another description. A given description can be made more specific by tightening the conditions on a dattr, by adding a dattr, by tightening the constraints of its structural description, or by explicitly conjoining (and'ing) it with another description. These operations applied to any finite set of situation descriptions induce a lattice structure of possible situation descriptions that can be formed by combinations of the elements of the initial set. We refer to this structure as the virtual lattice induced by a given set of situation descriptions, but do not expect this lattice to be stored. Instead, we expect the machine to store only a finite portion of this lattice with explicit connections from more specific to more general concepts. By processing this explicit lattice, we will be able to test any given description for membership in the virtual lattice and assimilate any new situation description into the explicit lattice in the appropriate place corresponding to its position in the virtual lattice.

In operation, any situation description about which information is explicitly stored will be entered into the explicit lattice. Any situation that the machine can understand is in some sense already

in the virtual lattice and needs only be "looked up" in it. Our task will be to develop efficient algorithms to tell whether a given situation can be understood in terms of the concepts of the lattice and if so, to construct its corresponding description and explicitly record its relations to other concepts in the explicit lattice. When this has been done, all of the necessary inheritance paths will have been established linking that description to all of the relevant stored advice at all levels of the network. Descriptions of all potentially relevant situations are placed in this lattice with associated rules to be executed when such situations are encountered. As the system carries out its assigned tasks, descriptions of its current situation are simultaneously being constructed and compared with this lattice.

1.7 An Example

As an example of the situation recognition process using marker propagation in a taxonomic lattice, let us consider a simple case of interpreting the intent of a simple English sentence. The example chosen is not complex enough to require all of the machinery discussed, but is presented here to illustrate the mechanism. The major features of the situation recognition mechanism only become critical in interpreting commands that require several sentences to build up, or which depend on the current context in complex ways, but such situations are difficult to illustrate.

For our example, suppose that the system contained a concept for requests to display a geographical region, and the user's input request were "Show me the eastern end of the Mediterranean." The concept [request] contains dattrs for the requestor, the requestee, a description of the state that the requestor desires, a form of request (demand, order, polite request, expression of preference, etc.), and perhaps others. Requests can take many forms. Assume that we have stored in the system a rule that says "Any sentence of the form: 'show me NP' is a request to display that NP." This rule could be stored in the lattice as a piece of advice associated with the concept "A sentence of the form: 'show me NP'," in such a way that when a sentence of the indicated form was found, an instance of a display request would be created. At that point, this resulting display request would be placed in the lattice in such a way that all more general concepts of which it is an instance would be activated, and in particular, the concept of a request to display a geographical region would be activated.

The parsing of the original sentence can either be done by an ATN grammar, or by a version of the taxonomic lattice itself (one that characterizes a taxonomy of sentence types). Let us assume here that it is done by an ATN grammar that is closely coupled to a taxonomic lattice, with the ATN representing the syntactic information about sentence form and the taxonomic lattice representing general semantic information. As the ATN grammar picks up constituents of the sentence, it reaches states where it makes

hypotheses about the syntactic roles that those constituents play in the sentence (e.g. "this is the subject", "this is the verb", etc.). Such hypotheses are then entered into the lattice, where they begin to activate the recognition conditions of concepts in the network. For example, in the taxonomic lattice there is a concept of an imperative sentence whose subject is the system, whose verb is "show", whose indirect object is the user and whose direct object is a displayable object.

As the parsing proceeds, the ATN will make assertions about the sentence it is building up, and it will not only be building up syntactic representations of constituents of the sentence, but will also be building up representations of possible meanings of those constituents. In particular, it will be building up a list of those concepts in the lattice of which the current constituent may be a restriction or instance and a list of the dattr-value pairings that have been found so far. If a parse path succeeds (i.e. reaches a POP arc), then a node in the taxonomic lattice corresponding to that hypothesis will be found or constructed. This node will have links to more general and more specific concepts, and will have its constituents linked to appropriate dattrs of those concepts. At the point when this concept node is found/constructed, a process of activation spreading will be launched in the lattice to find any advice that may be inherited by that concept. This process will also leave "footprints" in the lattice that will facilitate the detection of concepts of which the current one may itself be a dattr (or part of a structural cognition).

In the example above, when the parser has parsed the initial portion of the sentence "show me", it has built up in its internal registers the information corresponding to the hypothesis that the sentence is an imperative, with subject "you" and indirect object "me". Moreover, it knows that (in input sentences) "you" refers to the system itself, while "me" refers to the speaker. It also knows that the main verb is the verb "show". Let us suppose that at this point, the parser decides to activate the corresponding taxonomic lattice nodes for the concepts [the system], [the user], and [the verb show] (possibly with pointers to the syntactic hypothesis being constructed and/or the labels SUBJECT, OBJECT, VERB, respectively). Ignoring for now whatever information or advice may be found associated with these concepts or their generalizations, the footprints that they leave in the network will intersect at a node [display request] which has dattrs for requestor, requestee, form of request, and requested thing. They also intersect at other concepts such as [imperative sentence], [active sentence], [action], and a more specific kind of display request [region display request], whose requested thing is a geographical region. This latter concept was created and inserted into the lattice precisely to hold advice about how to display geographical regions, and to serve as a monitor for the occurrence of such situations. Fig. 1 is a fragment of a taxonomic lattice showing the concepts of interest. (For details of the notation, see Chapter 2.)

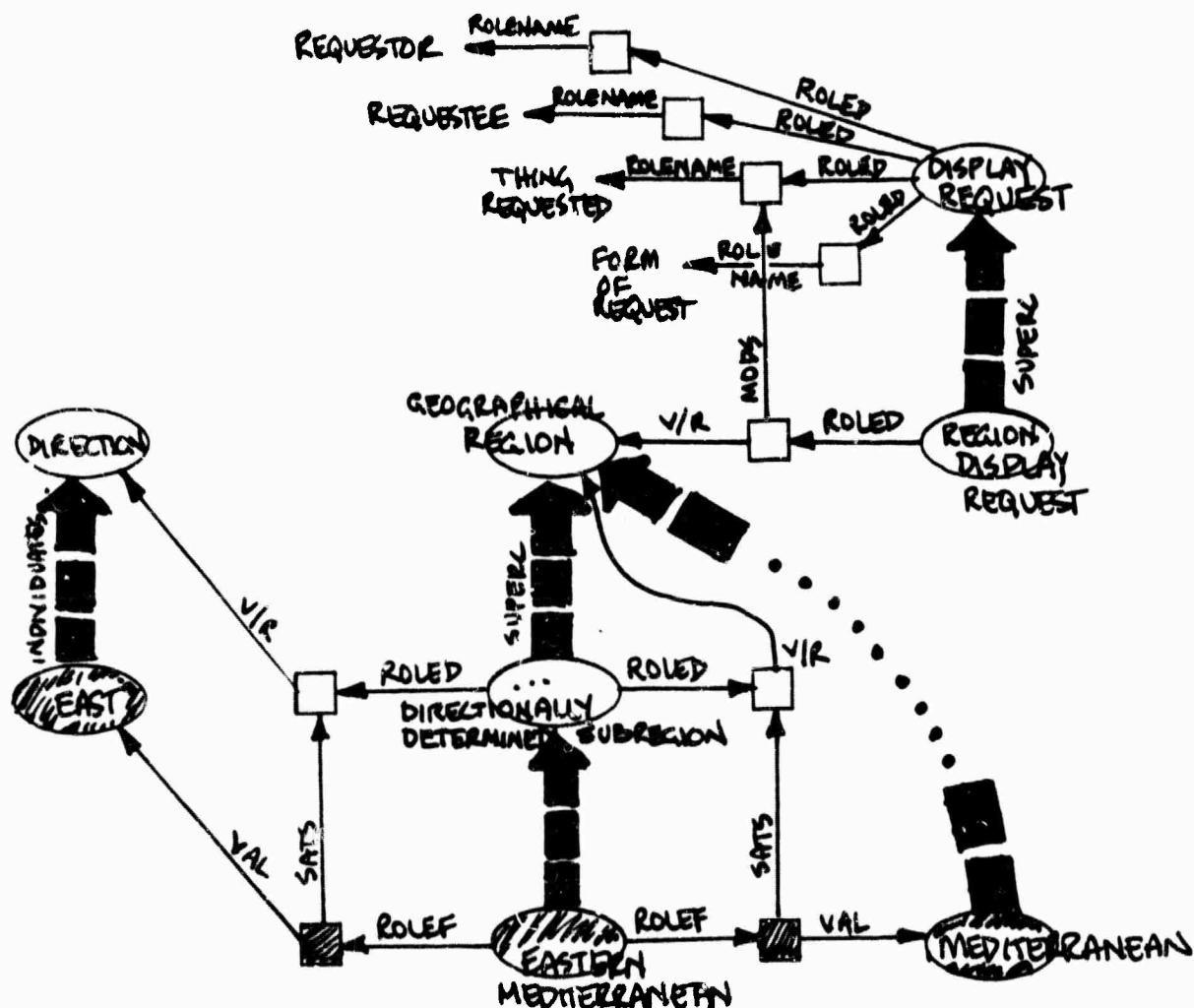


Fig. 1.

When the final noun phrase has been parsed and given an interpretation, the footprints that its activation leaves in the network will awaken the [region display request] node, which will then be fully satisfied, and the parser will create a corresponding instance node, with appropriate bindings for its dattrs. In

processing the noun phrase, the parser will discover the adjective "eastern" and the noun "Mediterranean" and will activate the corresponding nodes in the taxonomic lattice. The concept [east] is an instance of [direction], which, among other things, is the restriction for a dattr of a concept [directionally determined subregion] that defines the meaning of such concepts as "north eastern Idaho". Another dattr of this same concept has the restriction [geographical region], which is on the superc chain from Mediterranean. Hence, footprints from "eastern" and "Mediterranean" will intersect at the concept [directionally determined subregion], causing an instance of that concept to be constructed as a possible meaning of the noun phrase. The [directionally determined subregion] concept itself has a superc connection to [geographical region], which happens to be the restriction for the "requested thing" dattr of the concept [region display request] which has already received marks for its other dattrs. Thus, the intersection of footprints from the various constituents of the sentence at this concept node has served to select this node out of all the other nodes in the network. Since the more general concept [display request] is on a superc chain from [region display request], it will also be activated, and advice from both places will be considered.

1.8 Conclusions

As this example illustrates, the nodes in the taxonomic lattice structure serve as rendezvous points where footprints from various constituent elements of a concept can meet, thus detecting the

coincidence of related events, which in many cases will not be suggestive in isolation. The implementation of the kinds of operations described above involves a system of marker passing conventions for propagating the various "footprints" around the network, detecting coincidences, creating instance nodes, and propagating further markers when coincidences are found. A major portion of the current research project is the discovery of effective conventions for such marker passing operations.

The details of the marker passing algorithms are far from worked out. Issues include conventions for how far markers should propagate (amounting to decisions as to where to rendezvous), how much information a mark carries with it, to what extent marks are inherited, how a node remembers partial intersections of marks in such a way that it can incrementally extend them as additional marks accumulate, implications of the marker passing strategies on requirements for representational conventions, etc. These issues will be pursued in the coming months.

2. Structured Inheritance Networks

Ronald J. Brachman

This part of the current report is a preliminary specification of a knowledge representation scheme based on "Structured Inheritance Networks" [Brachman, 1977]. Many of the details of this structure, including the naming conventions for its various parts, are tentative and will almost certainly change as the ideas are more fully elaborated and put to the test. However, both as a report of progress, and as an explicit context for further elaboration and discussion, a detailed sketch of the system as it has so far evolved will be presented. We will refer to this particular version of the system as "SI-Net-1".

We view worlds that are to be represented in this scheme as composed of structured objects and relationships, as well as abstractions over those objects and relations. Each of these world entities has a corresponding "concept". The basic function of the networks we are concerned with is to describe and represent concepts -- especially concepts with internal structure -- and to represent explicitly the relations of generality and inheritance among such concepts and their parts. Our formal structures for concepts will reflect the structures that we perceive entities in the world to have. We think of them being composed of sets of "dattrs" that satisfy certain constraints and stand in specified relationships to each other. The notion of a dattr (short for defining/derived

attribute) is a generalization of a variety of attributes, characteristics, parts, etc. that a thing might be said to "have". For example, dattrrs will include such things as parts of structured objects (e.g. fingers of a hand), inherent attributes of objects and substances (e.g. color), arguments of functions (e.g. multiplicand and multiplier in a multiplication), and "cases" of verbs in sentences (e.g. "agent"). These dattrrs stand in certain relationships to the things to which they belong. We refer to these relationships as "roles", and think of instances of dattrrs as role fillers. Also associated with a concept is a structuring relationship that relates the individual dattrrs to each other and to the concept as a whole. Thus, a concept can be thought of as a constellation of dattrrs standing in a specified relationship to each other.

As a simple example, the notion of "arch" of AI vision fame, which consists of two vertical bricks supporting a horizontal brick above an open space, is a concept that describes a kind of thing with three (parts) dattrrs, two of which fill a vertical support role, and the third of which fills a lintel role. Also associated with this kind of entity are other (inherent attribute) dattrrs such as "vertical clearance", "overall height", "total mass", etc. The constraint on the role fillers in this case is that they are all bricks, while the structuring relationship specifies that the lintel must be supported by the two verticals and that the two verticals must have a space between them.

There are three basic types of formal SI-Net-1 entity, reflecting the above conception of the world (one important aspect of a representation is that its internal structure corresponds in a reasonable way to the structure of the domain that it represents). The central type is the Concept, which represents the structural unities of objects, relationships, and abstractions. Concepts have two types of parts: Role/Dattr Descriptions (which we will abbreviate to "Roles"), which describe the dattrr of a concept and the roles that they play within it, and Structural Descriptions (SD's), which express the interrelationships between the dattrr. Just as we think of an entity in the world as a constellation of dattrr standing in a specified relationship to one another, our Concept for that entity will have a set of Roles, and a set of SD's to express this relationship.

2.1 A Note on Notation

In this report, we will use a semantic network-like notation to express the structure of Concepts. To represent the formal SI-Net-1 entities (Concepts, Roles and SD's), we use different types of nodes. To indicate relationships between these nodes, we use connectors with explicit epistemological significance, called "epistemologically primitive links", or eplinks. This notation is used for illustrative purposes, and it should be borne in mind that the critical elements in SI-Net-1 are Concepts, Roles, and Structural Descriptions, and not the nodes and links with which we

draw characterizations of these epistemological constructs.* The reason that we use a network notation is that it provides an explicit place for each possible kind of connection between two entities, and forces us to account for every epistemological relationship implied by the Concept-Role-SD paradigm.

In the particular notation that we will use to represent a Concept, the existence of Roles standing in certain epistemological relationships to the Concept is indicated by a special kind of node, called a role-node, which is linked to a node for the concept with which it is associated by one of two kinds of eplink. A role-node contains links to information such as a constraint predicate on possible role fillers, and modality information indicating whether the Role that it represents must (may) be filled in any instance of the Concept, and if so how many times. Also associated with a concept-node is a set of sd-nodes (for "Structural Description") that specify the structural relationships that must hold among the individual dattr's if they are together to constitute an "instance" of the Concept.

All nodes and links in the notation have types. In diagrams of SI-Net-1 structures, the type of a node will be reflected by its shape, and the type of a link will be indicated by a mnemonic label

*This will be more apparent when the SI-Net-1 functional package is fully developed. The functions will manipulate Concepts, Roles, and SD's and will make no mention of nodes and links. While one possibility is to implement the epistemology in a semantic network, this is not meant to imply that there is any logical or necessary requirement to do so.

written next to the link. The set of types of nodes and links is fixed in advance by the language itself, and as a consequence, each node or link type has a fixed local syntax. For a node type, this means that the type of links that can emanate from it are predetermined; for a link type, the types of nodes connected by the link (the source and destination) are defined by the link's type. As one might infer, all eplinks are binary - they connect two, and only two, nodes. In addition, all eplinks can be followed both ways (i.e. from source to destination and vice versa). The arrowhead in a diagram indicates directionality of the relationship, and does not imply a "one-way link".* Table 1 illustrates the complete set of SI-Net-1 link and node types.

*Because certain parts of the conceptual structure are not stored in SI-Net-1 terms, some links will have to point out of the network into uncharted territory. It may be useful to point "back" into the network somehow from these items, in which case all links really would be considered two-way (e.g. PRINT-NAME if implemented as a link). Since no decision has been made on this, we should admit the possibility of certain one-way links. These will be marked specially.

NODE TYPES

Concept:
 generic
 individual
 parametric individual

Role:
 generic
 filled
 coref
 indirect

SD:
 sd

LINK TYPES

**Source node
 type**

**Destination node
 type**

Eplinks**1) Intra-concept**

RoleD
 RoleF
 Structure

generic-concept
 concept
 generic-concept

generic-role
 filled-,coref-role
 sd

2) Intra-role

RoleName
 V/R
 Number
 Modality

generic-role
 generic-role
 generic-role
 generic-role

[atom]
 concept
 [number predicate]
 [modality (OBL,
 INH, OPT; DER)]
 individual-concept
 generic-role,
 generic-,
 paraind-concept

Val
 CorefVal

filled-role
 coref-role

individual-concept
 generic-role,
 generic-,
 paraind-concept
 role
 role

Focus
 SubFocus

indirect-role
 indirect-role

3) Intra-sd

Check
 Derive
 NonActive

sd
 sd
 sd

paraind-concept
 paraind-concept
 paraind-concept

Table 1. SI-Net-1 Node and Link Types.

4) Inter-concept

SuperC	generic-concept	generic-concept
Individuates	individual-concept	generic-concept
ParaIndividuates	paraind-concept	generic-concept
BrotherC	generic-concept	generic-concept
<Nominalizations>	generic-concept	generic-concept

5) Inter-role

Sats	filled-role	generic-role
Mods	generic-role	generic-role
DiffS	generic-role	generic-role
CorefSats	coref-role	generic-role

6) Inter-sd

Preempts	sd	sd
----------	----	----

Hooks

Metahook	concept, role, sd	individual-concept
<Interpretive hooks (by situation)>	concept, role, sd	[procedure in interpreter language]

Table 1, continued.

2.2 Node Types

As shown in the table, SI-Net-1 notation has three basic node types. These are concept-nodes, role-nodes, and structural-description-nodes (sd-nodes). All concept-nodes are named with a unique identifier, either by the user or by the system (if the user supplies two identical names, the system will add a differentiating "gensym"). Other nodes are not named. SI-Net-1 graphical diagrams will depict a concept-node as an oval with the name enclosed. Role-nodes are depicted as small squares, and sd-nodes as diamonds.

2.2.1 Concept and Concept-node Types

There are three types of Concept in SI-Net-1: Generic Concepts, Individual Concepts, and Parametric Individual Concepts. Generic Concepts look like prototypical individuals (i.e. they have one part specification for each part that is to appear in an individual version), and implicitly define sets of individuals. These sets consist of all individuals that satisfy the generic description. The SI-Net-1 notation pictures a generic-concept-node as a simple oval.

Individual Concepts are intensional descriptions of individual entities. They generally do not have explicit SD's, but instead inherit them from more general Concepts that they individuate. The main thing, however, that makes an Individual Concept different from

a Generic Concept is its implicit uniqueness condition. Each Individual Concept is expected to have one and only one referent in the world being modeled*. We use a variant on our concept-node notation to indicate this difference from Generic Concepts -- individual-concept-nodes will be shaded ovals.

A Parametric Individual Concept (ParaIndividual) is much like an Individual Concept, but represents a variable individual. ParaIndividuals appear only in the SD's of Generic Concepts; a ParaIndividual represents a different individual for each instance of the Generic Concept. Some of the ParaIndividual's roles are "filled" not by particular entities, but by yet unspecified fillers of the Roles of the Concept in whose SD it appears. For each instance of the Concept, actual fillers for those roles will be indicated, and an individual version of the ParaIndividual can be fully specified. Thus, we can think of the individual "schema" that the ParaIndividual represents as parameterized by the Concept in which it is used: when the particular instance is given (i.e. the "parameter"), all of the Roles can be filled in. Parametric Individuals will be represented as ovals with two concentric borders.

- - - - -
*An individual concept is thus not merely a fully specified (i.e. all Roles filled) Generic Concept. Generic Concepts with all Roles filled can still describe many individuals, where the level of detail of the description is not sufficient to distinguish between those individuals.

2.2.2 Role- and SD-nodes

In our diagrams, role-nodes will be pictured as small squares. Role-nodes indicating Filled Roles (as opposed to general descriptions of Roles) will be shaded for perspicuity. In addition, there are two special kinds of role-nodes that we will distinguish. Coref-role-nodes will have double borders; these appear only in Parametric Individual Concepts, and will be explained later. Indirect-role-nodes, which are not directly attached to any concept, are indicated with dashed lines. Indirect-role-nodes allow one concept-node to point to a role-node of another (e.g. parts of parts of parts). These represent what we might call "ghost" Roles - Roles to which access is necessary, but which do not appear explicitly as Roles of the Concept from which the access is made.

Finally, sd-nodes indicate Structural Descriptions. There is only one type of sd-node: these will appear as small diamonds.

See Fig. 2 for a complete inventory of SI-Net-1 node types.

2.3 Eplink Types

Eplinks are used to tie together nodes; the epistemological relationships that they represent provide internal structure for Roles, SD's, and ultimately, Concepts. In addition, certain relationships between Roles and SD's of different Concepts, and between Concepts as wholes, are expressed by these links.

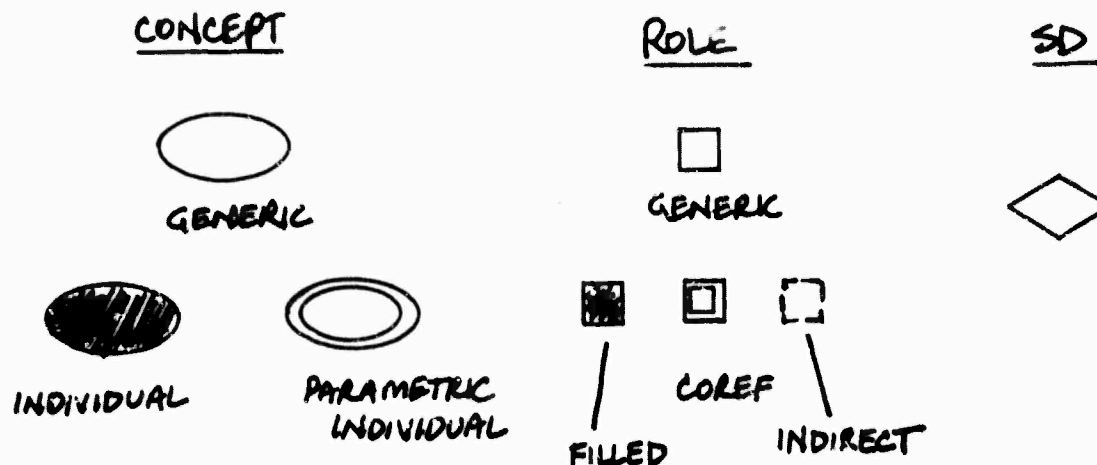
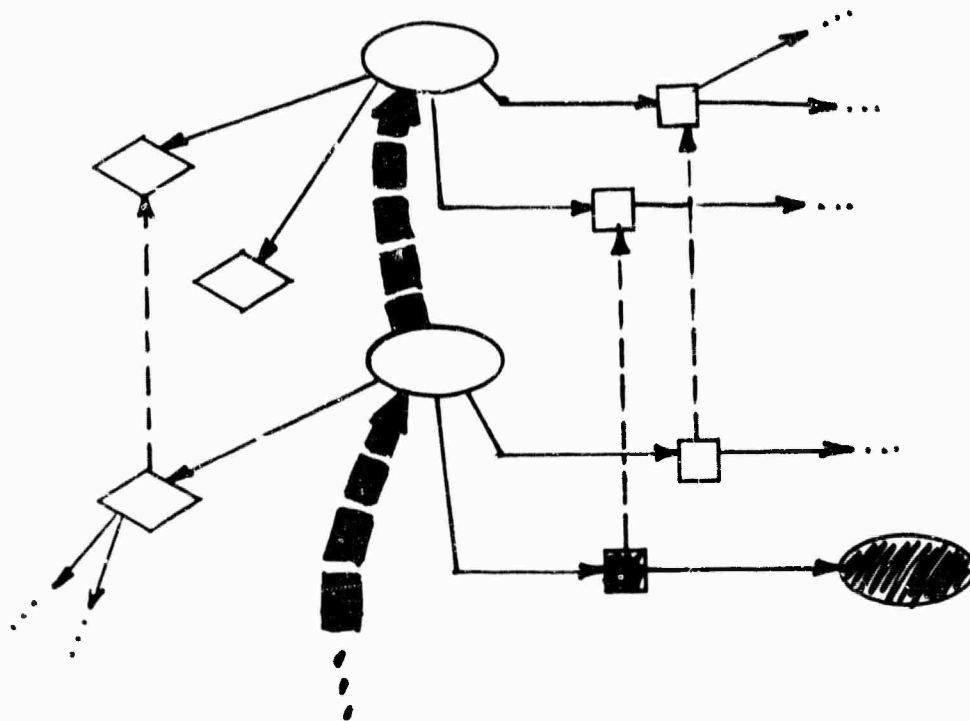


Fig. 2. Summary of SI-Net-1 Node Types.

Fig. 3 gives a general picture of six major varieties of eplink types. Inter-concept-links are used to express "inheritance" paths for Roles and SD's. Similar to the way that classes can be further and further divided into subclasses, Concepts can be further and further restricted into "Subconcepts". In SI-Net-1, aspects of a Concept that are shared by subvarieties of that Concept are represented at the most general Concept possible. Inter-concept-links connect Subconcepts to more general ones, and thus serve to pass the common aspects down to the more restricted description. In general, inter-concept-links are used to indicate that the Roles and SD's available at the "parent" Concept are to be considered available at the offspring Concept.

Intra-concept-links connect a concept-node to the role-nodes and sd-nodes that represent the Concept's Roles and SD's. Thus, they bind the Concept together into a structured definition.



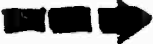
-  INTER-CONCEPT: INHERITANCE PATHS
-  INTRA-CONCEPT: INTERNAL STRUCTURE OF CONCEPTS
-  INTER-ROLE: ROLE CONSTRAINT INHERITANCE; ROLE ACCESS
-  INTRA-ROLE: PIECES OF ROLE CONSTRAINTS, FILLERS
-  INTER-SD: RELATIONS BETWEEN SD'S
-  INTRA-SD: PIECES OF STRUCTURAL DESCRIPTIONS

Fig. 3. General Classification of Eplinks.

Inter-role-links indicate relationships between Role constraints and relationships between constraints and their satisfiers in different Concepts. Intra-role-links structure the constraints themselves (within a single Concept); they are indicators of the internal structure of Roles. Similarly, inter-sd-links indicate relations between SD's, and intra-sd-links tie the internal pieces of an SD together.

In the next six subsections, we introduce the particular set of eplinks that embody the epistemology of the SI-Net-1 entities. First, we discuss the internal structuring links (intra-concept, intra-role, and intra-sd), and then we present those that express structured inheritance by connecting entities of the same type.

2.3.1 Intra-concept-links

These links connect a Concept to the SD's and Roles that constitute its definition. There is only one kind of link to an SD; this is called Structure. It simply indicates that the SD belongs to its corresponding Concept.

There are two types of link that may connect a Concept with one of its roles. RoleD means "Role/Dattr Description", and implies that the role-node pointed to is to be interpreted as an unsatisfied constraint. RoleF indicates a Filled Role, which is interpreted as a specification of the dattr that fills that Role.* When Roles are

*The critical thing to remember about a Role (and a Filled Role, in

inherited by other Concepts, RoleD indicates that the Role constraint is still to be satisfied, while RoleF indicates that the role is filled and the filler is to be inherited, per se.

Fig. 4 illustrates the use of these links. In it, the Concept FLIGHT10 has a generic Role description, D, and has as its only Structural Description, node S. These two indicate that every FLIGHT10 has a PILOT that must be a Person, and that every FLIGHT10 has a PILOT that must be a Person, and that every FLIGHT10

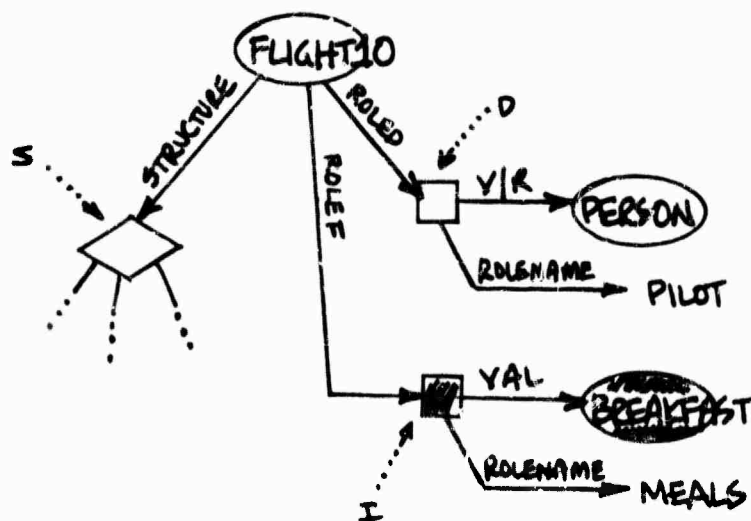


Fig. 4. Concept-Role Connections.

has the structure indicated by S, respectively. Node I, a filled-role-node, represents the fact that every FLIGHT10 has as its MEALS the particular Individual Concept, BREAKFAST.

particular), is that it explicitly accounts for not only the filler (datr), but also for the particular functional role filled, and the particular context (Concept) in which the datr fills that role. The RoleF link ties the context to the role-node, which is the place-holder for this three-way relationship. The other two aspects of the triad are discussed in the next section.

Occasionally, it may be necessary to "turn off" a Role from being inherited. For the moment, we propose to represent this by pointing from a role-node to the special primitive Concept, NOTHING. This is to be interpreted as "NOTHING fills this Role" - that is, a positive assertion that the Role is not filled.

2.3.2 Intra-role-links

Individual Role/Data Descriptions (Roles) specify properties that are expected to be true of the ultimate fillers of the functional roles of the Concept.* In SI-Net-1, there are four facets to such a description, each with its own link type:

- a) RoleName: names the role to be played by a dattr.
- b) V/R (Value Restriction): specifies a Generic Concept, of which any filler must be an instance. This Concept can be thought of as a predicate which must be true of any filler.
- c) Number: indicates the number of fillers of the particular Role to be expected. If no such link exists, assume "1".
- d) Modality: determines whether it is OBLIGATORY to have the Role filled in an instance; whether the Role is always expected to be filled, but is not critical to the definition of the concept (INHERENT); or whether the Role filler is OPTIONAL. In addition, the Modality indicates whether the filler is DERIVABLE from the structure of the Concept itself.

When no Modality facet is present, it is assumed that the Role is not OBLIGATORY and that the importance of the Role is encoded in the SD. For example, Context-sensitive criterialities (e.g. "there must be at least 2 of these 3"...) can be expressed in an SD.

*They also serve as place-holders for information about the roles themselves. The total constellation of all links pointing to a role-node from the Concept's SD's make up the definition of the role.

When a Role is filled, the filler is indicated by a Val link from the appropriate filled-role-node. In the case that such a filler is present, it must satisfy any constraints inherited from the Role descriptions that it satisfies. Those Role descriptions are indicated by a link called "Sats".

A special kind of Role, called a Coref Role, looks like a Filled Role, but serves a distinct function. This kind of Role occurs only in a ParaIndividual Concept, where it schematically specifies the filler of a Role without picking its particular value. This is achieved by a pair of links which are similar to the Val/Sats pair just described for ordinary Filled Roles. A CorefVal link parallels the Val link, but usually points to a role-node, rather than a concept-node.* It equates (intensionally) the filler of one Role with a potential filler of another. When an instance is formed, the filler of the Role pointed to by CorefVal also becomes the filler of the Role pointed from (in the SD). The link that parallels the Sats link is called CorefSats, and indicates which role it is that is to be played by the dattr eventually filled in for the instance. An example should help make all this clear. In Fig. 5, D is a ParaIndividual that is a version of the DISTANCE Concept. As such, it represents the fact that for every individual arch there will be a particular distance, appropriate to that arch,

*The sole exception occurs when the CorefVal link points to the Concept from whose SD it emanates. This indicates that when an instance is created, that instance as a whole will fill the role indicated by the CorefSats link from the Coref Role.

which is the distance between the LINTEL of the arch and the constant, GROUND. FR of D is an ordinary Filled Role that specifies the filler of the role, "TO", as the value, GROUND. CR is a Coref Role whose CorefVal is the role-node for LINTEL. It indicates that the role, "FROM" (which is indicated by the CorefSats link), of the particular distance in any instance of ARCH, is to be filled by the very entity that fills the "LINTEL" role of that arch instance.

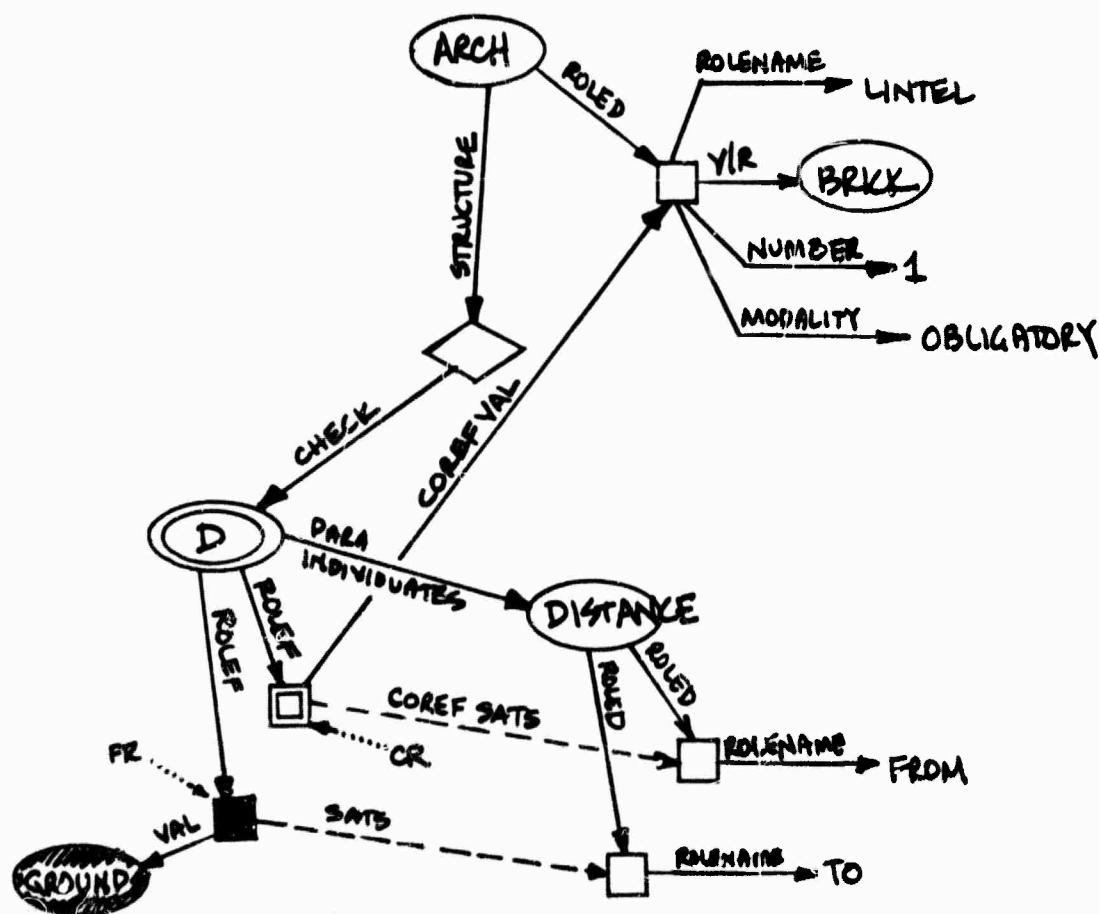


Fig. 5. Role Structure.

In the event that the V/R of a Role is itself structured (as will almost always be the case), we might want to reference one of its subparts, rather than reference it as a whole. To accomplish this, we use a Focus/SubFocus path. Such composite reference paths are constructed by creating indirect role-nodes not part of any concept, and pointing with Focus and SubFocus links to the "context" and desired Role, respectively. For example, in Fig. 6, node A is an indirect-role-node that stands for the NORMAL to the TOP of a

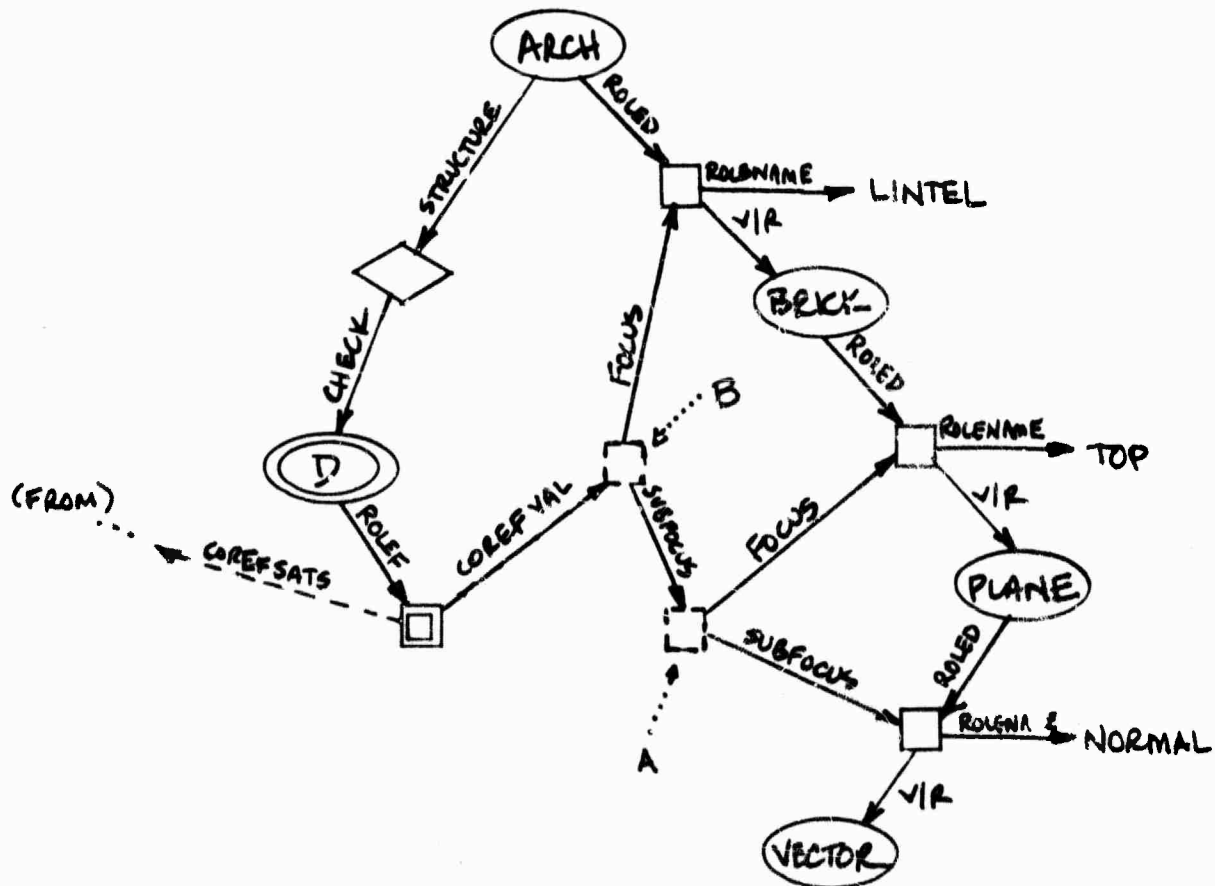


Fig. 6. Substructure Reference.

BRICK (as if it were a Role of BRICK itself). Node B applies that "indirect" Role to the LINTEL of an ARCH. Therefore, the filler of the FROM Role of the Concept D in ARCH is specified as the (potential) NORMAL to the TOP of the LINTEL of the ARCH.

2.3.3 Intra-sd-links

As mentioned previously, a Concept's internal structure is represented by a set of relational pieces called "Structural Descriptions" (SD's). An SD is generally composed of Parametric Individual Concepts that look just like those appearing elsewhere in the network, except that their role-nodes may point to role-nodes of the Concept being defined, as described above. A ParaIndividual does not have an independent existence outside of a Structural Description - it can be created only within an existing SD of some Generic Concept.

The intent of having several SD's in a single Concept is to allow separation of various relational aspects of a Concept (i.e. those involving more than one Role) into epistemologically meaningful chunks. We contemplate SD's for things like complex criterialities (involving several Roles), internal physical structuring relationships, external physical properties, "invited inferences", etc. Each SD, in addition, could make its own weighted contribution to the Concept as a whole, so that thresholds for recognition could be established.

SD's have elements (called "facets") that, for now, will be used to indicate their roles in processing. For the moment, let us consider three types of SD Facet: Check, Derive, and NonActive.

A Check facet is a set of relationships that are expected to hold between Role fillers of a recognizable instance. Every instance must pass the check for its corresponding Concept - that is, satisfy the relationship - before it is entered in the net as valid. Concepts in Check parts of SD's are considered to be predicates that must be verifiably true of the fillers of the instance's Roles. The Checks are in principle performed upon individuation, and when verified, the instance as a whole is considered to be "valid". This type of an SD facet is indicated by a "Check" link.

A second type of facet that we might use is a Derive facet, which is a set of functions that compute fillers for DERIVABLE Roles from those explicitly present in a valid instance. The Derive facets can be activated upon instantiation, after the successful completion of the Checks. The link to an element of this type is "Derive".

Finally, we might consider a more computationally neutral type of facet, which contains relationships between parts of the conceptual complex that may be used for inference, but which are not checkable or do not contribute to the derivation of new Role fillers. For now, we will call these, simply, "NonActive" facets.

These three types of facets might be used to factor the Concept of an ARCH as illustrated in Fig. 7. In the figure, the Check parts of SD's S1 and S2 would be used when forming an instance to ascertain that each UPRIGHT supported the LINTEL (S2), and that the two UPRIGHTs did not touch one another (S1). Once satisfied, the three candidate BRICKs would be considered to form an ARCH, and the one Derive part (of S2) would produce the particular VERTICAL CLEARANCE of that ARCH. Finally, the NonActive part (S3) would indicate that, if necessary, one could infer that the ARCH as a whole could support more weight than either of the UPRIGHTs could individually.

2.3.4 Abstraction Hierarchies and Inter-concept-links

One of the principal uses to which we will put the Structured Inheritance Network is that of an abstraction hierarchy. By this we mean a tree of Concepts, with more general Concepts "higher" in the tree. As mentioned in the first half of this report, this forms a partial ordering under the relationship of generalization (subsumption). The hierarchy thus ranges from very general, abstract Concepts, down through more and more specific ones, all the way down to those describing individuals. In SI-Net-1 terms, all Concepts that make up the intermediate nodes along such specialization chains are Generic, while the leaves, representing particular individuals, are always Individual Concepts.

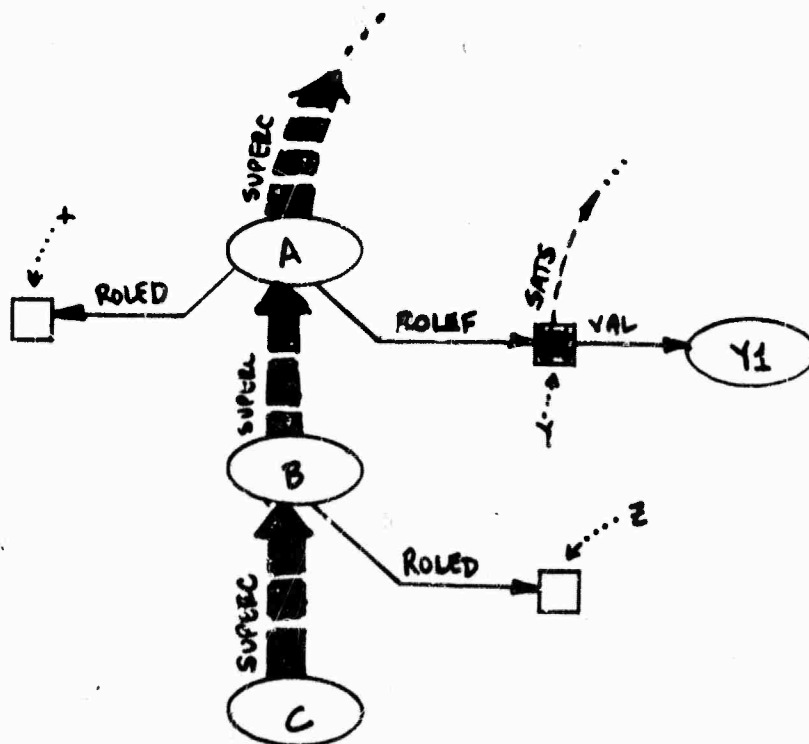


The standard biological taxonomic classification is an example of an abstraction hierarchy. In representing it, we would have a Concept for the cat "family", which subsumed all the more specific Concepts for particular cat genera (e.g. lion, tiger, leopard, jaguar). The Concept, CAT, would represent an abstraction over those genera, which are in some way deemed similar. Each genus Concept would itself have a set of more specific Subconcepts, describing species, and ultimately the chains would end with sets of particular cats, lions, etc.

The fact of the similarity of a set of Subconcepts (e.g. LION, TIGER, etc.) is expressed by their mutual connection to the same "super"-Concept (CAT, in this case). One can store facts and characteristics about all of these classes at the same time by associating that information at the most general Concept that covers the appropriate Subconcepts. The only requirement for the proper interpretation of this structure is an inheritance mechanism that allows all properties true of, say, cats in general to be visible, and assumed true, of all genera of cats, all species of those, and ultimately, LEO, a particular individual lion. One of the most important tasks of SI-Net-1 is to provide an adequate mechanism for achieving this inheritance. In SI-Net-1, the things that are inherited are Roles and SD's, and the backbone of the inheritance is provided by the inter-concept-links.

There are several types of links that one can use to indicate Role and SD inheritance from Concept to Concept. Three of these are

essentially "primitive", and the others can be considered abbreviations for combinations of these and other links (although their explicit use may add more information than the corresponding combination would). The most basic is SuperC. The Concept on the source (tail) end of this link is considered to inherit, intact, all Roles considered to be part of the Concept at the destination (arrowhead) end. In Fig. 8, Concept B inherits, via SuperC, Roles x and y of its parent Concept A. Concept C inherits, via SuperC to Concept B, those Roles, plus any new ones attached to B (z in this case). Notice that according to the rules of 2.3.1.2, y is inherited filled. The SuperC link provides the skeleton for a general taxonomic lattice structure.

Fig. 8. SuperC.

The second basic link is Individuates. This link performs the same inheritance role as SuperC, but arises out of an individual-concept-node. It implies the following: 1) for all Roles that are considered to be filled (RoleF) at the parent node, the offspring node inherits the fillers intact, as its fillers for the same Roles; 2) any Role of the parent that is not satisfied (RoleD) at any higher node can have an explicit Role filler indicated at the offspring; and 3) any Role not filled, with no explicit filler at the offspring a) must, if its Modality is OBLIGATORY or INHERENT, have some unknown filler (a variable) for the Role which satisfies that Role's constraint, or b) if OPTIONAL, may be considered to have no filler for the particular Role. These cases are illustrated in Fig. 9.

In the figure, Individual Concept I inherits X as the filler of its "C" Role; it has Y as the filler of its "D" Role; there is some Concept that fills the "B" Role, but is not yet known (see Section 1.7); and I has no "A" Role filler.

An inter-concept-link that emanates only from ParaIndividuals is "ParaIndividuates". This link acts much like Individuates, but allows the determining of actual satisfiers of Roles to be postponed until an instance of the Concept is constructed. The way that this is achieved is discussed above, in Section 2.3.2.

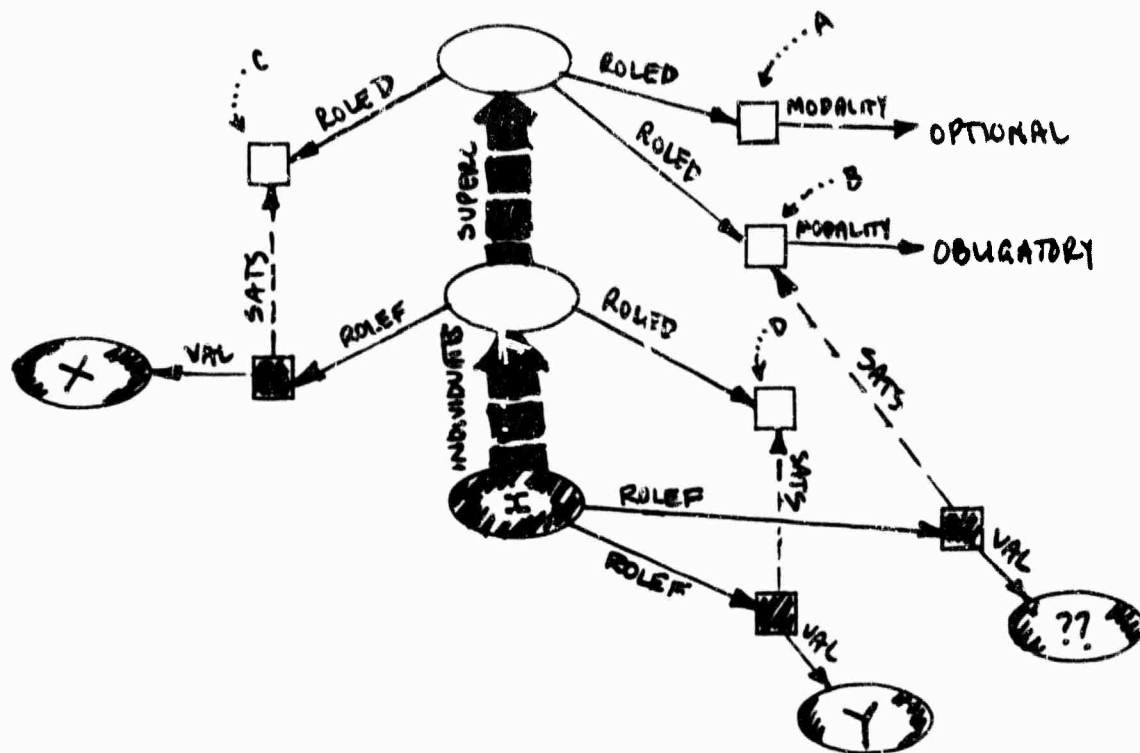


Fig. 9. Individuates.

2.3.5 Inter-role-links

As mentioned earlier, Role/Dattr Descriptions in part represent constraints on the potential fillers of the functional roles of a Concept. In an abstraction hierarchy or taxonomic lattice, Concepts will be connected to more general Concepts (by inter-concept-links), which will in turn be connected to still more general Concepts, etc. The notion of abstraction specifies that common Roles are introduced as "high" in the lattice as possible, so that shared information is stored only once. As we move "down" to more and more specific Concepts, those shared Roles may change to accommodate more restricted definitions. Inter-role-links are used

to build the constraints at lower Concepts based on those inherited from above.

When a Role is inherited by a Concept from a higher Concept, any or all of its facets may be modified. This is indicated by a Mods link from the role-node under construction to the original role-node being changed (it must be one inherited through a SuperConcept chain - see 2.3.4). If a facet is to be modified, it appears anew at the lower node, and its value is taken to be the new value for that facet. For example, in Fig. 10, Concept B inherits the "LINTEL" Role, but alters two of its facets. Anything that is a B now must have a LINTEL that is a WEDGE, whereas to be an A, having a LINTEL was not a necessity, and it took a BRICK to be the LINTEL. The Number of LINTELS of a B remains the same as for an A.

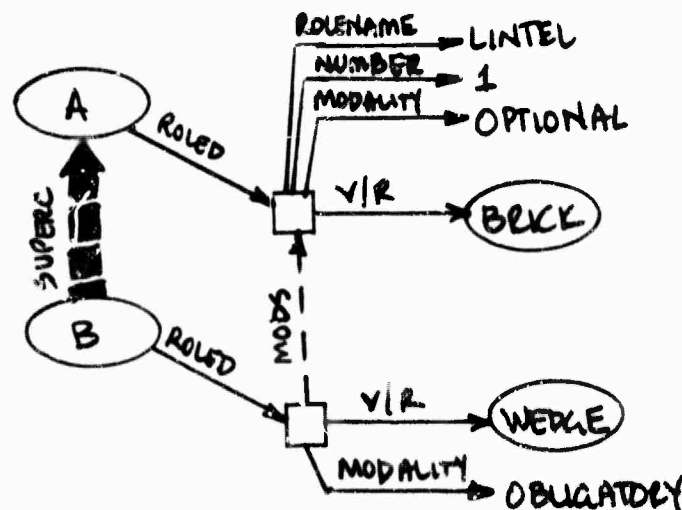


Fig. 10. Role Modification.

In some cases, a Role may give rise to specialized sub-Roles. In a general Concept, we may only be able to outline a Role in very

general terms, expressing only very sketchily what the Role filler does in the Concept, and not being able to determine how many fillers there are to be. In a more specific Concept, however, we may pin down the particular specific subroles and numbers of fillers. As seen in Fig. 11, this operation is indicated by the Diffs link. In this figure, we see that we define the Role of an ARGUMENT (to a function or command) in only very general terms, specifying that there are indefinitely many fillers of the Role. In PCMD (print command), we can pin down the two particular kinds of ARGUMENTs that we need. The two role-nodes of PCMD created from the single ARGUMENT role of CMD act as complete constraints unto themselves (e.g. no Number link means Number=1 rather than any inherited Number). However, the fillers of the new Roles in any particular PCMD (the MSG and the TEMPLATE) could still be considered as ARGUMENTs to the PCMD. They have in common the fact that they are sub-Roles of the same general ARGUMENT Role.

The other relationship between Roles expresses the filling of a Role (i.e. the specification of its value). As shown above, it is called "Sats", and means that the filler (indicated by the Val link) satisfies the inherited constraint (and fills the corresponding role). See Fig. 9, above.

2.3.6 Inter-sd-links

As mentioned above, SI-Net-1 Concepts can be connected to each other to form abstraction hierarchies. In this scheme, Concepts

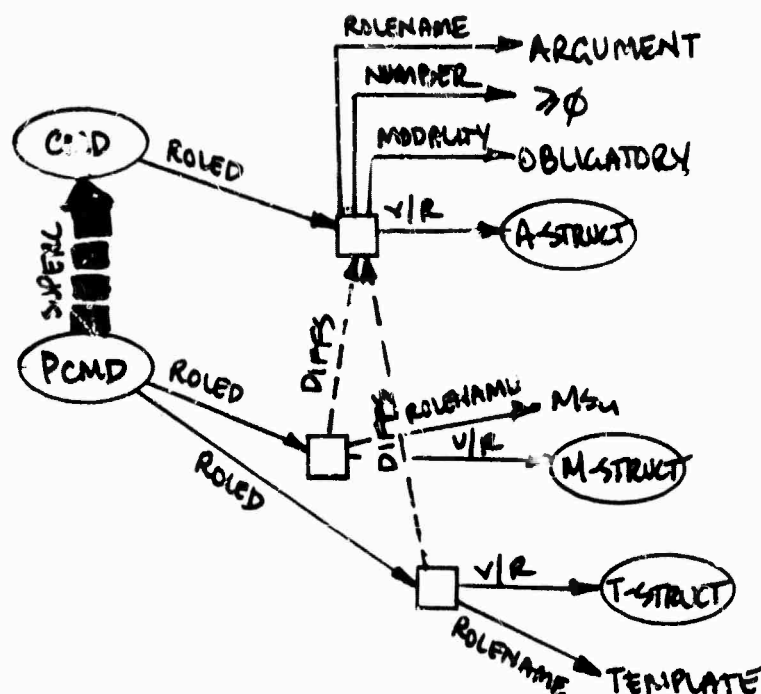


Fig. 11. Subroles.

inherit Roles and SD's from more general Concepts. In a more specific Concept, some of the inherited SD's may need to be altered, or "turned off", or new SD's may need to be added. To this end, SI-Net-1 has a single inter-sd-link, "Preempts", to connect SD's along an inheritance chain. When present in some Concept, S, the SD that is pointed to is no longer inherited by S, and the SD that does the pointing takes over instead. So, for example, in Fig. 12, SD S5 of X overrides S3, which normally would be inherited by S from Y. S4 overrides S2, essentially making it a nul' requirement, since no facets are present. S1 is inherited intact, since no inter-sd-link points to it. S6 is an SD applicable only to individuators of X and not to those of Y in general.

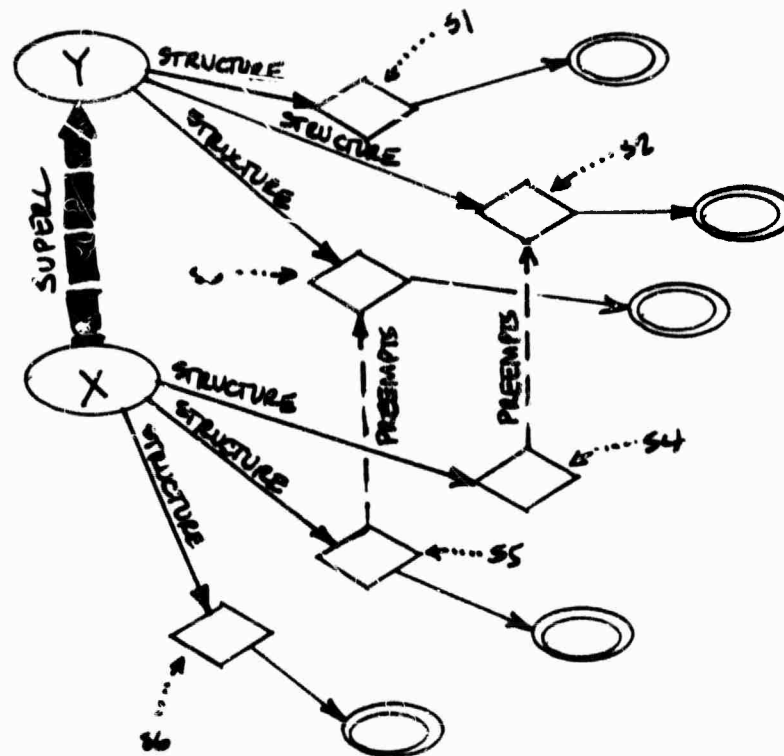


Fig. 12. Inheritance of SD's.

2.4 SI-Net-1 Primitive Concepts

A knowledge base built in SI-Net-1 will have at least a small number of primitive Concepts that are not themselves defined in terms of other Concepts. That is, the system provides to the user some primitive, "hard-wired" Concepts that s/he may use to construct definitions of other Concepts. These do not have examinable Structural Descriptions like regular Generic Concepts do, but instead are implemented directly in "executable" form. The primitive Concepts that the system provides are not intended to be a logically minimal set; one could conceive of alternative smaller sets of logical Concepts out of which all of the others could be

constructed. These sets of primitive Concepts are not different from others in any epistemologically significant way, and are therefore not distinguished notationally. This section summarizes a proposed set of primitive Concepts to be provided with the primitive node and link types.

(1) APPLY

Roles: Code (any LISP s-expression that evaluates to a function)

Argument [≥ 0] (an ordered list of values to be passed as arguments to Code)

Result (the result of APPLYing [in LISP] Code to Arguments)

Effect: when used in an SD, applies Code to Arguments and returns result in Result (just like LISP APPLY)

Note: this is the basic way of getting something "primitive" done, and is probably sufficient to account for all the other primitive Concepts.

There is presumably one subvariety of APPLY for each number of arguments that might be required. These subvarieties would have differentiated RoleNames for the ARGUMENT role (i.e. ARGUMENT1, ARGUMENT2, etc.).

(2) AND

Role: Conjunct [≥ 2] (any predicates)

Effect: when evaluated in Check part of an SD, returns T iff all Conjuncts evaluate to T (otherwise NIL)

(3) OR

Role: Disjunct [≥ 2] (any predicates)

Effect: same as AND, except returns OR(Disj1, Disj2, ...)

(4) NOT

Role: P (a predicate)

Effect: in SD Check, returns T iff P evaluates to NIL (otherwise returns NIL)

(5) EVERY

Roles: X (any Concept)
 R (a predicate)
 P (a predicate)

Effect: in SD Check, returns T iff, for all elements x of class X such that R(x) is T, P(x) is T (Woods' query language: FOR EVERY x/X : R(x) ; P(x)).

If X points to a Concept, R and P are applied to all instances of X; if X is a Coref Role, R and P are applied to all fillers of the Role indicated, when the EVERY is evaluated in any particular instance.

(6) SOME

Roles: X
 R - as in EVERY
 P

Effect: Same as (FOR SOME x/X:R(x); P(x))

(7) EQUAL

- useful only as a ParaIndividual

Roles: X
 Y

Effect: in SD Check, returns T iff X and Y are filled with the same entity (i.e. X's filler and Y's filler are the same)

(8) FILLED?

- useful only as a ParaIndividual

Role: Role

Effect: in SD Check returns T iff Role is filled (i.e. there is a Val link from the appropriate filled-role-node)

(9) BIND

- useful only as a ParaIndividual

Roles: Indep
 Dep

Effect: in SD Derivational part, takes value of Indep
(i.e. its filler), and makes it the filler of
Dep)

Other candidates, which are not explicitly proposed to be included here, but which would be useful to have, are programming functions (e.g. LOOP, IF, WHILE, etc.).

2.5 Meta-description and Procedural Attachment

It is anticipated that one important way to use SI-Net-1 conceptual structures will be to consider them as structured anchors - something akin to "conceptual coat racks" - for advice to the SI-Net-1 interpreter. The SI-Net-1 construct that supports the "hanging" of interpretive commentary (see [Smith, 1978]) on Concepts, Roles, etc. is called the "hook".

A hook is a kind of link, with properties significantly different from the intensional eplink. It does not constitute part of the structure of the concept from which it emanates.

There are two basic hook types: the "metahook" and the "interpretive hook". A metahook is used to represent knowledge about knowledge, and the descriptive information attached to it is expressed in SI-Net-1 notation. Metahooks always point to Concepts, but can point from any entity (i.e. Concept, Role, or SD). When emanating from, say, a Concept, a metahook tells the interpreter to consider that Concept as an entity itself (not as a denotative description of something else). In this way, information about when

the particular Concept itself was constructed, from whom it was learned, etc. can be expressed. For example, consider how Fig. 13 shows the "meta" nature of the name of a Concept. In this figure, the metahook is indicated by a jagged line. The node at the arrowhead end of the metahook represents ARCH as a Concept. That node describes ARCH as a Concept with its NAME role filled with "ARCH".

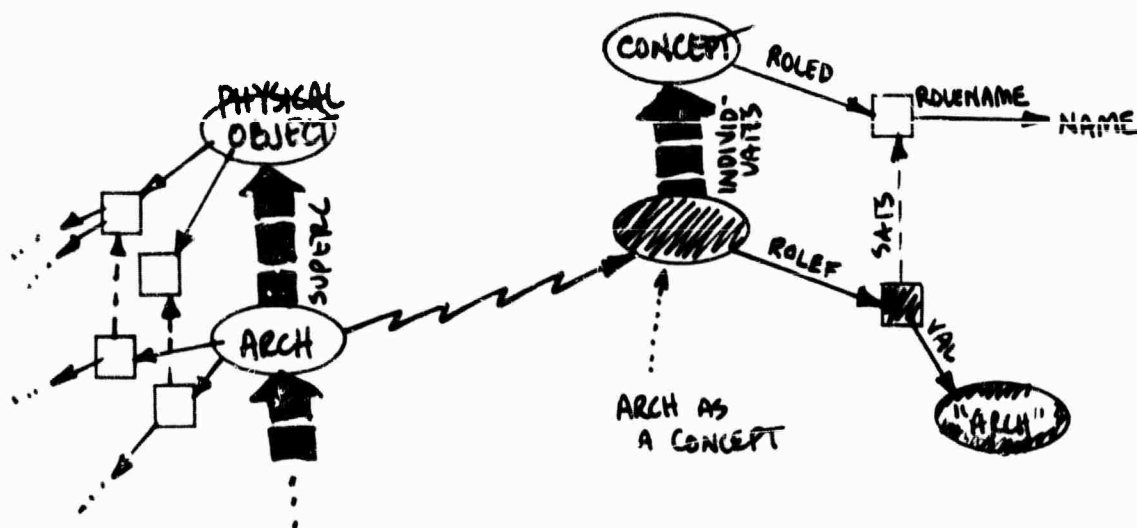


Fig. 13. Meta-description.

The intuitive interpretation of this kind of level jump between use of a concept to denote and meta-comment about the concept itself is as follows: the Concept, ARCH, acts as a description of all individual arches, and therefore passes descriptions from its superconcepts down to all individual arch descriptions (from PHYSICAL OBJECT, for example). ARCH, itself, however, can be described as a Concept with certain characteristics - but this description is not passed to any individual arch descriptions. In

order to clearly distinguish between these two types of description, we need to separate Roles that ARCH passes down to its instances and Roles that apply to ARCH as an individual. The normal inheritance relations (SuperC) accomplish the former, while the metahook achieves the latter. We could, perhaps, separate these by connecting ARCH with an Individuates link to CONCEPT. However, that would make ARCH a generic Concept and an Individual Concept simultaneously. Rather than make one node mean two different interpretations of the ARCH Concept, we use the metahook. Alternatively, one can think of the two ends of the metahook as essentially the same Concept (e.g. superimpose the two ovals), with the hook, however, cleanly separating the senses.

With this type of meta-description, various predicates, in SI-Net-1, can be asserted about Concepts as Concepts. For example, Fig. 14 illustrates how the SI-Net-1 assertion that Concept ARCH was constructed on Nov. 17, 1977 makes use of the Concept in a "meta" way. The metahook from ARCH equates the Concept with the value of the CONSTRUCTION Role in CONSTR1.

As Brian Smith points out [1978], there is another type of interpretive commentary that should be captured in a knowledge representation language. This commentary looks a lot like meta-description, but is encoded in the language of the interpreter itself. Often called "procedural attachment", this type of instruction is directly executed by the interpreter when appropriate. In the SI-Net-1 system, such code will be written in INTERLISP.

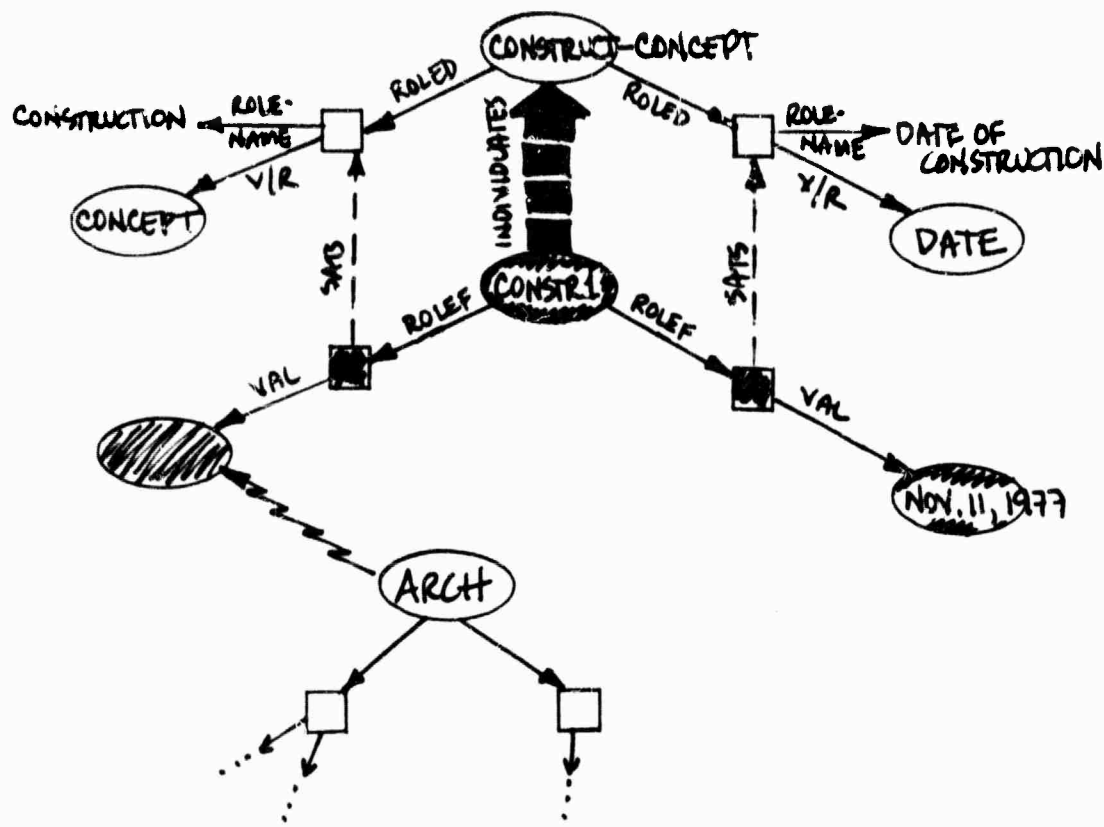


Fig. 14. Predication of Concepts.

To implement this we have, similar to the metahook, the interpretive hook (ihook). As should be evident, there are three important characteristics that the ihook needs to account for. First, naturally, there is the place at which the interpretive information is relevant. This, as was the case with metahooks, is indicated by the tail of the hook. Second, there is the situation in which the hook should be followed and used. Situations are things like "when this Role is filled", "before instantiating this Concept", etc. We will create a class of these to handle appropriate situations, and indicate the invoking situation by a

label on the ihook. And finally, the code to be executed is found at the head of the ihook. This code will be applied to a single argument, the entity that is being worked on when the ihook is invoked.

Figure 15 illustrates the places where ihooks can be attached, and indicates a few possible candidates for invoking situations. Invoking situations are generally expressed as before or after some interpreter action is taken on a node. These should be self-evident from the figure and the following list of possible ihook types:

<u>Place</u>	<u>Ihooks</u>
Concept	Before/After-Individuation; B/A-Subcategorization; B/A-Editing; B/A-Deletion
Role	B/A-Filling; B/A-Modification; B/A-Differentiation; B/A-Derivation; To-Fill; B/A-Editing; B/A-Deletion
Structural Description	B/A-Check; B/A-Derivation; B/A-Editing; B/A-Deletion; To-Check; To-Derive.

Using the ihook facility, we can implement the primitive Concepts of Section 2.4 in a straightforward fashion. Fig. 16 illustrates how the basic Concept, APPLY, is implemented: instead of an introspectable SD, there is a blank SD, attached to which is a direct call to the interpreter. The attached procedure is invoked when the SD would be used ToDerive a DERIVABLE Role. It makes use of the interpreter functions SatisfyRole, FindNamedRoles and GetRoleValue.

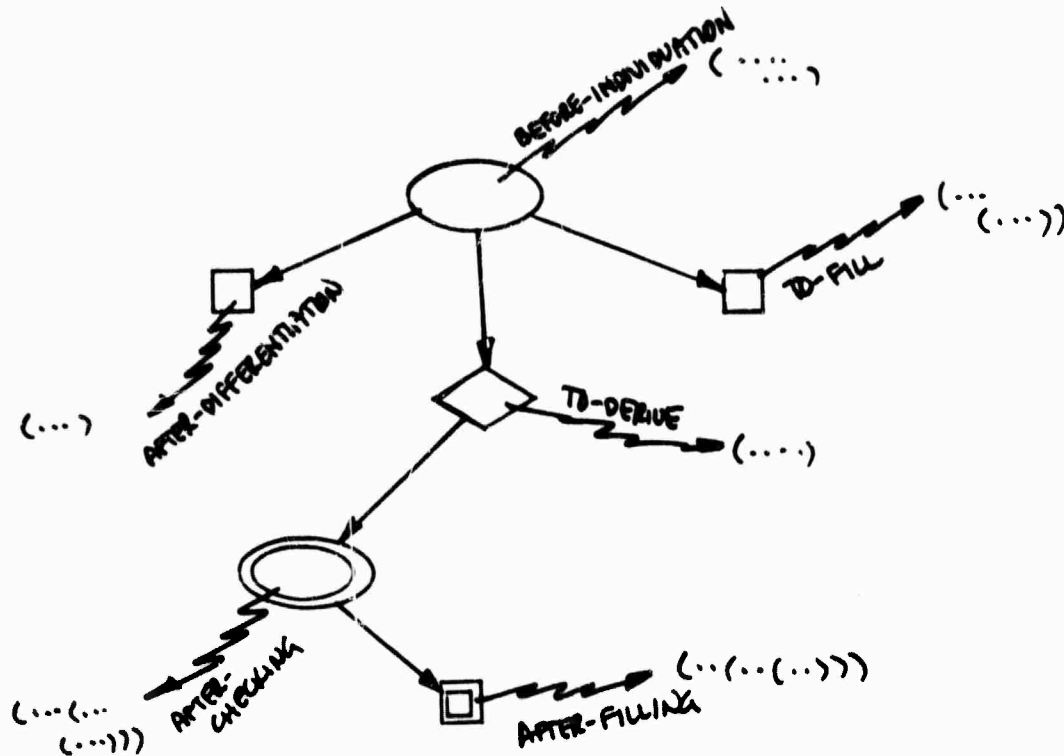


Fig. 15. Procedural Attachment.

2.6 The Individuation Process

The process of creating a representation of a particular object described in terms of a Generic Concept is called "individuation".* The basic aspect of SI-Net-1's individuation is the specification of fillers for all Roles inherited from the Generic Concept being individuated. To create an Individual Concept that is considered an "individuator" of a generic one, each OBLIGATORY Role must be assigned a filler. This can be achieved by the user in a single

 *This is usually called "instantiation", but we avoid the term here in order to avoid confusion over what an "instance" is. Recall that we chose to call an individual description related to a general one an "individuator" of the more general description.

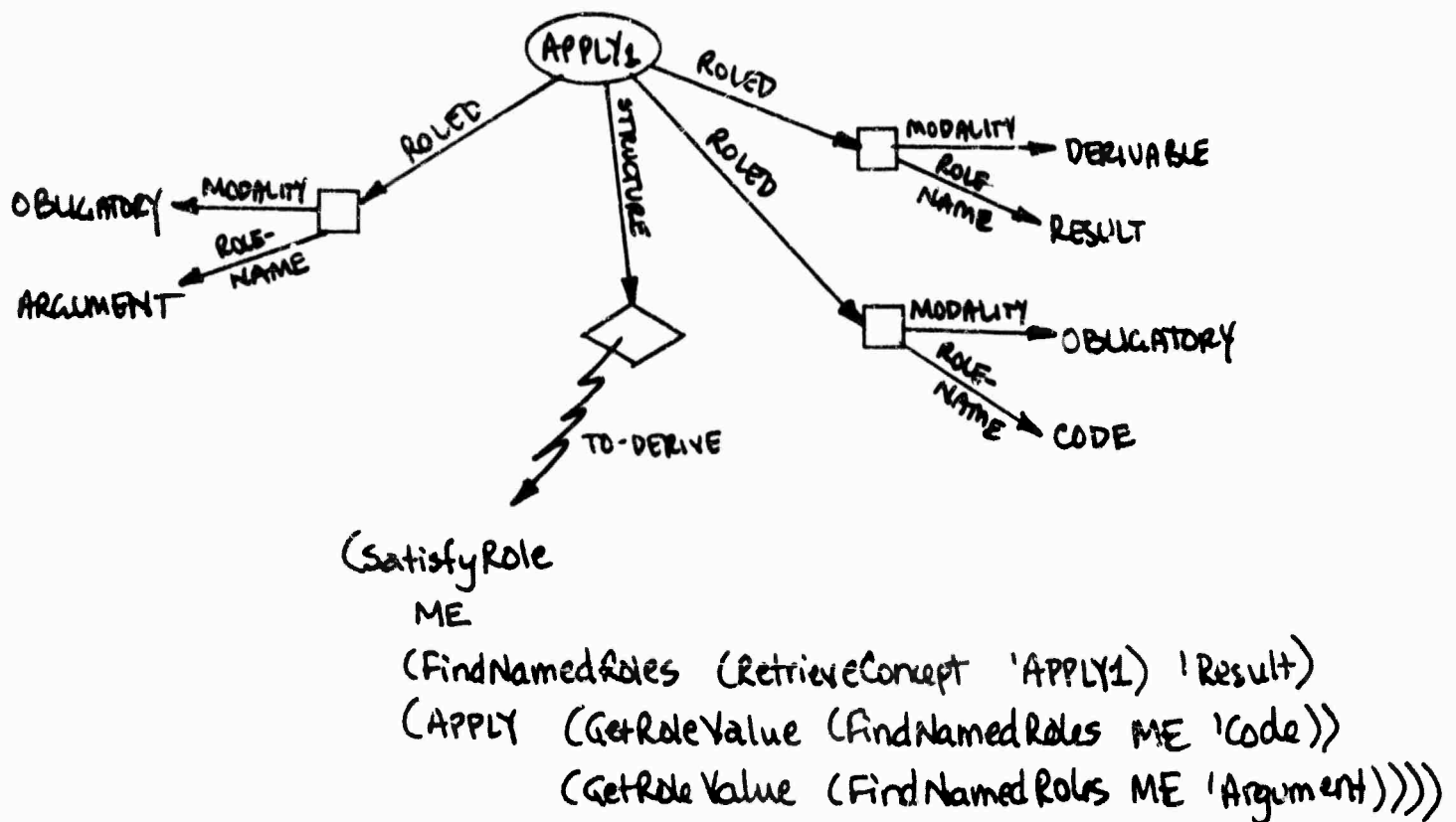


Fig. 16. The Primitive Concept, APPLY.

function call which specifies the fillers for each Role, or alternatively, the user might be interrogated by the system and asked to provide the appropriate number of fillers for each Role, one at a time.

In any case, once Role-filler bindings are specified, the procedure (call it "INDIVIDUATE") has the following simple form:

1) ascertain that

- each inherited Role that is filled is filled the appropriate number of times
- each filler satisfies the V/R constraint of its Role.
- all OBLIGATORY Roles are filled.

- 2) invoke the Check parts of all SD's.
- 3) invoke the Derive parts of all SD's.

Note that each step proceeds only upon successful completion of the previous one.

The procedure to be described below, if taken literally as an algorithm, is only one way to accomplish this "validation" process. The above-mentioned three steps (elaborated below) constitute, in some sense, the definition of what it means to be valid - if all three constraints are satisfied, no matter how it is ascertained that they are, then the instance (individuator) is validated. In particular, for many applications such as situation recognition, it may be desirable to evaluate (1) and (2) incrementally as Roles are filled in hypothetical conceptualizations of the input. It is probably better, therefore, to consider the following more as an abstract definition of validation, and less as a particular procedure to be run.

Step 1 is straightforward: the Number predicate of a Role is inherited from the first Role on a Mods chain in which a Number link appears. The fillers in the individuator are counted, and the Number predicate is applied to the count. If it succeeds, then the inherited V/R is found in the same manner, and the filler is checked to see that there is some path through one Individuates link, and possibly many SuperC links, to the V/R Concept. Finally, all OBLIGATORY Roles are checked to see if they have been assigned

fillers. If so, Step 1 has ascertained that, individually, the Role constraints have been met.

As mentioned earlier, the Check parts of the SD constitute a set of parametric tokens of Concepts in the network. For each token pointed to by a Check link, the following is done (call this procedure "INDIVIDUATOR/CHECK"):

- 2.1 Bind all Roles of the token as specified, creating an ephemeral "individuator" of the Concept of which the one in hand is a paraindividuator.

Binding works as follows:

- a) if some Role is specified as bound to the RESULT of some other Concept, recursively evaluate that ParaIndividual first (i.e. call INDIVIDUATE) to produce the RESULT; then bind the RESULT to the Role specified.
 - b) if some Role is specified as bound to a token of another Concept, invoke INDIVIDUATOR/CHECK on that token, and then bind as value T if INDIVIDUATOR/CHECK succeeded, or NIL if not.
 - c) if a Role points with a CorefVal link to one of the Roles of the enclosing Concept (this may be an indirect Role, in which case the Focus/Subfocus chain is followed to obtain the value), make the value of that Role to be the filler specified initially for the enclosing Concept.
- 2.2 Finally, evaluate INDIVIDUATOR/CHECK (recursively) on the Checks of the ephemeral instances created in Step 2.1; if this succeeds, then Step 2 succeeds.

The third Step of INDIVIDUATE (i.e. invoking derivations) works similarly to the second, except that when a RESULT is calculated, it may be bound to a Role of the enclosing Concept. This step produces bindings for the Roles of the Concept being built. Roles that can be filled in this way are what we have been calling "DERIVABLE."

The Individual Concept constructed during individuation is considered valid after Step 2 concludes successfully. Thus, the Concept just built may be used as a source of information in the Derivational Step (Step 3). This happens, for instance, when an Individual Concept, A, causes a DERIVABLE Role to be filled with an Individual Concept, B, one of whose Roles in turn is filled by A. The initial Individual Concept (A) causes a larger structure to be built as a consequence of its presence, and that larger structure encompasses the particular individual.

2.7 Further Refinements

While the above descriptions constitute a basic definition of the language, SI-Net-1, the specification is not yet complete. The following is a list of topics that need to be elaborated further.

- NOTHING - a Special Concept that works like "NIL" in LISP, and indicates explicit non-filling of a Role
- variables - indefinite Roles and Concepts
- default names for non-named Roles (more than one instance) - should the system supply these?
- always accept given over derived -- what the user says is more pertinent than what the system derives
- multiple superconcepts, superroles, and their inheritance problems
- Supra-SI-Net-1 Conventions - "Macro" patterns of use of SI-Net-1 structures will probably evolve as the system is used extensively. At some point, it may appear that certain of these patterns are so fundamental, that they should be the province of the language/system itself, rather than that of the application or user. Several of these are already apparent, and the decision has to be made as to their possible place in the SI-Net-1 epistemology. For the

moment, the decision to include or exclude knowledge representing conventions is being based on epistemological, rather than practical necessity.

The kinds of operations that might be universally useful, but that can be constructed out of already presented mechanisms, include functions, "basic types" (mutual exclusion hierarchies), "worlds" (contexts) and assertions relative to those worlds. Another useful convention is "cascaded derivation", in which an instance will cause the derivation of another instance, which will in turn cause the derivation of a third, etc.

- inverse Roles - are there some Roles which can be reached more easily from fillers than others? Are certain roles merely inverses of others (e.g. EMPLOYER/EMPLOYEE)? If so, how do we represent them?
- incremental Concept construction, and partial states of validity - we clearly need to be able to build Concepts piecemeal. How does this affect the notion of "validity"? At the moment, we are considering four possible states of knowledge: Valid, NotValid, Can'tTell, and NotTested.
- "tags" - provision for the user to attach data structures of his/her own design to SI-Net-1 entities. Like ihooks, but point to arbitrary data structures, not LISP code.
- nominalizations - there is a difference between the concept of a THIEF, and the "AGENT" Role of a STEAL event, but there is an important relation between them. Is that relationship epistemological?

3. References

Brachman, R.J. (1977)
"A Structural Paradigm for Representing Knowledge," Ph.D. dissertation, Division of Engineering and Applied Physics, Harvard University.

Fahlman, S.E. (1977)
"A System for Representing and Using Real-World Knowledge," Ph.D. dissertation, Dept. of Electrical Engineering and Computer Science, M.I.T.

Lowerre, B.T. (1976)
"The HARPY Speech Recognition System," Technical Report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa.

Quillian, M.R. (1966)
"Semantic Memory," Report No. AFCRL-66-189, Bolt Beranek and Newman Inc., Cambridge, Ma.

Quillian, M.R. (1968)
"Semantic Memory," in Semantic Information Processing (M. Minsky, ed.). Cambridge, Ma: M.I.T. Press., pp. 27-70.

Rieger, C. (1977)
"Spontaneous Computation in Cognitive Models," Cognitive Science 1, No. 3, pp. 315-354.

Smith, B. (1978)
"Levels, Layers, and Planes: The Framework of a System of Knowledge Representation Semantics," Master's thesis, M.I.T. Artificial Intelligence Lab, February.

Wolf, J.J. and W.A. Woods (1977)
"The HWIM Speech Understanding System," Conference Record, IEEE International Conference on Acoustics, Speech, and Signal Processing, Hartford, Conn., May.

Woods, W.A. (1970)
"Transition Network Grammars for Natural Language Analysis," CACM, Vol. 13, No. 10, October (reprints available).

Woods, W.A. (1973)
"Progress in Natural Language Understanding: An Application to Lunar Geology," AFIPS Conference Proceeding, Vol. 42, 1973 National Computer Conference and Exposition (reprints available).

Woods, W.A., M. Bates, G. Brown, B. Bruce, C. Cook, J. Klovstad, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue (1976)
Speech Understanding Systems - Final Report, 30 October 1974 to 29 October 1976, BBN Report No. 3438, Vols. I-V, Bolt Beranek and Newman Inc., Cambridge, Ma.

BBN Report No. 3742

Bolt Beranek and Newman Inc.

Woods, W.A. (1977)

"Semantics and Quantification in Natural Language Question Answering," to appear in Advances in Computers, Vol. 17, New York: Academic Press. (Also Report No. 3687, Bolt Beranek and Newman Inc., 1977).

Official Distribution List
Contract N00014-77-C-0378

	<u>DODAAD Code</u>	<u>Copies</u>
Defense Documentation Center Cameron Station Alexandria, VA 22314	S47031	12
Scientific Officer Program Director, Information Systems Office of Naval Research 800 North Quincy Street Arlington, VA 22217 Attn: Mr. Gordon D. Goldstein Code 437	N00014	3
Office of Naval Research Department of the Navy Code 1021P Arlington, VA 22217	N00014	6
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, MA 02210	N62879	1
Naval Research Laboratory Technical Information Division Code 2627 Washington, D.C. 20375	N00173	6
Director Advanced Research Projects Agency Information Processing Techniques 1400 Wilson Boulevard Arlington, VA 22209	HX1241	2
Administration Contracting Officer Defense Contracting Administration Services 666 Summer Street Boston, MA 02210 Attn: Mr. Anthony Lopez, ACO	S2206A	1