Bolt Beranek and Newman Inc.

ADA085917

Report No. 4274

Research in Natural Language Understanding

Annual Report

1 September 1978 — 31 August 1979

Prepared for:
Advanced Research Projects Agency

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 4274 / 2. GOVT ACCESSION NO. AD-A085917	3. RECIPIENT'S CATALOG NUMBER
4. ~~TITLE (and Subtitle)~~ 6 RESEARCH IN NATURAL LANGUAGE UNDERSTANDING. ~~Annual Report~~ 1 September 1978 - 31 August 1979	5. TYPE OF REPORT & PERIOD COVERED Annual Report 9/1/78 - 8/31/79
	6. PERFORMING ORG. REPORT NUMBER BBN Report No. 4274
7. AUTHOR(s) 10 William A./Woods R.J./Brachman R.J./Bobrow P.R./Cohen J.W./Klovstad ~~B.L. Webber, and W.A. Woods~~	8. CONTRACT OR GRANT NUMBER(s) 15 N00014-77-C-0378, ARPA Order-3414
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02238	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 11 31 Aug 79
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy Arlington, VA 22217	12. REPORT DATE 8/31/79
	13. NUMBER OF PAGES 174
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 12 178	15. 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.
14 BBN-4274

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
9 Annual rept. 1 Sep 78-31 Aug 79,

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Knowledge representation, structured inheritance networks, KL-ONE, cognitive science, artificial intelligence, natural language understanding, RUS, parsing, syntactic-semantic interactions, indirect speech acts, natural language graphics.

20. ~~ABSTRACT~~ (Continue on reverse side if necessary and identify by block number)
This report summarizes work done on BBN's ARPA-sponsored natural language understanding project during its second year (from 9/1/78 to 8/31/79). In it we describe in detail a prototype natural language understanding system oriented toward satisfying the information presentation needs of a commander in a command and control context. The system can be requested in English to produce displays of various sorts; it recognizes
cont'd. over

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

060100 [illegible]

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (cont'd.)

indirect as well as direct requests by hypothesizing a plan that the user is trying to carry out. The system is designed to be general in its handling of English syntax, indirect speech acts, and arbitrary conceptual descriptions of displays to be produced. It is built out of several substantial components, each of which is described in a chapter in this report:

(1) a general knowledge representation language (KL-ONE);

(2) a sophisticated state-of-the-art English parser (RUS) and semantic interpretation mechanism (the PSIKLONE Interface);

(3) a pragmatics/discourse component capable of plan recognition and understanding indirect speech acts;

(4) an intelligent graphics component with an explicit knowledge base describing various object types, display form (shape) types, and coordinate system transformation information.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 4274

RESEARCH IN NATURAL LANGUAGE UNDERSTANDING

Annual Report
1 September 1978 - 31 August 1979

Principal Investigator:
William A. Woods
(617) 491-1850 x4361

Prepared for:

Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

ARPA Order No. 3414

Contract No. N00014-77-C-0378

Effective Date of Contract:
1 September 1977

Contract Expiration Date:
30 September 1981

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 Defense Advanced Research Projects Agency or the U.S. Government.

TABLE OF CONTENTS

LIST OF FIGURES

PRECEDING PAGE BLANK - NOT FILMED

INTRODUCTION

BBN's ARPA-sponsored project in Natural Language Understanding is aimed at developing techniques for computer assistance to the decision-maker faced with a complex system or situation. In particular, we are working toward an environment in which the decision-maker (e.g., a military commander in a command and control context) can make use of a highly intelligent graphics display through natural language interaction. Our goal is to provide flexible generation of standard and innovative types of displays well-suited to the needs of a human attempting to understand some complex, multi-faceted situation.

Within the past year, we have designed and implemented a prototype natural language understanding system to test out our ideas on natural interaction in a command and control situation. This system combines many aspects of our research on knowledge representation and natural language: taxonomic lattice structures; structured inheritance; semantics-assisted parsing; attached procedure invocation; plan recognition; indirect speech act recognition and interpretation; and planning in a graphics context. This report is intended to describe in detail the prototype system, in order to illustrate how these various threads of research bear on one another.

First, Chapter 1 discusses the kind of command and control interaction that we would like to support. This discussion motivates the particular application area that we have chosen, and demonstrates the importance of natural language as the communication medium in our system. Chapter 1 also explains the critical part that **representation of knowledge** plays in the intelligent graphics system. In keeping with the importance of representation in our work, Chapter 2 is devoted to a description of the particular language we have developed for representing our system's knowledge, KL-ONE. KL-ONE is used throughout the natural language system to capture syntactic, semantic interpretation, domain structure, discourse, and graphical information, and constitutes in its own right a major thrust of our research efforts.

The remainder of the report discusses the structure and operation of the system we have built, and is organized thusly: Chapter 3 presents an overview of the system, illustrating how the pieces to be described in later chapters fit together. Chapter 4 examines the part of the system with which the user interacts directly - the natural language "front end". The **RUS Parsing System** is described, as is the **PSI-KLONE** interface, which allows RUS to communicate and be guided by the conceptually-based part of the system. By the end of this chapter, the reader will have a good idea of how English sentences get translated into

KL-ONE structures representing their literal semantic content. The province of Chapter 5 is the discourse component - the one that attempts to determine what the user has really meant in performing a certain utterance. By postulating a plan that the user is pursuing in his interaction with the system, it can look for utterances that fulfill its expectations, and interpret otherwise ambiguous expressions (anaphora, for example) in the appropriate way. In response to the user's input, this part of the system plans an appropriate response - either a verbal one, or some manipulation of the graphics display. The representation of graphics knowledge is for the most part embedded in KL-ONE[1] although communication between the planning part of the system and the graphics part is currently through an ad hoc LISP interface. Chapter 6, our final entry, describes the picture-drawing part of the system.

[1] This part of the system builds on another ARPA-sponsored system, AIPS (Advanced Information Presentation System). See [Zdybel, Yonke, and Greenfeld, 1980].

1. BACKGROUND: THE COMMAND AND CONTROL ENVIRONMENT

BBN's ARPA-sponsored project in Natural Language Understanding is aimed at developing techniques for computer assistance to a decision maker who is attempting to understand a complex system or situation using natural language control of an intelligent graphics display. A particular motivating need is that of a military commander in a command and control context both in strategic situation assessment and in more tactical situations - especially in crisis situations. Not only does the commander require certain information in order to make his decisions effectively, but in complex situations this requires the presentation of that information in a form that is matched to the abilities of human comprehension.

One of the underlying assumptions of this project is that in a crisis situation, the commander needs an extremely flexible system capable of manipulating large amounts of data in an underlying data base and presenting it on a graphical display in a variety of ways until the commander feels satisfied that he has a grasp of the situation. Such a display system would include many different kinds of map overlays, and would possess the abilities to change the kinds and amounts of detail shown in a display, to conveniently construct unique kinds of displays to suit the situation at hand, and to display tabular and graphical information and present textual material in ways that are easily comprehensible.

In a crisis circumstance, the description of the display that the commander wants and the specifications of the modifications to it that he will subsequently want to make must be made in a highly fluent and expressive language at a level of abstraction appropriate to the commander's intent. That is, one must not require the equivalent of a graphics systems programmer in order to obtain the displays required. Rather, one needs a system that is capable of accepting an abstract specification of the essential details of what should be in a display and then intelligently and effectively determining the remaining details necessary to actually produce that display. This is true whether or not the actual specification of requests to the computer system is done by the commander himself or by subordinate specialists.

If the language of such a system is to be matched well to human cognitive abilities, it should include a number of the features of ordinary natural language, including the use of pronouns and other anaphoric expressions. Other capabilities that the system should have to make it suitable for natural interaction are the ability to take an incomplete specification and fill in the details on the basis of prior knowledge, and the ability to take a specification that would be potentially ambiguous out of context and determine the intended meaning. Although it might be possible to design an artificial language which met the above needs, we believe that the best methodology for developing the required capability is to use natural English as the communication language for such systems. Although natural English has the advantage of minimizing the problems of learning and remembering special conventions, it is important to understand that the primary advantage of a natural language for this application is its convenience and flexibility in allowing a user to express fairly closely his conceptualization of the problem.

Understanding the commander's requests for information in the kinds of contexts that we envisage will require a number of capabilities that are significant research topics in knowledge representation and language understanding, a substantial number of which have not been adequately studied in the past. One of these is situation dependent interpretation of deixis and anaphora. The mechanism of anaphora permits one to make a subsequent reference to something that has previously been said in a dialog (e.g., using pronouns or definite noun phrases to refer to previously mentioned objects), and deixis involves references to things that have not been said, but are present in some way in the non-linguistic context of the conversation (e.g., in this case, what has just happened on the display screen). Anaphoric reference has been studied extensively in linguistics (although the problems are far from solved), whereas deixis of the kind that occurs in the display context is considerably less well understood.

The resolution of both deictic and anaphoric reference requires the system to perform certain kinds of common-sense inferences about possible referents determined by alternative possible interpretations and the plausibility of those alternatives. This in turn requires an ability to store and use considerable amounts of knowledge about the domain of discourse and the goals and objectives of the user.

In addition to linguistic devices, another aspect of the interpretation of the user's input depends even more critically on knowledge of the domain and the user. This is the process of filling in details that have presumably been intended but not literally said. Much of the time in communication with the system, the commander will not say literally what he intends, and there are good reasons not to require him to do so. The major one is that it is cognitively inefficient to be meticulously literal in one's communication (that is one reason that computer programming is a time consuming and expensive activity). One of the major activities in programming a computer to do a complex task is the systematic specification of all of the details that would be left unsaid if one were instructing a human to carry out the same task. We cannot afford in the command and control situations that we are considering to require this degree of literal specification of detail. Rather, the system must know enough about the objectives of the user that it can fill in details in reasonable ways, asking the user for clarification occasionally, but only when absolutely necessary.

Finally, the system should be able not only to use its general knowledge and the knowledge in its data base to go beyond doing merely what was requested, but in addition provide information that it can infer to be relevant to the user's goal and not otherwise known to the user. For example, when the commander asks how many of his interdiction fighters are equipped with a particular kind of radar during a mission planning operation, the system should volunteer information about how many of those radars are out of commission (unless it knows that the commander already knows that). That is, the system should go beyond the passive execution of the user's commands to infer the goal structure underlying those commands (where possible) and to volunteer additional relevant information (usually in accordance to standing instructions as to what kinds of additional information should be offered in what situations).

1.1 Knowledge Representation in Command and Control

The previous discussion alludes to the extent to which the representation and use of general world knowledge, knowledge of a particular domain, and knowledge of the goals and objectives of users are critical in the development of fluent communication and effective information display in the command and control context. The influence of these kinds of knowledge is fundamental in a variety of other artificial intelligence applications as well. Consequently, a major portion of our effort in this project has been and will continue to be devoted to fundamental problems of knowledge representation and use.

Central to using knowledge in understanding and appropriately responding to user requests is a problem that we call "situation recognition". At various points the system is in a state where it needs to determine which of a large number of possible rules of action are applicable to its current situation. The discovery of such rules can become a significant factor when the number of rules in the system becomes large. Consequently the development of representational structures and special algorithms for making such inferences efficient is especially important.

The work that we have done on knowledge representation has been guided by this need, and we have developed several concepts that we hope will provide sufficient speed and efficiency for making use of large knowledge bases. One of these is the view of network structures in our representation language as instances of cascaded generalized transition networks with advantages similar to those of ATN grammars [Woods and Brachman, 1978]. Another is the development of a class of marker passing algorithms for performing situation recognition operations on an abstract parallel automaton [Woods, 1979a]. These algorithms have a potential for massive parallelism and hold significant promise for providing real time operation of such systems in specialized VLSI computer systems.

2. AN INTRODUCTION TO KL-ONE

2.1 Preliminary Comments

KL-ONE is a uniform language for the explicit representation of conceptual information based on the idea of structured inheritance networks [Brachman, 1978a, b, 1979]. Several of its prominent features are of particular importance in the prototype natural language system we have built - its semantically clean inheritance of structured descriptions, taxonomic classification of generic knowledge, intensional structures for functional roles (including the possibility of multiple fillers), and procedural attachment (with automatic invocation).

This chapter presents a sketch of the version of KL-ONE used to support our natural language system. This representation language has been under development for several years, and now stands among the more powerful and useful paradigms in current knowledge representation technology. Our research on the language has proceeded considerably beyond the implementation used in the prototype system. This chapter presents a mix of new ideas and old implementation; we attempt not to mislead the reader, but rather to describe enough of the language to understand later chapters as well as to present a hint of things to come.

2.1.1 Language Structure and Philosophy

Before going into the details of KL-ONE structures, we will first paint the background of our philosophy in developing the language. As mentioned, KL-ONE is intended to represent general conceptual information. It is intended to allow construction of a knowledge base of a single reasoning entity, as opposed to being a repository for information from multiple sources. A KL-ONE network thus represents the beliefs about the world (and other possible worlds) as conceived by a single thinking being. Note that we are not intending to attempt to capture the world "as it really is" - only the conception of it by an individual perceiver.

KL-ONE is actually two sublanguages - a **description language** and an **assertion language**. The description language allows one to form a variety of description terms (e.g., general terms, individual descriptions) out of other description terms and a small set of primitive description-formation operators. The assertion language makes use of terms from the description language to make statements about the world. Assertions of this sort include statements of coreference of description in a particular context and of existence and identity of individuals in a particular context. In general, structures in the description language have no assertional import (but see Section 2.3).

In the past, most of our work on KL-ONE has been on description-formation; until recently very little attention was paid to making assertions. While we have now begun to focus more intensively on the assertion language, the natural language system described in this report was built almost exclusively out of elements of the description language (Concepts and Roles - see below). Therefore we shall concentrate the following discussion on those elements, commenting only briefly on an emerging conception of "how to say things with KL-ONE" that we feel will soon be a critical part of our system.

2.1.1.1 Epistemological Primitives

KL-ONE is an object-centered language. While its development has proceeded from traditional semantic networks, it does not base its structures on either propositions or sets as did several of the earlier semantic net systems (e.g., see [Schubert, Goebel, and Cercone, 1979; Hendrix, 1979]. Instead, the principle element of KL-ONE is the structured conceptual object, or Concept.

Our view of these objects comes from a careful analysis of early trends in semantic networks and recent trends in knowledge representation in general. As discussed in [Brachman, 1979] and [Woods, 1975], the history of network representations is fraught with imprecision on the meanings of nodes and links. We have found links in networks to represent implementational pointers, logical relations, semantic relations (e.g., "cases") and arbitrary conceptual and linguistic relations. Network schemes consistent with structures of any one of these "levels" (implementational, logical, conceptual, linguistic - see [Brachman, 1979]) can be compared and tested for adequacy, but unfortunately, most of the existing paradigms mix structures from two or more of these levels. This makes for confusing notation and difficulty in explaining a system's interpreter.

Realizing the value of consistency at a single level of network primitive, we have set out to capture an adequate set of primitive elements for structuring a broad spectrum of kinds of concepts. We are attempting to determine a reasonable set of underlying object and relation types for generalized knowledge-structuring. To the extent we can formalize the language of concepts in a grammar for well-formed conceptual structures, we have defined an "epistemology" - a theory of the nature of thought. Note that this is not a theory of any particular domain - one builds that on top of this level - but a generative theory of the structure and limits of thought in general. We address our research to the **epistemological level of network primitives** with the long-term goal of examining the scope of what is "thinkable".

KL-ONE thus comprises a set of "epistemologically primitive" structure types and structure-forming operations. We have attempted to understand the important epistemological features of the internal structure of concepts, and to embody them in a language that is expressively powerful and fairly natural to use.

2.1.2 JARGON

As will become evident momentarily, our primary conception of KL-ONE structures is graphical - the network structures are conveniently visualized as (at least) two-dimensional networks. Our implementation, however, does not yet have a two-dimensional editor or browser. The implemented KL-ONE system has as its primary interface a large set of INTERLISP functions that treat the object types of the system as essentially abstract data types. Our original functions were not constructed for user convenience, but instead were conceived of as the set of primitive operations one could use on a KL-ONE data base.

In order to make input and editing of a knowledge base easier, we have begun to adapt and use a language called **JARGON** for building knowledge structures. This language, initially developed under a companion ONR project, is a formalized, stylized subset of natural English. Although JARGON is similar to other "English-like" languages in that it makes some radical simplifications in the range of syntax it permits, it differs from many such languages in that it preserves rather faithfully the underlying conceptual structures of those sentences. (This, we feel, is to a large extent a reflection of the naturalness of the KL-ONE epistemology.) The JARGON language has been used as a tool for building the various knowledge networks sketched in Figure 10 (see Chapter 3), and, as will be pointed out in Chapters 4 and 5, has been called internally by parts of the natural language system.

JARGON's principal verbs are natural formalizations of the English words "be", "have", and "satisfy". It allows possessives and "of" phrases for specifying chains of subparts of Concepts, and has a flexible noun phrase mechanism. Rather than discuss the language at length here, we refer the reader to [Woods, 1979b] for a detailed presentation. In various figures in this chapter, the reader will find a KL-ONE graphical structure accompanied by an English-like phrase in quotes - such a phrase is the JARGON one might use to create that KL-ONE structure.

2.2 Concepts and Roles

The principal elements of KL-ONE descriptions are Concepts, of which there are two major types - Generic and Individual. Generic Concepts are arranged in an inheritance structure, expressing long-term generic knowledge as a taxonomy. A Generic

Concept is expected to act like the conceptual equivalent of a "general term" [Quine, 1960] - potentially many individuals in any possible world can be described by it. A single Generic Concept is a description template, from which individual descriptions (in the form of Individual Concepts) are formed. An Individual Concept can describe only a single individual in any possible world.

KL-ONE Concepts are highly structured objects. Each Concept's meaning is a combination of the meaning of one or several more general Concepts (its superConcepts) and its local internal structure expressed in Roles, which describe potential relationships between instances of the Concept and other closely associated Concepts (i.e., those of its properties, parts, etc.), and Structural Descriptions, which express the interrelations among the functional Roles. A superConcept serves as proximate genus, while the internal structure expresses essential differences, as in classical classificatory definition [Sellars, 1917].[2] The summum genus is represented by the Concept THING.[3]

[2] The intuitive form of this type of definition will be evident from the JARGON statements in figures to follow. By the way, parenthesized expressions in JARGON statements are meant to be suggestive, and are not literal JARGON.

[3] In the implementation described in this report, this Concept is called **ANYTHING**.

THING is depicted in Figure 1. The Concept itself is represented by the ellipse labeled with its name; the Role subpart is represented by the encircled square. As is evident here, Roles themselves have structure, including descriptions of potential fillers,[4] modality information, and names.[5] In Figure 1 we see that the Role named **subpart** has a Value Restriction link back to the Concept THING. This says that potential subparts of THINGs must themselves be THINGs (here a vacuous statement, since there is no other possibility). The Modality of "Obligatory" means that the subpart Role is part of the definition of THING.[6]

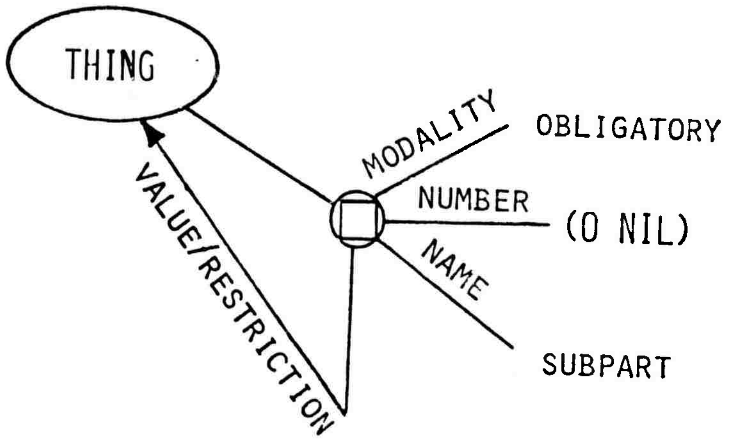
There are two Basic kinds of Roles in KL-ONE: **RoleSets** and IRoles. RoleSets capture the notion that a given functional role of a Concept (e.g., an upright of an arch, an officer of a company) can be filled in the same instance by several different entities. On a Generic Concept, a RoleSet generically describes

---

[4] These limitations on the form of particular fillers are called "Value Restrictions" (V/R´s). If more than one V/R is applicable at a given Role, the restrictions are taken conjunctively.

[5] Names are not used by the system in any way. They are merely conveniences for the user.

[6] A Modality of "Inherent" means that having the Role follows from the definition of the Concept, but is not part of it. "Optional" is the weakest Modality and simply allows the contingent possibility of the Role´s being associated with the Concept.



"A THING HAS PL SUBPART
(WHICH ARE PL THING)"

Fig. 1. The most general Concept, THING.

sets of individual intensions determined by that Role (e.g., the set of intensions "officer of a company"). Each instance of the Concept will then have its own corresponding set of intensional elements, each describing the binding of a given filler into the functional role (e.g., "the first-officer of the Enterprise"). IRoles are the KL-ONE structures representing these individual bindings.

Since functional roles defined by RoleSets can have multiple fillers, RoleSets have **Number Restrictions** to express cardinality information. At the moment, a Number Restriction is a pair of numbers (or NIL) defining the range of cardinalities for sets of role-player descriptions (NIL means "don't care"). Thus the Number Restriction in Figure 1 indicates that any THING can have arbitrarily many **subparts**.[7] The other facets on the Generic RoleSet description are applicable to every single **subpart** (e.g., each will have the functional role name **subpart** attributable to it, and each subpart must satisfy the Value Restriction).

A RoleSet on an Individual Concept stands for the set of individual intensions for that Concept only - it is not a generic description (e.g., the particular set of officers of a particular company). IRoles (for 'Instance Roles') appear only on Individual Concepts, and are used to represent particular bindings of Roles to Individual Concepts (e.g., the **president** of a particular COMPANY). (N.B. There would be one IRole for each officer position in a particular company, regardless of the actual number of people playing those Roles.)

Roles in general are thought of as reified intensional entities, which capture the bindings of individuals as functional

[7] This should literally be "any THING-description can have arbitrarily many subpart-descriptions."

role-players with respect to other individuals.[8] It should be emphasized that KL-ONE Roles are different from the fillers themselves. One person, for example, could conceivably be both the President and the Vice-President of a single institution. In KL-ONE, we would have two IRoles in an individual description of this company, one a place to store and access facts about the person (in his role) as the President, the other to store and access facts about him (in his role as) the Vice-President. Thus, the cardinality restriction mentioned above refers to the number of such IRoles, not the number of fillers.[9] Figure 2 illustrates both this case and another motivation for the reification of Roles. Contrast the intended referent of the word "it" in sentences a and b:

- The alligator's tail fell off.
  - a. It lay wriggling on the ground.
  - b. It will grow back again.

The "tails" that the pronouns are attempting to refer to are different in the two cases. In the first, the Role name is being used to refer to the previous filler. In the second, since it is

---

[8] Or, in more implementational terms, Roles are structures that express the binding of "slot-fillers" into the slots they fill.

[9] We intend to augment this with an additional restriction on the number of actual non-identical fillers.

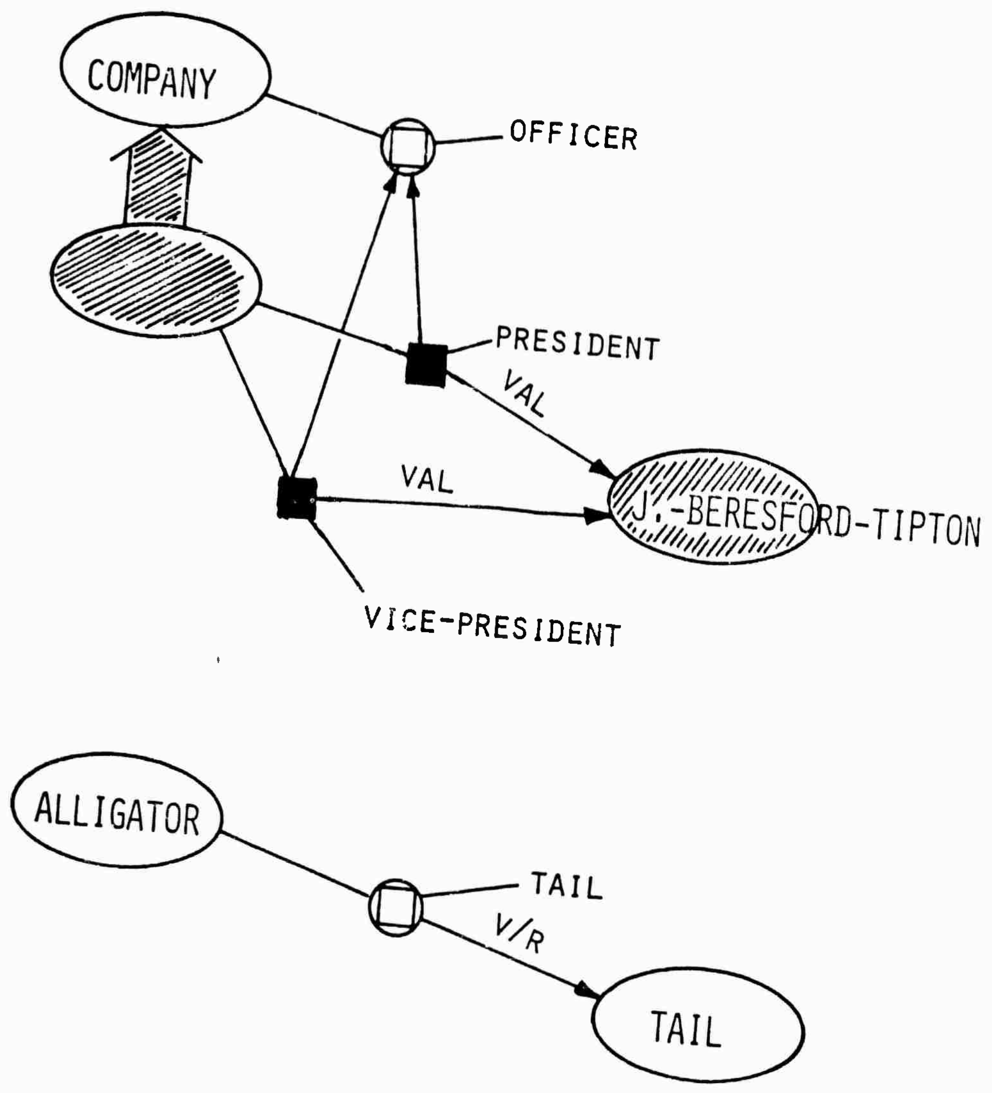


Fig. 2. A Role is not the same as a Role-player.

not the previous tail that will grow back, the name must be referring to something more abstract. We take the intent of this reference to be something like a KL-ONE Role.

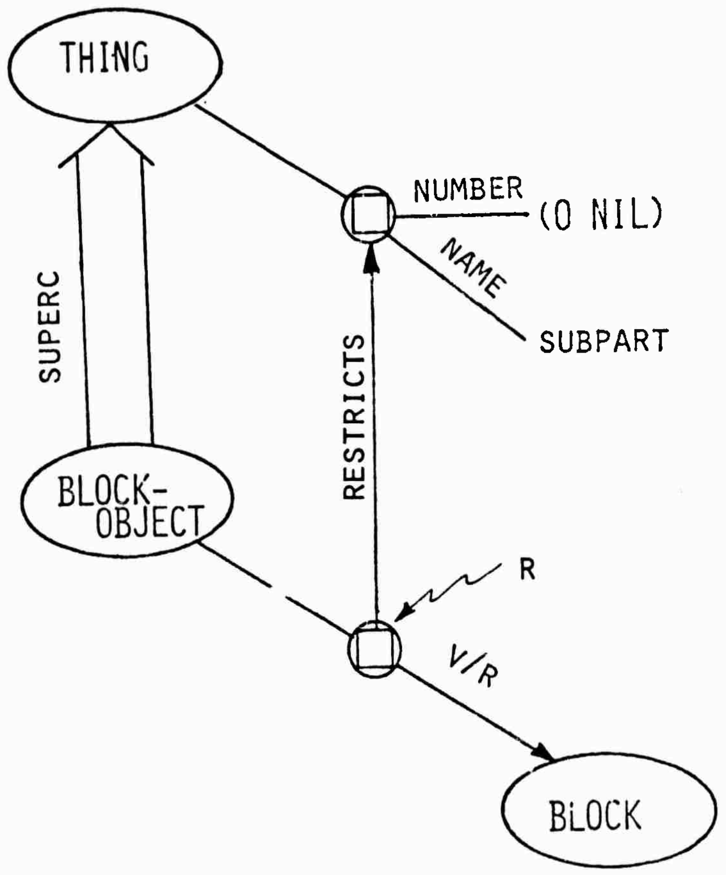
2.3 Derivative Definition

As mentioned above, definitional generic knowledge in KL-ONE is expressed in a taxonomic inheritance structure. The backbone of such a network is formed by inter-Concept inheritance Cables which pass structured definitions from Concepts to their subConcepts. The inheritance Cable is the primary description-formation operator of KL-ONE. It specifies how the meaning of a Concept is to be determined from the meaning of its superConcepts.

Figure 3 illustrates how to define a simple Concept from one we have already defined. The Cable, passing the meaning of THING to the lower Concept, is depicted as a double-shafted arrow. The lower Concept inherits that meaning, and constructs a more specific meaning from it by **restricting** the **subpart** Role - in this case, to have V/R BLOCK (which is, for now, an undefined type of THING). The **Restricts** (or **Modifies** in some places in this text) link[10] from the Role indicated "r" to the **subpart** role of THING indicates that the fillers of the **subpart** Role of the lower Concept are restricted to be BLOCKs. Role r **is** the **subpart** Role for the lower Concept, and thus inherits all of its meaning

[10] The link in actuality is closely associated with the SuperC Cable. It can be thought of as going "through" the Cable.

from that superRole. The new V/R is taken conjunctively with the old one, and the Name, Modality, and Number are inherited intact. As a result of the Cable (and its component Restricts link), the lower Concept is "a thing whose subparts are blocks".



"A BLOCK-OBJECT IS A THING WHOSE PL SUBPART ARE PL BLOCK"

Fig. 3. Basic description formation.

We can call this lower Concept "BLOCK-OBJECT", and a JARGON statement we might have made to create it is "A BLOCK-OBJECT IS A

THING WHOSE PL[11] SUBPART are PL BLOCK". This statement would define the new general term, BLOCK-OBJECT. Note that BLOCK-OBJECT is not some observationally determined class, but merely a new term in the language, defined to be nothing more than "A THING WHOSE PL SUBPART ARE PL BLOCK". Thus, the SuperC Cable between BLOCK-OBJECT and THING serves as a use of the latter in the definition of the former. Now, because BLOCK-OBJECT is defined in terms of THING, it must be true that (in any possible world) any BLOCK-OBJECT is a THING. However, the SuperC Cable means more than just the assertion of the subset relation between the two classes. It has as one component of its meaning that relation, but it goes on to say that it holds by virtue of meaning.[12]

2.3.1 Taxonomy and the MSS Algorithm

Since "BLOCK-OBJECT" means "A THING WHOSE PL SUBPART ARE PL BLOCK", any new term derived from BLOCK-OBJECT will of necessity carry "THING" as part of its meaning. By the same token, any

[11] PL is the JARGON morpheme for expressing the plural form of a noun. Read "SUBPARTS" for "PL SUBPART".

[12] We earlier stated that descriptive structure generally has no assertional import. The SuperC Cable, however, has a very strong one: because a BLOCK-OBJECT is a THING by virtue of meaning, then in any possible world the set of BLOCK-OBJECTS will be a subset of the set of THINGS.

entity with "a thing whose subparts are blocks" as part of its description will necessarily be a BLOCK-OBJECT. The KL-ONE system enforces this subsumption of Concepts by guaranteeing that a Concept entered in the network will be placed below all other Concepts that definitionally subsume it and above all Concepts that it subsumes.[13] This is one of the features of KL-ONE that makes it unique among current representation languages - the interpreter in a sense "understands" the Concept-formation language, and keeps all Concepts in a strict subsumption taxonomy based on their internal structures.

The algorithm that finds the most specific subsumers (MSS) of a Concept is documented in [Woods, 1979a] and will not be discussed further here. As we shall see in the description of the PSI-KLONE interface (Chapter 4), the taxonomic properties of KL-ONE networks plays a critical part in our natural language system.

2.4 Inter-Role Relations

The "Restricts" link of Figure 3 is only one of four types of KL-ONE links for expressing inter-Role inheritance

---

[13] In our implementation, this is currently only enforced if the JARGON interface is used.

relationships. Such relationships bear the brunt of the "structured inheritance" carried by SuperC Cables.

The four types of relationship are summarized here:

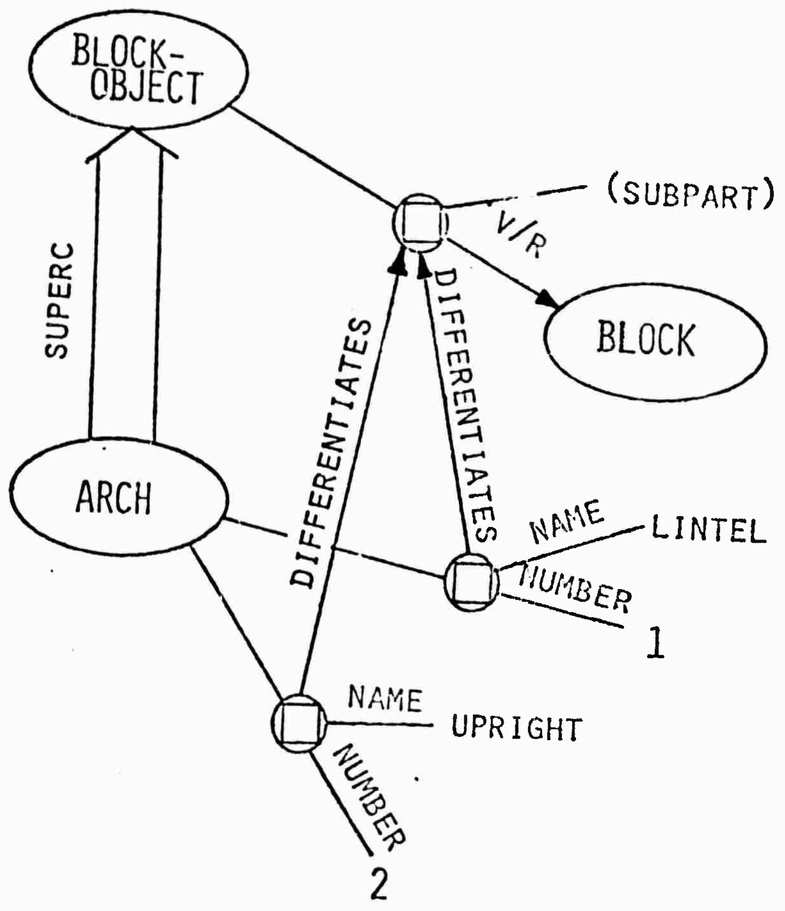
- restriction (of filler description and/or number); e.g., that a particular kind of COMPANY will have exactly three officers, all of whom must be over 45.
- differentiation (of a Role into subRoles); e.g., differentiating the officers of a COMPANY into president, vice-president, etc. This is a relationship between RoleSets in which the more specific Roles inherit all properties of the parent Role except for the Number Restriction (since that applies to the set and not the fillers);
- particularization (of a RoleSet for an Individual Concept); e.g., the officers of BBN are all COLLEGE-GRADUATEs; this is the relationship between a RoleSet of an Individual Concept and a RoleSet of a parent Generic Concept.
- satisfaction; this is the relationship between an IRole and its parent RoleSet.

Role differentiation is one of KL-ONE's unique features; Figure 4 illustrates its use. Since RoleSets have an associated cardinality, they can be divided into "sub-Role Sets". In the figure, we define the Concept of an ARCH as a BLOCK-OBJECT, one of whose subparts is its lintel, and two of whose subparts are its uprights. Since each of these Roles is also describable in the more general terms of its superRole, they each inherit all of the structure of that Role save for the Number Restriction. We intend to generalize the differentiation mechanism to allow multiple partitions of a RoleSet - at the moment we allow only one.

It should be noted that differentiation of a Role can occur within a single Concept. RoleSets are inherited through SuperC Cables, so an equivalent alternative to the previous definition of ARCH is that expressed in Figure 5. The Restricts link says that the RoleSet local to ARCH is the very same one as its superRole, with any modifications added on. Here there are none, so this is simply making explicit a structure that was for all intents and purposes already there. As far as KL-ONE is concerned, Role R is "as good as there" in Figure 4.

Also, we note that subRoles can themselves be modified (always) or differentiated (as long as their cardinality is greater than one).

Figure 6 illustrates the use of Cables and the structure of Concepts in a multiple level taxonomy. This figure includes some Concepts that contribute to the description of an ATN grammar. The most general Concept, ATN-CONSTITUENT, has two subConcepts - STATE and ARC. These each inherit the general properties of ATN constituents, namely, each is known to have a displayForm associated with it. The subnetwork below ARC expresses the classification of the various types of arcs in the ATN and how their conceptual structures vary. For example, a CONNECTING-ARC



"AN ARCH IS A BLOCK-OBJECT WHICH HAS A LINTEL
AND 2 UPRIGHT"

"THE LINTEL OF AN ARCH IS ONE OF ITS PL SUBPART
(AS A BLOCK-OBJECT)"

"THE UPRIGHT OF AN ARCH ARE SOME OF ITS PL SUBPART"

Fig. 4. Differertiation.

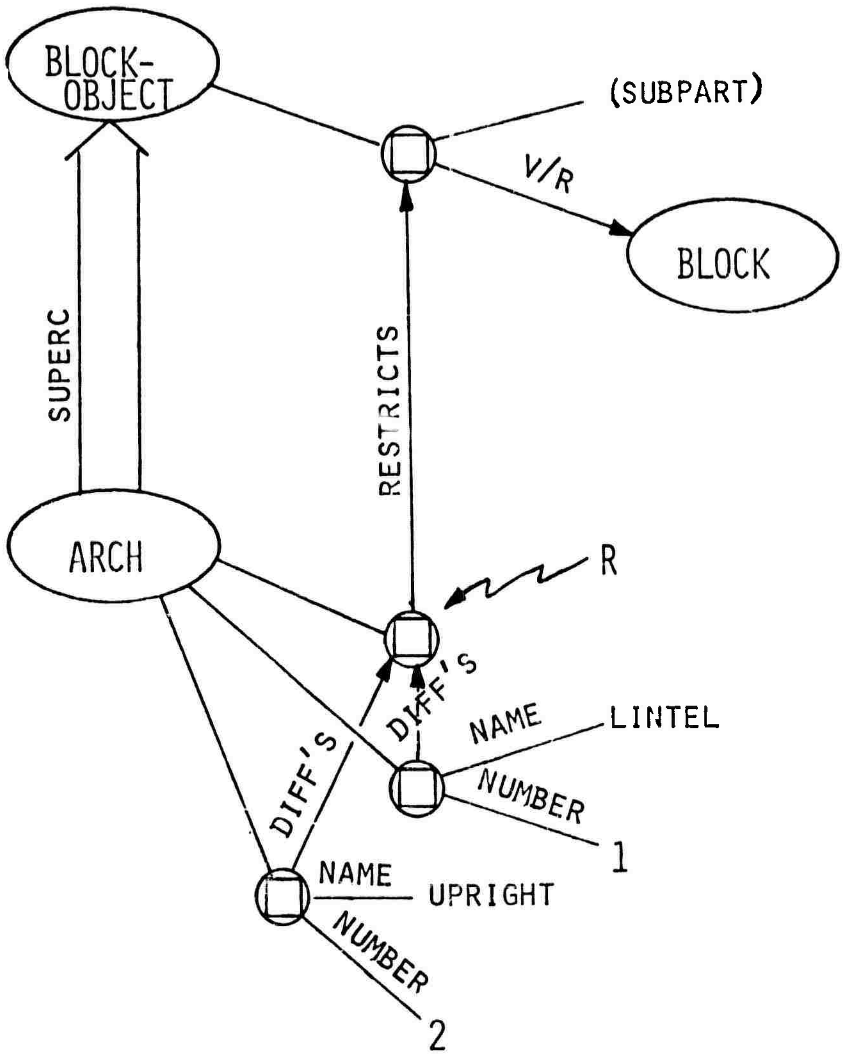


Fig. 5. Internal differentiation.

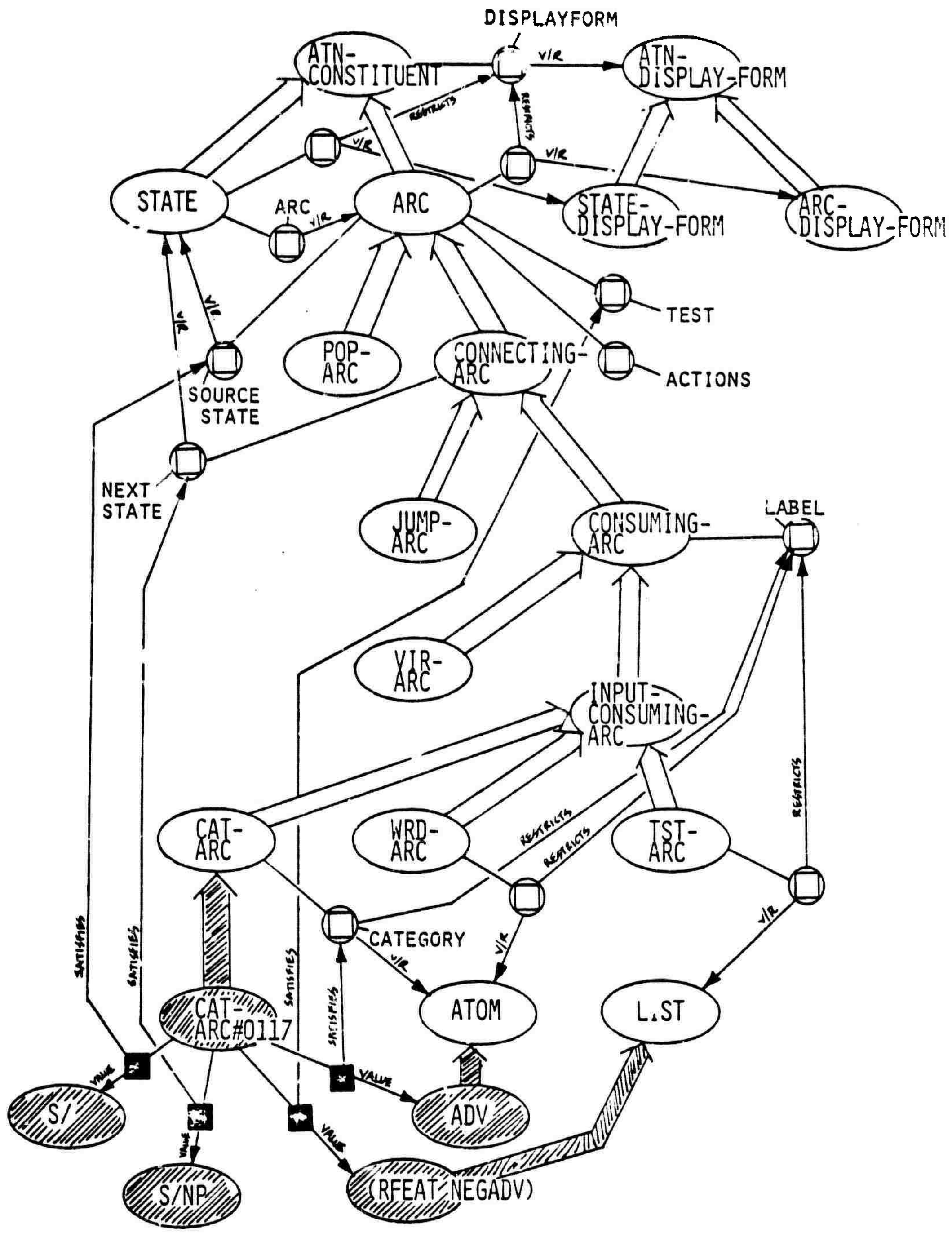


Fig. 6. A piece of a KL-ONE taxonomy.

has a **nextState** (the state in which the transition leaves the parsing process), while for POP-ARCs the term is not meaningful (i.e., there is no **nextState** Role). Links that connect the Roles of more specific Concepts with corresponding Roles in their parent Concepts are considered to travel through the appropriate Cables. Finally, the structure of an Individual Concept is illustrated by CATARC#0117. Each IRole expresses the filling of a Role inherited from the hierarchy above - because CATARC#0117 is a CAT-ARC, it has a category; because it is also a CONNECTING-ARC, it has a **nextState**, etc.

Finally, KL-ONE Concepts can have more than one superConcept. The inherited definitions are taken conjunctively.

2.5 Structural Descriptions

While KL-ONE Roles are assigned local "names", and inherit them from superRoles as well, these are meaningless strings as far as the system is concerned. Thus, in the structure as so far described, nothing gives the Role its meaning as a functional role description. In order to provide for the explicit representation of the roles that Role-fillers play, we complete the structure of a KL-ONE Concept with a set of **Structural Descriptions** (SD's). SD's express how the Roles of the Concept interrelate and how they relate to the Concept as a whole via the use of parameterized versions ("ParaIndividuals") of other Concepts in the network.

SD´s are actually special versions of a larger class of KL-ONE constructs, which we call **RoleSet Relations** (RSR´s). RSR´s express quantified relationships among RoleSets (not necessarily of the same Concept) as set mappings. Just as the Number Restriction discussed earlier deals with the cardinality of sets of IRoles, and not fillers, RSR´s are intended to describe mappings among IRoles. They do this by specifying the mappings at the Generic Concept level. SD´s are the subclass of RSR´s that map over the Roles of a single Concept. They are inherited through Cables and are restricted and particularized in a manner similar to that of Roles.

In addition to a very general type of RoleSet Relation, there is a special kind of structural relationship in KL-ONE called a RoleValueMap (RVM). This type of structure expresses a simple relationship between two sets of Role-fillers - either identity or inclusion. An RVM can equate the sets of fillers (N.B. not IRoles) of two Roles of the same Concept, or a Role of an Individual Concept with a Role of another Concept.[14] Figure 7

[14] If the other Concept is Individual, then the two particular sets of fillers are identical. If it is Generic, then all instances of that Generic satisfy the relationship.

illustrates the RVM structure. Imagine that we have augmented our ARCH description with the following:

1. AN ARCH HAS A NAME WHICH IS A STRING
2. AN ARCH MUST HAVE A DEDICATEE WHICH IS A PERSON
3. A PERSON MUST HAVE A NAME WHICH IS A NAME
4. A NAME MUST HAVE A FIRST WHICH IS A STRING AND HAVE A LAST WHICH IS A STRING.

These JARGON sentences define the non-dashed structure in the figure. Now, suppose we wanted to give some further indication of the meaning of an ARCH's name; in particular, that it is the same as the last name of the person for whom the arch is dedicated. The dashed structure in Figure 7 shows the appropriate RoleValueMap (the diamond). The RVM has two pointers: x, to the Role name of ARCH, indicating "THE NAME OF AN ARCH"; and y, a sequential pointer, indicating "THE LAST OF THE NAME OF THE DEDICATEE OF AN ARCH". The RVM is hung off of ARCH, since it is one of the Structural Descriptions of that Concept. Note that if any of the Roles in the y chain had potentially multiple fillers, that chain would "evaluate" in an instance to the complete set of STRINGs obtained by iterating over all dedicatees and all of their names, and all of their lasts.

Because the RoleValueMap in Figure 7 occurs within a single Concept, it means that each instance of ARCH satisfies the

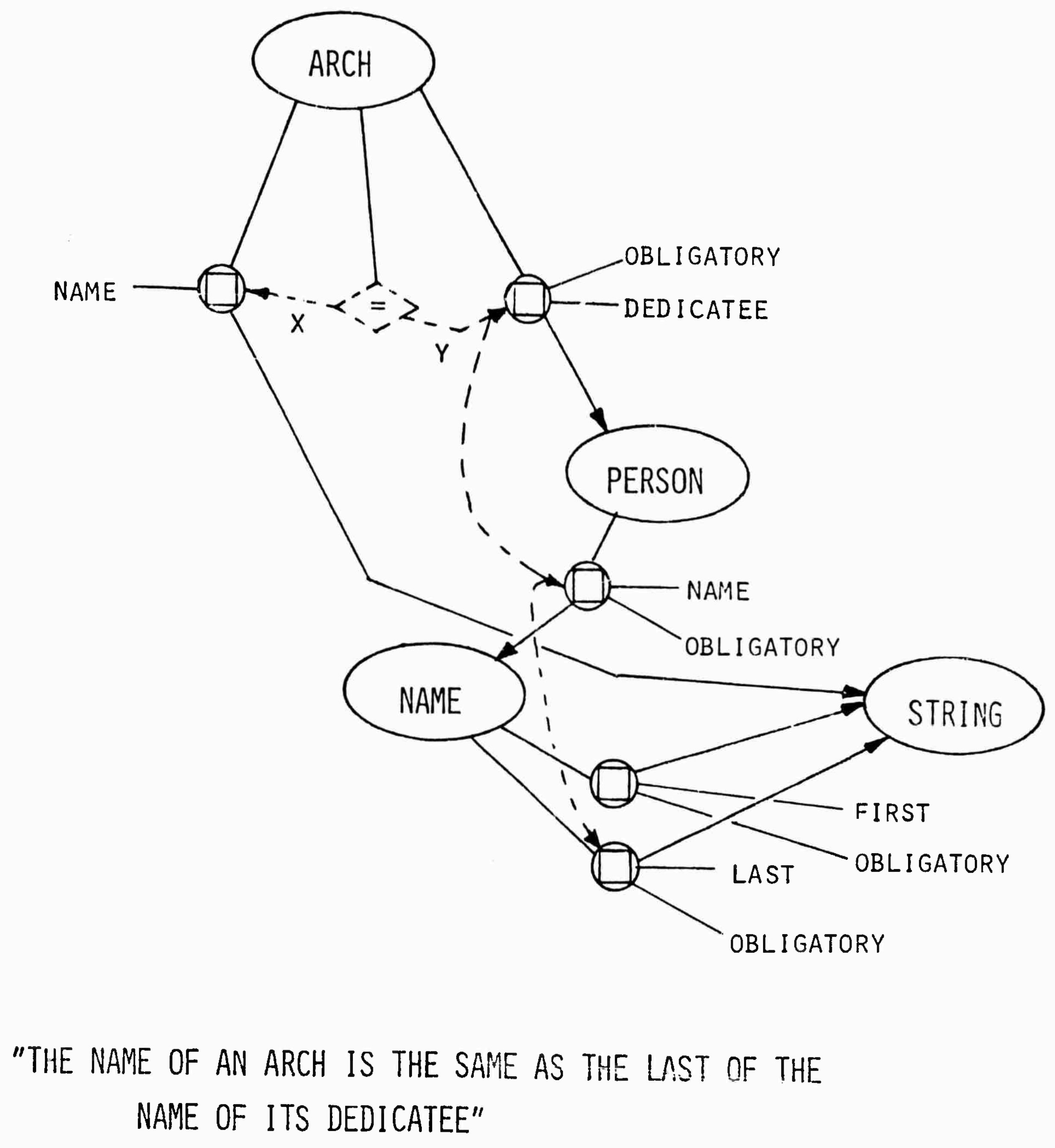


Fig. 7. A RoleValueMap.

generic relationship defined therein. That is, the set of

STRINGs obtained by retrieving the **names** of a particular ARCH is the same as the set retrieved as the **lasts** of the **names** of the dedicatees of the very same arch. In the kind of RoleSet Relation that uses parametric Concepts, a pointer to the enclosing Concept can be used to express the participation of the instance's "self" - that is, the thing as a whole - in a relationship. For example, the Concept of a HUSBAND would have as part of its definition a RoleSet Relation describing a MARRIAGE in which the **male-spouse** Role was to be filled by the HUSBAND itself. For each instance of HUSBAND, there would have to exist a MARRIAGE description whose **male-spouse** was that instance of HUSBAND (and not some other HUSBAND).

One final note on the RVM is needed to motivate the use of a chained pointer. If the RVM were to point directly to the ultimate Role in the y chain of Figure 7, that pointer to the last Role of NAME would happen to be unproblematic. However, if we had chosen to make the V/R of the **name** Role of ARCH be NAME as well, then the direct pointer would fail to disambiguate between the **last** of the name of the ARCH itself and the last of the name of its **dedicatee**. Thus, the chained pointer that starts at a Role of the enclosing Concept is necessary. Reading from the RVM out, the y pointer might be read as the "DEDICATEE'S NAME'S LAST", illustrating the prominent position of the **dedicatee** Role.

2.6 Contexts and Nexuses

As mentioned earlier, the KL-ONE description language has a complementary assertion language. We have tried carefully to distinguish between purely descriptional structure and assertions about coreference, existence, etc. All of the structure mentioned above (Concepts, Roles, and Cables) is **definitional.** All assertions are made relative to a **Context** and thus do not affect the (descriptive) taxonomy of generic knowledge. We anticipate that Contexts will be of use in reasoning about hypotheticals, beliefs, and wants.

Contexts are collections of structureless entities called Nexuses, which serve as loci of coreference statements. A Nexus is a simple object that holds together "wires" from various descriptions, all of which are taken to specify the same object in the world outside the system. The description wires that connect Nexuses to Concepts in the description language are also taken to be in the Context. Thus, a Context can act as a "possible world", which comprises a set of statements about description coreference[15].

---

[15] Co-"reference" is not quite the right term, since the objects "referred to" need not exist. **Co-specification of description** is probably a better term (see [Sidner, 1979])

2.7 Meta-description

Nexuses allow us to come as close to real "reference" to objects outside the system as is possible in this kind of representation environment. In addition to the use of Nexuses as "surrogates" for outside entities, KL-ONE allows reference to internal entities (e.g., Concepts) as well. Thus one can "meta-describe" a KL-ONE object in KL-ONE. Of course, to do this, the system needs to have the Concepts of a KL-ONE Concept, a KL-ONE Role, a KL-ONE RoleValueMap, etc.

In order to construct a meta-description one uses the same type of structure used in constructing a regular description. Each KL-ONE structure is considered implicitly to have a corresponding Nexus that is known to exist in the "KL-ONE base level" Context.[16] Meta-descriptions are simply descriptions (usually expressed in terms of the Concepts KL-ONE-CONCEPT, KL-ONE-ROLE, etc.) attached to those Nexuses by means of the description wire mechanism mentioned above. We intend to have the system eventually "understand" meta-descriptions so that one can influence the behavior of the KL-ONE interpreter with appropriately expressed KL-ONE meta-descriptions of actions and KL-ONE structures.

---

[16] We have on occasion called these Nexuses "meta-anchors".

2.8 Attached Procedures and Data

The final feature of KL-ONE relevant to our discussion is the ability to attach procedures and data to structures in the network.

The attached procedure mechanism is implemented in a very general way. Procedures are attached to KL-ONE entities by "interpretive hooks" (ihooks) (see [Smith, 1978]), which specify the set of situations in which they are to be triggered. An interpreter function operating on a KL-ONE entity causes the invocation of all procedures inherited by or directly attached to that entity by ihooks whose situations match the intent of that function. Situations include things like "Individuate", "Modify", "Create", "Remove", etc. In addition to a general situation, an ihook specifies when in the execution of the interpreter function it is to be invoked (PRE-, POST-, or WHEN-).

As mentioned previously, KL-ONE is used in several places in our language understanding system - these include the syntactic taxonomy used to constrain parsing and to index semantic interpretation rules, and the structures used in the syntactic/discourse interface to express the literal semantic content of an utterance. The parser uses KL-ONE to describe those syntactically correct structures for which there are known interpretation rules. Interpretation per se is achieved using attached procedures and data, with semantic projection rules expressed as data attached to Roles of the syntactic Concepts. Procedures attached to Roles near the top of the syntactic taxonomy specify how to use the projection rules to map syntactic structures into semantic Concepts and Roles.

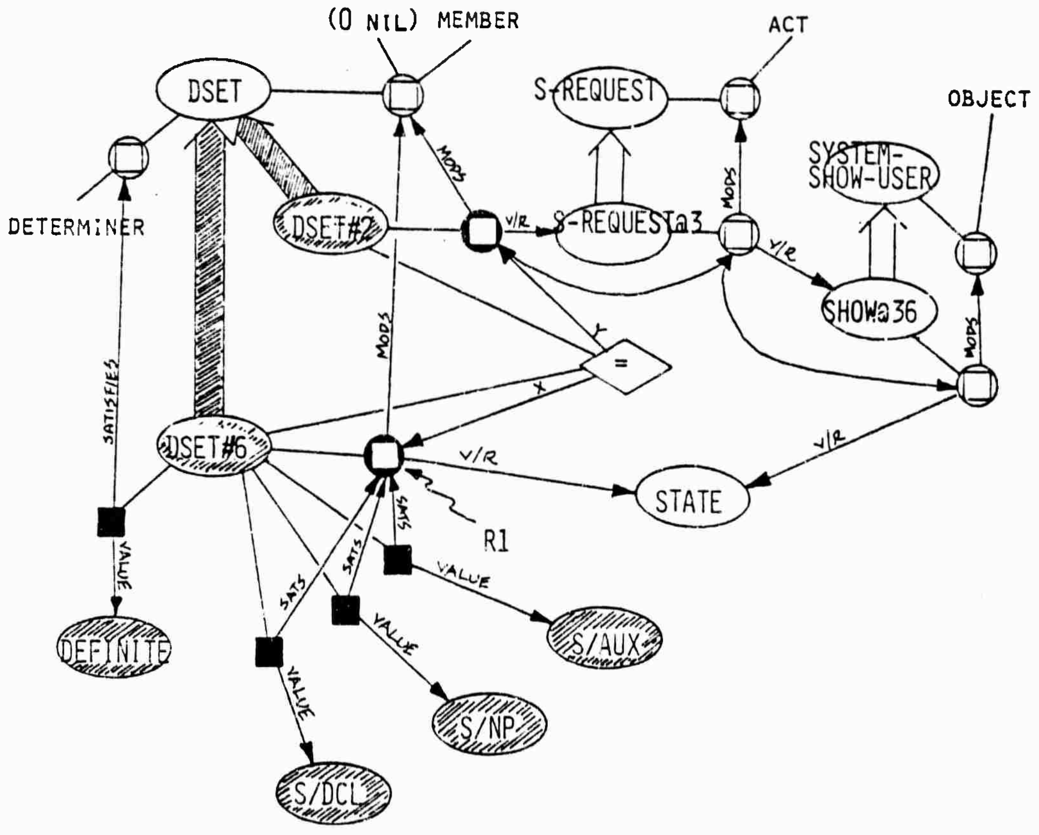
2.9 DSETs

As mentioned earlier, the literal semantic interpretation of a phrase produced by semantic interpretation is a KL-ONE structure "input" to the discourse component. An important element of this interface between the syntactic processor and the discourse component is that the parser/interpreter commits itself only to information explicitly present in the input phrase, and leaves all inference about quantifier scope, etc. to the discourse expert (see Chapter 5). Two kinds of representational structures support this. The particular Generic Concept DSET (for "determined set") is used extensively to capture sets implicit in noun phrases and clauses. DSETs use the inherent multiplicity of RoleSets to group together several entities under a single Concept, and to associate determiners (definite/indefinite, quantifiers, etc.) with such a set of entities. The former is accomplished using a single **member** RoleSet whose multiplicity is open-ended (between 0 and infinity); the latter is achieved by simply having a **determiner**

RoleSet whose number is restricted to be 1. A DSET can express the characteristics of a set of entities without enumerating them explicitly, or even indicating how many members the set is expected to have. We use RoleValueMaps to allow constraints between DSETs to be expressed in a general way. Such relations can be constructed without knowing in advance the cardinality of the sets or any of their members.

Figure 8 illustrates the use of these structures to express the literal semantic content of the sentence, "Show me states S/NP, S/AUX, and S/DCL." DSET#6 represents the interpretation of the noun phrase, "the states S/NP, S/AUX, and S/DCL". The generic DSET Concept has two Roles, member and determiner. The member Role can be filled multiply, and therein lies the "settedness" of the DSET. DSET#6 has a particularized version of the member Role: Role R1 represents the set of three states mentioned in the noun phrase, as a group. Thus, the Value Restriction of R1, STATE, applies to each member. The three IRoles of DSET#6, connected by "Satisfies" links to the particularized member RoleSet (R1), indicate that the particular states are the members of the set[17].

[17] The Value Restriction, STATE, is redundant here, since the members of this particular set were explicitly specified (and are known to be states). In other cases, the information is more useful. For example, no IRoles would be constructed by the parser if the sentence were "Are there three states?"; only one would be constructed in "Show me state S/NP and its two nearest neighbors". On the other hand, no Value Restriction would be directly present on Role R1 if the noun phrase were just "S/NP, S/AUX, and S/DCL".



"THE MEMBER OF DSET#6 ARE THE SAME AS THE OBJECT OF THE ACT OF THE MEMBER OF DSET#2"

Fig. 8. KL-ONE description of "Show me states S/NP, S/AUX, and S/DCL".

The other DSET in the figure, DSET#2, represents the clause-level structure of the sentence. The clause has been interpreted into something like "the user has performed what looks on the surface to be a request for the system to perform some number of showings to the user of some set of states". The

Concept S-REQUEST stands for "surface-request" and is considered to be the interpretation of the surface speech act of the sentence (see Chapter 5).

This captures several kinds of indeterminacy: (1) that the sentence may only be a request at the surface level ("Don´t you know that pigs can´t fly?" looks like a request to inform [a question]), (2) that there is more than one way to effect a "show" ("show" could mean redraw the entire display, change it slightly to include a new object, or simply highlight an existing one), and (3) that it is not clear how many operations are actually being requested (showing three objects could take one, two, or three actions). Therefore, the interpretation uses Generic Concepts to describe the kind of events appearing in the surface form of the sentence and makes no commitment to the number of them requested. The only commitment to "quantificational" information is expressed by the RoleValueMap. Its two pointers, X (pointing to the member Role of DSET#6) and Y[18] (pointing to the object of the requested act), indicate that

[18] Y is a chained pointer going first through the member Role of DSET#2, then through the act Role of S-REQUEST@3, and finally to the object Role of SHOW@36. It is considered to refer to the set of IRoles expressing the objects of all SHOW events ultimately S-REQUESTed, when it is determined exactly how many there are to be (i.e., when the IRoles of DSET#2 are finally specified). Thus, if there are ultimately two SHOWs, one of one state and the other of two, the Y pointer implicitly refers to the set of all three states shown.

the ultimate set of things to be shown, no matter how many particular SHOW events take place, must be the same as the set of members in the noun phrase DSET (namely, the three states).

3. INTRODUCTION TO THE NATURAL LANGUAGE SYSTEM

3.1 Natural Language and AIPS

In order to explore the knowledge representation and language understanding issues involved in the command and control graphics display context, we have implemented an experimental system that completes the cycle from user input in natural English to the generation of an image on a two-dimensional display.

As the component of our system that manipulates the display we have taken an existing system - the **Advanced Information Presentation System** (AIPS [Greenfeld and Yonke, 1979; Zdybel, Yonke, and Greenfeld, 1980]). AIPS has been built on top of KL-ONE; with it, one can represent explicitly all objects (ships, etc.) to be presented, their presentation forms (circles, text, etc.), descriptions of view surfaces on which to project presentations of the objects, and coordinate mappings between those surfaces. This explicit representation allows the AIPS user to flexibly alter at will the picture s/he sees by adding or moving display windows, changing size, shape, etc. of display forms, and adding and removing objects or object detail. The user changes the subject and form of what s/he sees by **describing** (in KL-ONE) what s/he wants displayed.

To a large extent, our overall task can be viewed as an attempt to provide a sophisticated natural language interface to something like the AIPS system. The addition of a natural language interface to AIPS yields more than just a convenient way to state explicit display changes. Now the display can be altered in response to a question (e.g., highlighting a ship to mean "there!" in response to a "where" question), or to an indirect speech act (e.g., "I want to see it" produces a display of the appropriate object). Further, natural language provides a convenient way to express standing orders of various types (e.g., "Display ships with radar as flashing triangles"; "whenever three ships are in the same convoy, and within 6 miles of each other, use a single task force symbol to stand for the set of ships").

3.2 The ATN Domain

For the sake of experimental convenience (so that the system designers can serve as genuine users of the system), we have replaced the geographical maps of the commander with an isomorphic domain consisting of an Augmented Transition Network (ATN) grammar [Woods, 1970] laid out spatially on a surface. In particular, we have taken as our domain of discourse the ATN grammar from the LUNAR natural language understanding system [Woods, Kaplan, and Nash-Webber, 1972]. Thus, instead of ships and land masses, the objects to be displayed are the states and arcs of the ATN, including state names, arc types, conditions, actions, etc.

The user of this system can make requests for portions of the grammar to be displayed, for the window to be zoomed in or out, for specified states or arcs to be made visible or invisible, and s/he can ask questions for details, some of which involve coordination of linguistic requests with pointing actions to objects on the display. The particular display setup in our implementation has three windows - for prompts, text interaction, and grammar display. At the moment, the size and placement of these windows is fixed; but these could be easily changed using the AIPS facility.

3.3 A Sample Dialogue

A simple dialogue will serve to show the blend of natural language and intelligent knowledge-based graphics that we envision in the command and control environment (note the use of user-pointing input as well as language). Figure 9, parts (a) through (d), illustrates the response of our system to each of the first four sentences:

1. Show me the clause level network.

   [System displays states and arcs of the S/ network] (Figure 9a)

2. Show me S/NP.

   [System highlights state S/NP] (Figure 9b)

3. Focus in on the preverbal constituents.

   [System shifts scale and centers the display on the preverbal states] (Figure 9c)

4. No. I want to be able to see S/AUX.

   [System "backs off" display so as to include state S/AUX] (Figure 9d)

5. Remove the highlight from this <user points to S/NP> state.

   [System removes highlight from S/NP]

At the same time, we would like to ask factual questions about the states, arcs, etc. of the ATN (e.g., "What are the conditions on this <user points> arc?"). Questions and commands addressed to the system typically (1) make use of elements of the preceding dialogue, (2) can be expressed indirectly so that the surface form does not reflect the real intent, and (3) given our graphical presentation system, can make reference to a shared non-linguistic context. The issues of anaphora, (indirect) speech acts, and deixis are thus of principal concern.

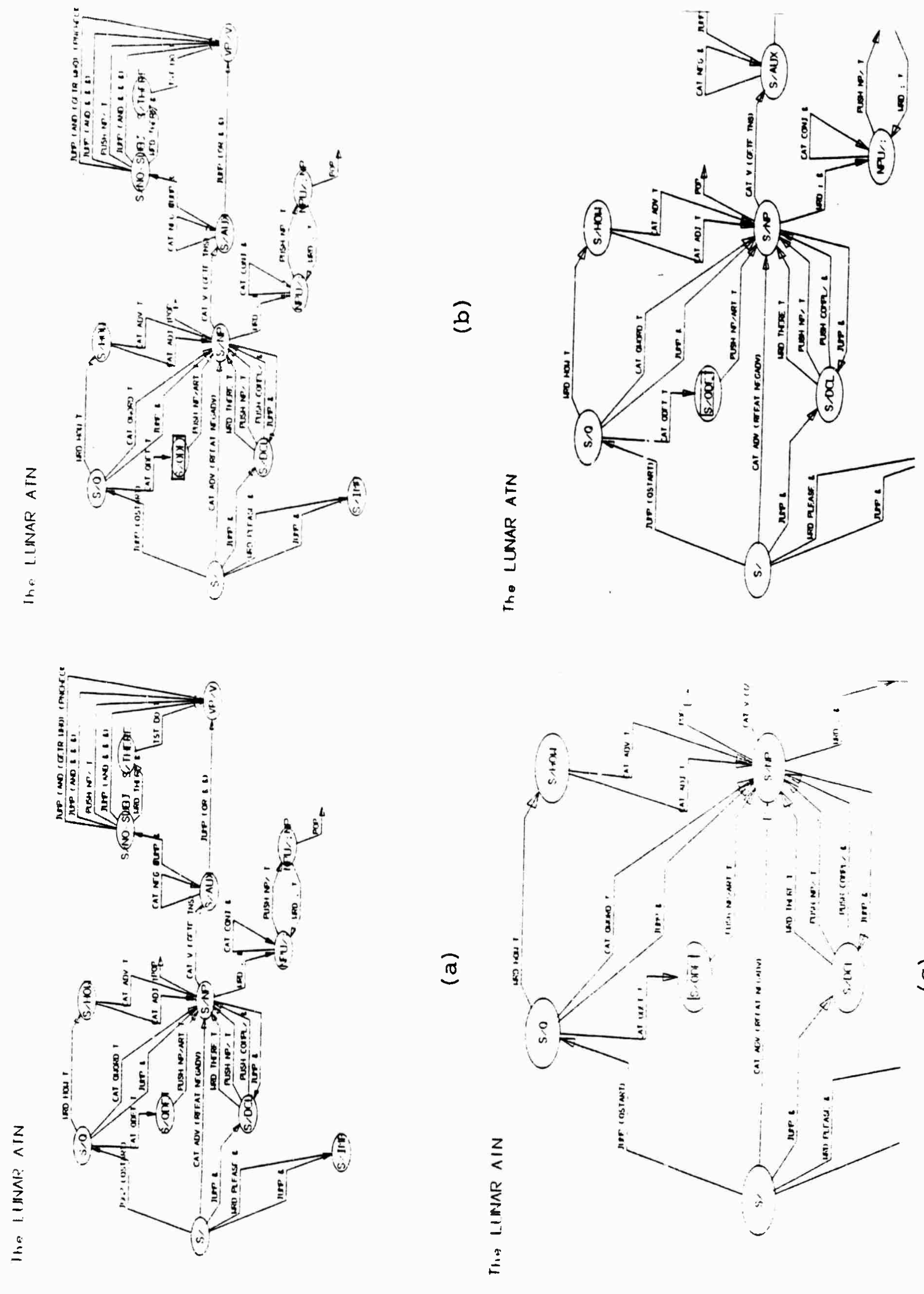


Fig. 9. Displays produced in sample dialogue.

3.4 System Structure

The natural language system is organized as illustrated in Figure 10. The user sits at a bit-map terminal equipped with a keyboard and a pointing device. Typed input from the keyboard (possibly interspersed with coordinates from the pointing device) is analyzed by a version of the **RUS Parsing System** - an ATN-based incremental parser that is closely coupled with a "case-frame dictionary". In our system, this dictionary is embodied in a syntactic taxonomy represented in KL-ONE. The KL-ONE knowledge representation system is in fact used at several points in this system - to represent a taxonomy of syntactic structures for organizing semantic interpretation rules, to maintain a taxonomy of speech acts for the speech act interpreter that determines user intent, and to represent the descriptions of the objects to be displayed and to organize the procedures that produce the display.

The parser produces a KL-ONE representation of the syntactic structure of an utterance. Incrementally along with its production, this syntactic structure triggers the creation of an interpretation. The interpretation structure - the literal (sentential) semantic content of the utterance - is then processed by a discourse expert that attempts to determine what was really meant. In this process, anaphoric expressions must be

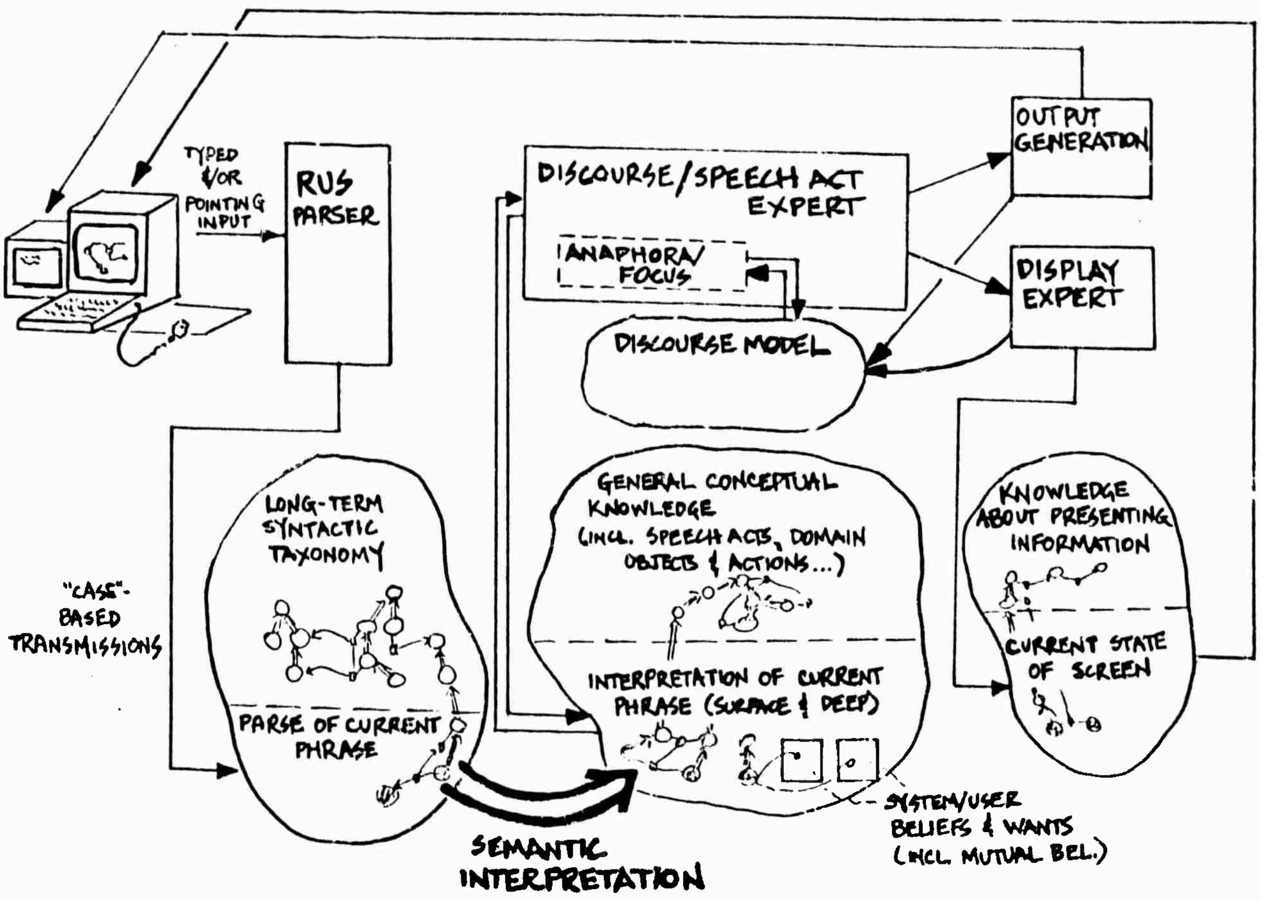

TYPED &/OR POINTING INPUT
RUS PARSER
DISCOURSE/SPEECH ACT EXPERT
ANAPHORA/ FOCUS
DISCOURSE MODEL
OUTPUT GENERATION
DISPLAY EXPERT
LONG-TERM SYNTACTIC TAXONOMY
PARSE OF CURRENT PHRASE
"CASE"-BASED TRANSMISSIONS
GENERAL CONCEPTUAL KNOWLEDGE (INCL. SPEECH ACTS, DOMAIN OBJECTS & ACTIONS...)
INTERPRETATION OF CURRENT PHRASE (SURFACE & DEEP)
KNOWLEDGE ABOUT PRESENTING INFORMATION
CURRENT STATE OF SCREEN
SYSTEM/USER BELIEFS & WANTS (INCL. MUTUAL BEL.)
SEMANTIC INTERPRETATION

Fig. 10. System structure
(highlighting types of knowledge involved.)

resolved and indirect speech acts recognized. Finally, on the basis of what is determined to be the intended force of the utterance, the discourse component decides how the system should respond. It plans its own speech or display actions, and passes them off to the language generation component (not yet implemented) or display expert (built on top of AIPS).

The various components of our system will now be discussed.

Each of the following chapters will point out how the corresponding component of the system makes use of explicit knowledge representation in its operation, as well as the sequence of activities it goes through as it parses, recognizes plans, or draws pictures.

4. THE PSI-KLONE INTERFACE

4.1 Introduction

This chapter[19] describes the syntactic and semantic processing components of the prototype natural language understanding system. There are several interesting features of this part of the system that we will highlight. The first is a framework for natural language parsing (called the **RUS Parser**) which combines the efficiency of a semantic grammar with the flexibility and extensibility of modular syntactic-semantic processing. The second (the **PSI-KLONE Interface**) comprises two descriptive taxonomies expressed in the KL-ONE formalism which represent, first, the system's knowledge of interpretable syntactic-semantic patterns, and, second, the system's semantic knowledge of possible objects, events and relationships. These taxonomies facilitate the two major tasks of the system's semantic processor:

1. providing feedback to the syntactic processor, and
2. providing semantic interpretations for individual phrases.

[19] This chapter is a revised version of a paper by Robert J. Bobrow and Bonnie L. Webber, to be presented at the third annual CSCSI/SCEIO Conference, Victoria, B.C., May 1980.

A third interesting feature of this part of the system will be touched upon only briefly - its treatment of natural language quantification in terms of a combinatoric problem to be solved, to whatever extent necessary, by a pragmatics/discourse component.[20]

Section 4.2 discusses the RUS parsing framework - first, the structure of its cascaded interactions with the semantic interpreter, then, techniques used to minimize backtracking in RUS. Section 4.3 discusses semantic interpretation in PSI-KLONE, with a detailed example of the dialogue that the parser and interpreter carry on in parsing a sentence and constructing the descriptive part of its semantic interpretation. Combinatoric aspects of a sentence's interpretation are discussed in the latter part of this section.

4.2 The RUS Natural Language Parsing Framework

4.2.1 Introduction

RUS is a framework for natural language processing that is as efficient as a semantic grammar, and as flexible and

[20] The pragmatics/discourse component of the natural language system is described in the next chapter.

extensible as a modular syntactic/semantic processor. It is based on a non-deterministic ATN parser, but it parses without backup in virtually all cases that Marcus's "deterministic parser" does [Marcus, 1978]. In addition, because of the ATN's ability to operate non-deterministically, RUS can handle phenomena not covered by Marcus' parser.

We have achieved this combination of efficiency and extensibility by **cascading** (see [Woods, 1980]) the syntactic and semantic processors - making calls to the semantic processor at significant points in the parsing process. The near-determinism results in part from two new arc-types - **GROUP arcs** and almost-GROUP arcs - and in part from a new control structure for ATNs.

The following two sections describe the features of the syntactic processor. Section 4.2.2 covers those features that are important for the cascaded interaction of syntax and semantics. Section 4.2.3 discusses the modifications to the grammar and the normal ATN control structure that increase the determinism of the parsing process.

4.2.2 Syntactic Labeling and Cascaded Interactions

4.2.2.1 Syntax and Functional Relations among Constituents

We view parsing as a mechanism for providing a functional description of the relations that hold among the pieces that form a syntactic unit (phrase). This description notes the phrase's constituent syntactic units, as well as labelling the functional relations that hold between the constituents and the phrase as a whole. These labels are based on a constituent's functional role in the higher phrase, and not simply on its internal syntactic structure. For example, a noun phrase (NP) can serve various functions in a clause, including logical subject (LSUBJ), logical object (LOBJ), logical indirect object (LINDOBJ), surface subject (SSUBJ), and first NP (FIRSTNP). It is up to the syntactic processor to determine which of these possible labels is appropriate for a given NP constituent of a clause.

These functional labels are primarily intended to provide information for semantic interpretation and discourse processing, and not to screen out ungrammatical constructions. Logical labels such as LSUBJ and LOBJ provide a coupling to the case relationships that are the basis of lexical semantics, while FIRSTNP helps determine discourse focus [Sidner, 1979] and SSUBJ constrains the use of a clause as the source of later verb phrase ellipsis [Webber, 1978].

4.2.2.2 Cascade Interaction between Syntax and Semantics

The parser does not interact with the semantic interpreter by sending it a complete syntactic analysis of a sentence labeled with the functional relations discussed in Section 4.2.2.1. Rather, the parser and interpreter engage in a dialogue consisting of a sequence of **transmissions** from syntax and responses from semantics.

An individual transmission consists of a **transmit triple**, which represents a proposal by syntax of the addition of (1) **a new constituent** with (2) **a label** indicating a particular functional relation to (3) **the phrase** currently under construction by both syntax and semantics. Semantics either rejects the proposal or returns a pointer to a data-structure which represents semantics' knowledge of the resulting phrase. These pointers are all that the RUS syntactic processor knows about the internal operation of the semantic component, and they are simply saved to act as part of the third component of later transmission triples. Thus the RUS framework has no commitment to any particular internal structure for semantic interpretations.

A transmission occurs as part of an arc action in the ATN, with the success of that arc depending on semantics' response to the transmission. The failure of an arc because of a semantic rejection is treated exactly like the failure of an arc because of a syntactic mismatch; alternative arcs on the source state are attempted, and if none are successful, a back-up occurs.

Transmit actions occur only when enough syntactic structure has been analyzed to confidently propose a functional label for the transmitted constituent. In particular, transmit actions are always postponed until after the head of the current phrase has been recognized. In a simple active sentence[21] like

"The three boys ate two pizzas"

the NP "The three boys" can be labeled as FIRSTNP immediately, and as SSUBJ and LSUBJ immediately after the head verb is recognized. In passive sentences like

"The dog was given a steak bone"
"The dog was given to the first boy who asked for it"

it is impossible to tell if the FIRSTNP "The dog" should be labeled LOBJ or LINDOBJ until the NP after the main verb is parsed.

Note that in this paradigm the parser does not **per se** produce a static syntactic structure. For any given path through the ATN the syntactic structure is implicitly represented in the sequence of transmissions, however, and a parse tree can easily be constructed from these transmissions.[22]

---

[21] That is, with the exception of sentences such as "John I like", or any active sentence in which topicalization or Y-movement has occurred.

[22] This is in fact the case in our natural language system. A parse tree of sorts is constructed in KL-ONE as a result of the transmissions from RUS.

Semantics' response to a transmission from syntax will be discussed in more detail in section 4.3.2.2. The important thing to note here is that this response is not necessarily the incremental interpretation of the phrase currently under construction. It may simply verify the existence of an interpretation (projection) rule (or rules) by means of which the interpretation of the phrase could be extended by the addition of the proposed new constituent. This buys efficiency by rejecting constructs that have no hope of semantic interpretation and not paying for the construction of a semantic interpretation until a phrase is syntactically checked.

4.2.3 Approaching Deterministic Parsing

The basic ATN is a non-deterministic parsing mechanism: when more than one arc leaves a state in the ATN, the parser must treat that state as a potential branch point. That is, the parser must select an arc to follow, and if its path from that arc becomes blocked, it must be prepared to back-up to previous branch points and try alternative arcs. A deterministic parser, on the other hand, must be able to treat a state with many arcs as a choice point, and make the correct choice of which arc to follow, without allowing for any back-up to that state.

By analyzing the back-ups that occurred in a typical non-deterministic ATN parser (i.e., an early version of the RUS system), we found them to have three major causes:

1. the existence of unnecessary branch points in the ATN,
2. the preponderance of "hypothesis-driven" (as opposed to "data-driven") characterizations of English grammar found in the ATN, and
3. the interaction of the normal depth-first control structure of the ATN with the capability for semantic rejection of constituents.

In a typical ATN there are many states that are not true non-deterministic branch points. That is, for any given sentence there is at most one acceptable arc from such a state. In those cases, the parser should be able to take the correct arc and not have to provide for back-up to that state. In the RUS parser, we have taken advantage of an extension to the normal ATN notation [Burton, 1976] that permits any set of arcs from a single state to be combined into one **GROUP** arc. The arcs within a **GROUP** are then treated as strict alternatives - at most one can succeed at any point in a parse, and so there is no need to allow for any back-up. In states where arcs could not be **GROUP**ed immediately, it was the case that by allowing arcs to examine not only the current word, but also one or two words ahead, those arcs could then be grouped.

We have also introduced the notion of an "almost-**GROUP**." This effectively splits a single node in two, with one **GROUP** splitting the situation into deterministic and non-deterministic cases, and another GROUP for the deterministic cases. This captures our intuition that most sentences could pass deterministically through a given state, and moreover, it would be easy to distinguish the sentences that had to be treated non-deterministically.

The second cause of back-up mentioned above has been pointed out by Marcus [1978] - the typical use of ATN's as a top-down, hypothesis-driven parsing mechanism. That is, when a point in the parsing is reached where it is possible for a constituent of type X to appear, the parser PUSHes to a network which actually looks for an X. In top-down analysis this is done purely on the sis of the structure found up to that point in the sentence. ck-up can be avoided if such PUSH arcs are not taken when it is ar that the current word (or the next few words) precludes such a constituent. For example, there are places in the analysis of a clause where a PP is optional. We do not want to PUSH for a PP there if the next word clearly precludes its presence (e.g., if the next word is not a preposition).

After analyzing situations where the RUS ATN PUSHed for constituents that were "obviously" not present, we inserted tests that blocked the offending PUSH arcs when the next words were inconsistent with the PUSH arc. These tests required looking no further than the next three words, and often no further than the next word. This is consistent with Marcus' "three chunk" look-ahead. Although there are cases where a three constituent look-ahead would have been required to completely avoid backup, three word look-ahead suffices to drastically reduce the back-up normally caused by top-down parsing.

The third source of back-up lay in the very heart of the RUS approach, namely the incremental semantic testing of constituents, coupled with the ATN's standard depth-first control structure. For example, PUSH actions admit the possibility that several constituents of the type PUSHED for (e.g., several PPs) are present at the given place in the string, differing in length or in internal structure. RUS may reject the first result of the PUSH because it is semantically unacceptable in the context of that PUSH. A "depth-first" control structure will produce all possible alternative constituents of the desired category before trying any alternatives to the PUSH.

However, as the parser becomes more nearly deterministic, the first semantically meaningful result returned from a PUSH is likely to be the best description of what actually occurs at that position. This is particularly true for optional constituents, such as prepositional phrase modifiers (especially those specifying location or time). A frequent case is where an embedded NP PUSHes for a PP, the parser finds one, and the semantic interpreter rejects it as a modifier of the NP. This situation can occur when the first PP found by the parser is actually a modifier of the matrix clause or NP.

For example, consider the sentence

> "That professor teaches undergraduates about languages for processing complex types of list structure."

When the parser is processing the embedded noun phrase "undergraduates", it will PUSH for a PP and find "about languages for processing complex types of list structure" as a semantically coherent PP. This is indeed the correct PP at this point in the string, as opposed to "about languages" or "about languages for processing", and so on, but it is not a PP that can modify "undergraduates." In this situation a depth-first control structure will generate useless parses of meaningful but irrelevant PPs before determining that in this sentence the NP "undergraduates" has no PP modifiers, and that "about languages for processing complex types of list structure" is actually a modifier of the clause.

To avoid this difficulty we have implemented a control structure that produces the first semantically acceptable result of each PUSH[23] but postpones branch points that might produce

---

[23] This usually is the longest semantically coherent constituent of the type PUSHed for.

alternative results for that PUSH (often by dropping off semantically acceptable but syntactically optional post-modifiers). When this control structure is combined with the well-formed substring facility (WFS) which is a normal part of the parser, we get an efficient technique for placing optional modifiers where they are semantically acceptable. If an optional constituent is semantically rejected because it was PUSHed for by the wrong level of network, it is stored in the WFS. If some other phrase then PUSHes for the same type of constituent at the same place in the string it will find that constituent in the WFS without any further parsing.

The net effect of these changes has been to remove almost all instances of backtracking in the operation of the parser. Most of the cases where the parser actually has to back up are ones which cannot be resolved on the basis of local evidence, and in which humans often garden path.

4.3 Semantic Interpretation in PSI-KLONE

4.3.1 Introduction

This section describes both the semantic interpretation assigned to an input sentence and the process by which it is assigned. As we indicated in Section 3.4, semantic interpretation is merely an intermediate stage in the processing of a sentence. The final stage is processing by a discourse component which has access to

- o the results of the syntactic analysis of the sentence
- o the semantic interpretation of the sentence
- o general pragmatic knowledge
- o evolving models of
    - the speaker's knowledge, beliefs and current focus
    - the objects, events and relationships under consideration in the current discourse

The semantic interpreter produces a representation of the input sentence based on the functional syntactic analysis of the sentence (see Section 4.2.2.1) and a knowledge of lexical semantics to be described here (a sample of such a representation was presented in Section 2.9). There are two distinct types of information included in the output of the semantic interpreter - combinatoric information and descriptive information. This distinction can be viewed as a generalization of the distinction between quantifiers and formulas with free variables (matrices) in quantified predicate logics, and we introduce it by means of an analogous distinction in typed-quantifier predicate logic.

Consider a typed-quantifier predicate logic with the following properties:

1. quantified variables are typed - each variable is limited to range over a particular domain, which is specified by a predicate,

2. variables are allowed to stand for sets as well as for individuals,

3. types are not limited to simple predicates on individuals or sets, but can be complex predicates that may themselves depend on the binding of other variables in the expression, and

4. expressions are written in Prenex Normal Form, with all quantifiers pulled out to the left, leaving an open formula to the right.

The advantages of such a logic as a representation for the semantics of English sentences are discussed by Webber [Webber, 1978]. The first three properties allow the information conveyed by noun phrases to be kept separate from the information conveyed in the clause. Properties 1 and 2 reflect the fact that in English one predicates attributes of a set, such as cardinality, in addition to predicating attributes of its members. Finally, property 3 provides for both explicit and implicit dependencies between noun phrases, by allowing the type-predicate for one variable to explicitly depend on the value of another variable.

To illustrate this, consider the sentence

"Each boy gave each girl he knew three peaches"

which we can represent by the typed predicate logic formula

```
(Ax: Boy)
  (Ay: LAMBDA(u: Girl)[Know x,u])
    (Ez: LAMBDA(w: Setof(Peach))[|w| = 3])
   Gave x,y,z
```

Here the representation of the clause is simply the open formula

"Gave x,y,z",

while the noun phrases correspond to elements in the quantifier prefix. The variable **x** is shown to range over individual boys, the variable **y** is shown to range, for each boy, over individual girls he knows - an explicit (non Skolem-function) dependency - while the variable **z** ranges over sets of individual peaches whose cardinality is 3. Note that cardinality is a property of sets rather than of individuals. (This particular notation is discussed further in [Webber, 1978], where its value is pointed out for understanding various anaphoric and elliptic phenomena. In PSI-KLONE, we are using the KL-ONE formalism, which provides these properties, as well as an inheritance hierarchy for the efficient indexing of relevant inference rules.)

The reason we have introduced this typed predicate calculus representation is that in Prenex Normal Form, the open formula to the right of the quantifier prefix can be viewed as a **pattern** - a way of describing a **set** of ground literal formulas by giving their **syntactic shape**. The literals in this set will vary according to how individual constants are substituted for the variables in the pattern. The quantifier prefix, on the other hand, can be viewed as a **combinatoric specification** which determines what ordered combinations of constants can be assigned to the variables to instantiate or stamp out copies of the pattern.

To summarize, we view a semantic representation as having both **a descriptive part** and a **combinatoric part**. In the representation we are using, the descriptive part of a semantic interpretation consists of an interlocking and interdependent collection of Generic descriptions in KL-ONE, to be instantiated to Individual Concepts in ways specified by the combinatoric part.[24] Among the combinatoric constraints on individual instantiations are dependency, distribution and cardinality. All of these will be discussed in section 4.3.3.

Finally, we believe that a quantified sentence like

"Which windows were delivered to each house?"

poses an **underconstrained** combinatoric problem that the listener

[24] These are not necessarily descriptions of things in the outside world, but rather of objects, events and relationships consistent with the system's long term semantic knowledge.

must solve in order to respond appropriately to the sentence. It is our view that semantic interpretation is only responsible for delineating the problem to be solved, whereas it is the responsibility of the discourse component - using whatever pragmatic and discourse information is available to it - to solve the problem to the extent required to respond appropriately. The procedure to be used by the pragmatic/discourse component to solve this problem is an active area of research.

4.3.2 Semantic Interpretation: Descriptive Information

4.3.2.1 Introduction

This section further describes the dialogue between syntax and semantics. There are two things that a cascaded or interactive semantics must do:

1. provide semantic interpretations for individual phrases, and
2. provide feedback to the syntactic processor.

If one considers two major existing models for computer based parsing - the framework used in the LUNAR system [Woods, Kaplan, and Nash-Webber, 1972], and semantic grammar framework [Burton, 1976] - one can see that in both cases there is one mechanism that checks properties of particular constituents and, if those constituents satisfy those properties, then there is another mechanism that shows how to build or add to the interpretation of the whole phrase depending on how those properties are satisfied.

In LUNAR, the pattern-match on the left-hand-side (LHS) of a semantic interpretation rule corresponds to the first mechanism, while the actions specified on the right-hand-side (RHS) of the rule correspond to the second mechanism. In a semantic grammar, on the other hand, PUSHing for a particular syntactic/semantically shaped constituent (e.g., "an NP which is interpretable as a measurement") corresponds to the first mechanism, while some "BUILD action" into a register corresponds to the second.

In PSI-KLONE, each interpretable syntactic/semantically shaped pattern corresponds to a KL-ONE Generic Concept. These Concepts are arranged into a KL-ONE taxonomy which can be used both as a discrimination net and as a mechanism for inheriting appropriate interpretation rules. Semantic checking of potential assignments of constituents to particular functional syntactic roles in a phrase involves information that may be used in building the interpretation of the completed phrase. On the other hand, semantic interpretation occurs only after the entire phrase has been recognized, and the possible rules for semantic interpretation have been collected.

4.3.2.2 Using KL-ONE Taxonomies to Build Semantic Interpretations

To illustrate the use of the taxonomy of syntactic/semantic shapes in the PSI-KLONE Interface, consider the sentence

"That professor teaches undergraduates about Lisp on Thursday."

Figure 11 shows a fragment of a possible syntactic/semantic taxonomy that covers some statements on teaching. We will concentrate on the activity at the clause level and ignore the details of parsing at the NP and PP levels. Figure 12 shows a fragment of a toy ATN that could be used as a (non-deterministic) parser of various types of clauses.

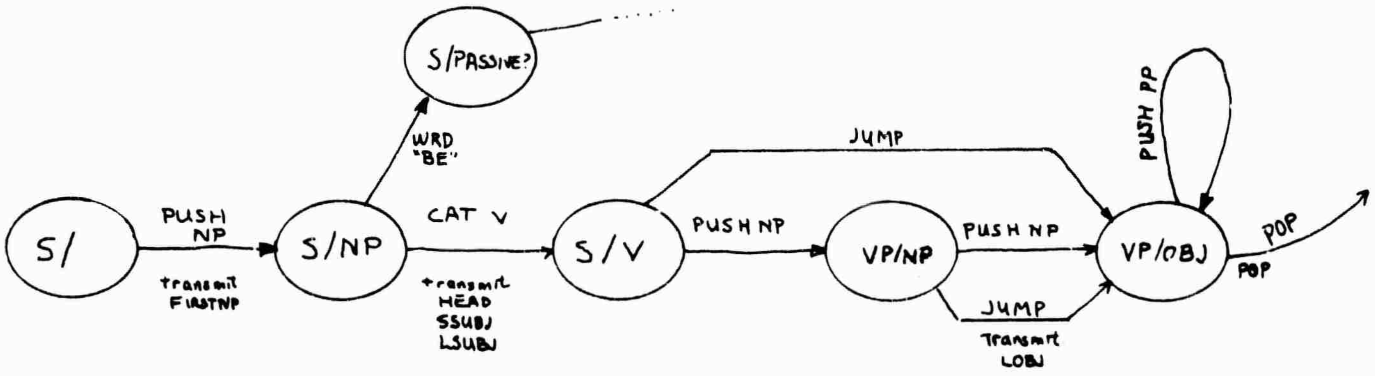


Fig. 12. A simplified ATN for clauses.

The first step in parsing the example is PUSHing for an NP. This parses the string "That professor" and produces the Individual Concept NP#1 which is an individuator of the Generic

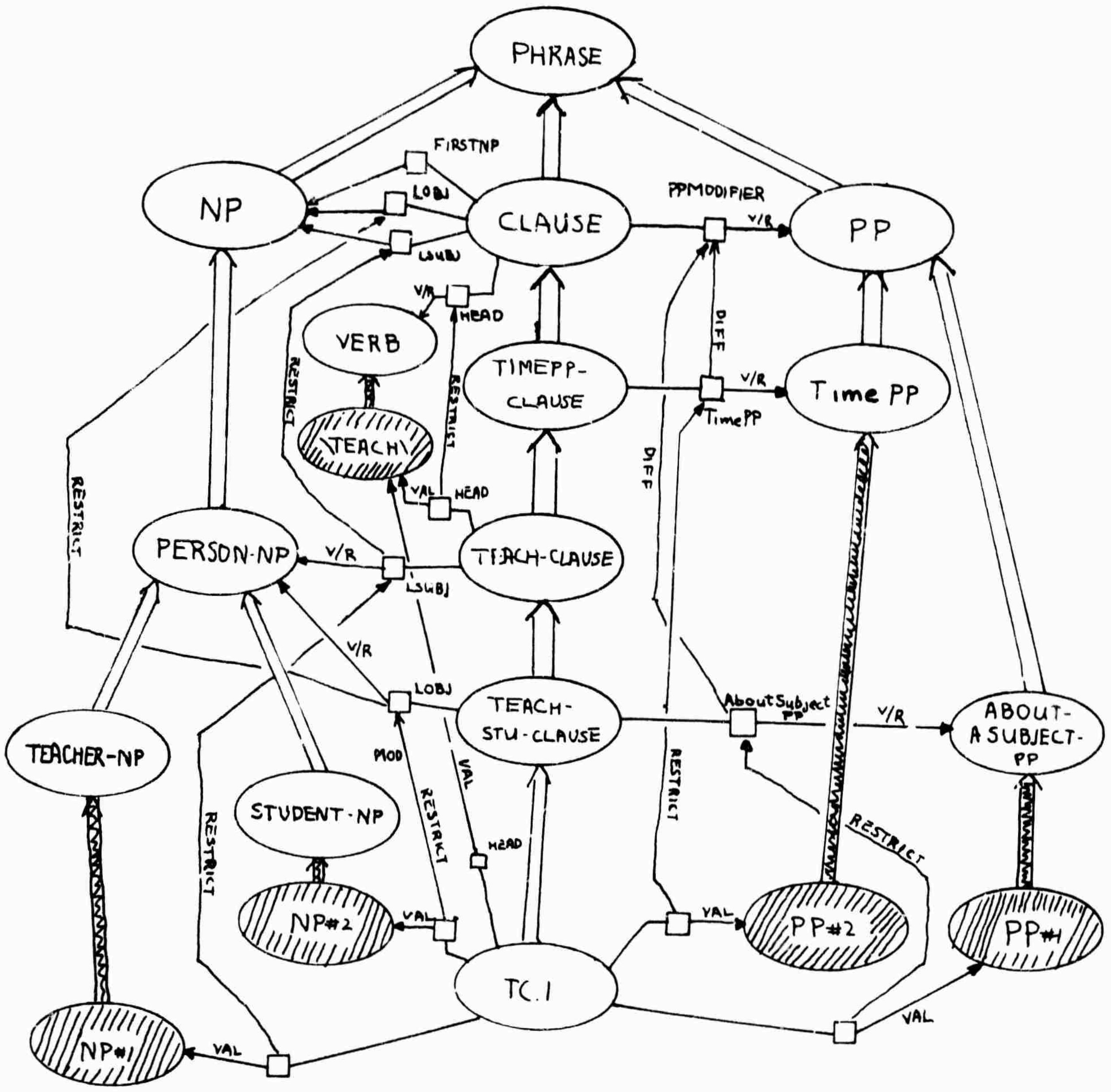


Fig. 11. A KL-ONE syntactic/semantic taxonomy.

TEACHER-NP[25] with an associated semantic interpretation not shown in this diagram.

At this point NP#1 is transmitted as the **FIRSTNP** of the (currently empty) clause, although the parser does not yet have enough information to decide on other Roles it fills.

The parser then discovers that "teaches" is the main verb of the clause, and transmits the Individual Concept \TEACH\ (we use the character "\" to bracket the names of Concepts that stand for morphological units, \TEACH\ corresponds to the morphological root of "teaching") as the **HEAD** of the current clause. The PSI-KLONE Interface can now begin to place the clause within the syntactic/semantic taxonomy, as a subConcept of TEACH-CLAUSE. This Generic Concept carries the information common to two types of "teach" clauses - those whose LOBJ is a subject of study like "John teaches calculus" (not pictured in the figure), and those (represented by TEACH-STU-CLAUSE) whose LOBJ is human (or at least sentient). The interpretation of a clause of either type is an individuator of the Concept TEACHING, and both types of clauses must have an LSUBJ whose interpretation is an

---

[25] The justification for having a special class of NPs which can be interpreted as teachers is based on the fact that various modifiers like "tenured" are specifically applicable to such NPs, and others, like "at Berkeley", may receive special treatment.

individuator of PERSON. Additionally, both clauses are examples of clauses that can take PP time modifiers. Such clauses correspond to the Generic Concept TIMEPP-CLAUSE and TEACH-CLAUSE is a subConcept of TIMEPP-CLAUSE.

The PSI-KLONE Interface responds to RUS with a pointer to a newly created subConcept TC.1 of TEACH-CLAUSE, with its **HEAD** Role filled by \TEACH\ and its **FIRSTNP** Role filled by NP#1. Since the clause is not passive, the parser transmits NP#1 as the **LSUBJ** of TC.1. From the point of view of semantics, since NP#1 is an instance of a PERSON-NP (by inheritance through TEACHER-NP), it can fill the **LSUBJ** Role of TC.1. Thus semantics fills the **LSUBJ** Role with NP#1 and returns a pointer to TC.1 to RUS.[26]

RUS then parses "undergraduates" as an NP, producing the Individual Concept NP#2 which individuates STUDENT-NP. The parser cannot transmit this NP yet, because it can function as either the LOBJ or LINDOBJ of a "teach" clause.

We have glossed over an interesting point here - the fact that it was a restriction on the **PPMODIFIERs** of STUDENT-NP that

[26] Actually, a new subConcept of TC.1 is created with its **LSUBJ** role filled by NP#1. This strategy facilitates sharing of information between alternative paths in the parser, but we will ignore it in the remainder of this example.

prevented "about Lisp" from being included as a PPMODIFIER of "undergraduates." This is an example of the use of semantic information to reject syntactically plausible parsings.

Once RUS determines that no NP directly follows "undergraduates" it can transmit NP#2 as the **LOBJ** of the clause TC.1. This is done on the JUMP arc between VP/NP and VP/OBJ. In this case, PSI-KLONE notes that there is a subConcept TEACH-STU-CLAUSE of TEACH-CLAUSE which allows a PERSON-NP as the filler of its **LOBJ** Role, and so PSI-KLONE makes TC.1 a subConcept of TEACH-STU-CLAUSE and fills in its **LOBJ** Role with NP#2.

RUS then parses "about Lisp" as a PP, producing PP#1, an individuator of an ABOUT-SUBJECT-PP. Although a PP in this position may play a special syntactic role in a clause, like a "by ..." PP in a passive clause, PP#1 does not, so the parser transmits it to PSI-KLONE as simply a **PPMODIFIER** of the clause TC.1. Since TC.1 is now a subConcept of TEACH-STU-CLAUSE, it can take such a **PPMODIFIER**.[27] In fact, there is a specialized version of the **PPMODIFIER** Role present at TEACH-STU-CLAUSE, the Role

---

[27] Note that TEACH-SUBJECT-CLAUSE could not take such a modifier, so that in a string like "a professor who teaches algebra about Lisp", the PP "about Lisp" would have to be a modifier of something other than the "teach" clause, as in "John told a professor who teaches algebra about Lisp" where "about Lisp" is unambiguously a modifier of the "told" clause.

AboutSubjectPP, which can accept PP#1 as a filler. The response to this transmission is a pointer to TC.1, which now has PP#1 filling its **AboutSubjectPP** Role.

Finally, RUS parses the PP "on Thursday", producing PP#2, an individuator of TimePP. This is transmitted as a **PPMODIFIER** to TC.1, and PSI-KLONE determines that it can fill the **TimePP** Role that TC.1 inherits from TIMEPP-CLAUSE. PSI-KLONE returns a pointer to TC.1 with its **TimePP** Role filled by PP#2.

At this point the parser is at the end of the clause (and string) and signals this by a transmit triple whose label is **POP**. This tells semantics to check that all necessary Roles are filled and that all inter-Role restrictions are satisfied. Now PSI-KLONE creates the descriptive part of the semantic interpretation of the clause by collecting the **projection rules** that TC.1 inherits by virtue of its position within the syntactic/semantic taxonomy. These rules are attached as data on various Roles and Concepts in the taxonomy.

PSI-KLONE expresses semantic projection rules in JARGON (see Section 2.1.2). There are two reasons for this: one, JARGON is easily read and understood, and two, its interpreter implements an algorithm - the **MSS algorithm** (see Section 2.3.1) - that automatically inserts KL-ONE Concepts described in JARGON at the appropriate place in the taxonomy of Concepts. This makes it possible for the descriptive part of a semantic interpretation to inherit all appropriate inference rules that are stored in the long-term semantic taxonomy.

A slightly simplified form of the JARGON phrase that describes the semantic interpretation of the sentence "that professor teaches undergraduates about Lisp on Thursday" is

> A TEACHING WHOSE TEACHER IS THE INTERP OF (LSUBJ) AND WHOSE STUDENT IS THE INTERP OF (LOBJ) AND WHOSE TIMEPREDICATE IS THE INTERP OF (TimePP).

In JARGON, phrases can refer to both Concepts and their Roles and can use variables for their arguments. For example, the construction "THE INTERP OF (LOBJ)" refers to the Role named INTERP of the Concept which is the value of the variable LOBJ.

The JARGON phrase given above is a conjunction of smaller parts, including "A TEACHING", "WHOSE TEACHER IS THE INTERP OF (LSUBJ)", and so on. Each part indicates the source of a particular piece of information on the syntactic side (e.g., "THE INTERP OF (LSUBJ)") and the Role that the piece of information is to fill in the semantic interpretation (e.g., "WHOSE TEACHER", which means the TEACHER Role of the semantic interpretation of the clause).

These pieces of JARGON constitute the semantic projection rules hung on Roles in the syntactic/semantic taxonomy. For example, the rule "WHOSE TIMEPREDICATE IS THE INTERP OF (TimePP)" hangs on the Role TimePP of the Concept TIMEPP-CLAUSE. TC.1, the Concept describing the syntactic/semantic shape of the sentence "that professor ...", is a subConcept of TIMEPP-CLAUSE, and has an explicit filler (PP#2) for the Role **TimePP**; therefore, it inherits the projection rule. Similarly, TC.1 inherits the rule "WHOSE TEACHER IS THE INTERP OF (LSUBJ)" from Role **LSUBJ** of TEACH-CLAUSE and the rule "WHOSE STUDENT IS THE INTERP OF (LOBJ)" from the Role **LOBJ** of TEACH-STU-CLAUSE, and so on.

When the RUS parser transmits the triple labeled **POF**, the PSI-KLONE interpreter creates a new Individual Concept - say, TC#1 - as an individuator of TC.1. This action triggers an attached procedure hung on the highest-level syntactic/semantic concept, PHRASE, which collects the projection rules inherited by TC#1 and forms a JARGON phrase. It then binds the variables occurring there to the fillers of the appropriate Roles (e.g., the variable LSUBJ is bound to NP#1, the filler of the **LSUBJ** Role of TC.1), and then calls the JARGON interpreter which builds the KL-ONE Concept described by the JARGON phrase, and inserts it at the proper position in the semantic taxonomy. Finally, it fills the **INTERP** Role of TC#1 with the Concept in the semantic taxonomy produced by the call to the JARGON interpreter.[28]

[28] For a sample of what such an interpretation might look like, see Section 2.9.

4.3.3 Semantic Interpretation: Combinatoric Information

There seems to be a point in the processing of a sentence where there is some indication of the type of events, objects and relationships being described, but where things have not been resolved into a form which can be represented in an unambiguous predicate calculus type of quantification. People often believe that they have understood the sentence without further elaboration of this p[illegible] of the interpretation, without realizing that there are remaining quantifier scope ambiguities [Van Lehn, 1978]. Van Lehn suggests that correlations between syntactic structure and quantifier scope interpretation are epiphenomenal - i.e., that there are no processes based purely on syntactic information that can disambiguate quantifier scope. Our belief is somewhat stronger - that there are few, if any, processes that can completely disambiguate quantifier scope simply on the basis of syntactic and semantic information, without making use of discourse-level and pragmatic information.

We believe that this is not accidental - i.e., not a performance error - but rather represents a natural split between the results of the syntactic/semantic component and the activity of later discourse and pragmatically based processes. That is, it is not at the level of the sentence that the information needed to resolve things is available; if it is available at all, it is at the level of the discourse. Moreover, the degree to which scope ambiguities will be resolved is itself dependent on the purposes of the discourse. In some cases in fact, those purposes can be met without raising the specter of ambiguity at all.

4.3.3.1 The Combinatoric Aspects of Semantic Interpretation

The purpose of this section is to illustrate the combinatoric aspects of a sentence's interpretation that should be identified by semantics, and, if necessary, resolved by pragmatics. Although the current system does not yet treat the combinatoric part of a sentence's interpretation in line with this presentation, we are currently designing a semantic component for PSI-KLONE which does.

There are qualitatively three types of combinatoric constraints embodied in an English sentence:

1. dependencies
2. iterations
3. cardinalities.

To illustrate these types of constraints, consider the following sentence

"Two windows were tested in each house"

and the situations in which someone might generate it. In any such situation, the use of "each house" indicates, at a syntactic level, that the speaker has in mind a definite set of houses. (This treats "each house" as equivalent to the phrase "each of the houses".) For this example, label the elements in this set of houses h1,...,hk. There is also something being said about some set (or sets) of two windows. "Two" is cardinality information about the number of windows in each set. What is not specified is how many sets there are. This can be determined only after implicit dependencies have been made clear. There are three possibilities: there is no dependency of one thing on anything else, or there is a minimal constraint (Skolem-functional) dependency or a discourse or definitional dependency on some other variable.

No dependency. In this case, the speaker has in mind two particular windows (call them w1 and w2). There is no dependency, since independent of house, it is w1 and w2 that were tested there. We might represent this in terms of ground literals as

```
    Tested-in(w1, h1)
    Tested-in(w2, h1)
    .
    .
    .
    Tested-in(w1, hk)
    Tested-in(w2, hk)
```

Notice that there are two terms for windows and k terms for houses. Pragmatically, there are as many referents for windows and houses as there are terms.

Minimal (or **Skolem-functional**) **dependency** on the one **iterative variable.** Here the speaker is iterating over houses. For any house hi, the two windows tested there - f_1 and f_2 - depend on the hi. (The skolem functions f_1 and f_2 capture these dependencies.) The only property assumed to hold of f_1 and f_2 is that for any house, the two windows are distinct from each other, but not necessarily from the windows tested in some other house.[29] In terms of ground literals,

$$\text{Tested-in}(f_1(h1), h1)$$
$$\text{Tested-in}(f_2(h1), h1)$$
$$\vdots$$
$$\text{Tested-in}(f_1(hk), hk)$$
$$\text{Tested-in}(f_2(hk), hk)$$

where $f_1(hi) =/= f_2(hi)$. Notice that there are 2k different terms for windows here and k different terms for houses.

[29] This may be clearer in the analogous sentence "Two songs were sung by each boy", in which it is possible that more than one boy sings some particular song.

However, all we know about the number of different referents for windows is that there are at least 2.

Discourse or definitional dependency on the one iterative variable. Here again the speaker is iterating over houses in the set. For any house hi, the two windows tested there ($w1_{hi}$ and $w2_{hi}$) not only depend on the house, but are members of some previously established, definite set of windows W(hi) that either belong to that house or have been associated with the house through the discourse.[30] In terms of ground literals this can be represented as follows:

Tested-in($w1_{h1}$, h1)
Tested-in($w2_{h1}$, h1)
.
.
.
Tested-in($w1_{hk}$, hk)
Tested-in($w2_{hk}$, hk)

where $w1_{hi}$ =/= $w2_{hi}$, and W is a function from a house to the

[30] For example, consider the sequence

"The contractor delivered some experimental windows to each house on the block.

Two windows were tested in each house."

set of windows belonging to (or associated with) that house. Here again we have 2k different terms for windows and k different terms for houses. Moreover, since W(hi) is a previously established set, one may have additional information by which the referents of $w1_{hi}$ and $w2_{hi}$ can be further constrained. For example, in the case of definitional dependency, pragmatic knowledge tells us that, since a window can only belong to one house, there are 2k different referents for windows.

4.3.4 Next Steps in Representing Combinatoric Semantics

An important goal of our current research in semantic interpretation is to develop a formalism in which there is a clean split between the descriptive and combinatoric aspects of semantic representation. The number of alternative ground level interpretations of a sentence increase rapidly as the number of noun phrases (and hence quantifiers) increases in the sentence, and as the number of possible dependencies among entities increases. It is inefficient to try to represent large numbers of such alternatives as an explicit disjunction, both because of the amount of space such a representation would normally require, and because of the complexity of the case analysis that would be necessary to reason forward from such a representation.

We want to provide an efficient representation for that part of the meaning of a sentence in a discourse that can be provided on the basis of its internal syntactic/semantic structure alone. This would include explicit information on the cardinality restrictions on variables associated with NPs, restrictions on which variables are likely to be iterated, and information on possible dependencies among variables, including those suggested by long-term semantic knowledge, and restrictions on dependence based on syntactic structure. We believe that it should be possible to represent such knowledge as a set of constraints on the set of ground level literals to which the sentence might possibly expand.

Such a representation would provide an input to the pragmatics/discourse component, which could refine it and add constraints based on discourse information and perhaps some variants of the heuristics suggested by van Lehn. It is not always necessary for the pragmatics/discourse component to totally disambiguate the combinatoric aspects of semantics in order to satisfy the requirements of the discourse.

We are investigating a number of possible representations in KL-ONE, in the context of a broader study of the use of **meta-description** - the use of KL-ONE structures to describe (classes of) other KL-ONE structures (see Section 2.7).

5. THE PRAGMATICS/DISCOURSE COMPONENT

5.1 Introduction

One of the principal arguments for developing natural language systems is the possibility for instructing machines in a manner convenient to the user. For instance, it is envisioned that a robust natural language query facility will allow casual users to access knowledge-based systems without having to learn specialized query languages. While natural language processing techniques to date have been somewhat successful at analyzing the literal import of users' queries and commands, it is well-known that people formulate their natural language "queries" imprecisely, and often the literal meaning of what they request is different from what they intend. For instance, a literal-minded information system is likely to reply "yes" or "no" to questions like "Do you have the damage report?" instead of (conditionally) printing it out.

While Kaplan and Joshi [1978] propose limited solutions to a few forms of "non-literal" queries, his view of man-machine dialogue is in terms of a query/response pattern. Instead we contend that significant theoretical and practical advantages would be gained by viewing natural language interaction with a machine as **cooperative goal-directed conversation**. This shift is not just a relabeling, but rather entails viewing conversation as a sequence of speech actions [Austin, 1962; Searle, 1969] that are planned to influence the "hearers'" beliefs and goals [Allen, 1979; Bruce, 1975; Cohen, 1978; Cohen and Perrault, 1979; Perrault and Allen, forthcoming]. On such a view, machines would be required to reason about how the users' actions and utterances fit into their plans, and then to be helpful by doing something (perhaps responding in language) to enable those plans to succeed.

Being helpful will be loosely defined here as "doing more than is literally required, though not too much more." At least four aspects of being helpful have been identified in building computer systems:

- The ability to correct the user's erroneous presuppositions, as noted by Kaplan and Joshi [1978]. For instance, it would be decidedly unhelpful to answer "Which ATN states have comments on them?" with "None", if the system believes that states cannot carry comments.

- The ability to process the "non-literal" interpretation of the user's input [Allen, 1979; Perrault and Allen, forthcoming]. For instance, a literal-minded system that responded to "No, I want to be able to see S/AUX" with "OK", rather than displaying S/AUX, would not be helpful. Though the user has stated his goal, that statement does not exhaust his purpose in making the utterance. Clearly, he wanted the machine to do something about his goal. People (even when speaking to computers) are so used to expressing their intentions indirectly that it would be extremely hard to force them to become precise. The alternative is to provide a facility to infer the user's closely-related intentions (whenever possible).

- The ability to volunteer useful information, or perform additional obviously useful actions. If a user asks "Can the Kennedy currently attain 35 knots?", a system could appropriately reply "No, but the Saratoga can" if it believes the user is trying to find a ship to fill a particular need.

- The ability to explain the system's actions to the user. If a system were asked why it printed both the maintenance status and speeds of a set of ships, it might helpfully reply "Because I thought you were trying to find out which ships that could reach 35 knots were available for duty."

A critical component to the solution of all of these problems is the identification of the goal or plan behind the user's utterance. For example, the determination of what, if any, information would be useful to the user depends on what he is trying to do - i.e., his plan. An attempt to recognize the plan behind the user's utterances leads to a uniform treatment of a diverse range of problems in a theoretically sound, yet practical way.

The importance of plan-recognition to communication has been explored in the philosophical work of Grice [1957] and Searle [1969]. Grice was the first to show that "simple" plan recognition, as an unseen observer might perform (cf. [Genesereth, 1978; Schmidt, Sridharan, and Goodson 1978; Wilensky, 1978]), is insufficient as a basis for defining communicative acts. Instead, speakers must plan that hearers recognize their plans, and hearers must recognize the plans they were intended to recognize.

Based on Grice´s analysis of the intentional requirements for communication, Searle proposed that speech acts be defined in terms of "intended recognition of intention". More specifically, a speaker who is performing a speech act (say, a request) must intend to produce the effect of that action (to get the hearer to want to perform the requested act) by means of getting the hearer to recognize the speaker´s intention to produce that effect.

Searle´s analysis laid the foundation for formal and computational models of speech act use. Bruce [1975] and Bruce and Schmidt [1974] attempted to describe speech acts in the form of planning operators [Fikes and Nilsson, 1971] with preconditions and effects, and modeled extended sequences of speech acts as "social-action paradigms" (akin to the "scripts" of Schank and Abelson [1977]). Cohen [1978] and Cohen and Perrault [1979] then argued that the sequential appearance of speech acts in discourse is a function of their causal relationship in speakers´ and hearers´ plans. They showed the feasibility of defining speech acts as operators in a planning system by developing a computer system that could plan its speech acts and thus decide what it would say. They also proposed adequacy tests for speech act definitions. Based on that research, Allen [1979] and Perrault and Allen [forthcoming] have developed the first theory of speech act interpretation as intended plan-recognition and have constructed a computer program that models that process. Together these programs were capable of planning and recognizing REQUESTs (that the hearer do some action), INFORMs (that some proposition is true), INFORMIFs (informing whether or not some proposition is true, and INFORMREFs (informing what the referent of a description is).

The current BBN language understanding system incorporates Allen and Perrault's approach into a larger-scale system. It is capable of simple speech act recognition and planning, employing a KL-ONE based user-model. In particular, it can recognize when certain apparent INFORMs are REQUESTs, when certain yes/no questions are really wh-questions, and when certain questions are actually REQUESTs for display actions. In handling these indirect speech acts [Gordon and Lakoff, 1975; Searle, 1975], this system can derive appropriate interpretations. but it attempts to "short-circuit" various chains of plan-recognition inferences wherever possible. It employs a conditional catalog of mappings from the effects of one speech act form to other inferable effects whereby the conditions on applying a mapping are derived from the conditions that must be satisfied in the underlying plan. Thus, the system tries to apply specific plan-recognition inference rules, but has a more general facility to fall back upon.

The next sections will sketch Allen and Perrault's theoretical model, describe the implementation in our natural language system, and finally illustrate its operation. In particular, we will refer to the set of five sample sentences illustrated in Section 3.3.

5.2 Plans

Formal descriptions of plans typically treat actions as **operators**, which are defined in terms of applicability conditions, called preconditions, **effects** that will obtain when the corresponding actions are executed, and **bodies** [Sacerdoti, 1975] that describe the means by which the effects are achieved. Since operators are representations, their preconditions, effects, and bodies are propositions to be evaluated relative to the problem-solver's beliefs. The system is able to maintain, as part of its model of the world, a symbolic description of the world model of its user (including its user's model of it). Our plan-based approach will regard speech acts as operators whose effects are primarily on the models that system and user maintain of each other.

This method of speech act definition attempts to incorporate the communicative nature of the act (in the Gricean sense) **into** the act's definition as its "body". Thus the definitions of speech and physical (display) actions will have the same form, thereby allowing a uniform planning system to include both in plans. Before we discuss speech act definitions, however, we must outline the foundations on which they are built.

5.3 On Models of Others

Speech acts definitions involve changes to speaker and hearer's beliefs and goals. Thus, any formal or computational scheme will have to represent and reason with formulas containing BELIEVE and WANT. Below, we outline the requirements for such a representation. Later sections will discuss the system's representation.

5.4 Belief

Apart from simply distinguishing the system's beliefs from its beliefs about the user's beliefs, the system's belief representation ought to allow it to represent the fact that the user knows **whether** some proposition P is true, without the system's having to know which of P or ~P it is that the user believes. A belief representation should also distinguish between situations like the following:

1 The user believes that the train leaves from gate 8.
2. The user believes that the train has a departure gate.
3. The user knows what the departure gate for the train is.

Thus, case 3 allows the system to believe that the user knows what the departure gate is without the system's actually knowing which gate the user thinks that is.

Following Hintikka [1969], belief is interpreted as a modal operator A BELIEVE(P), where A is the believing agent, and P the believed proposition.[31] This allows for a formal, albeit too strong, axiomatization and semantics for BELIEVE.

[31] The following axiom schemata will be assumed:

```
B.1 a BELIEVE(all axioms of the predicate calculus)
B.2 a BELIEVE(P) => a BELIEVE(a BELIEVE(P))
B.3 a BELIEVE(P) OR a BELIEVE(Q) => a BELIEVE(P OR Q)
B.4 a BELIEVE(P&Q) <=> a BELIEVE(P) & a BELIEVE(Q)
B.5 a BELIEVE(P) => ~a BELIEVE(~P)
B.6 a BELIEVE(P => Q) => (a BELIEVE(P) => a BELIEVE(Q))
B.7 Ex a BELIEVE(P(x)) => a BELIEVE (Ex P(x))
B.8 all agents believe that all agents believe B.1 to B.7
```

These axioms unfortunately characterize an idealized "believer" who can make all possible deductions from his beliefs, and doesn't maintain contradictory beliefs. Clearly, the logic should be weakened. However, we shall assume the usual possible worlds semantics of BELIEVE in which the axioms are satisfied in a model consisting of a universe U, a subset A of U of agents, a set of possible worlds W, and initial world WO in W, a relation R on the cross-product A x W x W, and for each world w and predicate P, a subset Pw of U called the extension of P in w. The truth functional connectives and, or, not, and => have their usual interpretations in all possible worlds. a BELIEVE(P) is true in world w if P is true in all worlds wl such that R(a', w,wl), where a' is the interpretation of a in w. Ex P(x) is true in world w if there is some individual i in U such that P(x) is true in w when all free occurrences of x in P are interpreted as i.

Following Allen [1979], knowing whether P or ~P (P is a proposition) will be represented as:

```
    A KNOWIF P = (P & A BELIEVE(P))
                      or
                 (~P & A BELIEVE(~P)).
```

A's knowing the referent of a description ixDx will, for the time being, be formalized as:

```
    A KNOWREF ixDx = Ey  (ixDx)=y & A BELIEVE
                                    (ixDx)=y
```

Finally, A KNOW P will abbreviate P & A BELIEVE P. This is an abbreviation and not a definition since A ~KNOW P will be P & A ~BELIEVE(P) rather than ~P & A BELIEVE(~P). We shall not be concerned here with "know" as philosophers care to discuss it.

A natural question to ask is how many levels of belief embedding are needed by a system capable of participating in a dialogue? Obviously, to be able to deal with a disagreement, SYSTEM needs two levels (SYSTEM BELIEVE and SYSTEM BELIEVE USER BELIEVE). If SYSTEM successfully lied to USER, he would have to be able to believe some proposition P, while believing that USER believes that SYSTEM believes P is false (i.e., SYSTEM BELIEVE USER BELIEVE SYSTEM BELIEVE(~P)). Hence, SYSTEM would need at least three levels. However, there does not seem to be any bound on the possible embeddings of BELIEVE. If USER believes SYSTEM has lied, he would need four levels. Furthermore, Lewis [1969] and Schiffer [1972] have shown the ubiquity of **mutual belief** in communication and face-to-face situations - a concept that requires something equivalent to an infinite conjunction of beliefs. Cohen [1978] shows how a computer program that plans speech acts can represent beliefs about mutual beliefs finitely. Our system uses this representation (see Section 5.8.1.2) expanded to three levels.

5.5 Want

The USER´s goals will be represented as propositions that the USER WANTs. Utterances using the words "want" or "desire" will map onto the concept WANT. WANT, however, captures more than simple "desire"; it also covers what follows from a desire - what one is "willing to put up with" given that one has a particular desire. Thus, subgoals that must be achieved in order to obtain some other goal will also be considered as WANTs. The formal semantics of WANT, however, are problematic.

Any representation of the USER´s goals must distinguish such information from USER´s beliefs, SYSTEM´s beliefs and goals, and (recursively) from USER´s model of someone else´s beliefs and goals. The representation for WANT must also allow for different scopes of quantifiers. For example, it should distinguish between the readings of USER wants to "dispatch a ship" as "There is a specific ship that USER wants to dispatch" or as "USER wants to dispatch any ship". Finally, it should allow arbitrary embeddings with BELIEVE. Wants of beliefs (as in "SYSTEM WANTS USER BELIEVE P") become the reasons for SYSTEM's telling P to USER, while beliefs of wants (i.e., SYSTEM BELIEVES SYSTEM WANTS P) will be the way to represent SYSTEM's goals P.

5.6 Speech Act Definitions

It is well known that an utterance can masquerade as one speech act while actually realizing another. Thus, "I want to see the subs as long ovals" could be interpreted as a standing order (an INFORM of the user's WANT), or as a REQUEST to display the subs. To allow for "indirect" means of performing REQUESTs and INFORMs, Perrault and Allen propose two "levels" of speech act operators -- illocutionary and surface. The following are the system's illocutionary operator definitions (following [Allen, 1979]) for REQUEST and INFORM.[32] It is assumed that the reader is

---
[32] The only speech acts we try to model are requests, informs, and questions since many of their important properties appear to be definable solely in terms of beliefs and goals. Requesting and informing are prototypical members of Searle's [1976] "directive" and "representative" classes, respectively, and are interesting since they have a wide range of syntactic realizations, and account for a large proportion of utterances in man-machine dialogues.

familiar with operator definitions as in STRIPS [Fikes and Nilsson, 1971]. "Delete lists" are not used (since the procedures used to make additions are also assumed to make necessary deletions), and Allen and Perrault have added "bodies" to operators [cf. Sacerdoti, 1975].

```
REQUEST (speaker, hearer, act)         INFORM (speaker, hearer, prop)

pr:                                    pr: speaker BELIEVE prop
eff: hearer WANT act(hearer)           eff: hearer BELIEVE prop
body: hearer BELIEVE                   body: hearer BELIEVE
        speaker WANT act(hearer)               speaker WANT
                                               hearer BELIEVE prop

INFORMREF (speaker, hearer,            INFORMIF (speaker hearer,
              description)                           prop)

pr: (KNOWREF speaker description)      pr: (KNOWIF speaker prop)
eff: (KNOWREF hearer description)      eff: (KNOWIF hearer prop)
body: hearer BELIEVE                   body: hearer BELIEVE
       speaker WANT                           speaker WANT
      (KNOWREF hearer description)            (KNOWIF hearer prop)
```

These speech act definitions are intended to capture illocutionary acts [Austin, 1962; Searle, 1969]. However, because they are to be viewed in the context of a planning system, some of Searle's "preparatory conditions", (for example "the hearer is able to do ACT") are not present. Instead their verification is assured by prior planning [Perrault and Allen, forthcoming; Cohen and Perrault, 1979].

A REQUEST thus is an attempt by the SPEAKER to get the HEARER to want to do ACT by means of getting the HEARER to believe the SPEAKER wants him to do it. An INFORM speech act is an attempt by the SPEAKER to get the HEARER to BELIEVE that PROP is true by getting him to BELIEVE the SPEAKER wants the HEARER to BELIEVE PROP.[33]

5.7 Surface Speech Acts

Surface speech acts represent what an utterance "looks like"; further analysis allows the system to infer what speech act it actually was. Surface-requests (S-REQUESTs) and surface-informs (S-INFORMs) are used to represent imperative and declarative utterances, respectively. As such they are determined solely by superficial aspects of the user's input, such as its mood. Questions are modeled as surface-requests that the hearer inform the speaker whether a proposition is true/false (INFORMIF) or what the referent of a description is (INFORMREF).[34] The definitions of S-REQUEST and S-INFORM are

[33] These definitions ignore Cohen and Perrault's [1979] "mediating acts" that would allow, for instance, for the hearer to then believe PROP. Furthermore, their "want" preconditions are missing. Neither of these omissions causes undue harm for the test examples.

[34] Since questions are not part of the sample dialogue, discussion of them here will be minimal. Our approach to them basically follows Allen [1979].

```
S-REQUEST (speaker, hearer, act)  S-INFORM (speaker, hearer  prop)

pr:  ATTEND(hearer, speaker)      pr:  ATTEND(hearer, speaker)
eff: hearer BELIEVE               eff: hearer BELIEVE
       speaker WANT                      speaker WANT
         act(hearer)                       hearer BELIEVE prop
```

Surface speech acts are "primitives" - they have no "body". Thus, user and system are assumed to perform only surface speech acts (rather than illocutionary ones). Notice that the body of a REQUEST, HEARER BELIEVE (SPEAKER WANT ACT(HEARER)) matches the effect of an S-REQUEST. That is, S-REQUEST is one way performing a REQUEST.

Using a hierarchical planning model [Allen, 1979; Perrault and Allen, forthcoming; Sacerdoti, 1975], illocutionary speech acts are first planned and then the body proposition is added to the plan. Numerous surface speech act operator combinations can then be planned to achieve the body, leading to indirect ways of performing the illocutionary act.

Conversely, the process of identifying which speech act(s) the user performed involves hierarchical plan recognition - recognition of how the observed surface speech act fits into the user's plan. In general, plan recognition is done by inferring that the user WANTed to perform some act because he WANTed its effect. Furthermore, he WANTed that effect because he thought it would enable other desired actions, etc. Hierarchical plan recognition also attempts to discover the action(s) that was (were) accomplished by means of the observed action. That is, the recognition process should discover which actions have the inferred proposition or action as part of their bodies.

The essence of Allen and Perrault's model, then, is that by defining speech acts in terms of the recognition of intention (as "hearer BELIEVEs speaker WANTs"), and by using hierarchical plan recognition, indirect speech acts can be analyzed. It turns out that, consonant with Grice [1957] and Searle [1969], the propositions involved in speech act bodies force the recognition process to perform intended plan recognition. The following speech act identification process indicates how this occurs:

1. Characterize the user's input as a surface speech act; declaratives are S-INFORMs, imperatives are S-REQUESTs, and interrogatives are S-REQUESTs to INFORMIF or INFORMREF.

2. Assume the user intentionally performed that action. That is, the system now believes the user wanted to perform that surface speech act.

3. Infer that the user wanted to perform that action because s/he wanted its effect. For an S-REQUEST that the system do some ACT (as in "Show me the clause level network") the proposition resulting from this first inference can be described as follows:

```
SYSTEM BELIEVE
  USER WANT
    effects of S-REQUEST (USER, SYSTEM, ACT(SYSTEM)),
```

   which expands to:

```
SYSTEM BELIEVE
   USER WANT (SYSTEM BELIEVE USER WANT ACT(SYSTEM))
```

Thus, the expansion of the user's goal to achieve the surface speech act's effects leads to a goal (ACT) that the user wants the system to think he has. This formula is abbreviated SBUW SBUW ACT(S). The recognition of goals whose prefix is SBUW SBUW is termed "intended plan recognition." The discovery of goals that the system believes the user wants (i.e., goals prefixed by "SBUW") is termed "simple plan recognition."

4. Apply hierarchical plan recognition to formulas generated from 3). When a goal is created that matches the body of a speech act, that speech act can be "identified." For example, expansion of the user's goal to issue an S-REQUEST (USER, SYSTEM, ACT(SYSTEM)) yields SBUW SBUW ACT(S). The "embedded" SBUW ACT(S) matches the body of a REQUEST, yielding SBUW REQUEST (U, S, ACT(S)). Thus, inferencing has proceeded from level "SBUW SBUW" to "SBUW" - from "intended" to "simple" goal recognition. Any further goals that the system recognizes (e.g., SBUW effects of REQUEST (U, S, ACT(S))) are found because the system is helpful, not because it was intended to recognize them.

Simple plan recognition, loosely speaking, keeps the prefix "SBUW" constant and allows the remainder of the proposition to be involved in plan recognition (either expanding an act by its effect, or finding some act enabled by that propositional remainder).[35] Non-literal interpretations can be obtained by not

---

[35] Perrault and Allen [forthcoming] provide a set of rules that indicate how intended plan recognition keeps these prefixes constant.

performing a "body" inference, but performing other expansions at level SBUW SBUW. For instance, assuming the user has performed an S-REQUEST that the system do ACT the recognizer could expand the literally requested ACT by its effect, keeping the "prefix" SBUW SBUW constant. At some later stage, a new goal of SBUW SBUW ACT2(S) may be inferred, at which point the body inference may apply and the user will be said to have requested ACT2.

Clearly any proposition of the form SBUW SBUW PROP can be expanded at either the simple or intended levels. To decide which level to expand, Perrault and Allen suggest the heuristic of "attributing intention wherever possible," i.e., prefer inferences that keep the prefix SBUW SBUW constant. They also propose an axiom of communication indicating that communicated plans should be simple - speakers do not intend for hearers to recognize that the speaker intends for them to be confused. In cases where multiple mutually exclusive intended inferences are proposed, the system should prefer simple plan-recognition inferences progressing to a literal reading. While the same problems are then likely to occur in trying to recognize (helpfully) the plan behind the literal speech act, the conclusion is not attributed to the user as intentional. The system could then plan another speech act to ascertain which of the alternatives actually was the user's goal (as in Allen [1979]).

The next section discusses the system's implementation of this general model.

5.8 The Discourse Component

Speech acts are defined in terms of speaker and hearer's beliefs and goals. We first discuss the system's representation of these, and then outline the control algorithm that selects and applies plan-based inference procedures attached to the KL-ONE taxonomy of speech acts.

5.8.1 Organization of Beliefs and Goals

5.8.1.1 Belief

The system's representation of belief basically follows that of Cohen [1978] - beliefs will be represented as propositions inside Belief Contexts. The latter are KL-ONE Contexts meta-described as containing a particular agent's beliefs (see Figure 13). The "meta-describes" link resides in some number of Contexts itself, affording a representation of one agent's beliefs about another's. The system's beliefs are meta-described within the same Belief Context, thus representing what it believes it believes (see Fig. 14). Assertions or propositions are made using individual descriptions that meta-describe Nexuses with respect to Belief Contexts (see Section 2.6). Currently, the representation of quantified beliefs (as in Ez John Believe (iyDy=z)) involves the description D describing a Nexus, which is intended to behave as Cohen's [1978] and Allen's [1979] "known constants." These did not occur in our sample dialogue since there were no wh-questions. We shall not dwell on the representational issues, other than to say that the representation is expected to change as a result of further research on descriptions, co-reference, and questions.

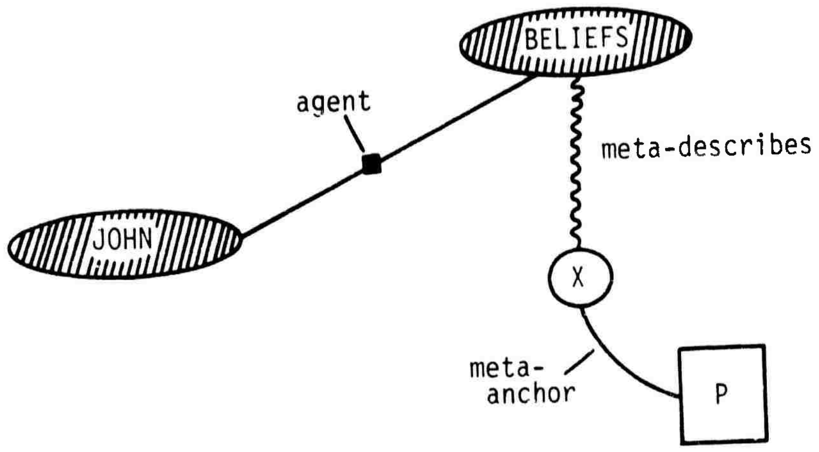


Fig. 13. Meta-describing a belief context.

5.8.1.2 Mutual Belief

Lewis [1969] and Schiffer [1972] discuss the concept of mutual knowledge that arises in face-to-face and communicative situations. Mutual knowledge between two parties C and D that P is true, is defined as

```
C KNOWS P &                     D KNOWS P &
C KNOWS D KNOWS P &             D KNOWS C KNOWS P &
C KNOWS D KNOWS C KNOWS P &     D KNOWS C KNOWS D KNOWS P &
        .                               .
        .                               .
        .                               .
```

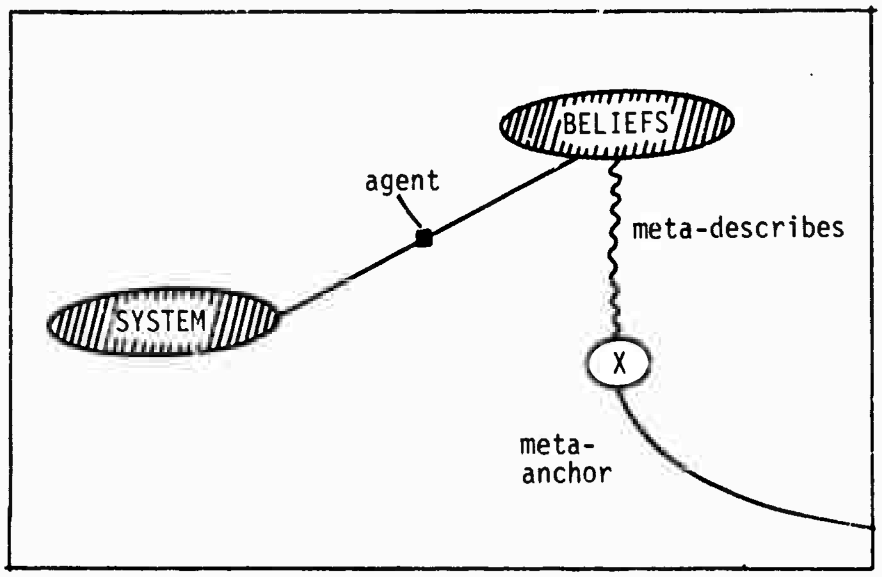


Fig. 14. The system believes that it believes.

Since we are dealing with "belief" as a primitive propositional attitude, we shall only discuss mutual belief. Mutual belief prototypically arises in face-to-face situations. For the current system, the objects on the screen are mutually believed (between the system and the user) to be visible.

The system models beliefs about what is mutually believed between itself and its user in a memory structured as in [Cohen, 1978]. For convenience, we shall incorporate an agent's beliefs about mutual belief into the definition of MB as

```
MB(C,D,P) = C BELIEVE P
          & C BELIEVE D BELIEVE P
          & C BELIEVE D BELIEVE C BELIEVE P
          & ...
```

Using arrows to abbreviate the "meta-describes/meta-anchor" links between the belief description and the Belief Context, MB(SYSTEM,USER,P) is represented as in Fig. 15. The Belief Contexts are given labels - SB (SYSTEM BELIEVES), SBUB (SYSTEM BELIEVES USER BELIEVES) and SBUBSB. Notice that the mutual belief representation equates SBUBSBUB with SBUB, and SBUBSBUBSB with SBUBSB. Cohen shows that, provided the system acquires no beliefs about the user's beliefs from sources other than the user, three Belief Contexts are sufficient to finitely represent mutual belief. As "private" beliefs about the user are created, the system has to create new "looped" Belief Contexts to keep mutual and private beliefs separate. The current system maintains three Belief Contexts, and using Cohen's algorithm, can create new levels as necessary, though this circumstance does not arise in the sample dialogue.

5.8.1.3 Want Contexts

As in Allen [1979] and Cohen [1978], the system's planning and plan recognition take place in various Want Contexts, as below:

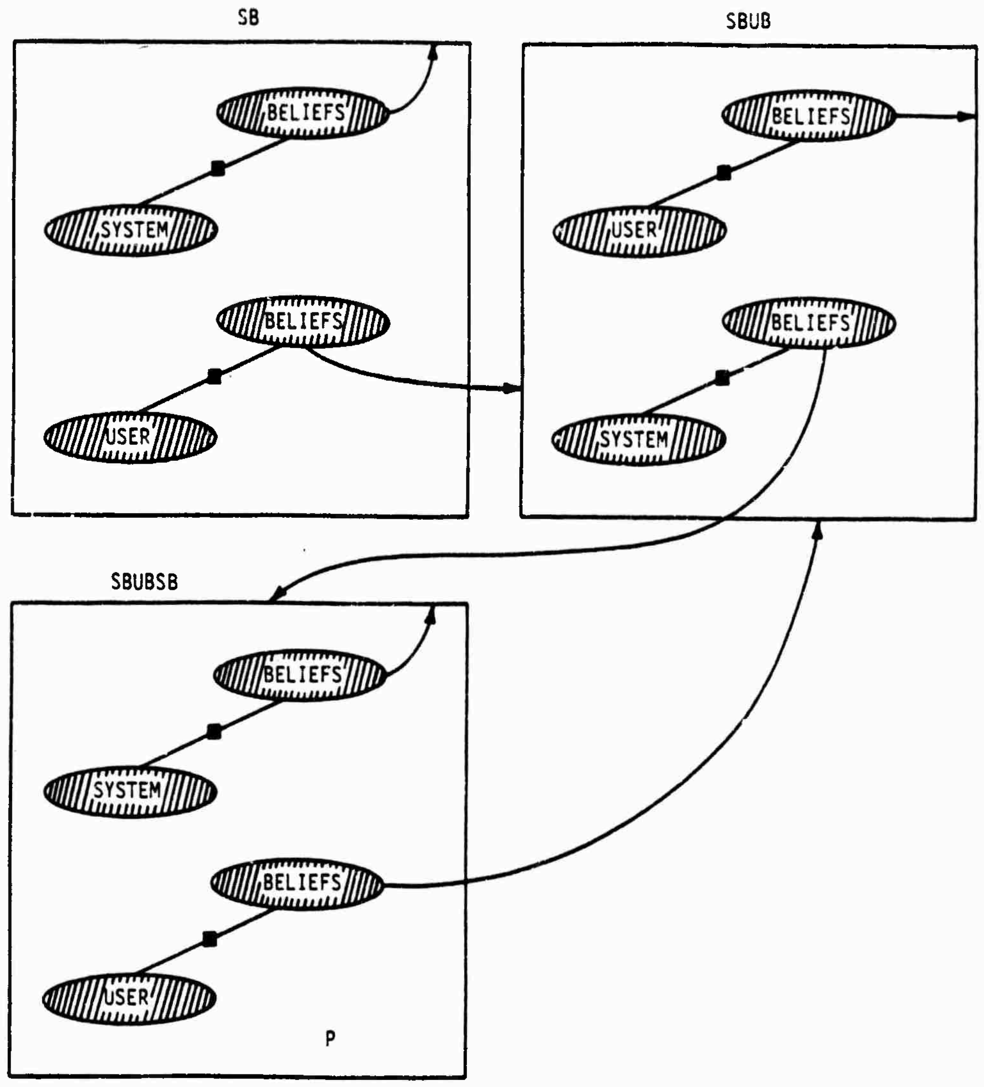


Fig. 15. The mutual belief representation.

Want Context	Knowledge Base
SW	SB
SBUW	SBUB
SBUWSBUW	SBUBSBUB
SBUWSBUWSW	SBUBSBUBSB

The system does its planning "in" SW (SYSTEM WANTS) and its recognition of the user´s plans in SBUW (SYSTEM BELIEVES USER WANTS). The preconditions and effects of acts the system is itself planning are tested with respect to what the system believes. Acts the system is hypothesizing the user to be performing are judged with respect to what the system believes the user believes.

Recognition of the plan the user wanted the system to recognize is performed "in" context SBUWSBUW (SYSTEM BELIEVES USER WANTS SYSTEM BELIEVE USER WANTS). Finally, the system can infer that it was intended to believe the user wanted the system to plan to achieve some goal. This Cortext (SBUWSBUWSW) is used in the analysis of sentences like "I want to be able to see..." It was stated earlier that the knowledge base for Context SBUWSBUW is SBUBSBUB. However, under the representation of mutual belief, using Cohen´s algorithm in the simplification discussed above, SBUBSBUB = SBUB. Similarly, the knowledge base for SBUWSBUWSW is actually SBUBSB.

Want Contexts whose label ends in "SW" are planning Contexts for the system - i.e., given a goal in that Context, the control algorithm will try to find act(s) that would achieve it. That is, reasoning proceeds "backwards". Want Contexts whose label ends in "UW" are plan recognition Contexts - given a goal, the control algorithm attempts to find acts enabled by that goal and to expand the plan being recognized by such acts and their effects. As we shall see, in the course of determining which speech act the user performed, the system may perform both planning and plan recognition.

5.8.2 Plan-related Inference Rules

Depending upon the Want Context in which it is operating, the inference control component will apply plan recognition or planning rules to the goal it is examining. For the current system, these rules are encoded as procedures attached at some level of the hierarchy of actions and states-of-affairs (not to be confused with ATN actions and states). Inference rules higher up in the hierarchy (e.g., attached to the Concepts S-INFORM and ACTION) are applied only if there are no applicable rules placed lower down (e.g., to an S-INFORM whose propositional content is a WANT).

The rules range from general ones - e.g., add the effect of the action to the plan - to specific ones encompassing a number of steps in a plan. The ability to use procedures to encode conditional, multi-step inferences is termed "short-circuiting", and enables us to capture commonly used forms of indirect speech acts (such as "I want...") quickly [cf. Morgan, 1977]. A formalism that provides a foundation for "short-circuiting" is described in [Cohen and Levesque, 1980]. Figure 16 shows our

KL-ONE taxonomy of speech acts and inference procedures for the demonstration system, and the appendix describes what each inference procedure does.

5.8.3 The Controller

The system's inference control algorithm follows Allen's [1979] in its separation of the discovery of applicable inference rules from its decision to employ any particular one. Plan recognition inference rules are rated, and the number of inferences applicable to any goal greatly affects the choice of any of them. This section will discuss the controller briefly, while the example, Section 5.9, will show how the ratings are used.

The controller is implemented in Woods' non-deterministic programming system, NDPROG. This system was initially developed as an abstraction of the central ideas from ATN processing systems. It is used here to keep track of disjunctive leaves of plans.

The controller has a number of states, each of which has access to the following registers: a GOAL, the RATING of that goal, the WANTCONTEXT in which that goal resides, and the BELIEFCONTEXT appropriate to that Want Context.[36] The process

[36] One state, EXPAND, also looks at the register INFERENCE.

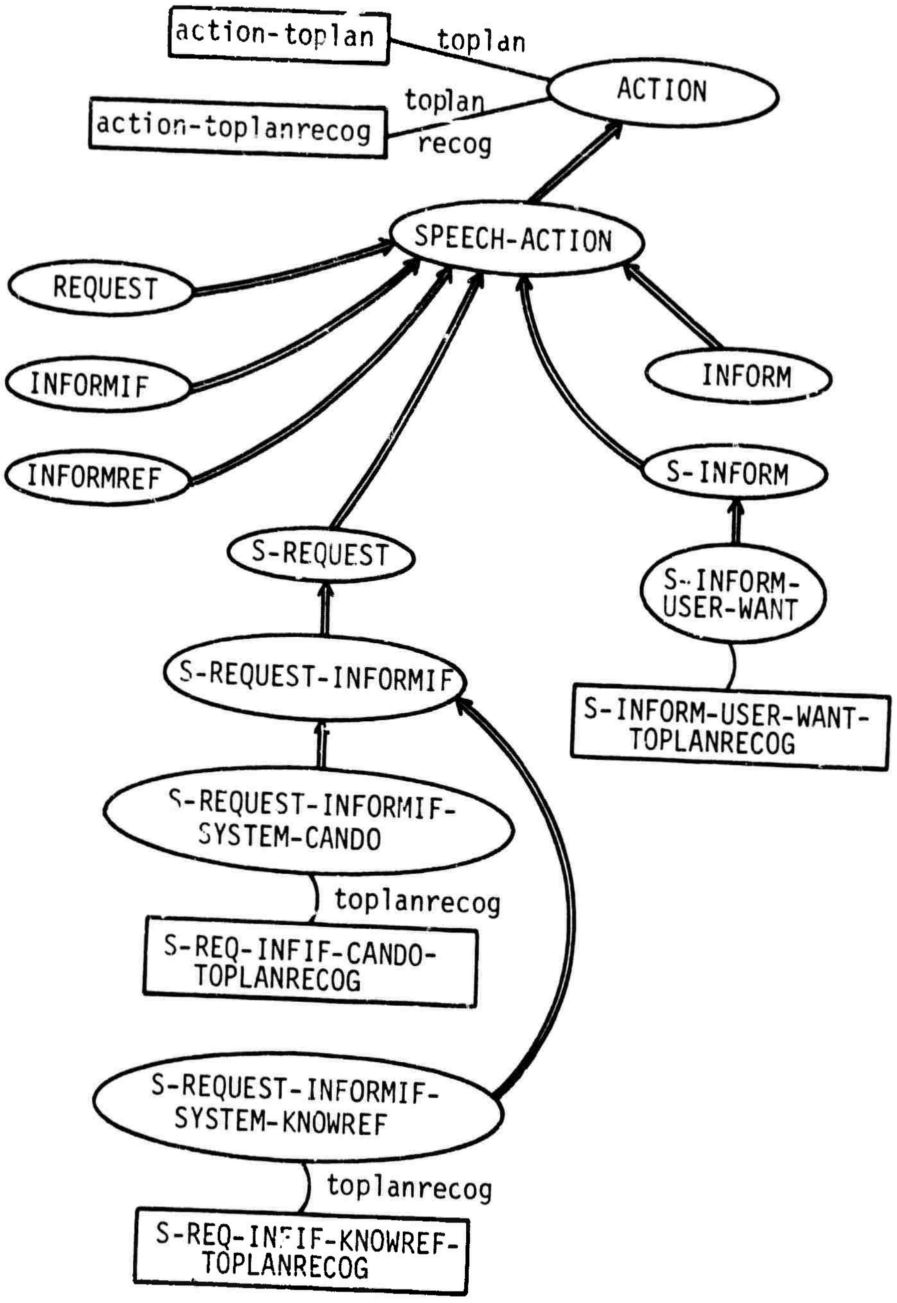


Fig. 16. Taxonomy of speech acts and attached procedures.

can be following several inference "paths" simultaneously. The "endpoints" of each path and the state of the inference engine are maintained by tasks, all residing on an agenda. The controller chooses the highest rated task, applies the code of its state using its associated register contents to produce a new task on the agenda (as an ATN would make a transition to a new state). The states are as follows (as in [Allen, 1979]):

- INFER: Depending upon the WANTCONTEXT, find all planning or plan recognition rules applicable to GOAL. If none, create an ACCEPT task to stop inferencing. Otherwise, for each matching inference rule spawn an EXPAND task with a new rating that is the current RATING divided by the number of applicable inferences. Thus, the rating of a path in the plan recognition process is diminished if there are other competing paths.

- EXPAND: Perform the inference to create a new GOAL, and perhaps change the WANTCONTEXT and/or BELIEFCONTEXT registers. EXPAND also establishes a MERGE task for this new GOAL.

- MERGE: Check to see if GOAL was expected. If so, increase its RATING by 50%. Otherwise, diminish RATING by 5% allowing a breadth-first expansion. Create an INFER task on GOAL.

- ACCEPT: If there is a task on the agenda rated 60% of GOAL or better, then diminish GOAL's rating by 50% and spawn an ACCEPT-FINAL task for that GOAL (at that lower rating). This allows competitors to become better rated. If there are no worthy competitors, then plan-recognition terminates successfully, with GOAL as the last leaf.

- ACCEPT-FINAL: If there are no competing tasks within 80% of GOAL, then terminate processing (successfully). Otherwise, create a CLARIFY task with GOAL and the GOAL of the competitor as alternatives. (This could lead to a clarification dialogue. It has not been implemented yet).

The important heuristics embodied in the controller are these (as in [Allen, 1979]):

1. The plan the system is intended to recognize should be direct. If there are multiple mutually exclusive intended inferences applicable to some goal, then they all lose in rating. The system will attempt to recognize utterances indirectly until a "split" occurs with WANTCONTEXT SBUWSBUW. In that event, an earlier task would have been set up to interpret the utterance literally, thus causing speech act identification and a "shift" from intended plan recognition (SBUWSBUW) to simple plan recognition (in Context SBUW). This earlier task would then be favored because the ratings of the competitors in SBUWSBUW would have been diminished drastically. Some other more direct route can then be followed. If there are no other direct routes, then prior equal division of ratings may lead to clarification dialogue.

2. Expected goals (cf. [Wilensky, 1978; Schmidt, Sridharan, and Goodson, 1978]) are favored by a 50% increase in rating. This is the only way ratings can rise. Without expectations, only intended plans without competitors will lead to indirect interpretations. Expectations can be situational (as in Allen's train station domain), normative [Schank and Abelson, 1977; Schmidt, Sridharan and Goodson, 1978; Wilensky, 1978] or

conversationally acquired, as we show in the example (see also [Bruce, 1975]).

Given this system organization, we need only discuss the operators for manipulating the display before proceeding to the example.

5.8.4 Interfacing with the Graphics Subsystem

Our task of flexible natural language control over a display system requires the system to maintain a model of its own capabilities. In general, the system should also have a model of what the user thinks the system can do. For our current system, it is assumed that the user knows the definitions of the system's display acts. This assumption plays a decisive role in determining which action the user has requested the system to perform.

Our current system has a model of its capabilities as shown in Fig. 17. Part of the display action taxonomy is shown in Figure 18. The Concept DISPLAY models the action of presenting a displayable object onto a blank screen. Various subConcepts of DISPLAY are used:

DISPLAY-STATE, where the object is an ATN-STATE.
DISPLAY-ARC, where the object is an ATN-ARC.
DISPLAY-STATE-ARC-COLLECTION, where the object is a grouping of states and arcs (e.g., ones that would be used in parsing "the preverbal constituents").

DISPLAY(object) precond: DISPLAYABLE(object) body: CLEAR(SCREEN) DISTINGUISH(object, VISUALLY) effect: VISIBLE(object)	INCLUDE(object) precond: DISPLAYABLE(object) INVISIBLE(object) body: DISTINGUISH(object, VISUALLY) effect: VISIBLE(cbject)
HIGHLIGHT(object) precond: DISPLAYABLE(object) VISIBLE(object) body: DISTINGUISH(object, VISUALLY) effect: HIGHLIGHTED(object)	FOCUS(object) precond: DISPLAYABLE(object) VISIBLE(object) body: CENTER(object) effect: CENTERED(object)

DISTINGUISH(object, mode)
 	
precond: DISTINGUISHABLE(object, mode)
 	
effect: DISTINGUISHED(object, mode)
	EXCLUDE(object)
	
	precond: DISPLAYABLE(object)
	VISIBLE(object)
	
	effect: INVISIBLE(object)

	SEE(USER, object) precond: DISTINGUISHED(object,VISUALLY)

Fig. 17. The system's model of its display actions.

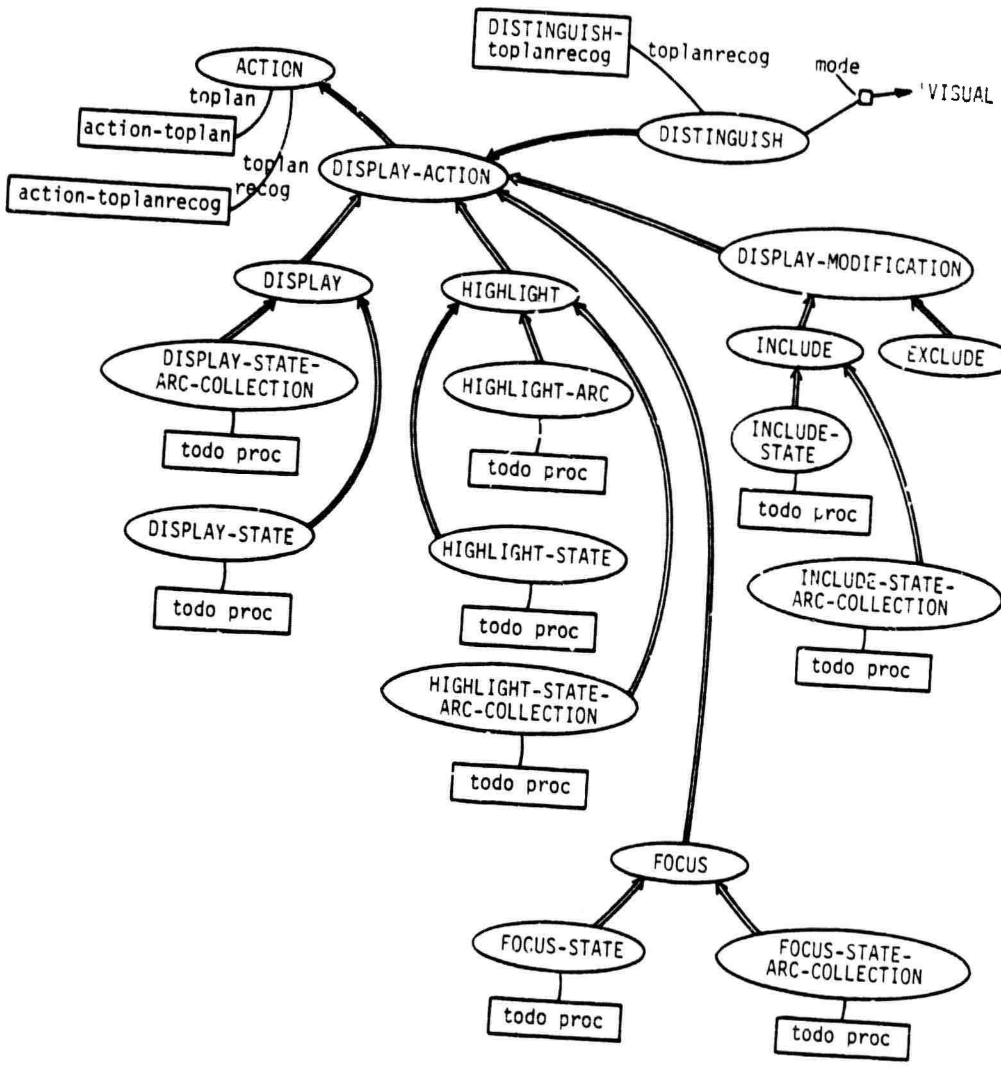


Fig. 18. Taxonomy of display actions.

Generic Concepts for displaying actions also have attached "todo" procedures that are used in invoking the graphics subsystem (Section 5.10.6). GRAPHICS produces the appropriate displays and returns to the DISPLAY procedure an indication of those states and arcs that are visible. Appropriate annotations of what is visible can then be made in the knowledge representation or in other efficient data structures.

Similarly, the system has an act to INCLUDE displayable objects onto the screen, which is subcategorized as INCLUDE-STATE-ARC-COLLECTION and INCLUDE-STATE. These acts are similar to DISPLAY, except that they do not require a clear screen; they instruct GRAPHICS to make the minimal modification needed to include the object (a STATE or STATE-ARC-COLLECTION) into the current display. The objects that were visible prior to an INCLUDE will also be visible afterwards (though this is implicit in the Concept definition). The system can center the display on a particular STATE or STATE-ARC-COLLECTION using the appropriate FOCUS act, can HIGHLIGHT STATES, ARCS, or STATE-ARC-COLLECTIONS that are already VISIBLE, and can EXCLUDE (erase) visible objects. DISPLAY, INCLUDE, and HIGHLIGHT use an action in their bodies called DISTINGUISH. When the user asks the system to show him something, the user wants the system to DISTINGUISH that thing from its "surroundings." If the object is not on the screen, making it visible will distinguish it (perhaps by doing a DISPLAY or an INCLUDE). Otherwise, when the screen is cluttered, highlighting may be necessary, or (in a future system) flashing may be required.

5.9 Example

The system's speech act identification process is best illustrated by the fourth sample sentence in our prototype dialogue (see Section 3.3) - "No, I want to be able to see S/AUX". The screen at that point in the dialog is shown in Fig. 19.[37] The sentence is analyzed as two speech acts - a REJECTion,[38] and an S-INFORM (USER, SYSTEM, (WANT USER (CANDO USER (SEE USER S/AUX)))).

These speech acts are processed sequentially. The REJECT sets up expectations that are used in analyzing the S-INFORM. On processing a REJECT, the system creates an expected goal that the user will want to modify the display. More specifically, a call to a search program - (JP NIL FIND AN INDIVIDUAL DISPLAY-MODIFICATION)[39] - is created and added to the list of EXPECTATIONS. This list, which is consulted during the MERGE

---

[37] Prior to uttering this sentence, the user had requested the system to "focus in on the preverbal constituents". The effect of that prior request was to produce the display of Fig. 19, which does not include S/AUX. Apparently, the user had expected that request to produce a display that included S/AUX.

[38] In the future, the "No" should be handled as a surface-rejection, since "No" could realize other speech acts, for instance a denial or a refusal.

[39] This call is expressed as an internal use of the JARGON parser via the function JP.

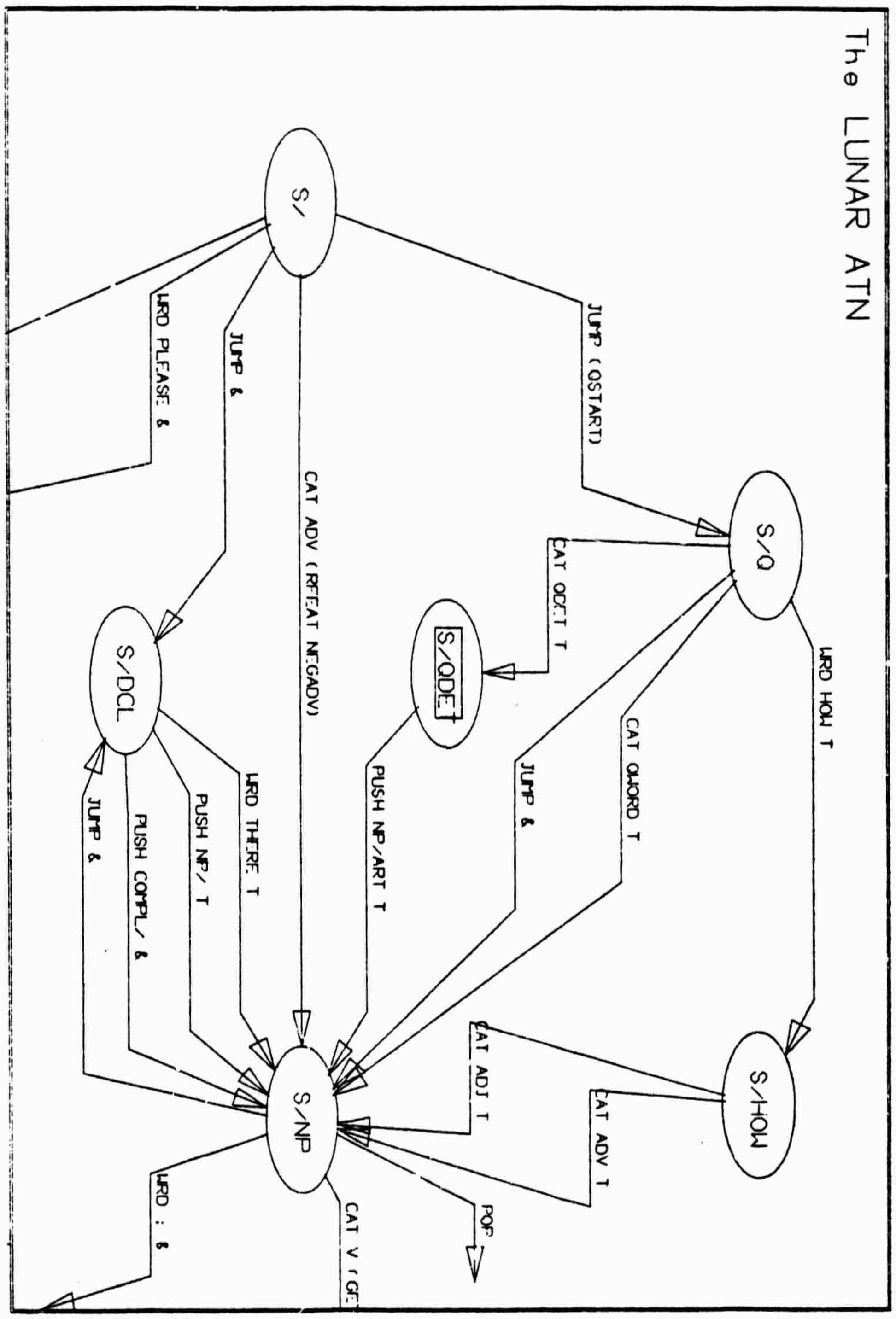


Fig. 19. The display scope after sentence 3.

state of the controller (see Section 5.8.3) is evaluated relative to the then current WANTCONTEXT to determine if the then current GOAL is expected. Those GOALs will be discovered during the processing of subsequent utterances.

5.9.1 "I want to be able to see S/AUX"

The parser's final structure for this utterance, which follows the rejection, appears in Fig. 20.

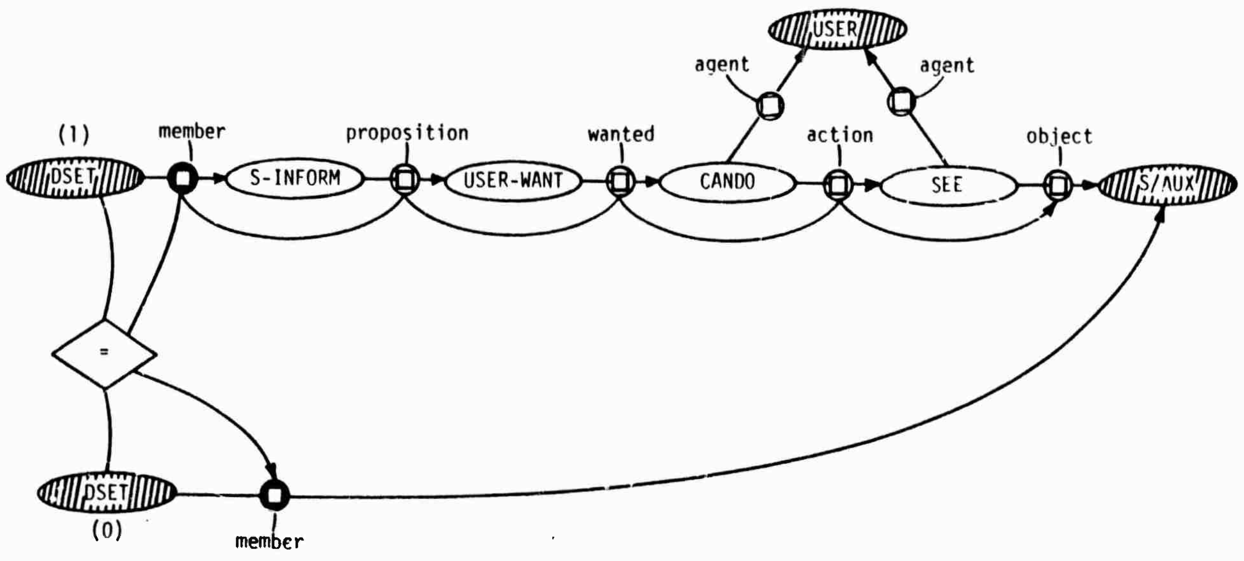


Fig. 20. Result of parsing "I want to be able to see S/AUX".

To begin the plan-recognition process, the parser invokes the inference controller with pointers to the 1) surface speech act found for the user's utterance, 2) Want Context SBUW, and 3) Belief Context SBUB. The controller, by placing the DSET (see

Section 2.9) corresponding to the user's surface speech act (DSET(1) in Fig. 20) into context SBUW, is initially processing under the assumption that the user wanted to perform the observed surface speech act. The agenda is configured to contain a task to INFER from this GOAL, with initial rating 100, WANTCONTEXT = SBUW, and BELIEFCONTEXT = SBUB.

Since SBUW is a recognition context, INFER attempts to find a plan recognition rule applicable to GOAL. As mentioned earlier, these inferences are implemented as procedures attached to various Concepts in the action/state hierarchy.

For the example, in creating the parse structure, the PSI-KLONE Interface created the Concept of an S-INFORM whose proposition is a WANT whose agent is the USER and whose wanted-concept is a CANDO, and linked it into a taxonomic lattice of speech act types. This made it a subConcept of an S-INFORM whose proposition is a WANT whose agent is the USER. Attached to this general Concept in the hierarchy is the S-INFORM-USER-WANT plan recognition procedure. Instead of calculating the effects of an S-INFORM, this procedure "compiles" a specific inference chain. In this case, the chain would have been

```
SBUW    S-INFORM (USER, SYSTEM, (WANT USER Q))
  |       effect
SBUW  SBUWSB   (WANT USER Q)
        | cause-to-want  (see Allen [1979])
SBUW  SBUWSW Q
```

That is, the system, is intended to think the user's goal is that the system wants Q (Q = USER can see S/AUX). The system's believing that the user wants some state of affairs is sufficient to cause it to want that state as well. The two-step chain is "short-circuited" by the S-INFORM-USER-WANT procedure, which expands the effect of S-INFORM to be SBUWSW Q. It returns a JARGON expression that, when evaluated by an EXPAND task, will place Q in WANTCONTEXT SBUWSBUWSW. This is currently an unconditional inference since our model of what it takes to get the system to want some goal is its simply believing the user wants it.[40] If there were any conditions under which the system could decide **not** to want some goal that the user desires, they would be checked before making this transition to SBUWSW Q. Short-circuiting therefore avoids having to search for applicable operators, and is similar to the use of MACROPS in STRIPS [Fikes, Hart, and Nilsson, 1972].

INFER places a task to EXPAND GOAL on the agenda; the WANTCONTEXT is shifted to SBUWSBUWSW, the BELIEFCONTEXT becomes SBUBSB (see Section 5.8.1.3 for justification of the use of this knowledge base), and the rating remains at 100.

---

[40] Future versions could check whether it is mutually believed that the goal in question does not conflict with a goal the system is already believed to have.

Since there are no other tasks on the agenda, this EXPAND is chosen, and the inference expression is applied to GOAL to produce a new GOAL, that the user CANDO SEE (represented by DSET(2), Fig. 21) in context SBUWSBUWSW. The new status of the interpretation is shown in Fig. 21.

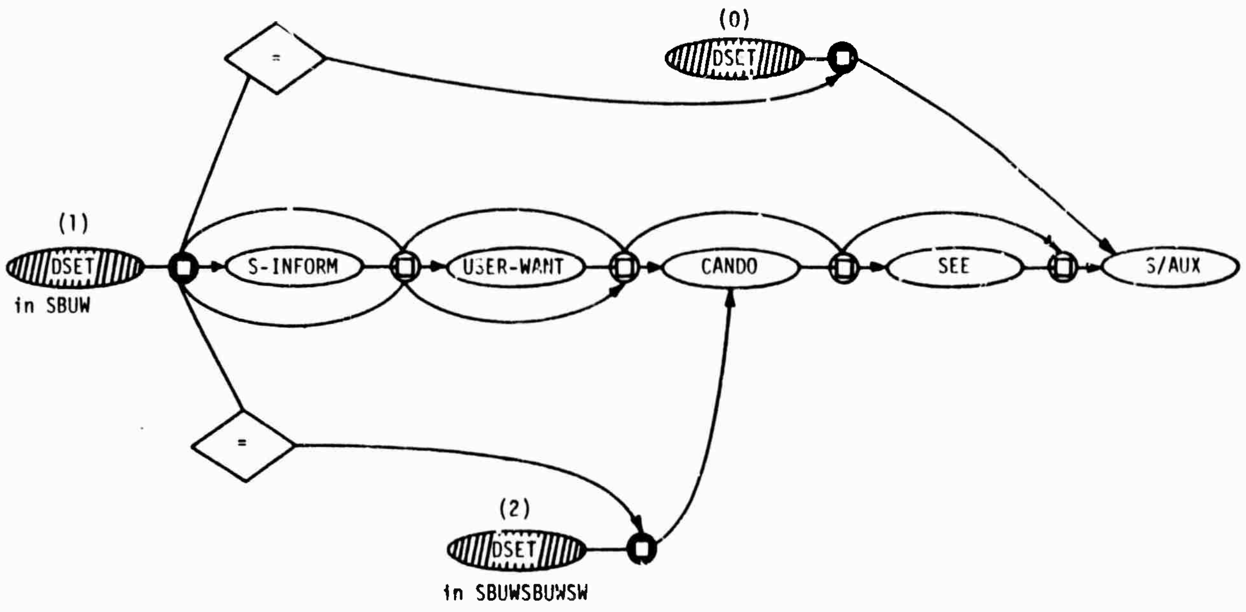


Fig. 21. Interpretation after the expansion of the S-INFORM.

EXPAND places a MERGE task on the agenda to see if this new GOAL is expected. Again, since no other tasks are on the agenda, the MERGE task is chosen, causing the expressions on the list of EXPECTATIONS to be evaluated relative to GOAL. None succeed since only modification acts are expected, and a new INFER task is spawned: RATING=95, GOAL=DSET(2), WANTCONTEXT=SBUWSBUWSW, and BELIEFCONTEXT=SBUBSB.

It is important to notice here that Context SBUBSBUWSW is a planning Context for the system. That is, the system will attempt to find actions that will achieve the GOAL of the user's being able to see S/AUX. Thus, INFER finds a planning procedure attached to the Generic CANDO that, when evaluated later by EXPAND, returns the precondition of the filler of the action Role of CANDO (i.e., the precondition of SEE). INFER suggests an EXPAND (RATING=95, GOAL=DSET(2), BELIEFCONTEXT=SBUBSB, WANTCONTEXT=SBUWSBUWSW, INFERENCE-RULE=(a JARGON expression to create a DSET of DISTINGUISHED whose object is (equated to be) S/AUX)).

EXPAND creates DSET(3) (in Fig. 22), and MERGE "passes", since S/AUX's being DISTINGUISHED was not an expected goal. Again, an INFER task is placed on the agenda: (RATING=91.3, WANTCONTEXT=SBUWSBUWSW, BELIEFCONTEXT=SBUBSB). INFER now finds a planning procedure attached to the generic DISTINGUISHED that returns a JARGON expression to insert a DISTINGUISH action into the WANTCONTEXT (see the domain operator definitions of Section 5.8.4).

EXPAND evaluates that JARGON expression and creates DSET(4) of DISTINGUISH in Context SBUWSBUWSW, and MERGE finds that DISTINGUISH is not expected (thus losing 5% of its rating of 86.7). At this point, there is a primitive action (i.e., one

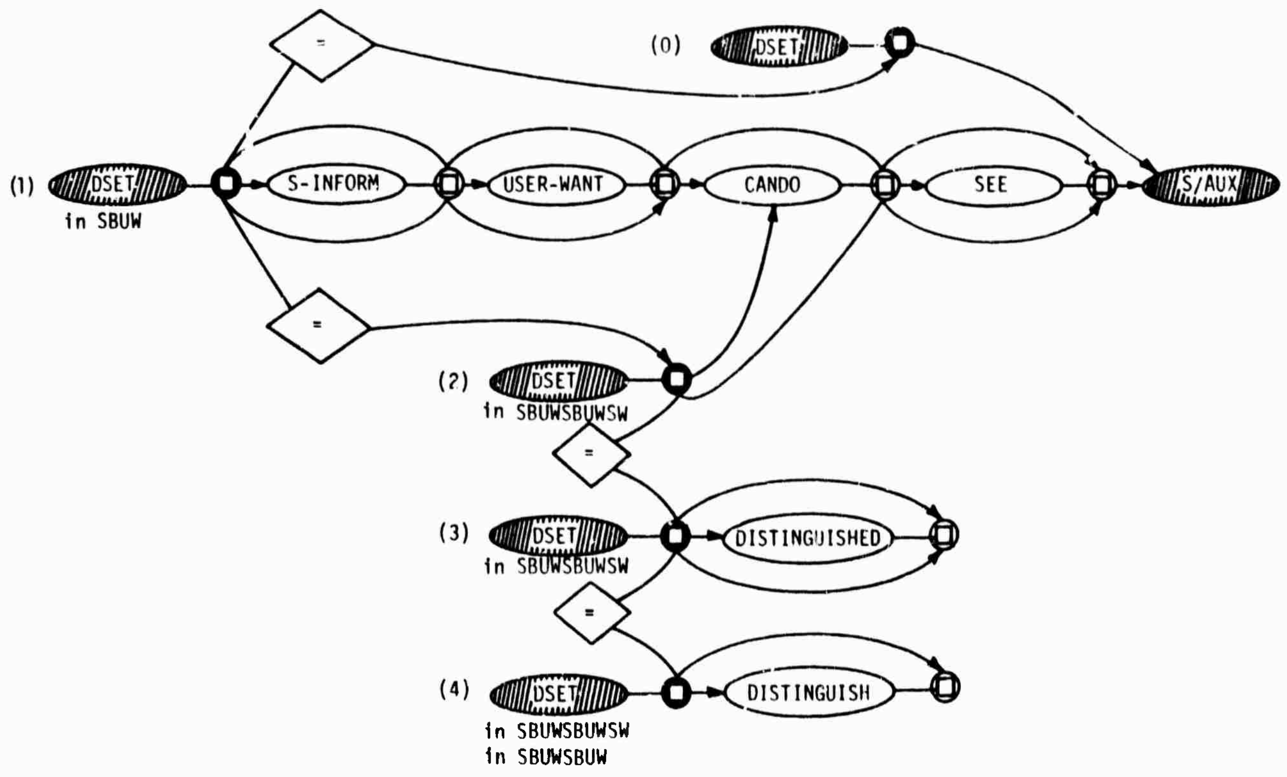


Fig. 22. Interpretation after recognizing the user intended for it to recognize it should plan to DISTINGUISH S/AUX.

having no "body") in want context SBUWSBUWSW. That is, the system has recognized the user wanted it to believe the user was planning for the system to want to perform a DISTINGUISH act on S/AUX. The next stage of the plan recognition process performs this inference: the user wants the system to want to do that action because the user wants it done - wanting the action is a precondition for doing it. Thus, DSET(4) is placed in Context SBUWSBUW by an INFER-EXPAND-MERGE cycle.

The agenda now contains an INFER task, GOAL=DSET(4), WANTCONTEXT=SBUWSBUW, BELIEFCONTEXT=SBUB, RATING=82.4. SBUWSBUW is a plan-recognition Context, INFER invokes a plan-recognition procedure attached to the generic DISTINGUISH. Figure 17 shows t DISTINGUISH is the body of actions HIGHLIGHT, DISPLAY, and INCLUDE. Ideally, hierarchical plan recognition should find those acts by a "body-inference" (as in Allen [1979]), and then rate th by the truth/falsity of their preconditions. Since there is as yet no general KL-ONE matcher, the body inferences are currently encoded in the plan-recognition procedure for DISTINGUISH.

That procedure tests the precondition for HIGHLIGHT - whether or not the filler of the object Role of DISTINGUISH is DISPLAYABLE and VISIBLE.[41] If S/AUX were visible (it's not), the procedure would return a JARGON expression to insert a HIGHLIGHT action into the Want Context. Instead, it returns two potential inferences - one to insert a DISPLAY of S/AUX and one to insert an INCLUDE of S/AUX. Since these are mutually exclusive intended inferences, each gets half the rating (82.4/2 = 42.1) and a copy

[41] This testing is performed by a "to test" procedure [Levesque, 1977] attached to VISIBLE, which traces through the equated RoleValueMaps to find the description used in the original utterance (DSET(0) in Fig. 20). The system thus tries to find coreferential descriptions only when it has to (since some descriptions are not intended to refer, as in "Call that ne map of the Mediterranean").

of the Want Context SBUWSBUW (to allow separate plans to be inferred). Each inference rule carries a rating adjustment, usually 1, that is multiplied by the current GOAL's RATING in determining the RATING of the resulting GOAL. In the absence of any other information, the system would prefer to DISPLAY (using a 25% higher rating) rather than INCLUDE.[42] This preference combines with the rating "split" to give the INCLUDE alternative a rating of 31.8.

The agenda now contains:

```
EXPAND - RATING = 42.4

(1)       GOAL    =   DSET(4)
          INFERENCE  = a JP to create a DISPLAY OF S/AUX
          WANTCONTEXT = SBUWSBUW'
          BELIEFCONTEXT = SBUB
EXPAND - RATING = 42.4 X 0.75 = 31.8

(2)       GOAL = DSET(4)
          INFERENCE = a JP to create an INCLUDE of S/AUX
          WANTCONTEXT = SBUWSBUW"
          BELIEFCONTEXT = SBUB
```

EXPAND(1) is chosen, since it is rated higher, and inserts DSET(5) (see Fig. 23) into Context SBUWSBUW' (a copy of SBUWSBUW). Again, MERGE does not find this DISPLAY act to be

[42] It is for this reason that "Show me the clause level network" is treated as a DISPLAY action.

expected (since it is not a DISPLAY-MODIFICATION), so it loses another 5% in rating and an INFER task is placed on the agenda. The agenda is now:

```
INFER - RATING   =   42.4 - 5% = 40.3
          GOAL = DSET(5)
          WANTCONTEXT = SBUWSBUW´
          BELIEFCONTEXT = SBUB

EXPAND - RATING = 31.8
          GOAL = DSET(4)
          INFERENCE = a JP to create an INCLUDE of S/AUX
          WANTCONTEXT = SBUWSBUW"
          BELIEFCONTEXT = SBUB
```

INFER finds a "body" plan-recognition rule applicable to the DISPLAY action in Context SBUWSBUW´, namely that of REQUEST.[43]

The body of REQUEST "absorbs" a level of SBUW, leaving a proposition SBUW REQUEST(USER, SYSTEM, DISPLAY(SYSTEM, S/AUX)).[44]

The agenda now contains:

[43] In addition, a more comprehensive system would expand DISPLAY by its effect and try to infer further.

[44] A body inference was not found earlier (for the DISTINGUISH action) since there were other intended inferences that could have been taken (applying Allen and Perrault´s "level of inference" heuristic).

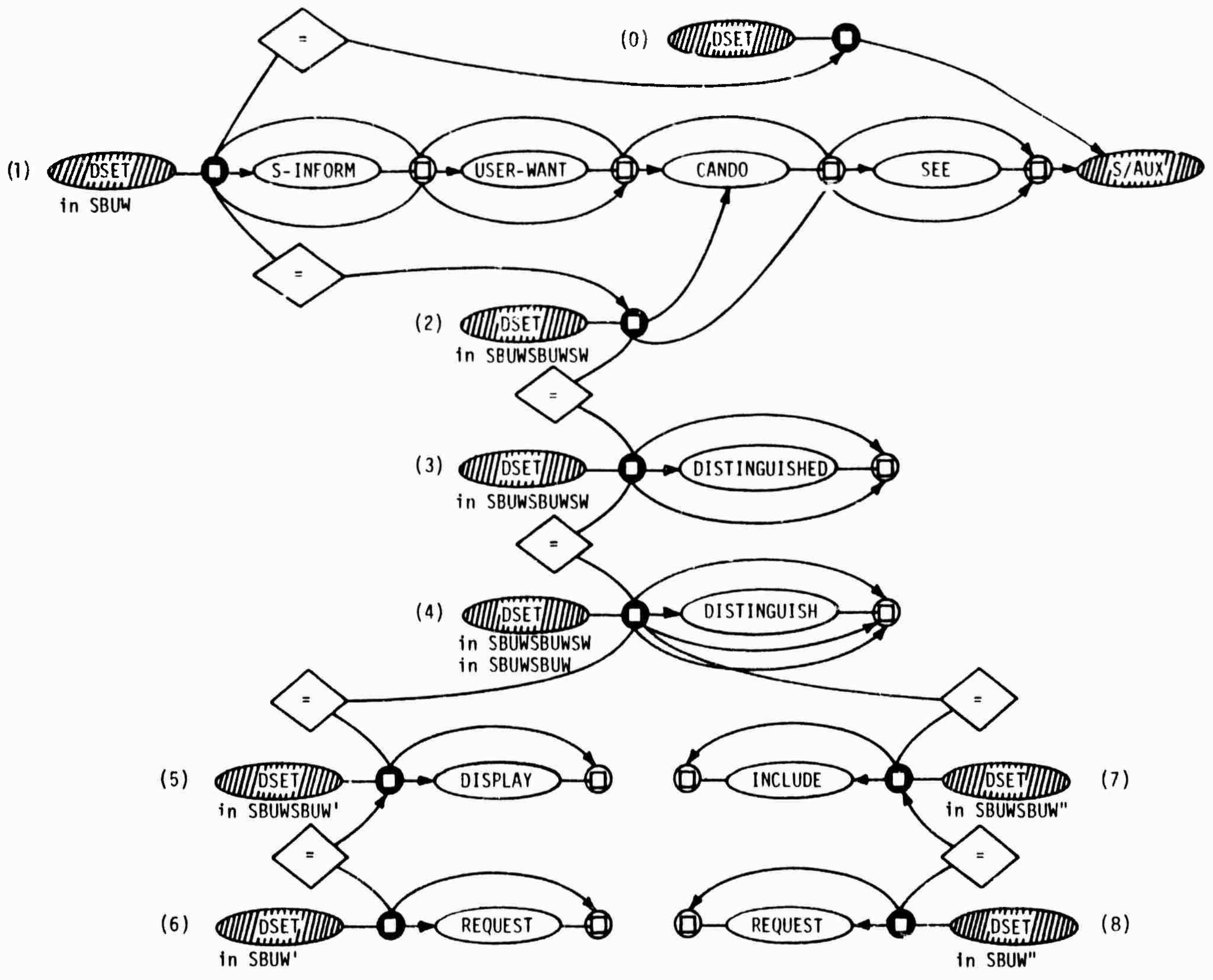


Fig. 23. The alternatives of DISPLAYing or INCLUDing.

```
EXPAND - RATING = 40.3
 (3)     GOAL = DSET(5)
         INFERENCE = a JP to create a REQUEST to DISPLAY S/AUX
         WANTCONTEXT = SBUW´
         BELIEFCONTEXT = SBUB

EXPAND - RATING = 31.8
         GOAL = DSET(4)
         INFERENCE = a JP to create an INCLUDE of S/AUX
         WANTCONTEXT = SBUWSBUW"
         BELIEFCONTEXT = SBUB
```

Summarizing the next few steps, EXPAND(3) is taken next. It creates the REQUEST to DISPLAY, which is not expected, beginning an INFER cycle again from REQUEST. The plan-recognition procedure for REQUEST short-circuits two inferences (the effect of REQUEST and the want precondition for the requested act) and inserts the DISPLAY act into SBUW´ (a copy of SBUW).

Finally, since the system can infer no more along this path, INFER suggests an ACCEPT task to terminate processing. The agenda is

```
ACCEPT - RATING = 38.2 (= 40.3 - 5%)
         GOAL = DSET(6) (of REQUEST to DISPLAY)
         WANTCONTEXT = SBUW´
         BELIEFCONTEXT = SBUB

EXPAND - RATING = 31.8
 (2)     GOAL = DSET(4)
         INFERENCE = a JP to create an INCLUDE of S/AUX
         WANTCONTEXT = SBUWSBUW"
         BELIEFCONTEXT = SBUB
```

ACCEPT terminates plan-recognition if there are no competing tasks within 60% of its score. Since this is not the case here, the rating of this alternative is halved (to 19.1) in order to allow other competitors the opportunity to complete the process. If the controller ever returns to this alternative again (in state ACCEPT-FINAL), other competitors will only be viable if they are within 80% of its score. Thus, the agenda becomes:

```
EXPAND - RATING = 31.8
 (2)     GOAL = DSET(4)
         INFERENCE = a JP to create an INCLUDE of S/AUX
         WANTCONTEXT = SBUWSBUW"
         BELIEFCONTEXT = SBUB

ACCEPT - RATING = 19.1
         GOAL = DSET(6)
         WANTCONTEXT = SBUW'
         BELIEFCONTEXT = SBUB
```

The remainder of the process proceeds as before, except that the INCLUDE that is created by EXPAND(2) is expected. Consequently, the rating is increased by 50% to 47.7. When the controller tries to ACCEPT this goal, the other alternative is well below the 60% threshold. Plan-recognition thus halts, having determined that the user has requested the system to INCLUDE S/AUX into the display. The system then executes the "to do" procedure associated with the INCLUDE-STATE Generic, which instructs the display manipulation program to perform the requisite operations[45] [46]. The screen is then reconfigured (see Fig. 24), the lists of visible states and arcs are updated, and the satisfied expectation is deleted.

The speech act interpretation process has thus recognized "I want to be able to see S/AUX" as a REQUEST to INCLUDE by recognizing that it was intended to **plan** an INCLUDE to make S/AUX "distinguished". The choice of INCLUDE was recognized as intended to be recognized since visibility information and the display action definitions are assumed to be mutually believed. The choice of INCLUDE over DISPLAY was made by virtue of expectations that were generated by the first half of the utterance ("No"). Had the user said "Now" instead of "No", the system would have DISPLAYed S/AUX by itself, provided "Now" were implemented as a topic-switching speech act that erases conversationally-induced expectations.

---

[45] Again, the system must trace through the equated RoleValueMaps to find the (original description of) the object that was requested to be displayed. Naturally, it is discovered to be the same as what the user said s/he wanted to be able to see.

[46] At this point, a more comprehensive system would examine the resulting plan, detect obstacles, and decide which act(s) to perform (cf. [Allen, 1979]).

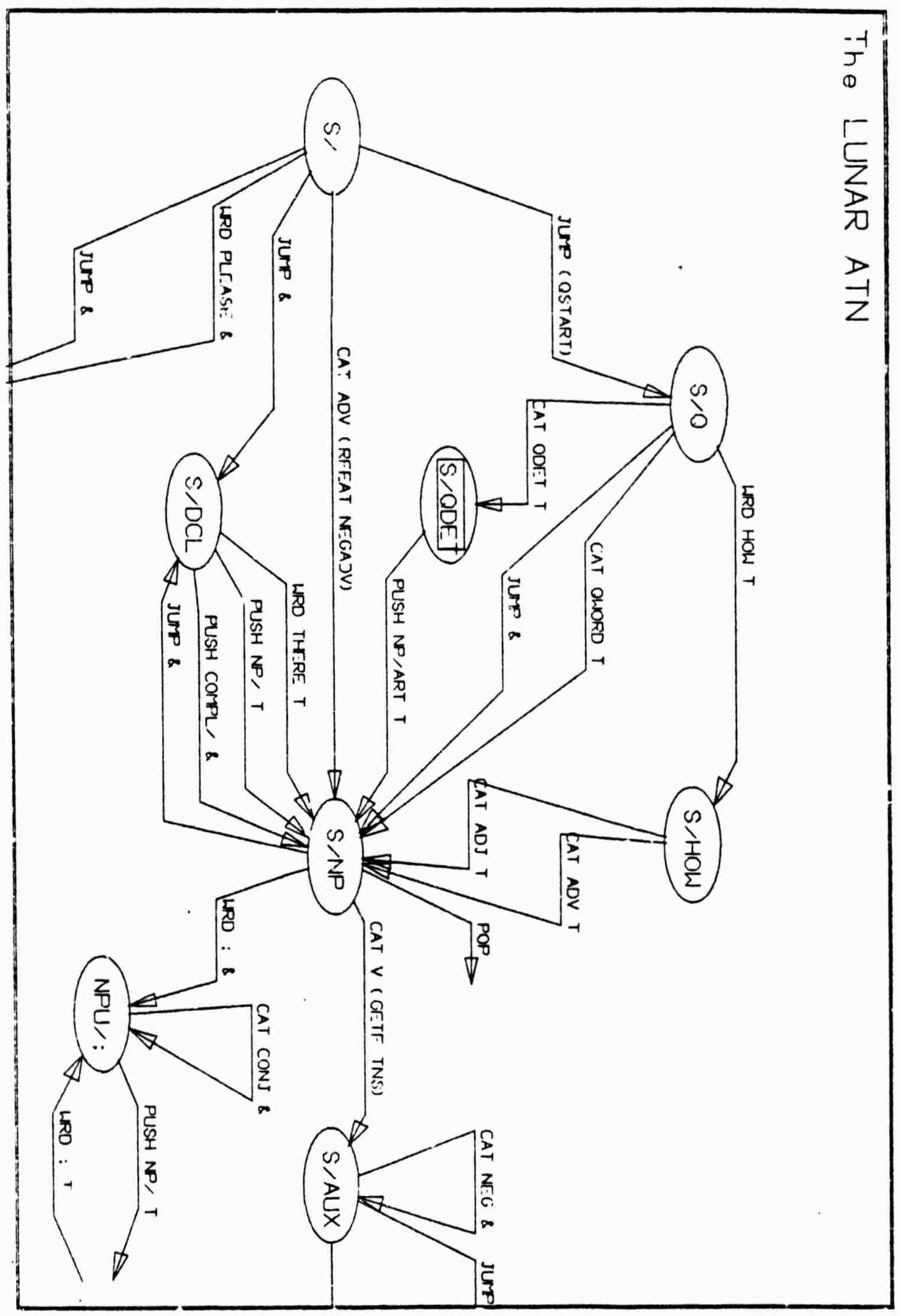


Fig. 24. The screen after sentence 4, "No, I want to be able to see S/AUX."

5.10 Limitations and Extensions

The current system has a number of theoretical and practical limitations. They are discussed here since they suggest obvious directions for further research. Practical limitations that arose because the current system was intended to be a "concept demonstration" will not be discussed.

5.10.1 The "Cycle" and Goal Recognition

As Perrault, Allen, and Cohen propose, helpful computer conversants should operate according to the following cycle:

Observe the user´s action.

Recognize the user´s plan.

Discover obstacles in the inferred plan by evaluating its steps against the system´s beliefs.

Adopt (at least one of) those obstacles as the system´s goal.

Plan to achieve that new goal.

Execute that plan, and go to step 1.

Currently, our system accomplishes step 2, but in a way that does not easily allow for the other capabilities. The system is inferring the user´s goals, but not linking those goals to the "premises" from which they were inferred. For example, given that the user wants to perform a particular action A, the system might infer that the use wants its effect E. However, the system does not explicitly represent that wanting E was inferred from wanting A (ard not from wanting B) because of its being A´s effect. Thus, the system does not recognize plans per se, but rather, goals.

Without the explicit intergoal "linkages", the system cannot examine or debug the user´s plans to be helpful. Previous implementations [Allen, 1979; Cohen, 1978] have made use of those linkages, and Allen was able to construct a helpful system of this sort, using concepts like "Enable", "Produce", etc.

However, there are still problems with such relationships. For instance, are the arguments to PRODUCE (as an action produces a certain state of affairs) both propositions, or an action and a proposition? If (an instantiated) action A´s effect is E, and the planner wants/needs E, and the planner then decides to want/need A, do we want to say that the planner wants PRODUCE(A,E) or PRODUCE (A(planner), E)?[47]

5.10.2 Expectation

Bruce [1975] suggested that for many domains one might find typical speech act sequences, from which one could derive

[47] This problem is related to the philosophical controversy over "intend" [Searle, 1979].

expectations about the future progress of domain-related conversation (see also [Horrigan, 1977; Mann, Moore, and Levin, 1977; Schank and Abelson, 1977]). The existence of such speech act sequences was assumed as our basis for the current system's expectation that after a rejection of the system's display act, the user will want to modify the display. However, the system creates an expectation as a call to a search program, which is attached to a list (EXPECTATIONS) that is examined during the MERGE state of the controller. Clearly, this is insufficient - EXPECT should be treated as a modal operator, perhaps equivalent to believing that some proposition will be true. Since we are dealing with the user's goals, expected goals will be beliefs that the user will want something. Future work should be addressed to representing beliefs and time, as a basis for a principled representation of expectations. Rules should be written to describe the behavior of EXPECT to create particular expectations, and to describe better how the recognition of goals that were expected should influence plan recognition.

5.10.3 Reasoning about Belief

As yet, no attempt has been made to reason about the system's and user's beliefs. Rather, the system, as conceived, is equipped to retrieve information from the network or other data structures. Future work should be addressed to this problem, perhaps following the approach of Moore [1979]. The belief mechanism might also employ a "truth maintenance" scheme [Doyle, 1978; McAllester, 1978] for keeping track of changes to the system's and users' beliefs. Such a scheme should also be tied to a belief/time representation and logic.

5.10.4 Planning and Plan Recognition

The system's planning/plan-recognition capabilities, though complex in the structures manipulated, were otherwise overly simple. Better models of the display actions need to be developed, including analyzing some display actions as the system's means for demonstrative presentation. The system will be required to use models of planning and plan recognition in order to map yes/no questions into their intended interpretations (handled by Allen and Perrault's KNOWIF(P) -> P inferences).

5.10.5 Language Generation

The system does not plan any speech acts for the current examples. After planning to the level of a surface speech act, the problem of actually producing utterances remains. Given a surface form the system must determine the content, and whether or not to transform that surface form in some way (e.g., for reasons of focus, [Grosz, 1977; Sidner, 1979]). The former involves the planning of referring expressions [Clark and

Marshall, in press; Perrault and Cohen, in press] and the choice of words. Regarding the latter, an algorithm like McDonald's [1978] may serve as a basis for utterance generation since it requires left-moving transformations to be pre-planned.

5.10.6 The Form of Speech Act Definitions

Finally, the speech acts need to be redefined. The current definitions embody Grice's [1957], Strawson's [1964], and Searle's [1969] recognition of intention conditions for communication. However, in a series of counterexamples, Schiffer [1972] showed that this kind of definition of communication is insufficient in that it (ultimately) requires an infinite regress of "intending that you recognize my intention that you recognize...to achieve P". Schiffer proposed instead that the recognition of intention be shared (mutual) knowledge. Following Schiffer, Perrault and Allen [forthcoming] suggest their definitions be refined to incorporate mutual beliefs. Cohen and Levesque [1980] use their idea to redefine surface speech acts, and show that illocutionary act definitions (e.g., REQUEST) can be **derived** as summaries of shared plans (not just goals). Summaries can be used to formalize our "short-circuiting" of inference paths.

In summary, the current system is able to analyze a subset of the class of indirect speech acts discussed in [Allen, 1979], using a combination of "short-circuited" inference procedures and more general goal/plan recognition. As the representational capabilities of KL-ONE evolve, other kinds of plans and speech acts (direct and indirect) will be generated and recognized. Our primary concern for the future is to incorporate the remainder of the "cycle" of cooperative problem-solving behavior in order to detect obstacles and be helpful, to model linguistic and demonstrative reference acts, and to interface with a representation of discourse focus [Sidner, 1979].

6. THE GRAPHICS DISPLAY COMPONENT

The application domain for the demonstration system, as discussed previously, is an isomorph of the command and control map display domain, but instead of a map uses a diagram of an ATN grammar. Under this isomorphism, states in the grammar correspond approximately to cities or ports and connecting arcs correspond to roads or routes. States are labeled with state names and arcs are labeled with various information depending on the kind of arc, the conditions and actions that may be associated with them, and a selected level of detail. This display of an ATN grammar can be used by a grammar designer in extending or debugging a grammar in a way roughly similar to the use of a map display system by a commander - that is, as a problem solving tool to present various portions of the map, to make different kinds of details visible or invisible, to access portions of the display by specification of content, etc. The focus of our research interests lies in the ways that a user will express such requests and the theory and technical devices necessary to handle them.

Both the knowledge of the ATN grammar to be displayed and the general knowledge of how to display ATN grammars are represented in KL-ONE. Moreover, the knowledge of how to display grammars is built on a KL-ONE representation of general knowledge about displaying objects on display surfaces. This body of general graphics knowledge was developed under a separate project [Zdybel, Yonke, and Greenfeld, 1980]. Thus, the knowledge base of the system consists of an integrated knowledge representation of facts about displaying things in general, displaying ATN grammars in particular, and knowledge of a particular ATN grammar to be displayed. Procedures attached to the KL-ONE network representing this information take care of the actual drawing of objects on the display screen and are associated with those objects by the KL-ONE mechanisms of inheritance.

6.1 Graphical Representation of ATN Grammars

In this section we will describe the representational conventions which are used to depict ATN grammars on a graphical display. The objects to be displayed consist essentially of states and arcs in an ATN grammar. In graphic form, a state is represented by an ellipse with the name of the state inside it, and has a coordinate position in addition to its other characteristics as an ATN state. Similarly an arc has positional information as well as its normal information as an ATN arc. For this implementation, we have selected a simple but convenient set of conventions for the graphic representation of an arc. Specifically, each arc that connects two states (called a ConnectingArc) has three segments. The middle segment is always horizontal and contains the label information associated with the arc; the other two segments connect the middle one with the states at the head and tail of the arc. Pop arcs, which have a source state but do not connect to another state, are represented by two segments, the second of which is horizontal and contains the label information, while the first connects to the source state.

The coordinates used for the position of a state are the coordinates of the center of the ellipse used to represent it. An arc segment connected to a state is aligned with the center of the ellipse but terminates at the point of intersection with it. To indicate the directionality of an arc, it is drawn so that the final segment of the arc has an arrowhead which points to the destination state (or away from the source state in the case of pop arcs). On each arc, the text used to identify the arc (the arc label, etc.) is positioned so that its left most character appears just above the leftmost end of the middle (horizontal) arc segment. Notice that this choice of positioning does not require any additional positional information for the text.

An interactive program has been written to facilitate the layout of an ATN grammar on a display surface. This program takes an ATN grammar in conventional notation (without any graphical information) and prompts a user for each state and each arc, allowing him to specify with a pointing device the location of the state and the locations of the ´knees´ of an arc. The user can defer consideration of any state or arc for later and can interactively specify what state or arc he wishes to place instead of responding to a prompt. He can also, if he wishes, dynamically add new states and arcs that are not in the original grammar, and he can pick up and move states and arcs that he has already placed as often as necessary to achieve a satisfactory layout. This program can thus serve as an interactive graphics ATN editor as well as an initial layout aid.

The location of an arc is specified relative to the center of the source state. This is true for both pop arcs and connecting arcs. It has the advantage of permitting a convenient mass movement of states by changing only the position of the states themselves. Specifically the location of one end of the middle section of the arc is specified by coordinates relative to the center of the source state. The length of the horizontal arc section is specified by a length. If the length is negative, the middle arc section is drawn from right to left.

The text that is used to identify an arc is determined at the time that the arc is initially placed (during the grammar layout). The amount of text is determined dynamically by changing the depth of print level used in the printing of the text so that the total length of the resulting text does not exceed the length of the middle arc segment.

One special device for the graphical representation of ATN's on a two dimensional surface is the concept of a virtual state (not to be confused with virtual arcs, which are a feature of ATN grammars in general). A virtual state is essentially a connector which allows one to show arc connections between states without excessive crossings of arcs. It permits a local reference to a state that is physically located somewhere else on the display surface. Any state in the pictorial representation of an ATN can have any number of virtual copies located at different points on the display surface, and arcs entering or leaving that state can be represented pictorially as entering or leaving either the state itself or any one of its virtual copies. This is frequently used so that an arc leaving a state in one region of the ATN and entering a state that is considerably removed from that region can be shown as entering a virtual copy of that state which is located conveniently in the region of the arc. Virtual states are shown in our pictorial representation as double ellipses containing the name of the state of which they are a virtual copy.

Although the graphics package that draws ellipses requires a specification of the major and minor axes of the ellipse, the

KL-ONE structure for the display forms of states specifies this information in terms of Roles for size and shape, where the size is the length of the x-axis and the shape is the ratio of the y-axis to the x-axis. Since it was decided that all states displayed would have the same basic shape, this permits states of different sizes by changing only one Role value.

The sizes of states are determined by a set of default conventions that are governed by inheritance mechanisms in KL-ONE. If a real state has explicit size or shape, these will take precedence. If a real state has no explicit size or shape the corresponding default values from the Concept State are used. If a virtual state has explicit size or shape roles, then they will take precedence. If a virtual state has no explicit size or shape, the corresponding default values to be used are those of its corresponding real state. This is accomplished by the execution of a default function which is hung on the Generic Concept for a virtual state as an attached datum. The defaulting function looks up the structured hierarchy one level at a time. If a Concept has an attached datum called "DefaultValue", then the value stored there is used. If a Concept has an attached datum called "DefaultFunction", the function stored there is evaluated and its value is used.

Figure 25 gives an example of a fragment of an ATN grammar

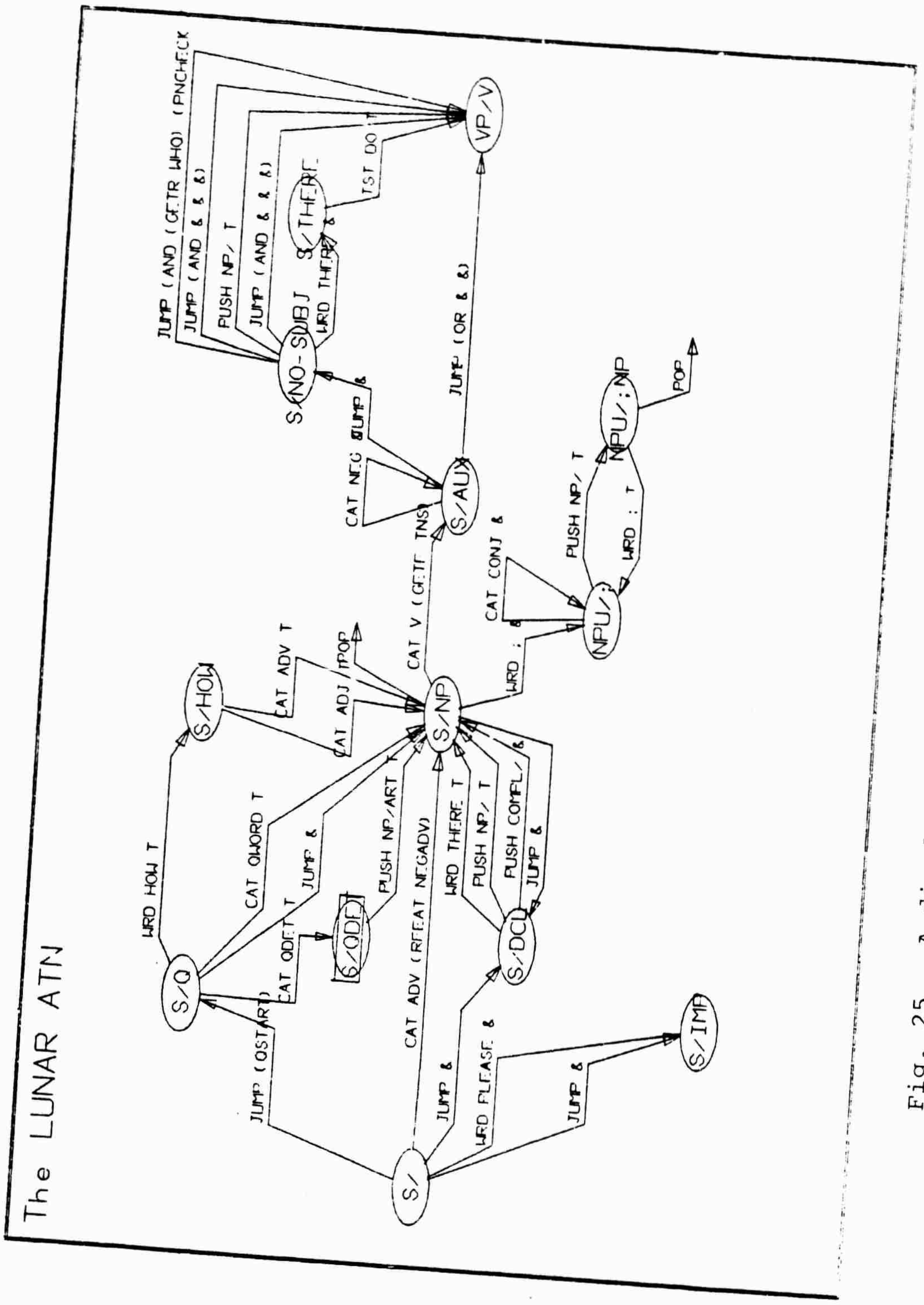


Fig. 25. A display of a fragment of an ATN grammar.

laid out with these conventions (it is also the output resulting from the first question in our sample dialogue - see Section 3.3).

A system has been written to take the same display specifications that drive the bitmap display and use them to drive a Calcomp plotter in order to obtain hard copies of the displays. This facility was used to produce the figures of the displays in this report.

6.2 The Graphical Display Knowledge Base

In the internal KL-ONE representation, the graphic information associated with the ATN is expressed by giving each state and arc a Role called **DisplayForm** which in the case of a state requires an ellipse and in the case of an arc requires a collection of three connecting lines. The information that the system knows about states and arcs and their associated display forms can be expressed by the following JARGON statements (recall that JARGON uses the special determiner PL to indicate plural nouns rather than using inflections on the nouns):

```
(A State IS AN ATNConstituent)
(AN Arc IS AN ATNConstituent)
(AN ATNConstituent HAS A DisplayForm)
(THE DisplayForm OF A State IS A StateDF)
(A State HAS PL Arc WHICH ARE PL Arc)
(AN Arc HAS PL Comment, A Test, AN ActionList,
 AN ArcType, AND A SourceState)
(ITS Comment IS A LISTP)
(ITS Test IS A LISTP)
(ITS ActionList IS A LISTP)
(ITS ArcType IS A STRINGP)
(ITS SourceState IS A State)
(ITS DisplayForm IS AN ArcDF)
(AN Arc WHOSE ArcType is "Pop" IS CALLED A PopArc)
(ITS DisplayForm IS A PopArcDF)
(IT HAS A BuildForm WHICH IS A LISTP)
(A ConnectingArc IS AN Arc)
(IT HAS A NextState WHICH IS A State)
(ITS DisplayForm IS A ConnectingArcDF)
(A ConnectingArc WHOSE ArcType IS "Jump" IS CALLED
 A JumpArc)
(A ConsumingArc IS A ConnectingArc AND HAS A Label)
(A ConsumingArc WHOSE ArcType IS "Vir" IS CALLED
 A VIRArc)
(IT HAS A PushState WHICH IS A State)
(ITS PushState IS ITS Label AS A ConsumingArc)
(AN InputConsumingArc IS A ConsumingArc)
(AN InputConsumingArc WHOSE ArcType IS "Push" IS CALLED
 A PushArc)
(IT HAS A PushState WHICH IS ITS Label AND IS A State)
(AN InputConsumingArc WHOSE ArcType IS "Cat" IS CALLED
 A CatArc)
(IT HAS A Category WHICH IS ITS Label AND IS AN ATOM)
(AN InputConsumingArc WHOSE ArcType IS "Wrd" IS CALLED
 A WrdArc)
(IT HAS A Word WHICH IS ITS Label)
(AN InputConsumingArc WHOSE ArcType IS "Mem" IS CALLED
 A MemArc)
(IT HAS A ListOfWords WHICH IS ITS Label)
(AN InputConsumingArc WHOSE ArcType IS "Tst" IS CALLED
 A TstArc)
(IT HAS A Test WHICH IS ITS Label)
```

```
(AN Ellipse IS A DisplayFormPrimitive AND HAS A Center
 (WHICH IS A 2DimensionalPosition), A Size (WHICH IS A
 NUMBERP), AND A Shape (WHICH IS A NUMBERP))
(AN ATNDF HAS AN Object WHICH IS AN ATNConstituent)
(A StateDF IS AN ATNDF AND HAS A State WHICH IS
 ITS Object AS AN ATNDF)
(ITS State IS A State)
(A StateDF IS ALSO AN Ellipse)
(A RealStateDF IS A StateDF)
(A VirtualStateDF IS A StateDF)
(AN ArcDF IS AN ATNDF AND IS ALSO A ConnectingLine)
(IT HAS A SourceState WHICH IS ONE OF ITS
 PL DisplayForm AS A ConnectingLine)
(ITS SourceState IS A StateDF)
(IT HAS A DeltaX (WHICH IS A NUMBERP), A DeltaY
 (WHICH IS A NUMBERP), A Length (WHICH IS A NUMBERP),
 AND A Text (WHICH IS A STRINGP))
(IT HAS AN Arc WHICH IS AN Arc AND IS ITS Object
 AS AN ATNDF)
(A ConnectingArcDF IS AN ArcDF AND HAS
 A DestinationState WHICH IS ONE OF ITS PL DisplayForm
 AS A ConnectingLine)
(ITS DestinationState IS A StateDF)
(A PopArcDF IS AN ArcDF)
```

Figure 26 shows a pictorial representation of most of the information contained in the above JARGON sentences (although some of the Value Restrictions have been removed and the Concepts for Ellipse and ConnectingLine have been removed for clarity). Note how statements about inherited Roles (such as the **DisplayForm** Role which an Arc gets by virtue of being an ATNConstituent) create local Roles that are linked to their parent Roles by Mods or Diffs links.

The information for actually drawing pictures of states and arcs is inherited from procedures attached to the display form primitives Ellipse and ConnectingLine, which are part of the AIPS graphics package [Zdybel, Yonke, and Greenfeld, 1980].

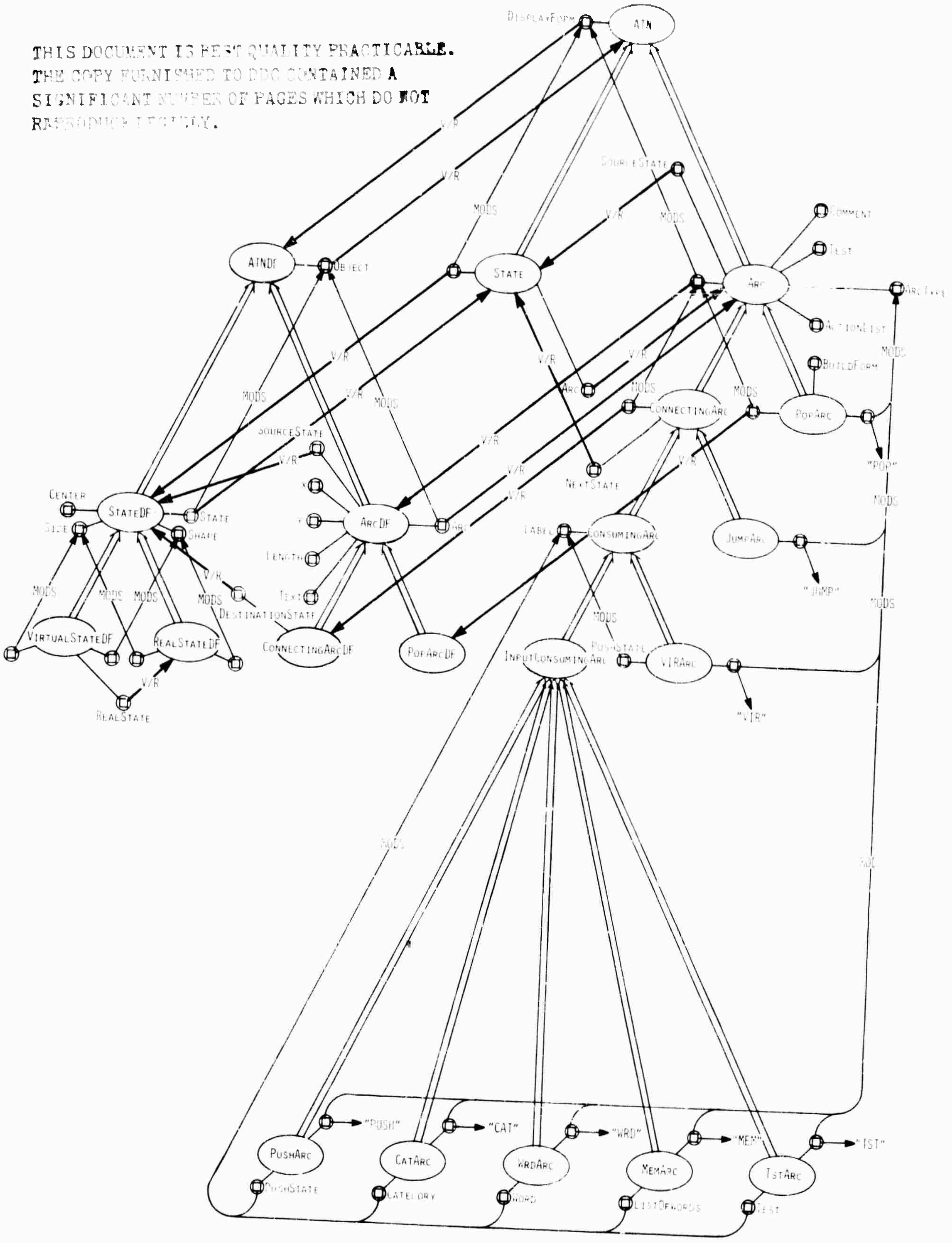


Fig. 26. KL-ONE Generic Knowledge about ATN Grammars.

6.3 Interactions with the Graphics System

The graphics display system is intended to respond to the requests of a user to show different portions of the grammar, to make states and arcs visible or invisible, to highlight specified states and arcs, and to move states and arcs from one point to another. Many of these operations are ones that would in fact be most conveniently performed with special display commands, light buttons, and pointing. They are provided for in English only for the sake of completeness. (The system also provides ways of doing them without use of English.) Other commands (such as requests to include states that are not currently visible on the screen) cannot be handled by pointing alone. These require a capability for retrieval by means of linguistic expressions. Highlighting is another example where pointing is not sufficient, since highlighting is usually used to find an object on a cluttered screen when one knows its name but does not know where it is on the screen. One of the major capabilities of the system with respect to linguistic input is the ability to construct a display to satisfy specified constraints. An example of this is a request for a display that will make a descriptively specified set of states visible.

The graphics system supports three basic kinds of operations: windowing, visibility setting, and highlighting.

Windowing programs determine what portion of the ATN is to be visible on the screen at any point in time. Thus if the states and arcs that the user wishes to see on the screen are known, it is the job of the windowing programs to determine a mapping such that those states and arcs (and not much else) will be visible. The algorithm for determining the windowing function is based on a three step procedure:

1. Determine the coordinates of the smallest rectangle which encloses the requested visible states.
2. Using the knowledge of the screen's aspect ratio, expand either the x or y coordinates of the rectangle so as to be the same shape as the monitor's screen surface.
3. Scale the area of the ATN world within the new rectangle in conjunction with some minimum padding constraints so that the states and arcs in the ATN world essentially fill up the screen. (The padding constraints insure that enough room is left around the edges of the display so that objects occurring at the edges of the display are not cut off.)

The windowing programs are used to determine the specific window parameters necessary to insure that a specified set of visibility constraints expressed by the user will be satisfied. They are used to determine the initial window for a display request, to respond to requests to focus on a subportion of a display, and to compute revised parameters in response to requests to include additional states in the display.

Visibility of states and arcs is handled by the specification of visibility lists and a system variable used to determine under what default conditions an arc ought to be visible on the screen. By making commands to the graphics system, the user can make specified states and arcs invisible even when they lie within the region covered by a window.

Highlighting of states and arcs is done by drawing a black background for the text of the state name (for states) or the arc label (for arcs). The black rectangle is drawn by the bitmap terminal in an "exclusive or" mode which has the effect of changing the normally black lettering (on a white background) into white lettering on a black background. (In the Calcomp plots, this highlighting is represented by a rectangular box around the text.)

In order to support the activities of the discourse component, the graphics system maintains a model of what is visible on the screen. This is usually presumed to be shared knowledge between the system and the user. (In the case of a highly cluttered screen, this presumption begins to break down, but the pragmatic interpreter can take this into account. In particular, if the system entertains the hypothesis that a user wants a state highlighted in order to see where it is, the system does not block this hypothesis on the grounds that its position on the screen is presumed to be shared knowledge.) In response to each request for action, the graphics component returns to the discourse component a list of the states and arcs which are visible on the screen after it performs the requested action.

REFERENCES

Allen, James A. Plan-Based Approach to Speech Act Recognition. Ph.D. Thesis. Technical Report No. 131/79. Dept. of Computer Science, University of Toronto, January, 1979.

Austin, J.L. How To Do Things With Words. J.O. Urmson (Ed.), Oxford University Press, 1962.

Brachman, Ronald J. [1978a]. A Structural Paradigm for Representing Knowledge. BBN Report No. 3605. Cambridge, MA: Bolt Beranek and Newman Inc., May, 1978.

Brachman, Ronald J. [1978b]. Theoretical Studies in Natural Language Understanding: Annual Report, 1 May 1977 to 30 April 1978. BBN Report No. 3888. Cambridge, MA: Bolt Beranek and Newman Inc., September, 1978.

Brachman, Ronald J. On the Epistemological Status of Semantic Networks. In Associative Networks: Representation and Use of Knowledge by Computers. N.V. Findler (Ed.). New York: Academic Press, 1979, pp. 3-50.

Bruce, Bertram C. Belief Systems and Language Understanding. Report No. 2973. Cambridge, MA: Bolt Beranek and Newman Inc., January, 1975.

Bruce, B., and Schmidt, C.F. Episode Understanding and Belief Guided Parsing. Presented at the Association for Computational Linguistics Meeting at Amherst, MA (July 26-27, 1974).

Burton, Richard R. Semantic Grammar: An Engineering Technique for Constructing Natural Language Understanding Systems. BBN Report No. 3453. Cambridge, MA: Bolt Beranek and Newman Inc., December, 1976.

Clark, H., and Marshall, C. Definite Reference and Mutual Knowledge. In Proceedings of the Workshop on Computational Aspects of Linguistic Structure and Discourse Setting. Joshi, A.K., Sag, I.A., and Webber, B.L. (Eds.). Cambridge: Cambridge University Press, in press.

Cohen, Philip R. On knowing what to say: Planning speech acts. Ph.D. Thesis. Technical Report No. 118, Dept. of Computer Science, University of Toronto, January, 1978.

Cohen, Philip R. and Perrault C. Raymond. Elements of a Plan-based Theory of Speech Acts, Cognitive Science, Vol. 3, 1979, pp. 177-212.

Cohen, Philip R. and Levesque, Hector J. Speech Acts and the Recognition of Shared Plans. In Proceedings of the Third National Conference of the CSCSI/SCEIO. Victoria, B.C.: Canadian Society for Computational Studies of Intelligence, May, 1980.

Doyle, Jon. Truth Maintenance Systems for Problem Solving. AI-TR-419. Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology, January 1978.

Feldman, Jerome A., and Sproull, Robert F. Decision Theory and Artificial Intelligence II: The Hungry Monkey. Cognitive Science, Vol. 1, No. 2, April, 1977, pp. 158-192.

Fikes, Raymond E., and Nilsson, Nils J. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 1971, 2.

Fikes, Richard E., Hart, Peter E., and Nilsson, Nils J. Learning and Executing Generalized Robot Plans. Artificial Intelligence, Vol. 3, 1972, pp. 251-288.

Genesereth, Michael R. Automated Consultation for Complex Computer Systems. Ph.D. Thesis, Harvard University, September, 1978.

Gordon, David, and Lakoff, George. Conversational Postulates. In Syntax and Semantics, Vol 3: Speech Acts. P. Cole and J.L. Morgan (Eds.). New York: Academic Press, 1975.

Greenfeld, Norton R., and Yonke, Martin D. AIPS: An Information Presentation System for Decision Makers. BBN Report No. 4228. Cambridge, MA: Bolt Beranek and Newman Inc., December, 1979.

Grice, H.P. Meaning. In The Philosophical Review. 1957, 66, 377-388. Reprinted in D.A. Steinberg and L.A. Jacobovits (Eds.), Semantics: An interdisciplinary reader in philosophy, linguistics, and psychology. New York: Cambridge University Press, 1971.

Grosz, Barbara J. The Representation and Use of Focus in Dialogue understandinG. Technical Note 151. Menlo Park: SRI International, July, 1977.

Hendrix, Gary G. Encoding Knowledge in Partitioned Networks. In Associative Networks: Representation and Use of Knowledge by Computers. N.V. Findler (Ed.). New York: Academic Press, 1979, pp. 51-92.

Hintikka, J. Semantics for Propositional Attitudes. In J.W. Davis et al. (Eds.), Philosophical Logic. Dordrecht-Holland: D. Reidel Publishing Co., 1969. Reprinted in L. Linsky (Ed.), Reference and Modality. New York: Oxford University Press, 1971.

Horrigan, M.K. Modelling Simple Dialogues. Technical Report No. 108. Computer Science Dept., University of Toronto, 1977.

Kaplan, S.J., and Joshi, Aravind K. Co-operative Responses: An Application of Discourse Inference to Data Base Query Systems. In Proceedings of the Second National Conference of the CSCSI/SCEIO. Toronto: Canadian Society for Computational Studies of Intelligence, July, 1980, pp. 196-205.

Levesque, Hector J. A Procedural Approach to Semantic Networks. Technical Report No. 105. Toronto: Department of Computer Science, University of Toronto, April, 1977.

Lewis, D.K. Convention: A Philosophical Study. Cambridge, MA: Harvard University Press, 1969.

McAllester, David A. A Three Valued Truth Maintenance System. A.I. Memo No. 473. Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology, May, 1978.

McDonald, David D. Subsequent Reference: Syntactic and Rhetorical Constraints. In Theoretical Issues in Natural Language Processing-2. University of Illinois at Urbana-Champaign, July 25-27, 1978.

Mann, William C., Moore, James A., and Levin, James A. A Comprehension Model for Human Dialogue. In Proceedings of the 5th International Joint Conference on Artificial Intelligence. Cambridge, MA: International Joint Conferences on Artificial Intelligence, August, 1977, pp. 77-87.

Marcus, Mitchell P. A Theory of Syntactic Recognition for Natural Language. Ph.D. Thesis, Massachusetts Institute of Technology, February, 1978.

Moore, Robert C. Reasoning about Knowledge and Action. Ph.D. Thesis. Artificial Intelligence Laboratory, Dept. of Electrical Engineering and Computer Science, MIT, February, 1979.

Morgan, J.L. Two Types of Convention in Indirect Speech Acts. Technical Report No. 52. Urbana: University of Illinois Center for the Study of Reading, July, 1977.

Perrault, C. Raymond, and Allen, James F. A Plan-based Analysis of Indirect Speech Acts. Forthcoming.

Perrault, C. Raymond, and Cohen, Philip R. Inaccurate Reference. In Proceedings of the Workshop on Computational Aspects of Linguistic Structure and Discourse Setting. Joshi, A.K., Sag, I.A., and Webber, B.L. (Eds.). Cambridge, MA: Cambridge University Press, in press.

Quine, Willard Van Orman. World and Object. Cambridge, MA: The M.I.T. Press, 1960.

Sacerdoti, Earl D. A Structure for Plans and Behavior. Ph.D. Thesis, Technical Note 109. Menlo Park, CA: Artificial Intelligence Center, Stanford Research Institute, Menlo Park, CA, August, 1975.

Schank, Roger C., and Abelson, R. Scripts, Plans, Goals, and Understanding. Hillsdale, NJ: Lawrence Erlbaum Associates, 1977.

Schiffer, S. Meaning. Oxford: Oxford University Press, 1972.

Schmidt, C.F., Sridharan, N.S., and Goodson, J.L. The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence. Artificial Intelligence, Vol. 11, Nos. 1,2, August, 1978, pp. 45-83.

Schubert, Lenhart K., Goebel, Randolph G., and Cercone, Nicholas J. The Structure and Organization of a Semantic Net for Comprehension and Inference. In Associative Networks: Representation and Use of Knowledge by Computers. N.V. Findler, (Ed.). New York: Academic Press, 1979, pp. 121-175.

Searle, John R. A Taxonomy of Illocutionary Acts. In Language Mind and Knowledge. K. Gunderson (ed.), U. of Minnesota Press, 1976.

Searle, John R. Intention and Action, ms., 1979.

Searle, John R. Indirect Speech Acts. In P. Cole and J.L. Morgan (Eds.). Syntax and semantics, Vol. 3: Speech Acts. New York: Academic Press, 1975.

Searle, John R. Speech Acts: An Essay in the Philosophy of Language. Cambridge: Cambridge University Press, 1969.

Sellars, R.W. The Essential of Logic. Cambridge, MA: The Riverside Press, 1917.

Sidner, Candace L. Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse. AI-TR-537. Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology, June, 1979.

Smith, Brian C. Levels, Layers, and Planes: The Framework of a Theory of Knowledge Representation Semantics. M.S. Thesis, Massachusetts Institute of Technology, February, 1978.

Strawson, P.F. Intention and Convention in Speech Acts. In The Philosophical Review, v:lxxiii, 1964. Reprinted in Logico-Linguistic Papers. London: Methuen & Co., 1971.

Sussman, Gerald J. A Computational Model of Skill Acquisition. AI-TR-297. Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology, August, 1973.

Van Lehn, Kurt A. Determining the Scope of English Quantifiers. AI-TR-483. Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology, June, 1978.

Webber, Bonnie L. A Formal Approach to Discourse Anaphora. BBN Report No. 3761. Cambridge, MA: Bolt Beranek and Newman Inc., May, 1978.

Wilensky, Robert. Understanding Goal-Based Stories. Research Report No. 140. New Haven: Dept. of Computer Science, Yale University, September, 1978.

Woods, William A. Transition Network Grammars for Natural Language Analysis. Communications of the ACM, Vol. 13, No. 10, October, 1970, pp. 591-606.

Woods, William A., Kaplan, Ronald M., and Nash-Webber, Bonnie L. The Lunar Sciences Natural Language Information System: Final Report. BBN Report No. 2378. Cambridge, MA: Bolt Beranek and Newman Inc., June, 1972.

Woods, William A. What's in a Link: Foundations for Semantic Networks. In Representation and Understanding: Studies in Cognitive Science. D.G. Bobrow and A. Collins (Eds.). New York: Academic Press, 1975, pp. 35-82.

Woods, William A., and Brachman, Ronald J. Research in Natural Language Understanding: Quarterly Technical Progress Report No. 1, 1 September 1977 to 30 November 1977. BBN Report No. 3742. Cambridge, MA: Bolt Beranek and Newman Inc., January, 1978.

Woods, William A. [1979a] Research in Natural Language Understanding: Quarterly Progress Report No. 6, 1 December 1978 to 28 February 1979. BBN Report No. 4181. Cambridge, MA: Bolt Beranek and Newman Inc., February, 1979.

Woods, William A. [1979b] Theoretical Studies in Natural Language Understanding: Annual Report, 1 May 1978 to 30 April 1979. BBN Report No. 4332. Cambridge, MA: Bolt Beranek and Newman Inc., April, 1979.

Woods, William A. Cascaded ATN Grammars. American Journal of Computational Linguistics, Vol. 6, No. 1, January-March, 1980, pp. 1-12.

Zdybel, Frank, Yonke, Martin D., and Greenfeld, Norton R. Application of Symbolic Processing to Command and Control, Final Technical Report. BBN Report No. 3849. Cambridge, MA: Bolt Beranek and Newman Inc., February, 1980.

APPENDIX A
PLAN-RELATED INFERENCE PROCEDURES

This section presumes familiarity with Chapter 5, especially Fig. 11 and the speech act definitions of Section 5.7 and 5.8. This appendix presents the speech act related inference procedures that short-circuit multi-step speech act plans.

If an inference procedure returns NIL, other procedures inherited from "above" in the taxonomy will be invoked. In particular, the more general inferential machinery will be invoked when the "to plan" or "to plan-recognize" procedures attached to ACTION are invoked.

A.1 ACTION-toplan:

Return an expression that, when evaluated, tests the instantiated preconditions (of the relevant action instance) in the planner's beliefs and creates new goals for those preconditions believed to be false. For those whose truth/falsity are unknown to the planner, the expression should also create goals that the planner KNOWIF those conditions hold.

A.2 ACTION-toplanrecog:

Tests the instantiated effects of the action instance and return expressions that insert them into the plan. Those effects already believed to be true will be given a lower rating (arbitrarily set at 15%).

A.3 S-REQUEST-INFORMIF-SYSTEM-CANDO-toplanrecog:

This procedure will be invoked for utterances like "Can you <do action>"?

```
   If  ~MUTUAL-BELIEF(HEARER, SPEAKER,
        (WANT HEARER  ~(effects of filler of ACT role of
                                    SYSTEM-CANDO)))
```

then return an expression to place the filler of the ACT role of the CANDO into context SBUWSBUW. A body inference to a REQUEST will then apply to this new GOAL yielding an interpretation of "Can you ACT" as a REQUEST to ACT. Potentially, a line of inference could leave SBUWSBUW ACT and yield SBUWSBUW ACT2, thus arriving at a REQUEST to do ACT2.

If it is shared knowledge that the user does not want the effects of the act, then the system is to treat this utterance literally. Consequently, this procedure fails, allowing the ACTION-toplanrecog to expand S-REQUEST by its effect and proceed as per usual.

A.4 S-REQUEST-INFORMIF-SYSTEM-KNOWREF-toplanrecog:

This procedure will be invoked for utterances like "Do you know <description>"? If it is mutually believed between the hearer (system) and speaker (user) that the hearer KNOWREF <description> (e.g., it might be shared knowledge that the system knows the sizes of the constituent networks), then this procedure returns an expression to put an INFORMREF act into SBUWSBUW. A body inference to REQUEST may then classify this goal as a REQUEST to INFORMREF of <description> and infer that the user asked the wh-question "what is <description>?"

If no such mutual belief exists between system and user, the procedure fails, allowing the more general routines to expand the S-REQUEST by its effect, leading to a literal interpretation.

A.5 S-INFORM-USER-WANT-toplanrecog:

If the filler of the WANTED role of WANT describes an action (ACT) to be performed by the system, then this procedure places that action ACT into SBUWSBUW. If ACT is to be performed by the user, then fail, thus allowing the standard machinery to analyze the utterance as a literal INFORM. If the filler describes a state-of-affairs (e.g., that an ATN-STATE be visible), then the filler is placed in SBUWSBUWSW. As the example of Section 5.17 shows, this leads to the system´s planning to achieve that state-of-affairs. Currently, this inference is unconditional since the system always accepts the user's goals as its own.

Official Distribution List
Contract N00014-77-C-0378

	Copies
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
Office of Naval Research Information Systems Program Code 437 Arlington, VA 22217	2
Office of Naval Research Code 200 Arlington, VA 22217	1
Office of Naval Research Code 455 Arlington, VA 22217	1
Office of Naval Research Code 458 Arlington, VA 22217	1
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, MA 02210	1
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, IL 60605	1
Office of Naval Research Branch Office, Pasadena 1030 East Green Street Pasadena, CA 91106	1
Office of Naval Research New York Area Office 715 Broadway - 5th Floor New York, NY 10003	1

Naval Research Laboratory 6
Technical Information Division
Code 2627
Washington, D.C. 20380

Naval Ocean Systems Center 1
Advanced Software Technology Division
Code 5200
San Diego, CA 92152

Dr. A.L. Slafkosky 1
Scientific Advisor
Commandant of the Marine Corps (Code RD-1)
Washington, D.C. 20380

Mr. E.H. Gleissner 1
Naval Ship Research & Development Center.
Computation & Mathematics Dept.
Bethesda, MD 20084

Captain Grace M. Hopper 1
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D.C. 20350

Mr. Kin B. Thompson 1
NAVDAC 33
Washington Navy Yard
Washington, D.C. 20374

Advanced Research Projects Agency 1
Information Processing Techniques
1400 Wilson Boulevard
Arlington, VA 22209

Captain Richard L. Martin, USN 1
Commanding Officer
USS Francis Marion (LPA-249)
FPO New York 09501

Director 1
National Security Agency
Attn: R54, Mr. Page
Fort G. G. Meade, MD 20755